Intra-Shard and Inter-Shard Collaborative Incentive Scheme for Shard-Based Permissionless Blockchain

Heng Quan
quanheng@bupt.edu.cn
School of Artificial Intelligence, Beijing University of Posts and Telecommunications, Beijing 100876, China, Key Laboratory of Universal Wireless Communications, Ministry of Education, Beijing University of Posts and Telecommunications, Beijing 100876
Beijing, China

Tianyu Kang
kangtianyulm@bupt.edu.cn
School of Artificial Intelligence, Beijing University of Posts and Telecommunications, Beijing 100876, China, Key Laboratory of Universal Wireless Communications, Ministry of Education, Beijing University of Posts and Telecommunications, Beijing 100876
Beijing, China

Li Guo*
guoli@bupt.edu.cn
School of Artificial Intelligence, Beijing University of Posts and Telecommunications, Beijing 100876, China, Key Laboratory of Universal Wireless Communications, Ministry of Education, Beijing University of Posts and Telecommunications, Beijing 100876
Beijing, China

ABSTRACT

Shard-Based blockchain technology is currently one of the effective ways to improve the throughput of permissionless blockchain. By distributing transactions into multiple shards, the validators in each shard verify transactions and reach in consensus in parallel, thereby increasing system throughput and scalability. However, most of the existing Shard-Based Blockchain schemes lack an incentive scheme that can promote mutual cooperation among the nodes in the shard and participate in consensus as much as possible, which leads some selfish nodes to adopt strategies that are not conducive to the maintenance of the blockchain system in order to obtain higher profits, thereby endangering the stability and security of the entire blockchain system. In this paper, we introduce the concept of node reputation to evaluate node behavior, and establish the correlation between reputation and profit. In that case, nodes with higher reputation can get more rewards, thereby achieving the fairness of reward distribution. In addition, we use the N-player static game model in game theory to analyze the changes in utility corresponding to different strategies for node, and design a dynamic transaction distribution scheme through genetic algorithm(GA) to ensure that the majority of nodes participate in consensus and shard liveness. Our research results show that our solution can better improve incentive fairness and liveness compared to the general transaction uniform distribution scheme.

CCS CONCEPTS

• **Computing methodologies** → **Distributed computing methodologies**; *Modeling and simulation.*

*Li Guo is corresponding author.

ICCIP 2021, December 16–18, 2021, Beijing, China

ACM ISBN 978-1-4503-8519-0/21/12…$15.00
https://doi.org/10.1145/3507971.3507995

KEYWORDS

Blockchain, Sharding System, Game Theory, Incentive Scheme, Reputaion, Genetic Algorithm, VRF

ACM Reference Format:
Heng Quan, Tianyu Kang, and Li Guo. 2021. Intra-Shard and Inter-Shard Collaborative Incentive Scheme for Shard-Based Permissionless Blockchain. In *2021 the 7th International Conference on Communication and Information Processing (ICCIP) (ICCIP 2021), December 16–18, 2021, Beijing, China.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3507971.3507995

1 INTRODUCTION

At present, the implementation of blockchain generally has defects in performance and capacity. For instance, the maximum transaction throughput of Bitcoin is about 7 transactions per second (tps), and it needs to wait for the trusted confirmation time of 6 blocks, while the transaction throughput of Ethereum is about 20 tps, with an average delay of 12 seconds [9]. Compared with the thousands of tps in centralized payment systems such as Visa, the existing blockchain systems still have a certain gap. Therefore, blockchain expansion technology has become one of the important directions of current blockchain research.

One of the on-chain expansion technologies is sharding [2, 7, 8, 18]. It improves the scalability of the system by dividing all nodes in the network into several sets, also called shards, and each verifies transactions and consensus blocks independently [10]. Therefore, sharding is regarded as one of the effective ways to improve the scalability of the blockchain and break through the performance bottleneck of the "impossible triangle" of the blockchain. However, at this stage, the security and efficiency issues of the consensus layer are more fully studied in sharding system, but there is less analysis and research on the incentive layer. In sharding system, nodes are evenly distributed in each shard, so most mainstream sharding schemes including Elastico [11], Omniledger [6], RapidChain [20] all adopt a classical Byzantine consensus protocol such as PBFT [1] to improve the scalability and consistency of the consensus. However, due to the lack of research on incentive scheme, when PBFT is applied to a permissionless blockchain system, it will face the problem of economic incentives, and it is impossible to ensure

that selfish nodes correctly participate in the consensus protocol. This problem is more prominent in the permissionless sharding system. Since the number of nodes in the shard is small, if most selfish nodes choose to invest in very few computing resources to participate in the consensus or not to participate in the consensus, then the shard probably fail to generate blocks in time, which will affect the efficiency, availability, and security of the entire sharding system.

In order to solve the problem that the permissionless sharding system cannot effectively incentivize nodes to participate more in consensus, we propose an intra-shard and inter-shard collaborative incentive scheme based on the analysis of selfish node strategy. The reward distribution scheme and the dynamic optimization distribution scheme of inter-shard transactions encourage selfish nodes to participate and invest more computing resources in the consensus process to the greatest extent, which will improve the incentive and liveness of the entire sharding system. Specifically, the contributions of this paper can be briefly summarized as follows:

- **Intra-shard reward distribution scheme:** We analyze the selfish node strategy through the static game model in game theory, and use the node consensus time to quantify the amount of computing resources invested by the node during the current round of block consensus, so as to guide the design of the intra-shard reward distribution scheme and incentivize nodes put more computing resources into system maintenance to improve the efficiency of consensus.
- **Inter-shard optimized transactions distribution scheme:** We search for the optimal transaction distribution by designing a multi-objective genetic algorithm(GA) to maximize incentives for selfish nodes in shards to implement consensus protocols, and at the same time make the number of consensus nodes participating in each shard relatively balanced, which will improve the fairness and liveness between shards.
- **Distributed consensus scheme of GA:** The Inter-shard optimized transactions distribution scheme introduces the verifiable random number generated by the VRF as the random seed of the GA to ensure that each node can verify the execution result of the GA, thereby ensuring the consistency and security.

2 RELATED WORK AND MOTIVATION

2.1 Shard-based Permissionless Blockchain Architecture

In sharding technology, assigning nodes to multiple shards (network sharding) is the most basic sharding method. For the early sharding protocols Elastico [11] and Zilliqa [15], PoW puzzle was used to generate the identity of the unauthorized node, and then the blockchain nodes were divided into different shards according to the last few bits of the node identity, that is, to ensure that the last few bits of the node's identity in the shard are the same. If the number of nodes in each shard reach a certain threshold range, the node will broadcast its identity to the entire network. In conclusion, the form of Elastico network sharding is a relatively random way, but this randomness is weakly random in nature and predictable, related scholars have proposed to use some verifiable functions in cryptography to generate random numbers. The method provides more secure randomness for network fragmentation. Omniledger [6] generates unbiased verifiable random numbers by combining verifiable random function (VRF) [13] and RandHound [14]. Similarly, Algorand [4] relies on the VRF-based cryptographic sortition to select the group of consensus validators. In addition, RapidChain [20] simply lets each node perform Verifiable Secret Sharing(VSS) [3] and using the combined secret shares as the resulting randomness.

In addition, on the basis of network sharding, how to distribute transactions submitted by users is a very important issue. In the account/balance model, most of the sharding systems such as Zilliqa are based on the transaction sender's address as a rule for distribution, while in the UTXO model, each transaction is assigned by the hash. However, no matter in which accounting model, transactions are distributed according to unbiased rules, which is essentially a random process, so the number of transactions distributed by each shard is theoretically evenly distributed. However, this scheme does not consider the impact of the number of transactions in the shard on the node strategy from the incentive layer, and also does not consider the impact of different transaction distribution schemes on the fairness and activity of the entire sharding system, so it cannot play an incentive role.

2.2 Game analysis of Shard-based Permissionless Blockchain

In Shard-based Permissionless Blockchain, one key research gap is a lack of understanding of the game analysis of rational nodes and transactions distribution strategy. Such an understanding is critical for designing an appropriate incentive scheme, which will foster the cooperation inter shards and the cooperation among nodes within the same shard [12]. Furthermore, assuming that each node in the shard is rational and selfish, then when the node's expected return (such as block reward, transaction fee, etc.) is less than the node input cost (such as consensus cost, validation cost, communication cost, etc.), the node may choose to be passive, so as not to participate in the verification of transactions within the shard or the consensus of the block. If this situation increases, it will seriously affect the consensus efficiency and block generation speed of the sharding system. To solve this problem, the GTSB scheme [12] proves that participation is a Nash Equilibrium if the reward is divided equally among all participants. Thus, it can encourage nodes to cooperate with each other and improve fairness when compared to the equal distribution. In the GTSB scheme, since it is assumed that the nodes have the same computing power and computing capacity, it does not consider the heterogeneity of the input of all participating cooperative nodes. However, in the actual blockchain system, the computing input of the nodes participating in the consensus is due to the equipment differences, network stability and other factors. Therefore, if the reward is evenly divided among participants, the node cannot be motivated to invest more computing resources, and it will also affect the fairness of the incentive.

2.3 Reputation based node performance evaluation model

As aforementioned, even if a node participates in the process of verification and consensus, the performance of each node is different

due to the different computing resources invested, so it is necessary to quantitatively evaluate the difference in node performance. At this stage, there are many studies that introduce the reputation scheme into sharding system to evaluate the consensus behavior of nodes. For example, TBSD [19] obtains the reputation of each node by recording and calculating the consensus voting results of each node. This is used for the next round of network sharding, thereby reducing the probability of malicious nodes gathering. Similarly, RepChain [5] propose a new double-chain architecture which includes transaction chain and reputation chain. The transaction chain is used to record transaction information and run Raft based synchronous consensus to form transaction blocks, while the reputation chain records votes on transactions by nodes. As a result, the reputation is calculated, thereby forming a reputation block and running PBFT for consensus.

However, the above-mentioned studies all calculate the reputation by recording the results of node consensus voting. In this paper, we pay more attention to how to quantify the difference in computing resources invested by nodes. According to the distribution of user processor frequencies published by Steam in April 2019, there are a large number of devices with different performances in the user group with a relatively uniform distribution [17]. This shows that in the actual blockchain system, the hardware costs invested by each node are different, which leads to different node computing capabilities and different speeds of verifying transactions. In addition, how much the node invests in network maintenance costs will also affect the stability of the network. If network delay occurs, it will directly affect the speed of consensus [16]. Based on the above analysis, we can quantify the amount of computing resources invested of the node by recording the time for transactions verification, thereby forming the node reputation as the basis for reward distribution.

2.4 Summary

From the above analysis, we need to use the computing resources invested by the node as a factor for game analysis, and use the time of the node to verify the transactions as evaluation input basis, quantify the performance of the node to form the node reputation, and guide the design of the reward distribution scheme in the shard. In addition, we also need to find the optimal distribution of transaction among shards based on the above reward distribution scheme, so as to reduce the impact of the low activity and incentive caused by the uniform distribution of transaction.

3 SYSTEM MODEL

3.1 Shard model

A Shard-Based Blockchain partitions the nodes in the network into smaller shards. Each shard processes a disjoint set of transactions in parallel and group into blocks, which will overcome the scalability limits of blockchain systems. In this paper, we consider that there are $k+1$ shards, i.e., $S = \{S_1, S_2, \ldots, S_{k+1}\}$ where $S_{1 \le i \le k}$ are normal shards and S_{k+1} is a service shard. In brief, normal shard is mainly responsible for transaction verification, the record of each node's verification time and the generation and consensus of sub-blocks. Service shard is mainly responsible for network shard, transaction distribution, node reputation calculation, final block generation and consensus. Specially, service shard is a semi-centralized architecture in which nodes are fixed and authorized to serve normal shards. Therefore, we do not consider the selfishness of the nodes in service shard. However, as for nodes in normal shards, we assume that all nodes are unauthorized, selfish (i.e., profit-driven) and non-malicious. Further, we assume that the computing resources invested by all nodes are different, that is, each node has different computing capabilities.

In this paper, We learn from the consensus protocol of NRSS [17] and some other shard-based blockchain architectures [6, 11] to form our shard model. For ease of exposition, the main steps at an epoch are summarized from our shard model as in Fig 1:

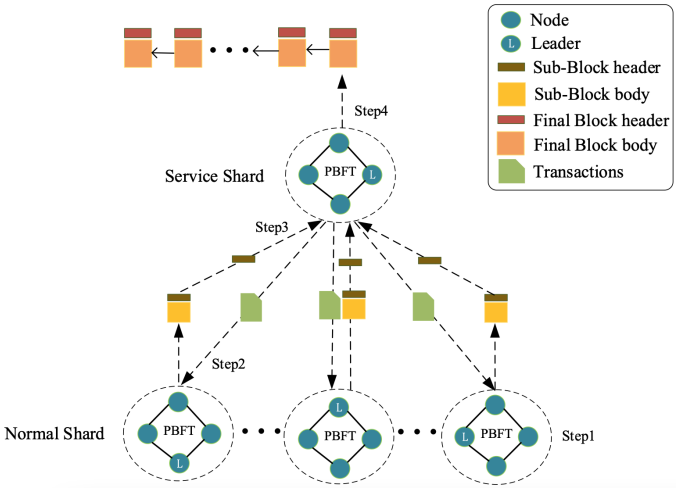

Figure 1: Shard Model

(1) Shard Formation: Service shard runs the VRF algorithm based on public information to generate a public unbiased verifiable random number for this epoch. In the previous epoch, the registered nodes generate a public verifiable identity by solving the PoW puzzle generated by the current random number. After that, each node is randomly and evenly distributed to each normal shard based on the last few bits of PoW result information, and the nodes belonging to the same shard are discovered by communicating with other nodes in a fully connected way.

(2) Transactions Distribution: In this paper, we assume that the total transaction received by the service shards is T, and the proportion of the transaction received by each shard is $\partial_{1 \le i \le k}$. When the service shard receives the transaction sent by the user, it needs to run the VRF-based GA algorithm to find the optimal transaction distribution strategy (i.e., $\partial_{1 \le i \le k}$), and allocate the transactions to normal shards.

(3) Sub-Block generation: Each normal shard will run the PBFT consensus protocol to verify the transaction after receiving that. When the leader receives most valid signatures, the formed sub-block will be sent to service shard to form final block. However, in this paper, we need to record the verification time of each node to form reputation, so as to characterize the resources which have been invested in the current round of block generation. Therefore, we learn from the consensus process of NRSS [17]. When the leader

packs the transaction and sends it to other nodes, it needs to record the timestamp t_{send} at this time. Then, when each node returns the signature to the leader, the leader will record the received time r_{recv}, so as to obtain the time length $r_{recv} - t_{send}$ for each node to verify the transactions. This duration will also be sent to the service shard along with the sub-block header.

(4) Final-Block generation: After the service shard receive the sub-block headers returned from each normal shard and the verification time length of each node, they will be integrated and the reputation corresponding to each node will be calculated according to the unified reputation calculation model, and then PBFT will be run to package the reputation and generate final block, then it will be broadcast throughout the network. At the same time, the nodes in normal shards update the reputation of other nodes simultaneously, and get corresponding rewards according to the reward distribution scheme.

3.2 Reputation model

The reputation is obtained by recording and calculating the behavior information of the node, which is used to characterize the node's credibility and confirm the discourse right of node. In this paper, reputation is calculated by service shard through the verification of transaction time information sent by each normal shard. The formula is shown in (1):

$$r^{*i}_j = \log\left(1 + \frac{\overline{t_i}}{t^i_j}\right) \tag{1}$$

where

$$t^i_j = t_{recv} - t_{send} \tag{2}$$

$$\overline{t_i} = \frac{\sum_{j=1}^{n_i} t^i_j}{n_i} \tag{3}$$

In (1), r^{*i}_j represents that the original reputation of node j in this round of block generation in shard S_i. In (2), t^i_j represents the time for node j to verify the transaction, t_{send} denotes the time stamp when the leader sends the transaction to node j, and t_{recv} denotes the time stamp when the leader receives the signature of the transaction from node j. In (3), $\overline{t_i}$ denotes the average time of all nodes verifying transactions in shard S_i, and n_i is the size of S_i. In addition, we use the logarithmic formula to appropriately reduce the variance of the reputation and avoid some extreme situations.

However, the reputation also needs to be normalized, so that the reputation of different nodes are in the same order of magnitude, which is more convenient for comparison. Therefore, we use Min-Max Normalization to normalize the reputation, as shown in (4):

$$r^i_j = \frac{r^{*i}_j - r^i_{min}}{r^i_{max} - r^i_{min}} \tag{4}$$

We can easily observe that r^i_{max} is the highest reputation in shard S_i, and r^i_{min} is the lowest reputation in shard S_i. At this point, the nodes whose signatures are included in the final block all have a quantifiable reputation that can evaluate the amount of computing resources they have invested in this round after the block is generated. The remaining nodes that have not participated in the consensus process or whose signatures have not been packaged into the block by the leader have no reputation. But for the leader in each shard, we consider their reputation to average, because they do other work (i.e., packaged transactions).

3.3 Game analysis of rational nodes

From the consensus process in our hypothetical sharding model, it can be seen that after receiving the transaction sent by the leader, each node makes a strategic choice at the same time and independently, that is, first choose whether to participate in this consensus process. If you choose to participate, then the node need to choose how much computing resources to invest, so our hypothetical game is essentially a Non-cooperative static game G.

Further, we need to clarify the utility function of each node when choosing different strategies, so we need to assume and define the reward and loss of the node. For loss part, it is divided into necessary loss L^0 and unnecessary loss L^1. Among them, L^0 refers to the loss in the shard generation process, which is the loss that a node joins the sharding system. In our sharding system, nodes need to solve the PoW puzzle to enter the shard, so we assume that the loss of this part is L^S. After that, the node communicates with other nodes in a fully connected network, and finds other nodes that belong to the same shard. In this process, we assume that L^I is the loss of identity authentication between individual nodes, and the final identity authentication loss is related to the number of nodes in the shard. We assume that n_i is the number of nodes in the shard S_i, so the loss during the identity authentication process of the node joining the sharding system is $L^I(n_i - 1)$, and the necessary loss L^0 is shown in (5):

$$L^0 = L^S + L^I(n_i - 1) \tag{5}$$

In shard-based consensus protocol, After the node receives transactions sent by leader, it needs to make a strategic choice, that is, whether to participate in the consensus process and how much computing resources to invest. If the node chooses not to participate, it does not have to bear the non-essential loss L^1, otherwise it needs to bear it. In the process of transaction verification and consensus, we consider that L^1 is proportional to the amount of computing resources invested by the node, that is, the more computing resources the node invests, the more L^1 will increase accordingly. In addition, the number of transactions verified by the node also directly determines the size of L^1. We assume that the node input computing resources is x, the verification loss for each transaction is L^V, and the consensus loss is L^C. At the same time, it can be known from the sharding model that the transaction volume of the shard S_i is $\partial_i T$, so the unnecessary loss L^1 is shown in (6):

$$L^1 = x\partial_i T\left(L^V + L^C\right) \tag{6}$$

In sharding system, if the node participates in consensus, we consider that the reward that node receives has two parts, namely the block reward and the transaction fee. The block reward is generated after the final block is generated. Let's say it is R_b. The transaction fee is charged by the user when the user conducts a cryptocurrency transaction. It is used to reward the verification and consensus of the transaction by the node. Assuming the cost of a transaction is f. In addition, we also consider the benefits of nodes not participating in the consensus. This part of the revenue is generated by computing resources that should be invested in the consensus process to

do other tasks. We assume that the revenue generated by a unit of computing resource is σ, Therefore, the overall benefit of this part is σx, and the utility function when the node chooses not to participate in the consensus strategy is shown in (7):

$$U_j^i(I) = \sigma x - \left[L^S + L^I\left(n_i - 1\right)\right] \tag{7}$$

However, how to distribute the rewards generated in sharding system is a key issue in the study of incentive schemes. In the GTSB scheme [12], it is verified that the fair distribution of rewards among all nodes participating in the consensus is fairer and more motivating than the equal distribution among all nodes. Therefore, if we adopt this fair distribution method, the utility function when nodes choose to participate in the strategy is as shown in (8):

$$U_j^i(P) = p_s\left(\frac{\partial_i T f}{p_i} + \frac{R_b}{\sum_{i=1}^{k} p_i}\right) - \left[L^S + L^I\left(n_i - 1\right)\right] - x\partial_i T\left(L^V + L^C\right) \tag{8}$$

Since there is currently no perfect way to find and uncontroversially prove who is not the node that actually participates in the consensus in the process of running PBFT, there are cases where the leader intentionally or unintentionally does not package the signature of the consensus node. So we assume that p_i is the number of nodes whose signatures are packaged by the leader in the shard S_i, which is used to represent the approximate number of nodes participating in the consensus, and p_s is the probability that the signature is selected by the leader.

Further, when we derive $U_j^i(P)$ with respect to the input computing resource x, we get (9):

$$\frac{dU_j^i(P)}{dx} = -\partial_i T\left(L^V + L^C\right) \tag{9}$$

In summary, $\frac{dU_j^i(P)}{dx}$ is a negative number, so the utility value when a node chooses to participate in the strategy decreases with the increase of the input computing resource x. In other words, using the least computing resources is a Nash Equilibrium strategy for selfish nodes, which will slow down the node verification transaction time and affect the efficiency of the entire sharding system. Therefore, in the following section we will propose a new type of reward distribution scheme, which can safely and effectively incentivize nodes to invest more computing resources and improve the system consensus speed and block generation efficiency.

3.4 Reward allocation within shard

Fair distribution of rewards cannot motivate nodes to invest more computing resources. The essence of the problem is that the correlation between rewards and computing resource input is not established. If the node invests more and gets more rewards, then selfish nodes will do their best to invest in order to obtain benefits. In the section of reputation model, we designed a node reputation calculation model based on transaction verification time to accurately quantify and evaluate the amount of computing resources invested by nodes. Therefore, we can incorporate the node reputation into the consideration factor of reward distribution. At the same time, we also need to consider the security of reward distribution, that is, we need to consider the situation of nodes being malicious in the incentive layer.

Based on the above analysis, we will propose a fairer and safer intra-shard reward distribution scheme. Specifically, we believe that the reward distribution of the leader and other nodes in the shard is different (specific analysis is in the Security Arguments). For other nodes, the reward is only the transaction fee and will be based on the proportion of each node's reputation. The utility function is shown in (10):

$$R_{j\neq L}^i = p_j^i \partial_i T f \tag{10}$$

where

$$p_j^i = \frac{r_j^i}{\sum_{j=1}^{n_i} r_j^i - r_{j=L}^i} \tag{11}$$

Specially, p_j^i represents the proportion of the reputation for node j after the previous round of block generation in the shard S_i. It is worth noting that the reputation after the current round of block generation needs to be calculated by the service shard. Therefore, when the node makes a strategy choice, the reputation after the current round of block generation cannot be predicted, so it is impossible to judge the utility of participating in consensus. Therefore, the reputation of the previous round is selected as the basis for this round of reward distribution to ensure the availability of game analysis. On the other hand, when calculating the proportion of reputation, the total reputation of nodes in the shard does not include the reputation of leader, because if it is included, the leader may tamper with the timestamp of other nodes for profit.

For the leader in the shard, the reward is only the block reward, which is related to the number of signatures packaged into the sub-block, and when setting the parameters, it is necessary to ensure that the block reward obtained by each leader is greater than the transaction fee obtained by other nodes, such as (12)(13) shows:

$$R_{j=L}^i = q_j^i R_b \tag{12}$$

$$s.t.\ R_{j=L}^i > R_{j\neq L}^i \tag{13}$$

where

$$q_j^i = \frac{s_j^i}{\sum_{i=1}^{k} s_j^i} \tag{14}$$

Among them, s_j^i is the number of signatures packed into sub-blocks by the leader in the shard S_i, and q_j^i represents the proportion of the number of signatures collected by the leader among the leaders of all normal shards. Such a design can maximize the incentive for the leader to pack as many signatures of other nodes as possible, and reduce the influence of collusion between the leader and other nodes. At the same time, the design that allows the leader to obtain more revenue than other nodes will ensure the fairness of the reward distribution, because the leader does more tasks (i.e., packaged transactions) than other nodes in the consensus process.

Security Arguments: Since we assume that the nodes are all selfish, there are nodes that do malicious things in the incentive layer of our sharding system in order to get more rewards, and implement self-interested but unfavorable behaviors for other nodes, which affects the ecology of the entire system. In this paper, the time for each node in the shard to verify the transaction is determined by the leader. In essence, whether the leader is honest or not directly affects the reward of other nodes. Therefore, we need to focus on the leader's behavior, so that the leader's behavior is

not profitable or the profit is small, so as to indirectly ensure the interests of each node. Specifically, we consider the leader's two evil modes, namely:

Mode A: The leader maliciously tampered with the timestamp of the transaction verified by other nodes.

Mode B: The leader colludes with other nodes and deliberately does not include the signatures of some honest nodes.

For Mode A, in the reward distribution scheme we designed, it is a good guarantee that the leader has no incentive to maliciously tamper with the timestamp of other nodes. Specifically, the reward allocated to the leader is only the block reward, and the transaction fee only allocated to other nodes. Therefore, even if the leader changes the timestamp of other nodes, it will not affect its reward. In contrast, if the leader also divides the transaction fee based on the reputation, then the leader can increase its reputation by maliciously modifying the timestamps of other nodes to obtain more rewards.

For Mode B, the method of conspiracy to be malicious is almost difficult to be discovered due to its concealment. Therefore, we cannot identify it through consensus, and can only reduce the bad influence of this way through incentives. In short, we consider that the final block reward obtained by the leader is positively correlated with the number of packaged signatures. At the same time, the block reward is greater than the transaction fees obtained by other nodes. In this way, the best way for a leader to increase reward is to pack as many signatures from other nodes as possible, instead of conspiring to be malicious and sharing rewards with partner nodes.

After the above analysis, we propose a fair and safe reward distribution scheme, which realizes an effective incentive to invest more computing resources in nodes within the shard. But in the sharding system, in order to design a good incentive scheme, it is necessary to find a reasonable transaction distribution scheme among shards. Therefore, in the next section, we will pay attention to the inter-shard incentive and then propose a new transactions distribution protocol of VRF-based GA, which securely encourages selfish nodes in the shard to participate in the consensus protocol.

3.5 Transactions distribution protocol of VRF-based GA

In this section, our next goal is to find a transaction distribution strategy that meets the expectations. Using GA's algorithm characteristics, we only need to define fitness function expressions, chromosomes, algorithm parameters and other information to find the optimal transaction distribution strategy through iterations. In terms of expectations, we hope to maximize the number of nodes participating in the consensus in the system and the relative balance of the number of nodes participating in the consensus among shards, so that on the one hand, more nodes can be encouraged to participate in the consensus, and on the other hand, it can be guaranteed the fairness of incentives among shards and has a similar number of nodes participating in the consensus, thereby improving the availability of the system.

However, there are two significant issues that need to be discussed before implement our algorithm. The first issue is how the service shard determines the strategy choice of each node in the current consensus process. Second, GA is a nondeterministic algorithm. Each node in the service shard will calculate different results separately. Therefore, how to safely and effectively ensure the consistency of GA results is very important.

With regards to the first issue, service shard can predict the strategy adopted by the nodes in the current round of consensus by playing the game instead of each node. Specifically, after the analysis of the reward distribution scheme in the previous section, we can finally get the utility function of nodes in the consensus process to choose different strategies, as shown in (15) (16):

$$U_j^i(P) = P_s P_j \partial_i T f - \left[L^S + L^I (n_i - 1)\right] - x\partial_i T \left(L^V + L^C\right) \quad (15)$$

$$U_j^i(I) = \sigma x - \left[L^S + L^I (n_i - 1)\right] \quad (16)$$

In the game process, we assume that the common information parameter value faced by the node is certain, then the only thing that actually affects the game result is that the node invests in computing resource x, the proportion of reputation P_j, and the proportion of transaction distribution ∂_i. For x, assuming that through our reward distribution scheme and appropriate parameter settings, it can be guaranteed that the more nodes invest, the more benefits they get, so it can be assumed that x is a fixed value. For P_j, since we use the reputation after the last round of block generation as the basis for the current round of reward distribution, the proportion of the reputation is a certain value that all nodes know. And ∂_i is the value we need to determine through GA. Therefore, service shard can predict the strategy of nodes in the current round of consensus based on the utility functions of different strategies.

With regards to the second, the non-determinism of GA is caused by the randomness when we initialize chromosomes and perform chromosome crossover and mutation. However, this randomness is essentially a pseudo-random. Therefore, as long as the random seed used in the GA process is determined, the entire operation of GA and calculation results are deterministic, so that service shard can form a consensus on the transaction distribution strategy output by GA. However, the random seed generation environment must be secure and have unbiased and verifiable characteristics. Therefore, we select the random number generated based on the VRF when used to form the shard as the random seed of the GA to ensure the distributed operation result of the GA consistency.

After discussing the two issues above, we have a clearer solution. Next, we will analyze and design the GA model. First of all, the most important thing is to define the chromosome. The goal of service shard to run GA is to find the optimal transaction distribution strategy, so this needs to be transformed into the form of chromosome, as shown below:

$$C = (C_1, C_2, \ldots, C_k) = (\partial_1, \partial_2, \ldots, \partial_k) \quad (17)$$

$$s.t. \sum_{i=1}^{k} \partial_i = 1.0 \quad (18)$$

C represents the chromosome, which is composed of the proportion of the transaction distributed to each shard ∂_i. After having the chromosome, we have to define another important part of the fitness function. Since there are two objective expectations, the GA we designed is essentially a multi-objective optimization problem. Multi-objectives can be merged into a single objective through

mathematical methods. The most commonly used method is the weight coefficient conversion method, which assigns each objective function weights and combine them into one objective function, then the single objective function after fusion is:

$$\min F = \omega_1 F_1 + \omega_2 F_2 \tag{19}$$

where

$$F_1 = \sqrt{\frac{\sum_{i=1}^{k} (p_i - \bar{p})}{k}} \tag{20}$$

$$F_2 = \frac{\sum_{i=1}^{k} v_i}{\gamma} \tag{21}$$

In (19), F is the fitness function expression of GA, F_1 and F_2 are two different objective functions, ω_1 and ω_2 are the corresponding weight coefficients. In (20), F_1 represents the root mean square error (RMSE) of the nodes participating in the consensus among shards, where p_i is the number of nodes participating in the consensus of the shard S_i, and $\bar{p}$ is the average number of nodes participating in the consensus of the sharding system. In (21), F_2 represents the total number of nodes which choose invalid in the entire sharding system. Among them, v_i is the number of invalid nodes in the shard S_i, and γ is a control variable used to control the range of F_2.

In summary, after the definition of chromosome and fitness function, we can design and implement the transactions distribution protocol of VRF-based GA. The detailed process of the protocol is shown in Algorithm 1.

Algorithm 1 Transactions distribution protocol of VRF-based GA

function SEED($message$)
  $HashValue \leftarrow HASH(prikey, message)$
  $Randomness \leftarrow HASHTOINT(HashValue)$
  return $Randomness$
end function
function GA ($Randomness$, M, P_C, P_M, ϕ)
  $Seed \leftarrow Randomness$
  Population initialization process
  Randomly generate M chromosomes to form parent set
  $F \leftarrow (C_1, C_2, \ldots, C_M)$
  Populations iteration process
  while $N_U > \phi$ **do**
    Crossover process
    Randomly select chromosomes with P_C as the probability for gene crossover
    Output $F \xrightarrow{\gamma} O$
    Mutation process
    Randomly select chromosomes with P_M as the probability for gene mutation
    Output $O \xrightarrow{\delta} O^*$
    Selection process
    Select the M chromosomes with the lowest fitness as the parent set for the next iteration
    Output $O^* \xrightarrow{\epsilon} F^*$
  end while
  return C_O
end function

In Algorithm 1, we assume that the chromosome crossover operator is γ and the probability is P_C; the chromosome mutation operator is δ and the probability is P_M; the population size is M; the selection operator is ϵ; the optimal chromosome after each iteration is C_O, The number of times the fitness is unchanged is N_U, and the threshold is φ; the set of parents is F, the set of offspring after crossover is O, the set of offspring after mutation is O^*, and the next round of parent set after selection is F^*. Further, the calculation process of the protocol consists of two functions, one part is using random number generated based on VRF as the random seed of GA; the other part is the population iteration process of GA, which mainly includes crossover, mutation, and selection of chromosomes. When the fitness of the optimal chromosome no longer rises, that is, when the number of invariants N_U reaches the threshold φ, the final result will be output.

4 PERFORMANCE EVALUATION

In this section, we combine Golang and Python to implement a simulation model and analyze the transactions distribution protocol of VRF-based GA performance. Specifically, our analysis focuses on three evaluation metrics: (i) Changes in fitness during GA process; (ii) RMSE of the nodes for participating in the consensus among shards; (iii) Consensus participation rate of nodes in the sharding system. Then, we compare the performance of our protocol with uniform distribution of transactions. The main related simulation parameters are shown in Table 1.

According to the simulation results, the performance of GA is evaluated mainly to demonstrate the incentive of our protocol.

Table 1: Fixed Parameters Used In The Simulation

	Parameter	Value	Description
Game Related	σ	2.5	The profit for doing other things
	x	0.8	The investment of Computing resource
	n_i	100	The number of nodes in a shard
	∂_i	0.0-1.0	The ratio of transactions in a shard
	T	10000	The total number of transactions
	f	0.25	The fee for per transaction
	P_s	0.67	The probability of signature being selected
	P_j	0.0~1.0	The proportion of node reputation in the shard
	L^I	0.01	The loss of P2P Identity authentication
	L^S	1.0	The loss for solving PoW puzzle
	L_V	0.0005	The loss of validation for per transaction
	L_C	0.0003	The loss of consensus for per transaction
GA Related	M	30	The number of chromosomes
	P_C	0.6	The probability of crossover
	P_M	0.2	The probability of mutation
	φ	200	The threshold of invariant fitness

Figure 2 shows the changes in GA's fitness during the population iteration process. We assume that the reputation of each node is randomly generated. At the beginning, the fitness is about 5.678. At this time, the number of consensus nodes in five shards is (65, 56, 63, 57, 68) respectively. With the iteration of the population, the chromosomes with high fitness are gradually eliminated, so the optimal fitness in the population gradually becomes smaller, and stabilizes after 150 generations and no longer becomes smaller, until the threshold of constant fitness is met. Finally, the transaction

distribution scheme we found makes the number of participating consensus nodes in the five shards (63, 64, 63, 64, 63) respectively. It can be seen that compared with the initial result, the total number of participating consensus nodes and the index of the RMSE between shards have been significantly improved.

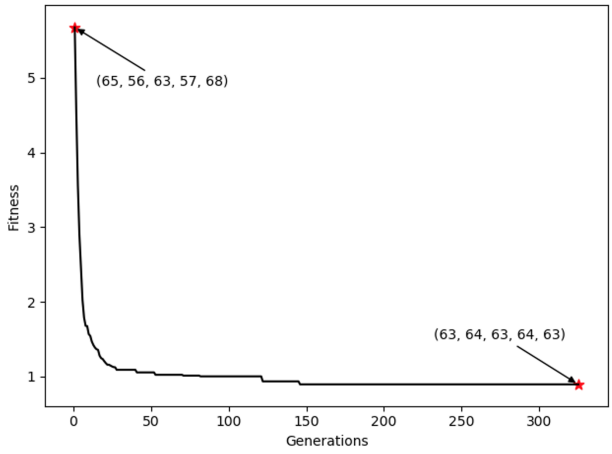

Figure 2: The change of fitness in GA

Figure 3 shows the change of the RMSE of participation in the consensus between our scheme and the uniform distribution scheme as the reputation RMSE changes. The experimental results show that our scheme can dynamically maintain the balance of the number of consensus nodes in each shard when the reputation between shards is unevenly distributed. Specifically, as the RMSE of reputation increases, the uniform distribution scheme will increase the RMSE of the number of nodes participating in the consensus among the shards, that is, it will become more and more uneven. In contrast, the scheme we propose will gradually stabilize at a very low range as the RMSE of reputation increases, which can ensure the fairness and liveness of the shards.

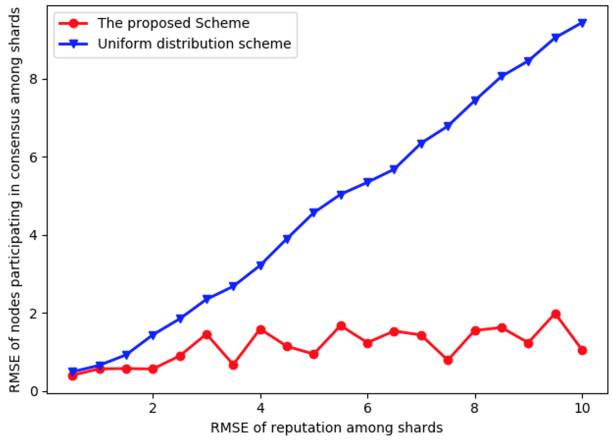

Figure 3: The results of balance among shards experiment

Figure 4 shows the process of our scheme and uniform distribution scheme in the consensus participation rate of the entire sharding system as the reputation value RMSE changes. The experimental results show that our scheme can always ensure that more nodes participate in the consensus compared with the uniform distribution scheme, and has a better incentive effect, which is beneficial to the maintenance of the entire sharding system.

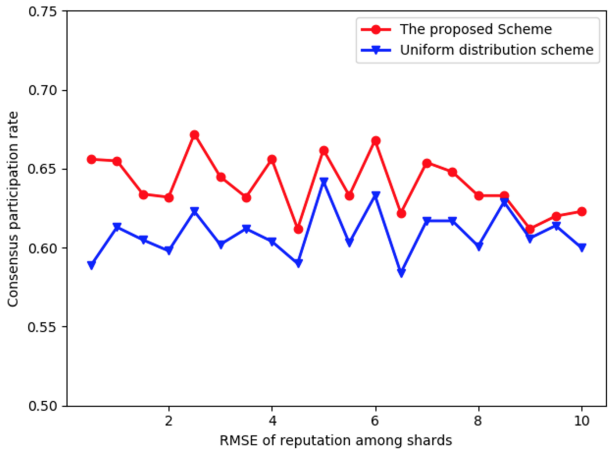

Figure 4: The results of consensus participation rate experiment

From above analysis, the simulation results displayed in Figure 2 to Figure 4 validate the advantages of the proposed scheme by comparing with the uniform distribution scheme.

5 CONCLUSION

In this paper, we found that the shard-based permissionless blockchain system lacks a safe and efficient incentive scheme. Therefore, we have proposed an intra-shard and inter-shard collaborative incentive scheme for this problem. Specifically, for intra-shard incentives, we analyze selfish node strategies based on a static game model, and use reputation to quantify the amount of computing resources that nodes invest in this round of block consensus, guide the realization of a fairer reward distribution scheme, and encourage nodes to invest more computing resources to the consensus process, improve the efficiency of block generation. For the inter-shard, we design a multi-objective genetic algorithm to search for the optimal transaction distribution strategy to maximize the incentives for nodes in the shards to participate in the consensus, and at the same time make the number of nodes participating in the consensus among the shards relatively balanced, improving fairness and activity. In addition, we use the verifiable random number generated by the VRF as the random seed of the genetic algorithm to ensure that each node can verify the execution result of the genetic algorithm, which meets the requirements of consistency and security. Finally, we simulated and analyzed the transactions distribution protocol of VRF-based GA. The experimental results show that our scheme is more able to motivate nodes to participate in consensus than the transaction uniform distribution scheme, while maintaining dynamic balance of participation in each shard and improves the fairness and activity of sharding system.

ACKNOWLEDGMENTS

This work is supported by the National Key Research and Development Program of China under Grant No.2019YFB1406500, Beijing Natural Science Foundation under Grant No.L192032, and the Key Project Plan of Blockchain in Ministry of Education of the People's Republic of China under Grant No. 2020KJ010802.

REFERENCES

[1] Miguel Castro, Barbara Liskov, and Barbara Llskov. 2002. Practical Byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems* 20, 4 (November 2002), 398–461. https://doi.org/10.1145/571637.571640

[2] George Danezis and Sarah Meiklejohn. 2016. Centrally banked cryptocurrencies. In *23rd Annual Network and Distributed System Security Symposium*. NDSS.

[3] Paul Feldman. 1987. A practical scheme for non-interactive verifiable secret sharing. In *Proceedings of the 28th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, Washington, DC, USA, 427–438. https://doi.org/10.1109/SFCS.1987.4

[4] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. 2017. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. *Cryptology ePrint Archive* (2017). https://doi.org/10.1145/3132747.3132757

[5] Chenyu Huang, Zeyu Wang, Huangxun Chen, Qiwei Hu, Qian Zhang, Wei Wang, and Xia Guan. 2020. RepChain: A Reputation-based Secure, Fast and High Incentive Blockchain System via Sharding. *IEEE Internet of Things Journal* (2020).

[6] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ewa Syta, and Bryan Ford. 2018. OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding. In *2018 IEEE Symposium on Security and Privacy (SP)*. 583–598. https://doi.org/10.1109/SP.2018.000-5

[7] Merve Can Kus Khalilov and Albert Levi. 2018. A Survey on Anonymity and Privacy in Bitcoin-Like Digital Cash Systems. *IEEE Communications Surveys & Tutorials* 20, 3 (2018), 2543–2585. https://doi.org/10.1109/COMST.2018.2818623

[8] Wenting Li, Alessandro Sforzin, Sergey Fedorov, and Ghassan O. Karame. 2017. Towards scalable and private industrial blockchains. In *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*. ACM, 9–14.

[9] Xiaoqi Li, Peng Jiang, Ting Chen, Xiapu Luo, and Qiaoyan Wen. 2020. A survey on the security of blockchain systems. *Future Generation Computer Systems* 107 (2020), 841–853. https://doi.org/10.1016/j.future.2017.08.020

[10] Lailong Luo, Deke Guo, Ori Rottenstreich, Richard T.B Ma, Xueshan Luo, and Bangbang Ren. 2019. The Consistent Cuckoo Filter. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*. 712–720. https://doi.org/10.1109/INFOCOM.2019.8737454

[11] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. 2016. A secure sharding protocol for open blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 17–30.

[12] Mohammad Hossein Manshaei, Murtuza Jadliwala, Anindya Maiti, and Mahdi Fooladgar. 2018. A Game-Theoretic Analysis of Shard-Based Permissionless Blockchains. *IEEE Access* 6 (2018), 78100–78112. https://doi.org/10.1109/ACCESS.2018.2884764

[13] S. Micali, M. Rabin, and S. Vadhan. 1999. Verifiable random functions. In *40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039)*. IEEE Computer Society, 120–130. https://doi.org/10.1109/SFFCS.1999.814584

[14] Ewa Syta, Philipp Jovanovic, Eleftherios Kokoris Kogias, Nicolas Gailly, Linus Gasser, Ismail Khoffi, Michael J. Fischer, and Bryan Ford. 2017. Scalable Bias-Resistant Distributed Randomness. In *2017 38th IEEE Symposium on Security and Privacy (SP)*. 444–460. https://doi.org/10.1109/SP.2017.45

[15] The Zilliqa Team. 2017. The zilliqa technical whitepaper. https://docs.zilliqa.com/whitepaper.pdf.

[16] Luming Wan, David Eyers, and Haibo Zhang. 2019. Evaluating the Impact of Network Latency on the Safety of Blockchain Transactions. In *2019 IEEE International Conference on Blockchain (Blockchain)*. Atlanta, GA, USA, 194–201. https://doi.org/10.1109/Blockchain.2019.00033

[17] Jianrong Wang, Yangyifan Zhou, Xuewei Li, Tianyi Xu, and Tie Qiu. 2019. A Node Rating Based Sharding Scheme for Blockchain. In *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*. Tianjin, China, 302–309. https://doi.org/10.1109/ICPADS47876.2019.00050

[18] Haaroon Yousaf, George Kappos, and Sarah Meiklejohn. 2019. Tracing Transactions Across Cryptocurrency Ledgers. In *28th USENIX Security Symposium*. 837–850.

[19] Jusik Yun, Yunyeong Goh, and Jong-Moon Chung. 2019. Trust-Based Shard Distribution Scheme for Fault-Tolerant Shard Blockchain Networks. *IEEE Access* 7 (2019), 135164–135175. https://doi.org/10.1109/ACCESS.2019.2942003

[20] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. 2018. RapidChain: Scaling Blockchain via Full Sharding. Cryptology ePrint Archive, Report 2018/460. https://ia.cr/2018/460.