User Distributions in Shard-based Blockchain Network: Queueing Modeling, Game Analysis, and Protocol Design

Canhui Chen
School of Computer Science and Engineering
Sun Yat-sen University, Guangzhou, China
chench296@mail2.sysu.edu.cn

Qian Ma
School of Intelligent Systems Engineering
Sun Yat-sen University, Shenzhen, China
maqian25@mail.sysu.edu.cn

Xu Chen
School of Computer Science and Engineering
Sun Yat-sen University, Guangzhou, China
Pazhou Lab, Guangzhou, China
chenxu35@mail.sysu.edu.cn

Jianwei Huang
School of Science and Engineering, Shenzhen Institute of Artificial Intelligence and Robotics for Society
The Chinese University of Hong Kong, Shenzhen, China
jianweihuang@cuhk.edu.cn

ABSTRACT

Sharding is one of the most promising and practical methods to achieve horizontal scalability of blockchain networks. However, the increasing number of cross-shard transactions in blockchain sharding protocols may degrade the system throughput. In this paper, we investigate how to distribute users properly in the shard-based blockchains to boost the system transaction performance. We first build an open Jackson queueing network model to capture users' transaction dynamics on shards. Then we cast users' interactions as a shard-based blockchain game, wherein each user aims to minimize its transaction confirmation time and transaction fee. We investigate the equilibrium of the game, and design a polynomial-time algorithm to find efficient equilibria with good system performance. We further design a novel sharding protocol with dynamic user distribution for the permissionless blockchain, and the protocol can maintain good performance in long-term dynamic environment. Extensive numerical results using realistic blockchain transaction data demonstrate that the proposed algorithm and the designed protocol can achieve superior performance for shard-based blockchains.

CCS CONCEPTS

• **Networks** → *Network protocol design*; **Network performance modeling**; **Network performance analysis**.

KEYWORDS

blockchain network, sharding protocol, queueing, game theory

This work was supported in part by the National Natural Science Foundation of China under Grant 62002399, the National Science Foundation of China (No. U20A20159, No. U1711265, No. 61972432); the Program for Guangdong Introducing Innovative and Entrepreneurial Teams (No.2017ZT07X355); the Pearl River Talent Recruitment Program (No.2017GC010465); the Shenzhen Institute of Artificial Intelligence and Robotics for Society. The corresponding author is Xu Chen.

MobiHoc '21, July 26–29, 2021, Shanghai, China

ACM ISBN 978-1-4503-8558-9/21/07…$15.00
https://doi.org/10.1145/3466772.3467051

ACM Reference Format:
Canhui Chen, Qian Ma, Xu Chen, and Jianwei Huang. 2021. User Distributions in Shard-based Blockchain Network: Queueing Modeling, Game Analysis, and Protocol Design . In *The Twenty-second International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc '21), July 26–29, 2021, Shanghai, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3466772.3467051

1 INTRODUCTION

Blockchain plays an important role in many fields such as finance, supply chain, and IoT services, thanks to its great advantages of decentralization, persistence, pseudonymity, and auditability [14]. However, the well-known trilemma of decentralization-security-scalability in blockchain severely affects its wide adoption [27]. Faced with this challenge, sharding [12] is one of the most promising and practical methods to achieve horizontal scalability of the blockchain network. The shard-based blockchain, i.e., breaking up the blockchain network into individual segments (or shards), allows multiple parallel transactions to be processed at the same time, which has a great potential to boost the system throughput.

Shard-based blockchains, however, face the key challenge of handling cross-shard transactions, which need to execute transaction operations asynchronously in different blocks in different shards. As the number of shards increases, the cross-shard transactions will dominate. Since executing cross-shard transactions leads to excessive cross-shard communication overhead, it would exhaust the network bandwidth, diminish the scalability benefit of sharding and even degrade the system transaction performance. Moreover, when a user initiates a transaction, he pays a transaction fee to the miner or validator who processes the transaction. Extra efforts in confirming cross-shard transactions would lead to higher transaction fees.

To reduce cross-shard transactions, one major approach is to distribute users properly into shards. Existing sharding protocols often distribute users into shards randomly [19]. This user distribution policy has the advantage of simplicity and robustness. However, it cannot optimize the system performance, e.g., minimizing users' transaction latency and transaction fees. In this paper, we aim at addressing the following fundamental question.

Key Question 1: *How good is such random user distribution in terms of system transaction performance?*

To improve the performance in the shard-based blockchain network, we further explore the user distribution strategies that take into account the transaction patterns. A simple strategy is to group users with high mutual transaction frequencies into one shard. However, such a strategy may result in some shards with large user sizes and crowed transactions, which lead to the bottleneck of intra-consensus of the shard-based blockchain [27]. Moreover, the blockchain is a decentralized system, and a user would join a shard that maximizes its own benefit. Therefore, it is essential to investigate users' selfish behaviors through a game-theoretic approach. The second fundamental question that we would like to address in this paper is:

Key Question 2: *Is there a transaction-aware user distribution that achieves a better system performance than the random user distribution and no user has the incentive to deviate from?*

To address these two key questions, we first develop an open Jackson queueing network model to capture the impact of user distribution on the system-wide transaction performance. We derive the closed-form average transaction confirmation time and the average transaction fee.We then develop a novel shard-based blockchain game to model users' selfish behaviors of choosing the shards to join. We show that randomly and uniformly distributing all the users into all the shards is an approximate equilibrium. However, such distribution will result in massive cross-shard transactions, which will degrade the system performance. Hence, we further explore other possible equilibrium states under different transaction patterns to see if it is possible to achieve more efficient system-wide performance.

To find efficient equilibria that can achieve outstanding system performance, we propose a polynomial-time algorithm with several heuristic searching rules by taking into account users' transaction patterns. Furthermore, based on the aforementioned theoretical analysis and practical algorithm, we design a novel sharding protocol with dynamic user distributions in the permissionless blockchain. The proposed algorithm ensures the self-stability property of the equilibrium, which guarantees users to achieve a stable and mutually-satisfactory state without any incentive to deviate. The dynamic user redistribution makes the system robust to dynamic environment. Simulations demonstrate that our proposed protocol can maintain a good long-term performance in a dynamic environment.

The key contributions of the paper are listed as follows:

- We propose an open Jackson queueing network model to characterize the transaction dynamics of the shard-based blockchain, and derive the average transaction confirmation time and the average transaction fee.
- We present a novel shard-based blockchain game and explore various equilibria under different transaction patterns. This provides practical insights for the transaction-aware user distribution in the sharding protocol, which is a key challenging issue and future trend as emphasized in the survey paper [27].
- We develop an efficient equilibrium finding algorithm of low complexity, which can achieve an equilibrium with a near-optimal system performance.
- We accordingly design a novel sharding protocol with dynamic user distributions, which can maintain a good long-term performance in a dynamic environment. This protocol is designed from user perspective and can be applied to most existing sharding protocols.

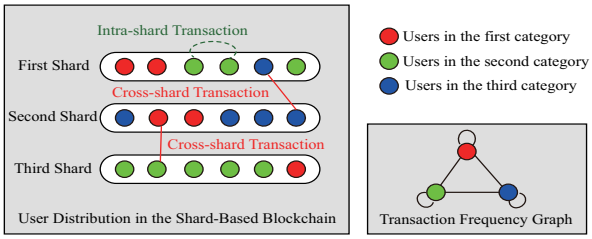


Figure 1: A shard-based blockchain system model

Due to space constraints, some of the proofs are presented in [4].

2 SYSTEM MODEL

We consider a shard-based blockchain with S shards and N users. We divide users into K categories based on their transaction patterns [27]. Figure 1 shows a shard-based blockchain with 3 shards and 3 categories of users.

2.1 Participants in Shard-based Blockchain

There are two types of participants in the shard-based blockchain:

- **Validator**: A blockchain validator plays a key role in the system consensus, as the validator is responsible for storing data, processing transactions, and adding new blocks to the blockchain to maintain the system secure.
- **User**: A blockchain user is someone who does not involve in system consensus, but he uses the blockchain applications, i.e., initiates or receives transactions.

In most sharding systems [1, 3, 12, 28], validators are randomly distributed to different shards periodically for system security. However, users can be distributed to different shards via some known and even deterministic rules such as using the least-significant bits of users' addresses [3, 24]. Most of the previous work study the shard-based blockchain through the validator perspective [1, 9, 12, 28]. Along a different line, in this paper we mainly focus on the user perspective for system performance analysis and optimization.

2.2 Dynamic Shard Selection

Efficient user distribution on shards (i.e., shard selection) has been recently emerging as an important research issue for next-generation shard-based blockchain design [23]. In some existing blockchains, even though users are allocated to different shards using some fixed rules (e.g., based on users' addresses), a user still has partial control regarding which shard to join, for instance:

(1) A user can generate a number of addresses and pick up the address that is allocated to the shard that the user wants to join in. This process is similar as the mining process in PoW-based blockchain [16]. As generating a number of addresses is computationally cheap, the user can implement such a shard selection strategy even on his mobile device.

(2) If a user who is in shard i and and wants to change to shard j, he should first generate a number of addresses and pick up the address in shard j, then initiate a cross-shard transaction with a certain transaction fee to transfer his assets from the address in shard i to the address in shard j.

2.3 User Distributions and Transactions

Let $R \in \mathbb{R}^{S\times K}$ denotes the distribution of users, where $R(i,k)$ denotes the fraction of users of category k joining shard i. Besides, let $\mathbf{r} = [r_1 \ \cdots \ r_K]$ be the population fraction vector, where r_k is the population fraction of user of category k. Hence, we have $\mathbf{r} = \mathbf{1}_S^T R$.

Users in different categories have different transaction frequencies. Let the real square matrix $T \in \mathbb{R}^{K\times K}$ be the transaction frequency matrix. Specifically, $T(k,l)$ denotes the frequency of transactions that are initiated by a user of category k and sent to a user of category l. We assume that the transaction frequency between users in the same category is higher than or equal to that between users in different categories, i.e., $T(k,k) \geq T(k,l),\ \forall k,l \in \{1,\ldots,K\}$.

As illustrated in Section 7, such transaction matrix can be obtained based on users' transaction records using some data mining tools such as community detection. We define $\mathbf{q} = T\mathbf{1}_K = [q_1 \ \cdots \ q_K]^{\mathrm{T}}$, where $q_k = \sum_{l=1}^{K} T(k,l)$ denotes the total transaction frequency initiated by a user of category k. Besides, we define $\mathbf{p} = \mathbf{1}_K^{\mathrm{T}} T = [p_1 \ \cdots \ p_K]$, where $p_l = \sum_{k=1}^{K} T(k,l)$ denotes the total transaction frequency received by a user of category l.

We divide transactions in the shard-based blockchain into *intra-shard transactions* and *cross-shard transactions*. An intra-shard transaction is a transaction that happens in a single shard. The intra-shard transaction are executed in one block in a single shard synchronously with a short transaction confirmation time and a low transaction fee f_{intra}. A cross-shard transaction is a transaction sent from one shard to another. The cross-shard transactions need to be executed in different blocks in different shards asynchronously, which lead to a longer transaction confirmation time and a higher transaction fee f_{cross}. To achieve tractable analysis in the following study, we assume that each shard of the blockchain adopts the same transaction protocol, such that the intra-shard transaction fee f_{intra} on different shards is homogeneous and the cross-shard transaction fee f_{cross} across different pairs of shards is the same. We also assume that $f_{\text{intra}} < f_{\text{cross}}$, i.e., the cross-shard transaction fee is higher

3 INTRA- AND INTER-SHARD TRANSACTION MODELING

In this section, we formulate users' transaction confirmation time based on queueing theory and discuss users' transaction fee.

3.1 Queueing Model for Transaction Confirmation Time

We first introduce the following two terminologies[1].

- **Transaction inclusion time**: the time from the transaction being issued to its first inclusion into a block in a shard.
- **Transaction confirmation time**: the time from the transaction being issued to its last inclusion into a block in a shard.

For intra-shard transactions, the transaction confirmation time is equal to the transaction inclusion time, since its operations are synchronously executed in one block in one shard.

For a cross-shard transaction that involves shard i and shard j, the transaction confirmation time is approximately equal to the sum of the transaction inclusion time in shard i and shard j according to the mechanisms for handling cross-shard transactions[2]. The transaction confirmation time of a cross-shard transaction where user A in shard i sends 10 coins to user B in shard j can be calculated as follows:

- The transaction is included in a block in shard i, which will subtract 10 coins from user A's account.
- After validation in shard i, the transaction will be forwarded to shard j.
- The transaction is included in a block in shard j, which will add 10 coins to user B's account.

Notice that the transaction forward time is negligible, since the size of a transaction is small enough to be quickly broadcast to the network. Hence, the transaction confirmation time of this cross-shard transaction can be well approximated by $Q^{(i)} + Q^{(j)}$, where $Q^{(i)}$ and $Q^{(j)}$ are the transaction inclusion time in shard i and shard j.

To calculate the transaction confirmation time, we model the shard-based blockchain as an open Jackson queueing network [7] shown in Figure 2. Specifically, we model each shard as an M/M/1 queue [21] and all shards with cross-shard transactions form an open Jackson queueing network. According to the Jackson's theorem [7], we can model the intra- and cross-shard transactions among the shards as multiple independent M/M/1 queues with different aggregate arrival rates.

3.1.1 Arrival Process of a Shard. We assume that the transactions arrive at shard i according to a Poisson process with an arrival rate λ_i [8, 21]. As illustrated in Figure 2, λ_i consists of two components. The first component λ_i^{I} is generated by the users initiating transactions in the shard, including the intra-shard transactions and cross-shard transactions initiated by users in shard i, that is,

$$\lambda_i^{\mathrm{I}} = N \sum_{j=1}^{K} R(i,j) \sum_{k=1}^{K} T(j,k). \tag{1}$$

The second component λ_i^{II} is related to the received cross-shard transactions initiated by the users in other shards, that is,

$$\lambda_i^{\mathrm{II}} = N \sum_{m=1,m\neq i}^{S} \sum_{k=1}^{K} R(m,k) \sum_{j=1}^{K} T(k,j) \frac{R(i,j)}{r_j}. \tag{2}$$

Combining (1) and (2), for an open Jackson queueing network with the utilization less than one at every queue[3], the total transaction arrival rate λ_i of shard i is

$$\lambda_i = \lambda_i^{\mathrm{I}} + \lambda_i^{\mathrm{II}}. \tag{3}$$

3.1.2 Service Process of a Shard. Each arriving transaction to a shard can be regarded as entering the queue buffer of the shard

[1] When defining the transaction confirmation time, we ignore the additional confirmations. In the existing blockchain system, a transaction will be visible to its payee once it is packed into a block in a shard (first confirm) [24]. To be secure against double spending [16], it will be secured after $n-1$ successive blocks appended (n-th confirm).

[2] Different blockchain systems have different mechanisms for handling cross-shard transactions. Monoxide handles cross-shard transactions by relaying transactions across asynchronous zones. Ethereum 2.0 handles cross-shard transactions using the beacon chain. We consider an abstract but general model here.

[3] The utilization of a M/M/1 queue with the arrival rate as λ and the service rate as μ is $\rho = \lambda/\mu$. And the utilization less than one implies the stability of the queue.

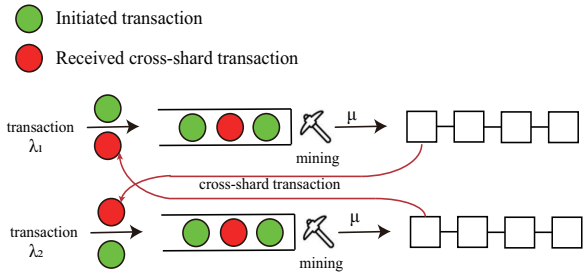


Figure 2: Illustration of the queueing model for shard-based blockchain. Each shard is modeled as an M/M/1 queue, and two shards form an open Jackson queueing network.

and waiting for processing according to the First Come First Service (FCFS) policy[4]. The transaction service process ends when the transaction is verified and included in a block by a miner who successfully solves the mining puzzle. And the block will be appended to blockchain of that shard. For simplicity, we assume that each block contains exactly one transaction, which captures the block size constraint in reality and has been adopted in previous studies [5, 10]. In many blockchain systems such as Bitcoin [16], Ethereum [26], and Monoxide [24], blocks are generated according to the proof-of-work (PoW) consensus mechanism or its variants, and the process of generating blocks has been shown to follow the Poisson process [2]. Hence, we assume that the mining capability of each shard is homogeneous and the block generation of a shard follows the Poisson process with a rate μ. Note that we only consider the stable case where the utilization is less than one at every queue. Based on Pollaczek-Khinchin formula [22], the transaction inclusion time of a shard with transaction arrival rate λ is

$$g(\lambda) = \begin{cases} \frac{1}{\mu-\lambda}, & \text{if } \lambda < \mu, \\ \infty, & \text{otherwise.} \end{cases}$$

Specially, the transaction inclusion time $Q^{(i)}$ in shard i is

$$Q^{(i)} = g(\lambda_i). \tag{4}$$

As a result, we can derive the average transaction confirmation time for a user in the following lemma.

LEMMA 1. *The average transaction confirmation time for a user of category k in shard i under user distribution state R is*

$$W(i,k,R) = \frac{1}{q_k}\left(\sum_{m\neq i}^{S}\sum_{n=1}^{K}\left(Q^{(m)}+Q^{(i)}\right)\frac{R(m,n)}{r_n}T(k,n) + \sum_{n=1}^{K}\frac{R(i,n)}{r_n}T(k,n)Q^{(i)}\right). \tag{5}$$

The proof is given in Appendix A in the technical report [4].

3.2 Transaction Fee Model

The average transaction fee of a user depends on both the intra-shard transactions and the cross-shard transactions. We calculate the average transaction fee for a user in the following lemma.

[4]In the realistic blockchain, the transaction waiting time in a queue is greatly depends on its transaction fee [8]. Since we only investigate the system performance here, the First Come First Service (FCFS) policy is a good approximation [21].

LEMMA 2. *The average transaction fee for a user of category k in shard i under user distribution state R is*

$$F(i,k,R) = \frac{1}{q_k}\left(\sum_{j=1}^{K}T(k,j)\frac{R(i,j)}{r_j}f_{intra} + T(k,j)\left(1-\frac{R(i,j)}{r_j}\right)f_{cross}\right).$$

The proof is given in Appendix B in the technical report [4].

4 SHARD-BASED BLOCKCHAIN GAME AND PERFORMANCE ANALYSIS

In this section, we analyze the scenario where each user chooses a shard to join to maximize its own benefit. We model users' interactions as a shard-based blockchain game, and study its equilibrium. We further investigate the system performance at the equilibria.

4.1 Shard-based Blockchain Game

We formulate the shard-based blockchain game as a non-cooperative game $\mathbb{G} = (\mathcal{N}, \mathcal{K}, \mathbf{r}, \mathcal{S}, \mathcal{U})$, where $\mathcal{N}$ is the set of users, $\mathcal{K} = \{1, 2, ..., K\}$ is the set of user categories, and $\mathbf{r} = \{r_1, r_2, ..., r_K\}$ is the population proportion of the proportions of users of each category. Furthermore, $\mathcal{S} = \{1, 2, \ldots, S\}$ denotes the common set of strategies for all users, where a strategy $i \in \mathcal{S}$ represents a user joining shard i. Finally, $\mathcal{U}$ is the set of the user utilities. Specifically, the utility of a user of category k, $k \in \mathcal{K}$, achieved by choosing shard i, $i \in \mathcal{S}$, depends on the transaction confirmation time and the transaction fee under the user distribution state R as follows:

$$U(i,k,R) = -W(i,k,R) - \beta F(i,k,R), \tag{6}$$

where β represents the relative sensitivity of the transaction fee compared to the transaction confirmation time.

In the following analysis, we focus on the case where there is a large population of users and a fixed number of strategies, i.e., N is large while S and K are much smaller than N. This is consistent with the practical shard-based blockchain [3, 24], where the number of users is quite large, and the number of shards and the number of user categories are relatively small compared to the number of users. Due to the large population size, in the following we will also adopt the population state (i.e., user distribution state) R to describe the equilibrium state of the shard-based blockchain game.

4.2 Equilibrium Analysis

In this subsection, we investigate the equilibria of the shard-based blockchain game. Let $R_{(k,i,j)}$ denote the updated user distribution where a user of category k under the distribution R moves to shard j while other users remain unchanged. Thus we have

$$R_{(k,i,j)}(\hat{i},\hat{k}) = \begin{cases} R(i,k) - \frac{1}{N}, & \hat{i} = i, \hat{k} = k, \\ R(j,k) + \frac{1}{N}, & \hat{i} = j, \hat{k} = k, \\ R(\hat{i},\hat{k}), & \text{otherwise.} \end{cases}$$

First, we introduce the definition of Nash equilibrium.

DEFINITION 4.1. *(Nash Equilibrium [25]). A population state R^* is a Nash equilibrium of the shard-based blockchain game if for any $R^*(i,k) \neq 0$ we have*

$$U(i,k,R^*) \geq \max_{j\in\mathcal{S}} U(j,k,R^*_{(k,i,j)}), \forall k \in \mathcal{K}, i \in \mathcal{S}. \tag{7}$$

The Nash equilibrium is a steady state under which users can achieve a mutually satisfactory solution and no user has the incentive to deviate. Note that in this study we only consider the pure strategy Nash equilibrium. The mixed strategy Nash equilibrium is impractical since a user needs to switch its shard selection frequently, which leads to a significant cost due to cross-shard transactions to transfer its profiles and assets across different shards.

For the shard-based blockchain game, we have the following result.

THEOREM 1. *A shard-based blockchain may not possess a Nash equilibrium.*

The proof is given in Appendix C in the technical report [4].

Motived by this, we will also consider a relaxed concept of approximate Nash equilibrium for the shard-based blockchain game, which is defined as follows.

DEFINITION 4.2. *(Approximate Nash Equilibrium [25]). A population state R^* is an ϵ-approximate Nash equilibrium of the shard-based blockchain game if for any $R^*(i,k) \neq 0$ we have*

$$U(i,k,R^*) \geq \max_{j \in \mathcal{S}} U(j,k,R^*_{(k,i,j)}) - \epsilon, \forall k \in \mathcal{K}, i \in \mathcal{S}. \quad (8)$$

Here $\epsilon \geq 0$ is the gap from a (precise) Nash equilibrium, which can also be understood as the maximum switching cost that a user can tolerate for transferring to another shard, e.g., transaction fee.

The following theorem formulates the equilibria of the shard-based blockchain game.

THEOREM 2. *Let $\mathcal{R}$ be the set of the feasible user distributions,*

$$\mathcal{R} = \{R | \mathbf{1}_S^T R = \mathbf{r}, \sum_{i=1}^{S}\sum_{k=1}^{K} R(i,k) = 1, \forall i,k, 0 \leq R(i,k) \leq 1, \lambda_i < \mu\}.$$

Define $\mathcal{E}$ as a subset of $\mathcal{R}$ as follows:

$$\mathcal{E} = \{R \in \mathcal{R} | \forall i,k, R(i,k) \neq 0 \Rightarrow \forall i', U(i,k,R) \geq U(i',k,R)\}.$$

Then $\forall R \in \mathcal{E}$ is an $O(1/N)$-approximate Nash equilibrium.

The proof is given in Appendix D in the technical report [4].

The user distribution $R \in \mathcal{R}$ is a feasible user distribution with the stability of the blockchain system[5]. Moreover, the condition that $\forall i,k,R(i,k) \neq 0 \Rightarrow \forall i', U(i,k,R) \geq U(i',k,R)$ implies that under the user distribution $R \in \mathcal{E}$, any user of any category exactly stays at the shard that maximizes its utility. When the population size N is large enough, the effect of any user changing to another shard is negligible. Thus it is an $O(1/N)$-approximate Nash equilibrium.

The following theorem shows that the random and uniform user distribution can be an approximate Nash equilibrium.

THEOREM 3. *Randomly and uniformly distributing all categories of users into S shards is an $O(1/N)$-approximate Nash equilibrium of the shard-based blockchain game, when the blockchain system is stable.*

The proof is given in Appendix E in the technical report [4].

Theorem 3 reveals the game theoretical basis for random and uniform distribution in the shard-based blockchain. We next investigate the user aggregation phenomenon wherein users only choose a subset of shards with the remaining shards being empty.

[5]The stability of the blockchain transaction queue implies that the transactions in the queue cannot be accumulated to infinity. The stability of the shard-based blockchain system implies that the transaction queue of each shard is stable.

THEOREM 4. *Randomly and uniformly distributing all categories of users into s shards ($s < S$) is an $O(1/N)$-approximate equilibrium of the shard-based blockchain game when*

$$\lambda \leq \mu - \frac{\mu(s-1)}{s + \beta(f_{cross} - f_{intra})\mu}, \quad (9)$$

where λ is the transaction arrival rate of the selected shards, which can be calculated according to (3).

The proof is given in Appendix F in the technical report [4].

Theorem 4 implies that when the current shards are crowded, i.e., $\lambda > \mu - \frac{\mu(s-1)}{s+\beta(f_{\text{cross}} - f_{\text{intra}})\mu}$, building new shards is necessary, since users are motivated to join the new shards, which helps to maintain the stability of the blockchain system. However, when the current shards are not so crowded, i.e., satisfying condition (9), building a new shard is pointless, since no user will spontaneously the new shard.

We next analyze the equilibrium in a special case where the transaction frequency between the users in the same category is strictly dominant, i.e., $T(k,k) > \sum_{j=1, j\neq k}^{K} T(k,j), \forall k \in \mathcal{K}$.

THEOREM 5. *Distributing the users in the same category in one shard is an $O(1/N)$-approximate Nash equilibrium of the shard-based blockchain game when the blockchain system is stable and*

$$\frac{T(k,k)}{\sum_{j=1, j\neq i}^{K} T(k,j)} \geq 2\frac{\max_{i \in \mathcal{S}} Q^{(i)}}{\min_{i \in \mathcal{S}} Q^{(i)}} - 1, k \in \mathcal{K}, \quad (10)$$

where $Q^{(i)}$ is calculated according to (4).

The proof is given in Appendix G in the technical report [4].

Theorem 5 shows that when users in the same category have much higher transaction frequency than that of users in different categories, i.e., satisfying condition (10), distributing users in the same category in one shard is an equilibrium.

4.3 Performance Analysis

We then analyze the performance achieved at the equilibrium of the shard-based blockchain game. We quantify the performance by the important metric: the average transaction confirmation time (ATCT) of the shard-based blockchain, which is defined as follows

$$\text{ATCT} = \frac{\sum_{i=1}^{S}\sum_{k=1}^{K} R(i,k)W(i,k,R)q_k}{\sum_{k=1}^{K} r_k q_k}. \quad (11)$$

Intuitively, a shorter average transaction confirmation time implies a better performance of the blockchain system with greater throughput and a better user experience. According to (11), we obtain the following result.

LEMMA 3. *When distributing users in different shards randomly and uniformly, the ATCT of the shard-based blockchain is*

$$ATCT = \frac{2S-1}{S}Q,$$

where $Q = 1/\left(\mu - \frac{N}{S}\sum_{k=1}^{K} r_k q_k + \frac{N(S-1)}{S^2}\sum_{k=1}^{K} r_k p_k\right)$.

Lemma 3 shows that when adopting the strategy of randomly and uniformly distributing users into all shards, the ATCT decreases with the number of shards. Note that, when $S \to \infty$, we have ATCT $\to 2/\mu$, which shows that random and uniform distribution is a strategy

that can achieve the scalability of the blockchain network when the number of shards is large enough. However, we will show that random and uniform distribution will result in massive cross-shard transactions, which will degrade the system performance.

To measure the impact of cross-shard transactions on the performance of shard-based blockchain, we define the cross-shard transactions ratio (CSTR) as the ratio between the number of cross-shard transactions and the number of total transactions in below:

$$\text{CSTR} = \frac{\sum_{i=1}^{S}\sum_{j=1}^{K} R(i,j)\sum_{k=1}^{K} T(j,k)\left(1-\frac{R(i,k)}{r_k}\right)}{\sum_{j=1}^{K} r_j \sum_{k=1}^{K} T(j,k)}. \quad (12)$$

According to (12), we have the following result.

Lemma 4. *When distributing users in S shards randomly and uniformly, the CSTR of the shard-based blockchain is*

$$CSTR = \frac{S-1}{S}.$$

Lemma 4 shows that when adopting the strategy of random and uniform distribution over all the shards, the CSTR increases with the number of shards in the shard-based blockchain. Indeed, when $S \to \infty$, we have CSTR $\to 1$, which implies that the number of the cross-shard transactions is a critical issue in the sharding protocol under random user distribution.

5 POLYNOMIAL-TIME ALGORITHM FOR EFFICIENT EQUILIBRIUM SEARCHING

In the previous section, Theorem 3 shows that randomly and uniformly distributing users is an $O(1/N)$-approximate Nash equilibrium. Nevertheless, Theorems 4 and 5 imply that we can exploit the structural characteristic of the shard-based blockchain systems to explore equilibria which can be more efficient in terms of the system-wide performance. To find the (approximate) equilibrium states with the best performance (e.g., the smallest ATCT), we need to explore the space of all equilibrium states. However, exploring all the equilibria of the game is computationally complex and infeasible. As a result, based on the equilibrium analysis in Section 4.2, we propose several heuristic rules to narrow down the search space.

Suppose that users of category k are distributed in the set of shards $\mathcal{X}_k$ and $|\mathcal{X}_k|$ denotes the number of shards. Then, we introduce the following heuristic rules:

(1) Users of the same category should be randomly and uniformly distributed in their selected shards, that is,

$$\forall k \in \{1,\dots,K\}, \forall i \in \mathcal{X}_k, R(i,k) = \frac{r_k}{|\mathcal{X}_k|}.$$

(2) Users of different categories should be distributed in different shards, otherwise, they should be distributed identically, that is,

$$\forall j,k \in \{1,\dots,K\},\ \mathcal{X}_j \cap \mathcal{X}_k = \emptyset \text{ or } \mathcal{X}_j = \mathcal{X}_k.$$

Intuitively, the rules (1) and (2) above imply that we should partition the set of shards into multiple groups and for each group of shards we choose the proper subset of user categories to distribute their users. Moreover, we still randomly and uniformly distribute the users within each group of shards to achieve the equilibrium for the ease of implementation.

Denote the search space after applying the heuristic rules above as Ω, then we have the following lemma.

Algorithm 1 Search for efficient equilibrium

1: Sort the user category list such that we have $r_1 q_1 \geq r_2 q_2 \geq \cdots \geq r_K q_K$.
2: **for** $k = 1$ to K **do**
3: $T = \emptyset$.
4: **for** $n = 1$ to $S - |\cup_{i=1}^{k-1} \mathcal{X}_i|$ **do**
5: $C = \{|\cup_{i=1}^{k-1} \mathcal{X}_i| + 1, \dots, |\cup_{i=1}^{k-1} \mathcal{X}_i| + n\}$.
6: Distribute users in category k into shards C based on the heuristic rule (1) and construct the distribution R_C.
7: **if** $T \neq \emptyset$ and $U(*, i, R_T) > U(*, i, R_C)$ **then**
8: **break**
9: **end if**
10: **if** $U(*, i, R_C) > -\infty$, i.e., the system is stable **then**
11: $T = C$.
12: $u = \sum_{i=k+1}^{K} U(*, i, R')r_i / \sum_{i=k+1}^{K} r_i$, where R' is the user distribution with users in category $k+1, \dots, K$ distributing in shards $\{|\cup_{i=1}^{k-1} \mathcal{X}_i| + n + 1, \dots, S\}$.
13: **if** $U(*, k, R_C) > u$ **then**
14: **break**
15: **end if**
16: **end if**
17: **end for**
18: $\mathbb{X} = \{\mathcal{X}_1, \dots, \mathcal{X}_{k-1}\} \cup \{T\}$.
19: $\mathcal{X}_k = \arg\max_{\mathcal{X} \in \mathbb{X}} U(*, k, R'')$, where R'' is the distribution after distributing users in category k into shards $\mathcal{X}$ based on heuristic (1).
20: Distribute users in category k into shards $\mathcal{X}_k$ based on heuristic (1).
21: **while** there exists a user who has the incentive to move from shard i to shard j **do**
22: $\mathcal{P} = \{m | i \in \mathcal{X}_m \text{ or } j \in \mathcal{X}_m, m = 1, \dots, K\}$.
23: $\mathcal{X} = \cup_{m \in \mathcal{P}} \mathcal{X}_m$.
24: Redistribute users of category m into shards $\mathcal{X}$ based on rule (1) and update $\mathcal{X}_m = \mathcal{X}, \forall m \in \mathcal{P}$.
25: **end while**
26: **end for**
27: **return** R, where R is the final user distribution.

Lemma 5. *The size of the set of all feasible user distributions, i.e., $|\mathcal{R}|$, is $O(S^N)$. By applying the heuristic rules above, the size of the search space $|\Omega|$ is $O(S^K)$.*

The proof is given in Appendix H in the technical report [4].

Since $N \gg K$, the heuristic rules can significantly narrow down the search space. To find an efficient equilibrium, we can then apply some heuristic optimization algorithms (e.g., evolutionary algorithm [20]) over the pruned search space. Nevertheless, when S and K are large, exploring the search space can be still computationally intensive. Hence we propose a polynomial-time algorithm (Algorithm 1) to find an efficient equilibrium by progressively improving users' utilities through proper user distributions. Notice that based on the heuristic rules (1) and (2), the expected utilities of users of the same category should be the same. For convenience, we denote $U(*, k, R)$ as the utility of a user of category k under the distribution R, i.e., $\forall k \in \{1, \dots, K\}, \forall i$ s.t. $R(i,k) \neq 0$, we have $U(*, k, R) = U(i, k, R)$.

Based on the greedy principle, we first sort the user category list by the users' transaction frequency (Line 1). Then the algorithm proceeds in rounds. In each round, we distribute a new kind of users into shards. According to the heuristic rule (2), a new kind of users, i.e., users in category k, can choose to join the empty shards with no users (Line 5) or the shards that the previous users have chosen,

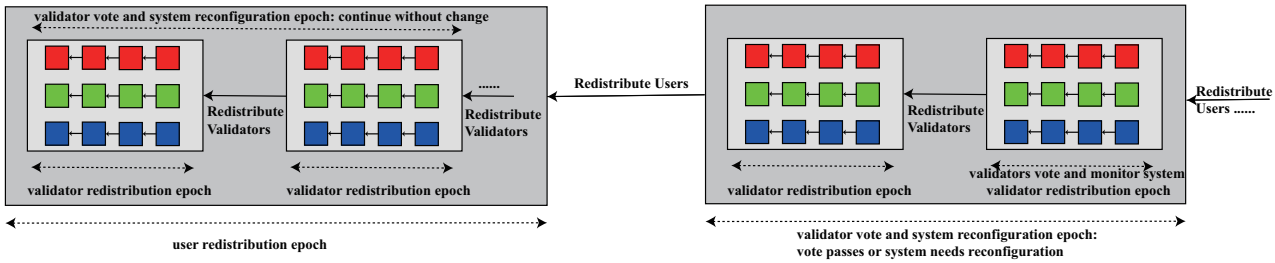


Figure 3: Sharding protocol with dynamic user distributions

i.e., $\mathcal{X}_i, i < k$, (Line 18). When users in category k choose the empty shards to join, the algorithm will encourage the users to choose a proper number of empty shards such that the blockchain system is stable and there is enough space for the remaining users (Lines 4-17). Then users in category k will choose to join the shards that can maximize their utilities under the constraints of the heuristic rules (1) and (2) (Lines 18-20). When the user distribution changes, there may exist a user who has the incentive to move from shard i to shard j. In this case, the algorithm will redistribute all the users who have joined shard i and shard j and until it achieves an equilibrium state, i.e., no user has the incentive to move to another shard (Lines 21-25). Then one round ends. After proceeding K rounds, the algorithm will output an efficient equilibrium R (Line 27).

Based on the analysis above, we have the following theorem.

THEOREM 6. *Algorithm 1 computes an efficient $O(1/N)$-approximate equilibrium in polynomial time with the computational complexity of $O\left(SK^3(S + \min\{S, K\})\right)$.*

The proof is given in Appendix I in the technical report [4].

6 PROTOCOL DESIGN WITH DYNAMIC USER DISTRIBUTIONS

In this section, we design a sharding protocol with dynamic user distributions in the permissionless blockchain based on the aforementioned theoretical analysis and practical algorithm.

6.1 Dynamic Sharding Protocol

As shown in Figure 3, there are three different epochs in this system:

- **Validator redistribution epoch**. In each validator redistribution epoch, validators will be randomly redistributed to different shards, which is a common practice in current shard-based blockchains [1, 9, 12, 28].
- **User redistribution epoch**. In each user redistribution epoch, users will be redistributed to different shards based on our proposed algorithm.
- **Validator vote and system reconfiguration epoch**. In each validator vote and system reconfiguration epoch, validators in all shards will vote to decide whether to redistribute users, and collect the block statistics to decide whether to change the number of shards in the system.

Similar to most current shard-based blockchains, the validator redistribution epoch is short, e.g., 10 min in Elastico [12], due to the system security concern. But the user redistribution epoch might be as long as several months to adapt to the long-term evolution of the users' transaction patterns. Besides, the validator vote and system reconfiguration epoch is longer than the validator redistribution epoch but is shorter than the user redistribution epoch, which can make the system adaptive to the short-term dynamic changes.

6.2 User Distribution Table

Different from the current shard-based blockchain, our system will distribute users into different shards in a dynamic way instead of the traditional fixed and deterministic way. Specifically, all the validators maintain a global user distribution table (UDT), which maps each user address to his shard. Note that the UDT does not need to record the mapping information of all the users, it only records the mapping information of the users that initiated or received at least b ($b \geq 1$) transactions in the previous user distribution epoch. Therefore, the UDT is small enough to be stored in the memory cache. Thus, the validators can easily find out which shard the user is allocated to, when processing cross-shard and intra-shard transactions. The address that does not lay in the UDT, will be allocated to the shard in the fixed and deterministic way, for example, using the least-significant bits of the address to determine which shard to join in.

6.3 User Redistribution Process

The user redistribution process goes as follows:

(1) The validators in shard i ($i = 1, 2, \ldots, S$) generate a statistics block and broadcast it to the other validators. There are statistics records of transactions initiated from shard i, begining from the last user redistribution, in the statistics block. A statistics record is a triple of $(address_A, address_B, \#transaction)$, i.e., the number of transactions sent from the $address_A$ in shard i to the $address_B$ in this period is $\#transaction$.
(2) After receiving all the statistics block, validators in each shard construct a transaction network with the user addresses as nodes and the transactions as edges.
(3) All the validators run the community detection algorithm [6] to classify users into different categories.
(4) All the validators run Algorithm 1 to redistribute users to different shards and construct the user distribution table. Validators are encouraged to broadcast the user distribution table to make it consistent in the system.

6.4 Validator Vote Process

In each validator vote and system reconfiguration epoch, the validator vote process goes as follows:

(1) The validators in shard i ($i = 1, 2, \ldots, S$) vote in the shard. If more than 50% validators vote for redistribution, then shard i will be marked as "vote for redistribution".
(2) If more than $2/3$ of shards are marked as "vote for redistribution", the user redistribution process will be conducted.

Consider the situation when the transactions are concentrated in a few shards, leaving most of the shards with few transactions. Without user redistribution, this skew transaction distribution will degrade the system throughput. The validators in the shards with many transactions can obtain high transaction fee rewards, while the validators in the remaining shards can only obtain low transaction fee rewards. Therefore, validators in those shards with few transactions are motivated to vote for redistribution, which helps to rebalance the transactions and improve the system performance.

6.5 System Reconfiguration Process

In each validator vote and system reconfiguration epoch, the validators will decide whether to change the number of shards as follows:

(1) The validators in shard i ($i = 1, 2, \ldots, S$) collect the historical block and transaction statistics from the last time of user redistribution, and broadcast it to other validators.
(2) If more than 80% blocks in shard i exceed 80% of the maximum block size, and the average transaction fee is at least 50% higher than that in the previous user redistribution epoch, shard i will be marked as "busy shard".
(3) If more than 80% blocks in shard i are below 20% of the maximum block size, and the average transaction fee is at least 50% lower than that in the previous user redistribution epoch, shard i will be marked as "idle shard".
(4) If more than 80% shards are "busy shard", the system will double the number of shards[6] and redistribute users.
(5) If more than 80% shards are "idle shard", the system will halve the number of shards[7] and redistribute users.

We estimate the system using the two important metrics, i.e., block size and transaction fee. Generally, when the system is busy like (4), i.e., there are too many unconfirmed transactions, the validators will try to pack as more as possible transactions in a block to obtain a higher transaction fee reward, and users need to raise their transaction fee to increase his transaction priority. On the other hand, when the system is idle like (5), i.e., there are few unconfirmed transactions, validators can only pack a few transactions in a block, and users are motivated to decrease their transaction fee. When the system is busy, increasing the number of shards can help boost the system performance. When the system is idle, decreasing the number of shards can help to increase the utility of the system, maintain the system security and guarantee the validators' transaction reward, which can motivate them to maintain the system consensus.

7 PERFORMANCE EVALUATION

In this section, we conduct extensive experiments based on the historical transactions in Ethereum to evaluate the impact of user distribution on the performance of the shard-based blockchain.Besides, we conduct several simulations to demonstrate the high efficiency of the proposed algorithm and the designed sharding protocol.

[6]The number of shards has its maximum for system security concern.
[7]There is at least one shard in the blockchain system.

7.1 Experiments Design and Configuration

We evaluate the impact of the user distribution by playing back the complete historical normal transactions [29] from the begining up to the block height 1,000,000 in Ethereum [26], which includes 1,674,262 transactions and 55,350 unique user addresses. Furthermore, we model the transaction network as an undirected weighted graph, which contains 55,350 nodes and 92,653 edges with their weights denoting the numbers of transactions between two users.

We analyze the system performance under different user distributions. Particularly, the transaction processing rate of each shard is $\mu = 1$. The intra-shard transaction fee is $f_{\text{intra}} = 1$ and the cross-shard transaction fee is $f_{\text{cross}} = 2$. The sensitivity of users to transaction fee relative to the transaction confirmation time is $\beta = 0.5$. Our experiments involve a set of users that continuously issue transactions from the dataset to the system at a predefined rate.

7.2 Community Detection of User Categories

Due to the anonymity of the blockchain, it is difficult to get a user's category directly from the user's address and transaction records. Here, we adopt the Louvain algorithm [6] to maximize the modularity of the transaction network and determine communities within the blockchain. Each community corresponds to a user's category.

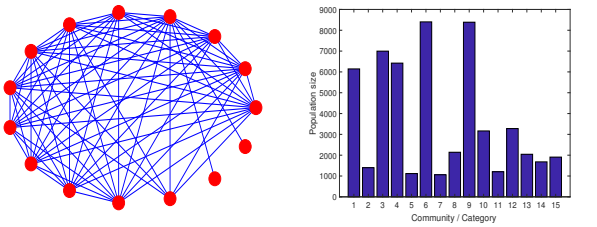
After using the algorithm and merging a small number of communities, we find that there are 15 communities corresponding to 15 categories, and the numbers of users in each category are 6144, 1399, 6997, 6422, 1117, 8403, 1060, 2141, 8387, 3165, 1205, 3278, 2044, 1680, and 1908. The modularity of this partition is 0.600. Figure 4 (a) shows the graph where nodes are the communities. Figure 4 (b) shows the population size of each community.

7.3 Performance of Different User Distributions

In this subsection, we derive insights on how different user distributions affect the performance of shard-based blockchain. We compare the following three user distributions:

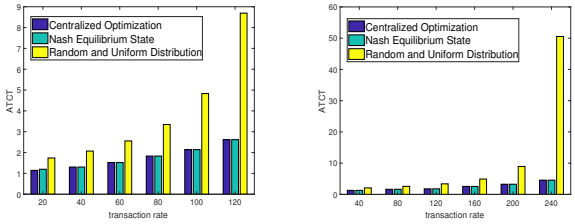
- Random and uniform distribution: Randomly and uniformly distribute all the users in all the shards.
- Efficient equilibrium state: Find the equilibrium user distribution with good performance using Algorithm 1.
- Centralized optimization: Using the "fmincoin" in MATLAB [15] to do the local search to find the local optimum (which may not be an equilibrium state). To find the near-optimal solution, we repeat each experiment 100 times.

Figure 5 shows the ATCT of the shard-based blockchain under different user distributions, from which we find that the system performance under the efficient equilibrium state is near-optimal, while random and uniform distribution has longer ATCT and worse system performance. Figure 6 shows the queue lengths under different user distributions where there are 4 shards in the shard-based blockchain with the transaction rate equal to 250, from which we find that the efficient equilibrium state will lead to higher performance with less congestion than the random and uniform distribution.



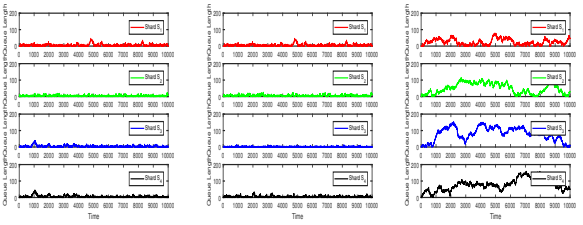
(a) Induced graph where nodes are the communities (b) Population size of each community

Figure 4: Community detection results



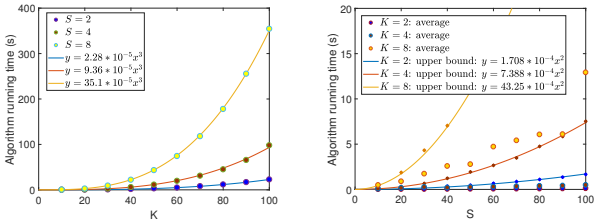
(a) ATCT with 2 shards (b) ATCT with 4 shards

Figure 5: Performance of the shard-based blockchain



(a) Centralized optimization (b) Efficient equilibrium state (c) Random and uniform distribution

Figure 6: Queue lengths of the shard-based blockchain



(a) Running time of Algorithm 1 with fixed S (b) Running time of Algorithm 1 with fixed K

Figure 7: Running time of Algorithm 1

7.4 Running Time of Algorithm 1

Theorem 6 shows the computational complexity of Algorithm 1 is $O\left(SK^3(S+\min\{S,K\})\right)$. Here, we investigate the running time of Algorithm 1 with different K and S. Figure 7 demonstrates the running time of Algorithm 1, which is implemented in Python 3.6.3 performed on a personal computer equipped with Intel(R) Core(TM) i7-7700HQ 2.80GHz CPU, 8.00GB memory, and a Windows 10 operation system. Figure 7 (a) shows that when fixing the value of S or $S \ll K$, the running time of Algorithm 1 increases cubically as K increases, which implies that the computational complexity of Algorithm 1 is $\Theta(K^3)$ if S is a constant. Figure 7 (b) shows that when K is a constant, the computational complexity is $O(S^2)$. On average, the running time of Algorithm 1 with fixed K is approximately linear because the loop in Lines 4-17 proceeds $O(S)$ rounds in total in most cases, which shows the high efficiency of Algorithm 1

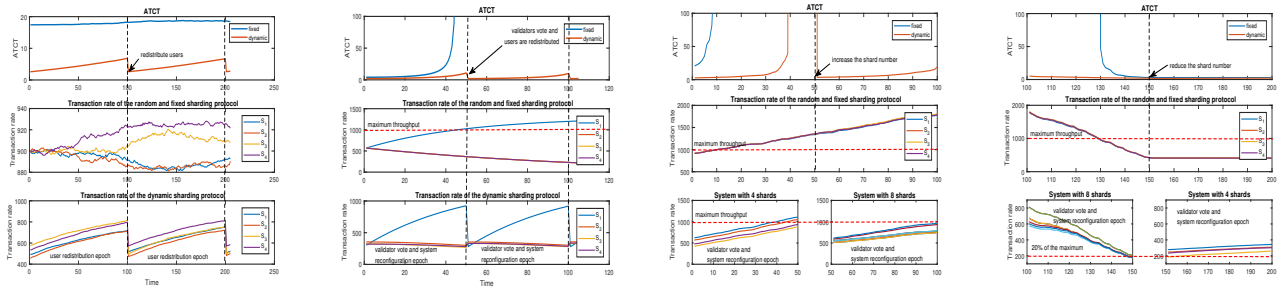
7.5 Performance of the Sharding Protocol with Dynamic User Distributions

In this subsection, simulations are conducted to investigate the performance of the proposed dynamic sharding protocol.

Figure 8 (a) shows that as the user transaction pattern in the blockchain system evolve gradually, the ATCT of the system increases and the system performance becomes lower. Compared to the fixed sharding protocol with random and uniform user distribution initially, the dynamic sharding protocol with periodical user redistribution can help to system maintain the good performance with low ATCT. Figure 8 (b) illustrates the situation when the the transactions are gradually concentrated in the first shard, leading to the infinite queueing time of that shard and infinite ATCT of the system. As mentioned in Section 6.4, in this situation, the validators in the first shard can obtain a high transaction fee reward, leaving the other validators in other shards with low transaction fee reward. Therefore, validators in other shards are motivated to vote for user redistribution, and the user will be redistributed to balance the transaction fee reward and the improve the system performance. Figure 8 (c) illustrates the situation when the transaction rate exceeds the system maximum throughput. In this situation, the validators will double the number of shards and redistribute users, which can help to maintain the system stability and boost the system performance. As shown in Figure 8 (d), as the transaction rate decreases, the validators will halve the number of shards and redistribute users, which can help to increase the system utility and maintain the system security.

8 RELATED WORK

Several blockchain sharding protocols, such as Elastico [12], Monoxide [24], OmniLedger [9], Rapidchain [28], Chainspace [1] and Ethereum 2.0 [3], have been proposed to achieve horizontal scalability of the blockchain system. The increasing number of cross-shard transactions is one of the key challenges in blockchain sharding protocols. Several studies have been presented to solve the problem [17, 19]. Due to the decentralization of the blockchain network, game theory [25] is an ideal modeling tool and has been utilized to analyze users' interactions within the blockchain network [11, 13, 18]. Besides, queueing theory has been recently adopted to analyze the transaction confirmation time in blockchain network [8, 21]. Our work puts forwards a more comprehensive model based on Jackson queueing network for shard-based blockchains. As emphasized in



(a) User redistribution with the evolution of transaction pattern

(b) Validator vote and user redistribution with the concentration of transactions

(c) System reconfiguration and user redistribution with the increase in transactions

(d) System reconfiguration and user redistribution with the decrease in transactions

Figure 8: Performance of the dynamic sharding protocol

a recent survey paper [27], transaction-aware user distribution in shard-based blockchains remains a key challenging issue. As a thrust towards this direction, we investigate the transaction-aware user distribution in the shard-based blockchain through queueing modeling and game theoretical approach and design a sharding protocol with dynamic user distributions to boost the system performance.

9 CONCLUSION

In this paper, we study the impact of user distributions on the performance of the shard-based blockchain network. We propose a Jackson queueing network model to formulate the average transaction confirmation time and the transaction fees. We also develop a shard-based blockchain game to capture the decentralized nature of blockchain for users to optimize their utilities. We further define a novel metric of ATCT to quantify the system performance, and propose an polynomial-time algorithm to find an efficient equilibrium. Besides, we propose a novel sharding protocol with dynamic user distribution based on our theoretical analysis and proposed algorithm. Experimental results demonstrate the efficacy of the algorithm and the high performance of the proposed protocol.

REFERENCES

[1] Mustafa Al-Bassam, Alberto Sonnino, Shehar Bano, Dave Hrycyszyn, and George Danezis. 2017. Chainspace: A sharded smart contracts platform. *arXiv preprint arXiv:1708.03778* (2017).

[2] Rory Bowden, Holger Paul Keeler, Anthony E Krzesinski, and Peter G Taylor. 2018. Block arrivals in the Bitcoin blockchain. *arXiv preprint arXiv:1801.07447* (2018).

[3] V. Buterin. 2020. *Ethereum Sharding FAQ*. Retrieved May 13, 2020 from https://github.com/ethereum/wiki/wiki/Sharding-FAQ

[4] Canhui Chen, Qian Ma, Xu Chen, and Jianwei Huang. [n.d.]. *Online technical report*. https://www.dropbox.com/s/3hhodpw0tynz8jh/technical%20report_User%20Distributions.pdf?dl=0 [Online].

[5] David Easley, Maureen O'Hara, and Soumya Basu. 2019. From mining to markets: The evolution of bitcoin transaction fees. *Journal of Financial Economics* 134, 1 (2019), 91–109.

[6] Santo Fortunato. 2010. Community detection in graphs. *Physics reports* 486, 3-5 (2010), 75–174.

[7] Erol Gelenbe and Guy Pujolle. 1998. *Introduction to queueing networks*. Vol. 2. Wiley New York.

[8] Shoji Kasahara and Jun Kawahara. 2016. Effect of Bitcoin fee on transaction-confirmation process. *arXiv preprint arXiv:1604.00103* (2016).

[9] 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)*. IEEE, 583–598.

[10] Juanjuan Li, Yong Yuan, Shuai Wang, and Fei-Yue Wang. 2018. Transaction queuing game in bitcoin blockchain. In *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 114–119.

[11] Ziyao Liu, Nguyen Cong Luong, Wenbo Wang, Dusit Niyato, Ping Wang, Ying-Chang Liang, and Dong In Kim. 2019. A survey on applications of game theory in blockchain. *arXiv preprint arXiv:1902.10865* (2019).

[12] 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*. 17–30.

[13] 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.

[14] Swan Melanie, Potts Jason, Takagi Soichiro, Witte Frank, and Tasca Paolo. 2019. *Blockchain Economics: Implications Of Distributed Ledgers-Markets, Communications Networks, And Algorithmic Reality*. Vol. 1. World Scientific.

[15] Holly Moore. 2017. *MATLAB for Engineers*. Pearson.

[16] Satoshi Nakamoto. 2019. *Bitcoin: A peer-to-peer electronic cash system*. Technical Report. Manubot.

[17] Lan N Nguyen, Truc DT Nguyen, Thang N Dinh, and My T Thai. 2019. OptChain: optimal transactions placement for scalable blockchain sharding. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 525–535.

[18] Zhengwei Ni, Wenbo Wang, Dong In Kim, Ping Wang, and Dusit Niyato. 2019. Evolutionary Game for Consensus Provision in Permissionless Blockchain Networks with Shards. In *IEEE ICC*. 1–6.

[19] Naoya Okanami, Ryuya Nakamura, and Takashi Nishide. 2020. Load Balancing for Sharded Blockchains. *engrXiv preprint engrXiv:fyqar* (2020).

[20] Christos H Papadimitriou and Kenneth Steiglitz. 1998. *Combinatorial optimization: algorithms and complexity*. Courier Corporation.

[21] Saulo Ricci, Eduardo Ferreira, Daniel Sadoc Menasche, Artur Ziviani, Jose Eduardo Souza, and Alex Borges Vieira. 2019. Learning blockchain delays: a queueing theory approach. *ACM SIGMETRICS Performance Evaluation Review* 46, 3 (2019), 122–125.

[22] János Sztrik. 2012. Basic queueing theory. *University of Debrecen, Faculty of Informatics* 193 (2012), 60–67.

[23] Gang Wang, Zhijie Jerry Shi, Mark Nixon, and Song Han. 2019. Sok: Sharding on blockchain. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*. 41–61.

[24] Jiaping Wang and Hao Wang. 2019. Monoxide: Scale out blockchains with asynchronous consensus zones. In *16th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 19)*. 95–112.

[25] James N Webb. 2007. *Game theory: decisions, interaction and Evolution*. Springer Science & Business Media.

[26] Gavin Wood et al. 2014. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper* 151, 2014 (2014), 1–32.

[27] Guangsheng Yu, Xu Wang, Kan Yu, Wei Ni, J Andrew Zhang, and Ren Ping Liu. 2020. Survey: Sharding in blockchains. *IEEE Access* 8 (2020), 14155–14181.

[28] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. 2018. Rapidchain: Scaling blockchain via full sharding. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 931–948.

[29] Peilin Zheng, Zibin Zheng, Jiajing Wu, and Hongning Dai. 2020. Xblock-ETH: Extracting and Exploring Blockchain Data from Ethereum. *IEEE Open Journal of the Computer Society* (2020).