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

Canhui Chen, Qian Ma, Xu Chen, and Jianwei Huang

- Canhui Chen, Qian Ma, and Xu Chen are with Sun Yat-sen University, Guangzhou, China
- Jianwei Huang is with The Chinese University of Hong Kong, Shenzhen, China

Framework

1.Background and Motivation

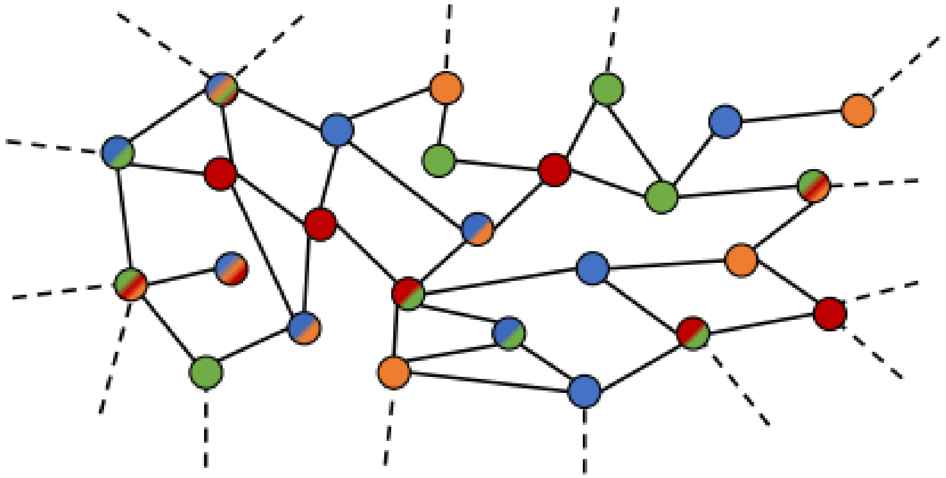
2.System Model

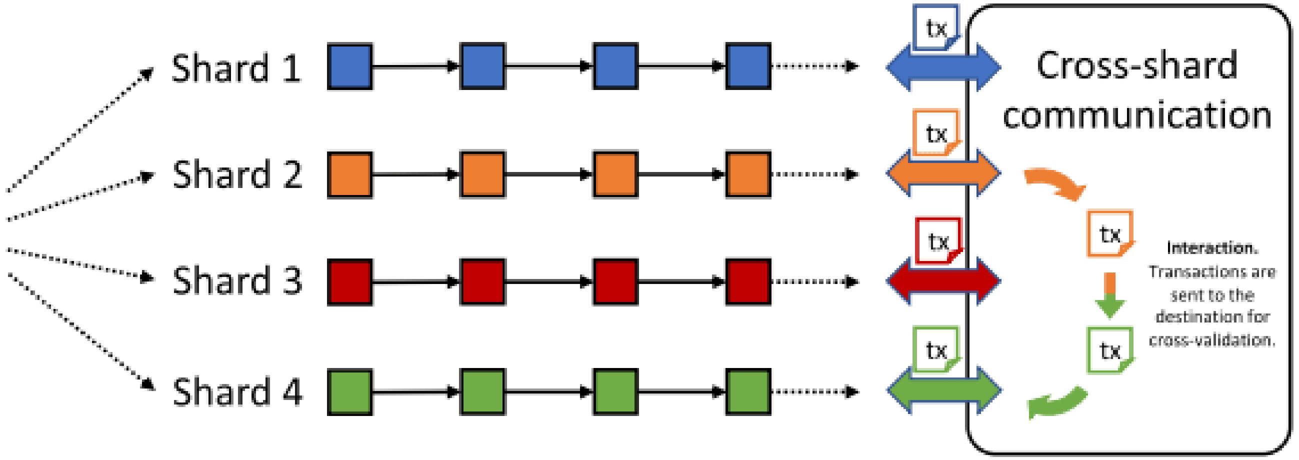
3.Shard-based Blockchain Game

4.Dynamic Sharding Protocol

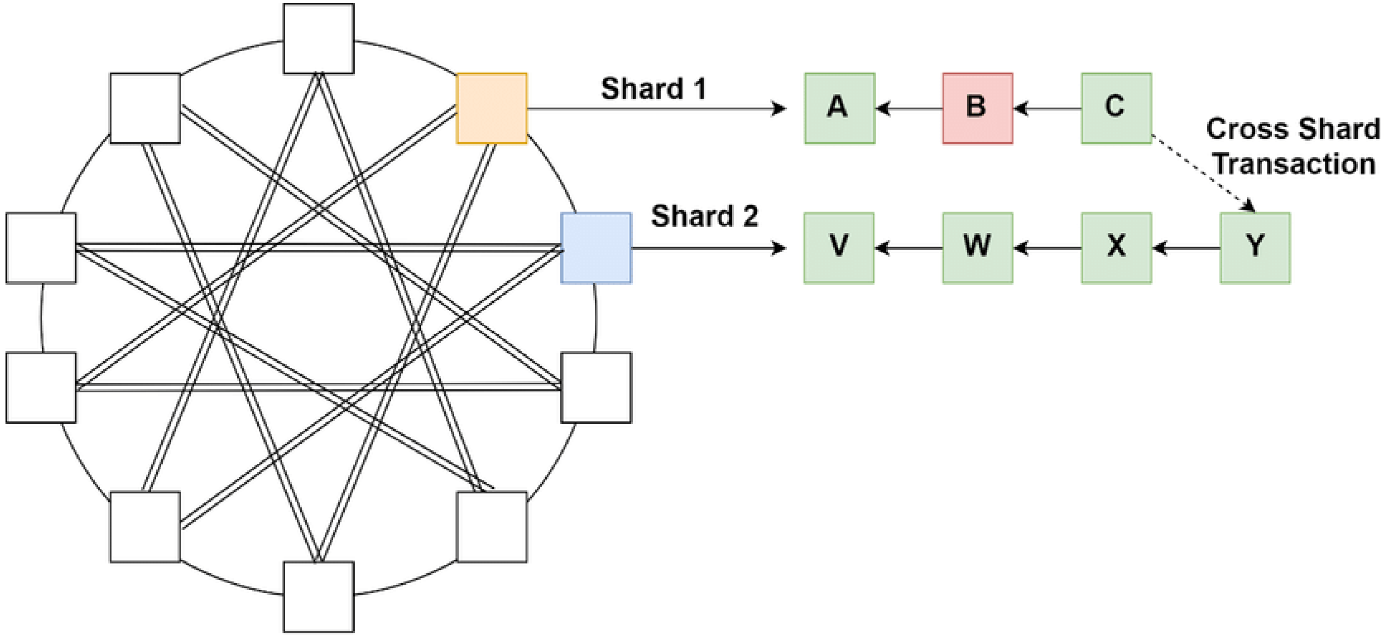
5.Conclusion

Background and Motivation





Cross-shard Transaction Problems



How to reduce the cross-shard transaction?

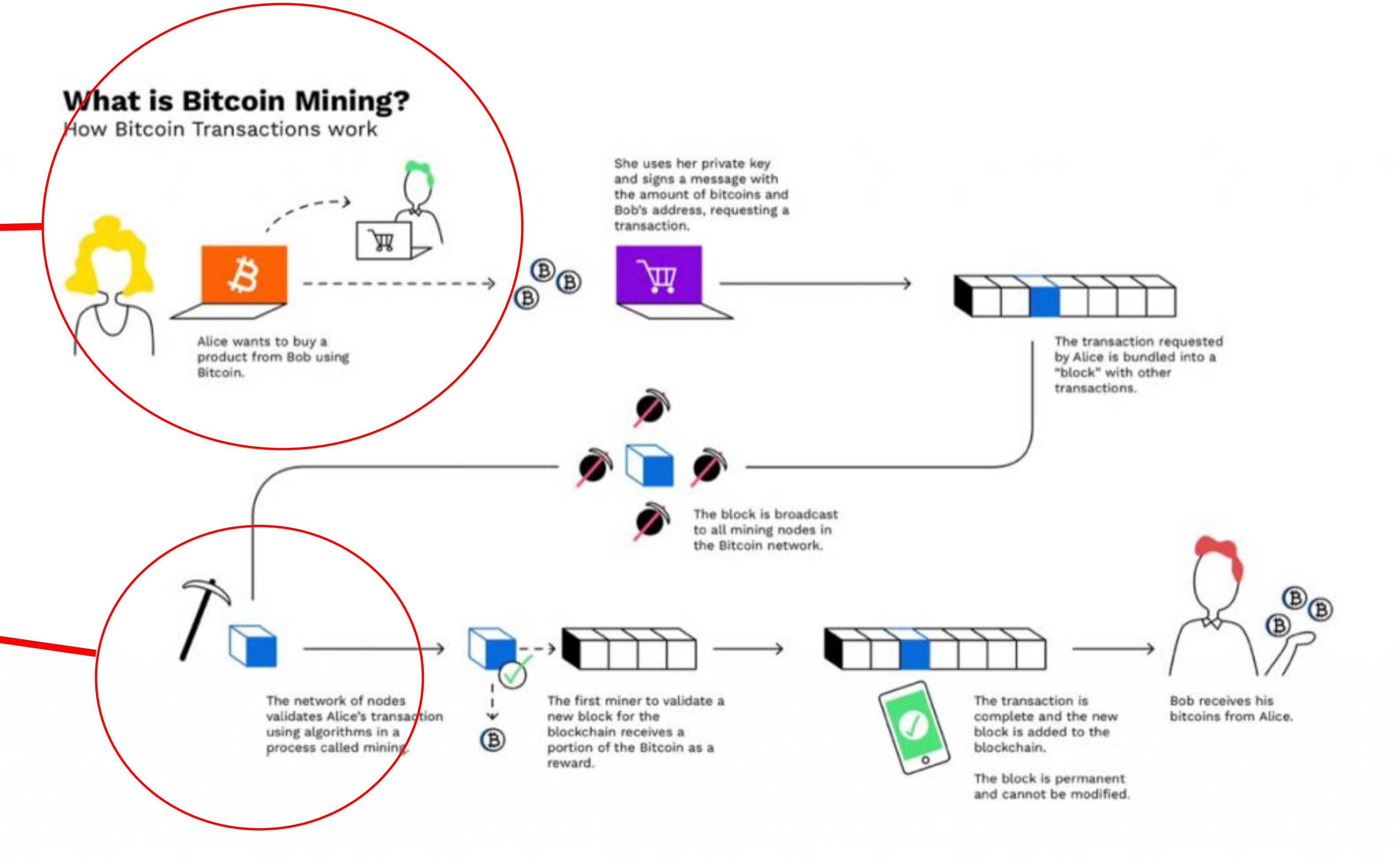
- Longer transaction confirmation time
- Higher transaction fee

Participants in Shard-based Blockchain

Difference: Our work focus on the user perspective!

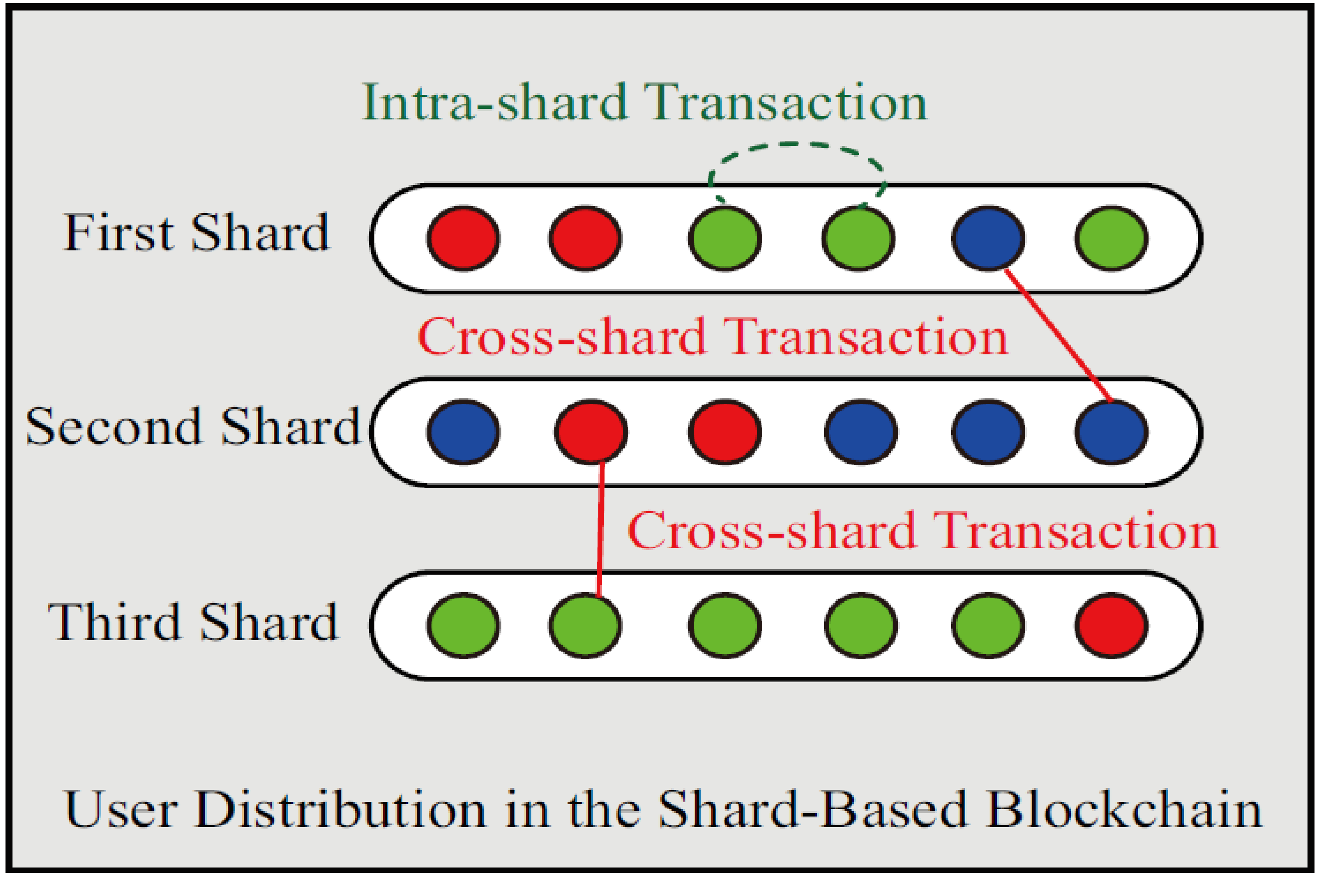
User
initiate or receive transactions

Validator
store data, process transactions, add new block



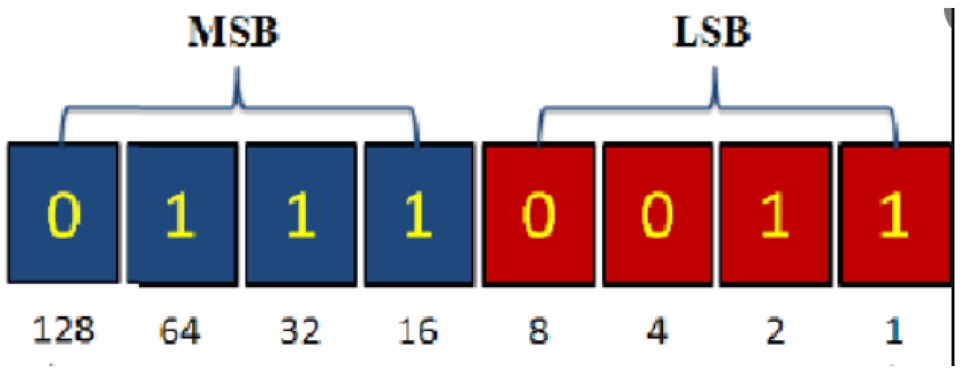
Participants in Shard-based Blockchain

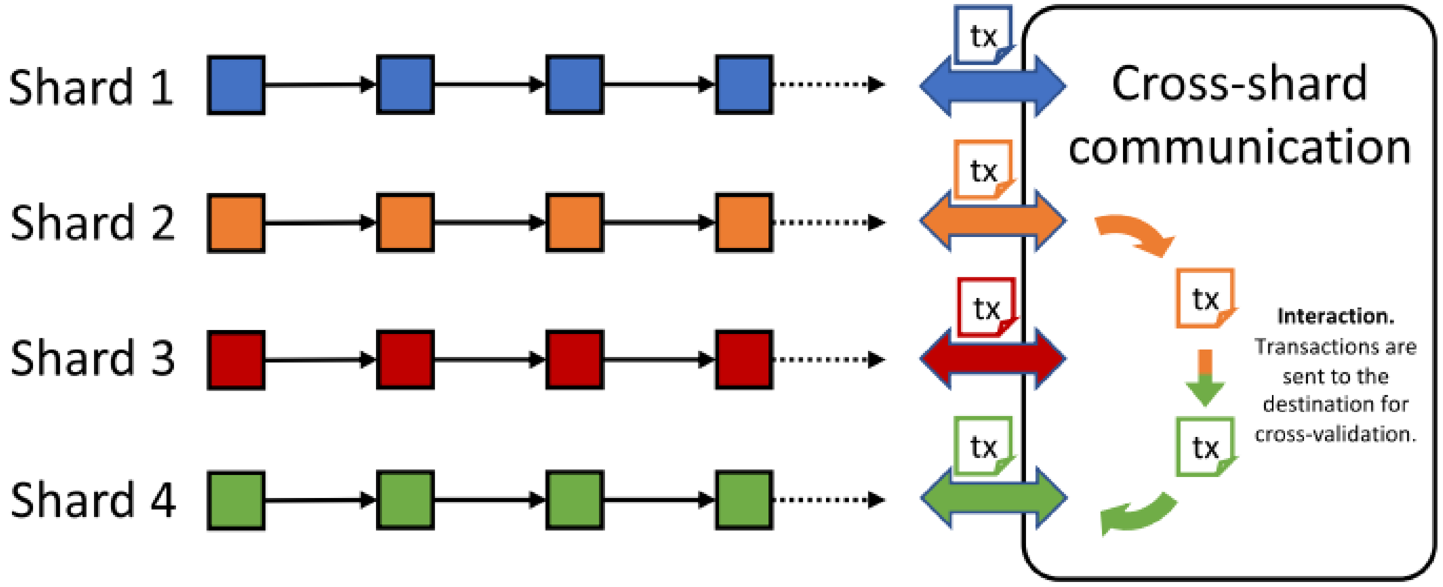
User distribution will directly affect the number of cross-shard transactions



Current User Distribution in Shard-based Blockchain

- A user is represented by its address.
- A user is allocated to the shard using some fixed rules (e.g., based on the least significant bits).
- This allocation is similar to the random and uniform distribution.

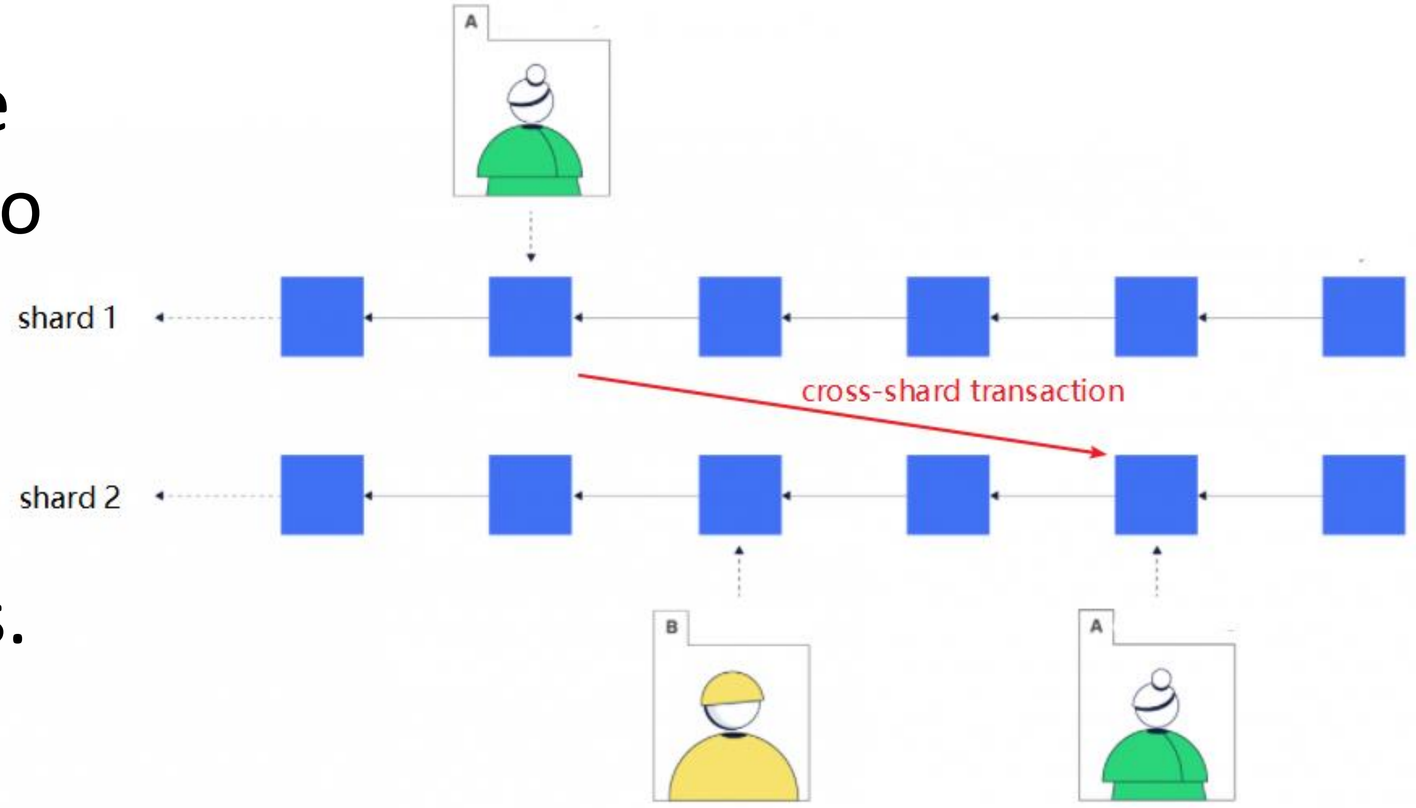




As the number of shards increase, the cross-shard transactions will dominate.

Dynamic Shard Selection

- Generate a number of addresses and pick up the address that is allocated to the certain shard.
- Initiate a cross-shard transaction to transfer assets to the new address.

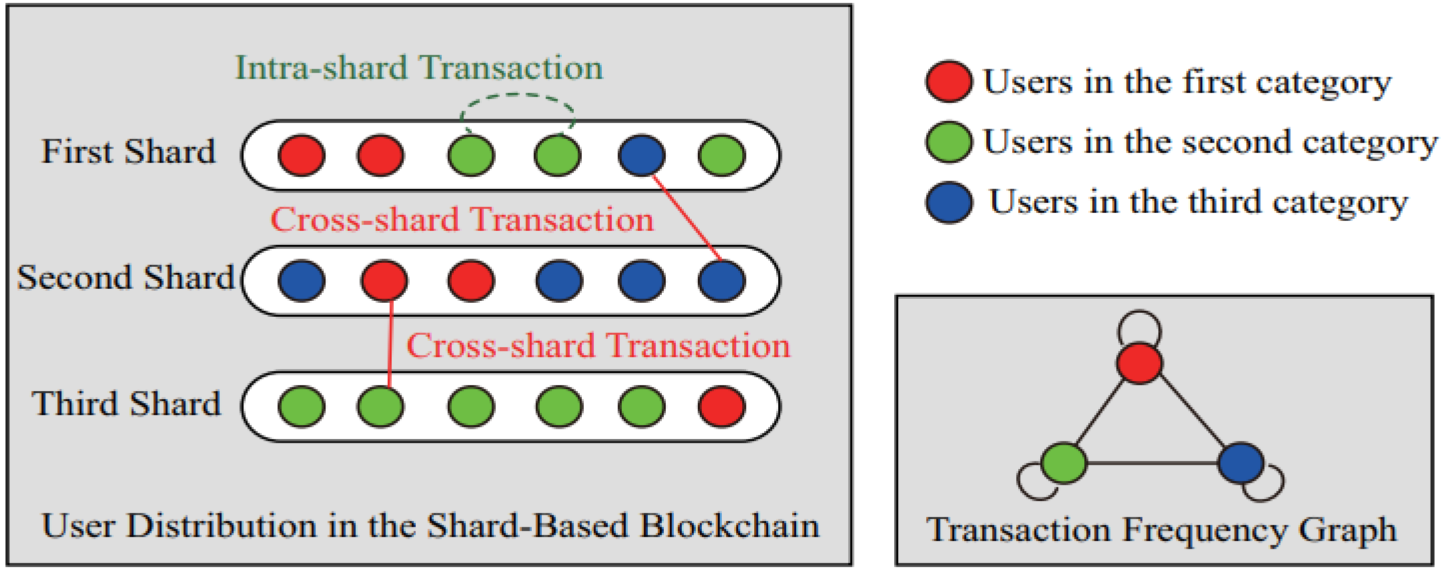


Key Questions

- **Key Question 1**: How good is such random user distribution in terms of system transaction performance?
- **Key Question 2**: Is there a transaction-aware user distribution achieves a better system performance than the random user distribution and no user has the incentive to deviate from?

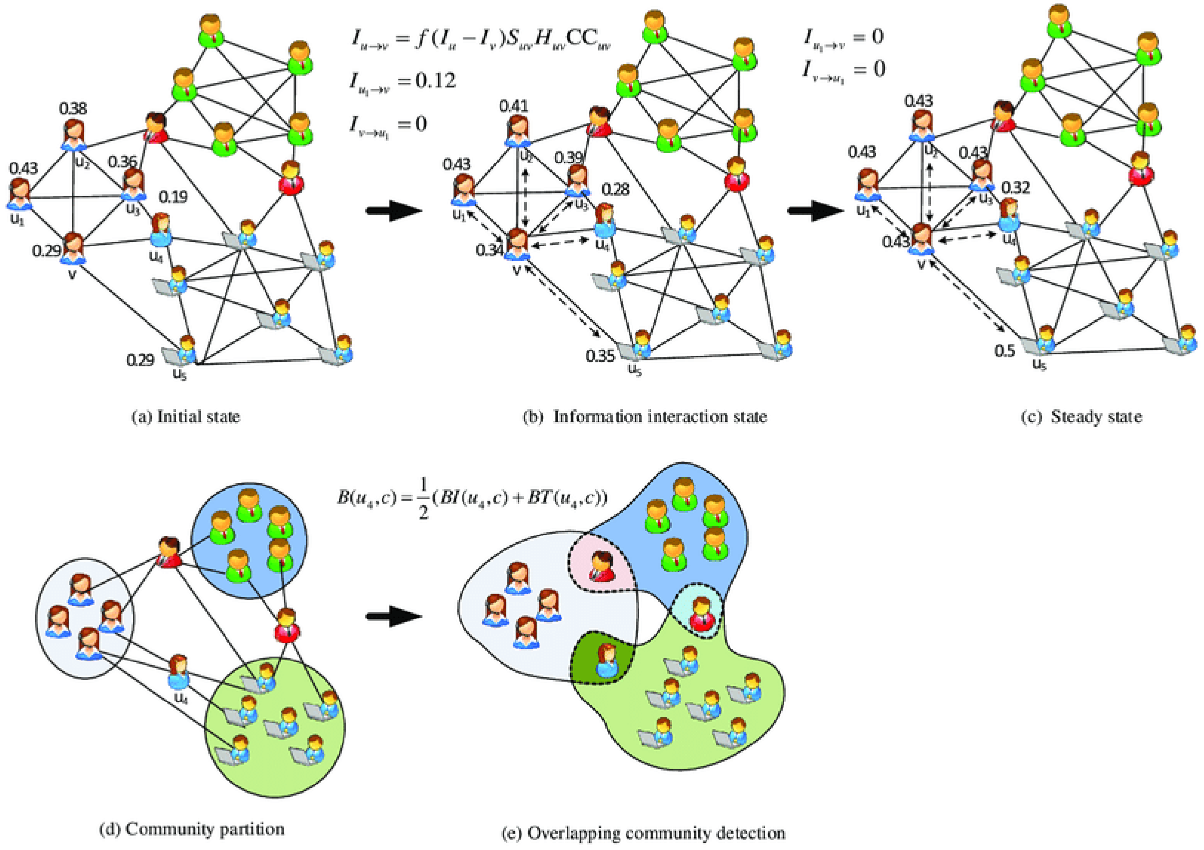
Outline

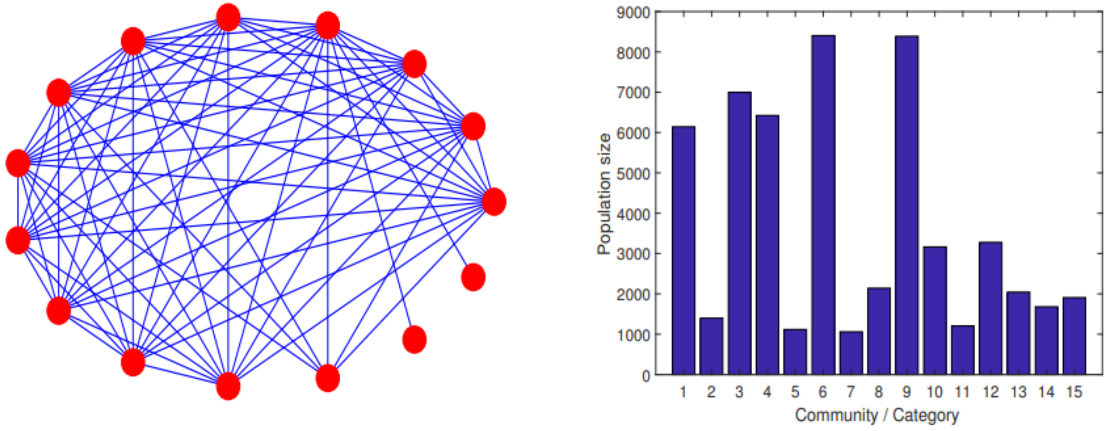
System Model



A shard-based blockchain system model

Community Detection





(a) Induced graph where nodes are the communities

(b) Population size of each community

Community detection results

Queueing Model

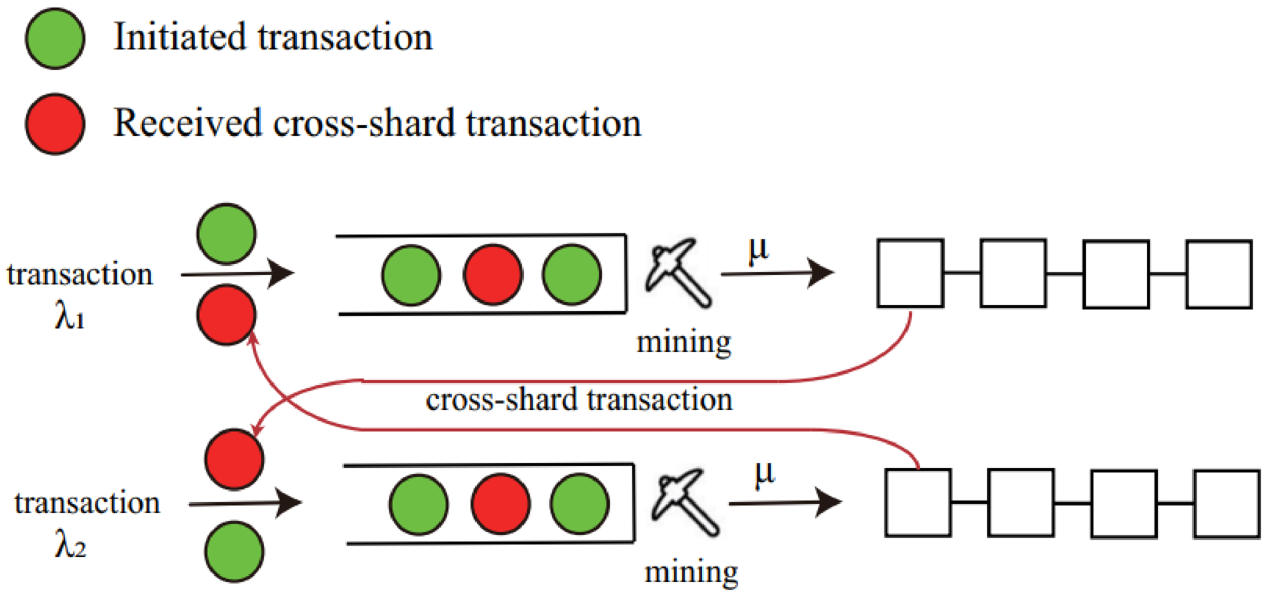


Figure Illustration of the queueing model for shard-based blockchain. Each shard is modeled as an M/M/1 queue, and two shards form an open Jackson queueing network.

Average Transaction Confirmation Time and Transaction Fee

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

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

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 average transaction confirmation time and the average transaction fee is related to

- Current user distribution
- User's category
- User's shard selection

The users' dynamic shard selection can be modeled as a game

Outline

Shard-based Blockchain Game

- Users' utility is related to
  - Average transaction confirmation time
  - Average transaction fee

$$U(i,k,R) = -W(i,k,R) - \beta F(i,k,R),$$

- Users have partial control regarding which shard to join.
- The users' dynamic shard selection will affect the queueing performance of each shard, leading to variable transaction confirmation times and transaction fees, which thus forms the **shard-based blockchain game**.

Equilibrium Analysis

Definition 1 *(Nash Equilibrium). A population state R^* is a Nash equilibrium of the shard-based blockchain game if for any $R^*(i,k) \neq 0$ we have*

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

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

Definition 2 *(Approximate Nash Equilibrium). 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*

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

$\epsilon \geq 0$ is the gap from a Nash equilibrium, and can be understood as the maximum switching cost that a user can tolerate for transferring to another shard, e.g., cross-shard transaction fee.

Algorithm for Efficient Equilibrium

Heuristic rules for equilibrium distribution

- Users of the same category should be randomly and uniformly distributed in their selected shards.

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

- Users of different categories should be distributed in different shards, otherwise, they should be distributed identically.

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

Algorithm for Efficient Equilibrium

Algorithm 1 Search for efficient equilibrium — Greedy

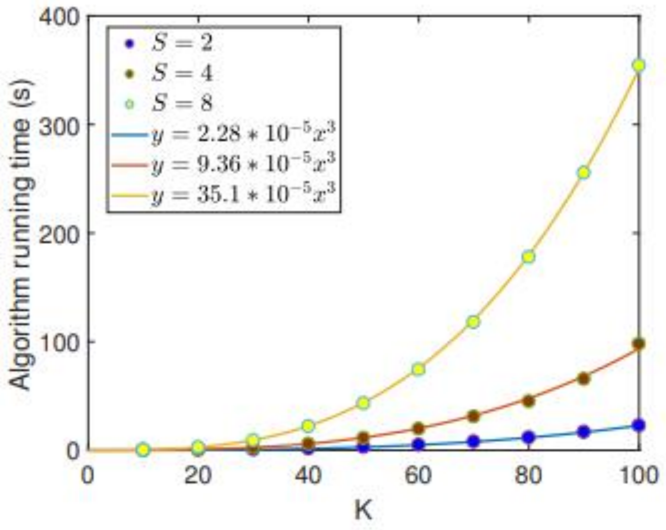
Sort the user category list such that we have $r_1 q_1 \geq r_2 q_2 \geq \cdots \geq r_K q_K$.

for $k = 1$ to K **do**
  $T = \emptyset$.
  for $n = 1$ to $S - |\cup_{i=1}^{k-1} \mathcal{X}_i|$ **do**
    $C = \{|\cup_{i=1}^{k-1} \mathcal{X}_i| + 1, \ldots, |\cup_{i=1}^{k-1} \mathcal{X}_i| + n\}$.
    Distribute users in category k into shards C based on the heuristic rule (1) and construct the distribution R_C.
    if $T \neq \emptyset$ and $U(*, i, R_T) > U(*, i, R_C)$ **then**
      break
    end if — Congestion avoidance
    if $U(*, i, R_C) > -\infty$, i.e., the system is stable **then**
      $T = C$.
      $u = \sum_{i=k+1}^{K} U(*, i, R')r_i / \sum_{i=k+1}^{K} r_i$, where R' is the user distribution with users in category $k+1, \ldots, K$ distributing in shards $\{|\cup_{i=1}^{k-1} \mathcal{X}_i| + n + 1, \ldots, S\}$.
      if $U(*, k, R_C) > u$ **then**
        break
      end if
    end if
  end for
  $\mathbb{X} = \{\mathcal{X}_1, \ldots, \mathcal{X}_{k-1}\} \cup \{T\}$. — Constraint of heuristic (2)
  $\mathcal{X}_k = \arg\max_{\mathcal{X} \in \mathbb{X}} U(*, k, R'')$, where R'' is the distribution after distributing users in category k into shards $\mathcal{X}$ based on heuristic (1).
  Distribute users in category k into shards $\mathcal{X}_k$ based on heuristic (1).
  while there exists a user who has the incentive to move from shard i to shard j **do**
    $\mathcal{P} = \{m | i \in \mathcal{X}_m \text{ or } j \in \mathcal{X}_m, m = 1, \ldots, K\}$.
    $\mathcal{X} = \cup_{m \in \mathcal{P}} \mathcal{X}_m$.
    Redistribute users of category m into shards $\mathcal{X}$ based on rule (1) and update $\mathcal{X}_m = \mathcal{X}, \forall m \in \mathcal{P}$.
  end while — Guarantee for equilibrium
end for
return R, where R is the final user distribution.

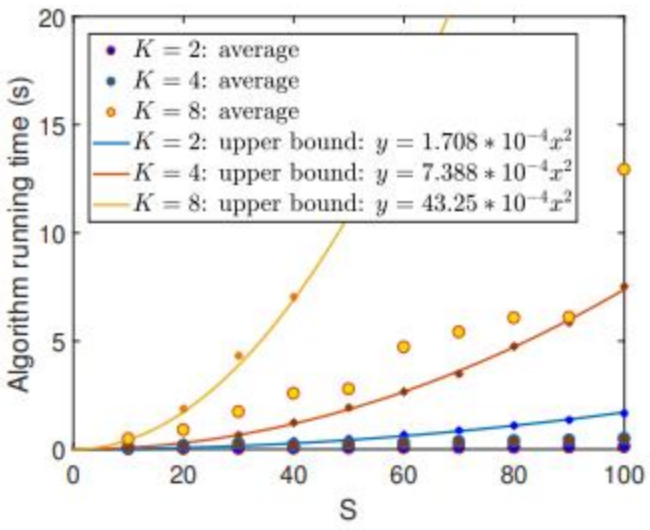
Experiment Results

Performance of Algorithm 1

- Polynomial time
- Low computational complexity and high efficiency



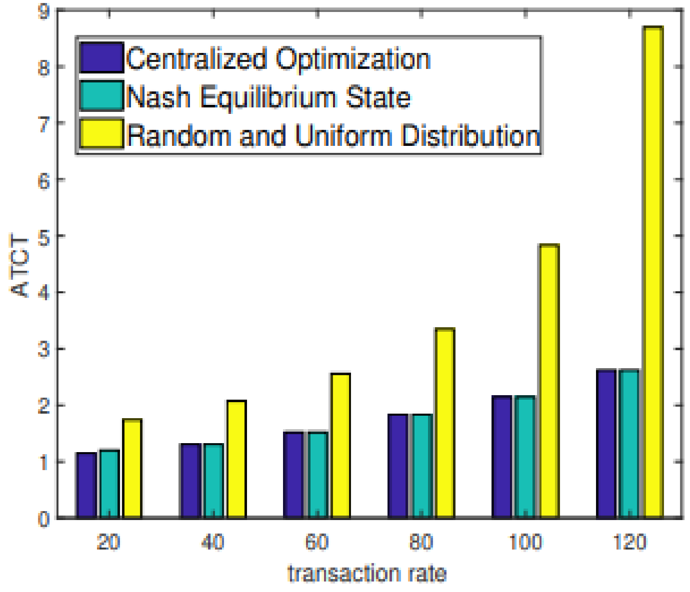
(a) Running time of Algorithm 1 with fixed S



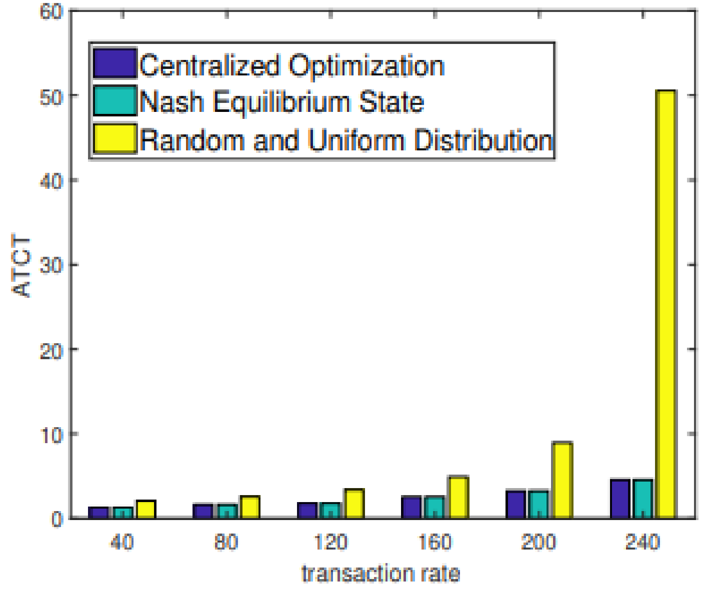
(b) Running time of Algorithm 1 with fixed K

Experiment Results

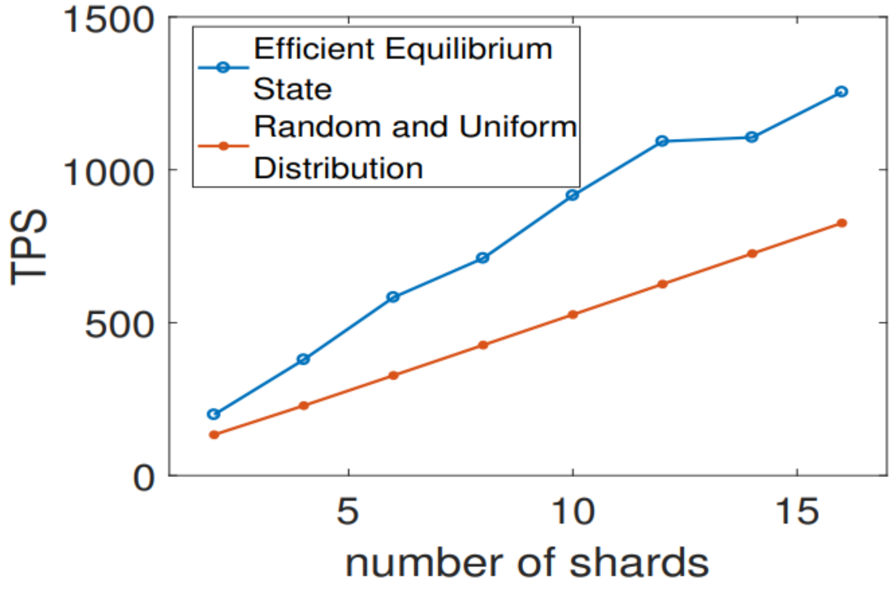
- Lower average transaction confirmation time (ATCT)
- Higher system throughout (TPS)
- Near-optimal performance



(a) ATCT with 2 shards



(b) ATCT with 4 shards



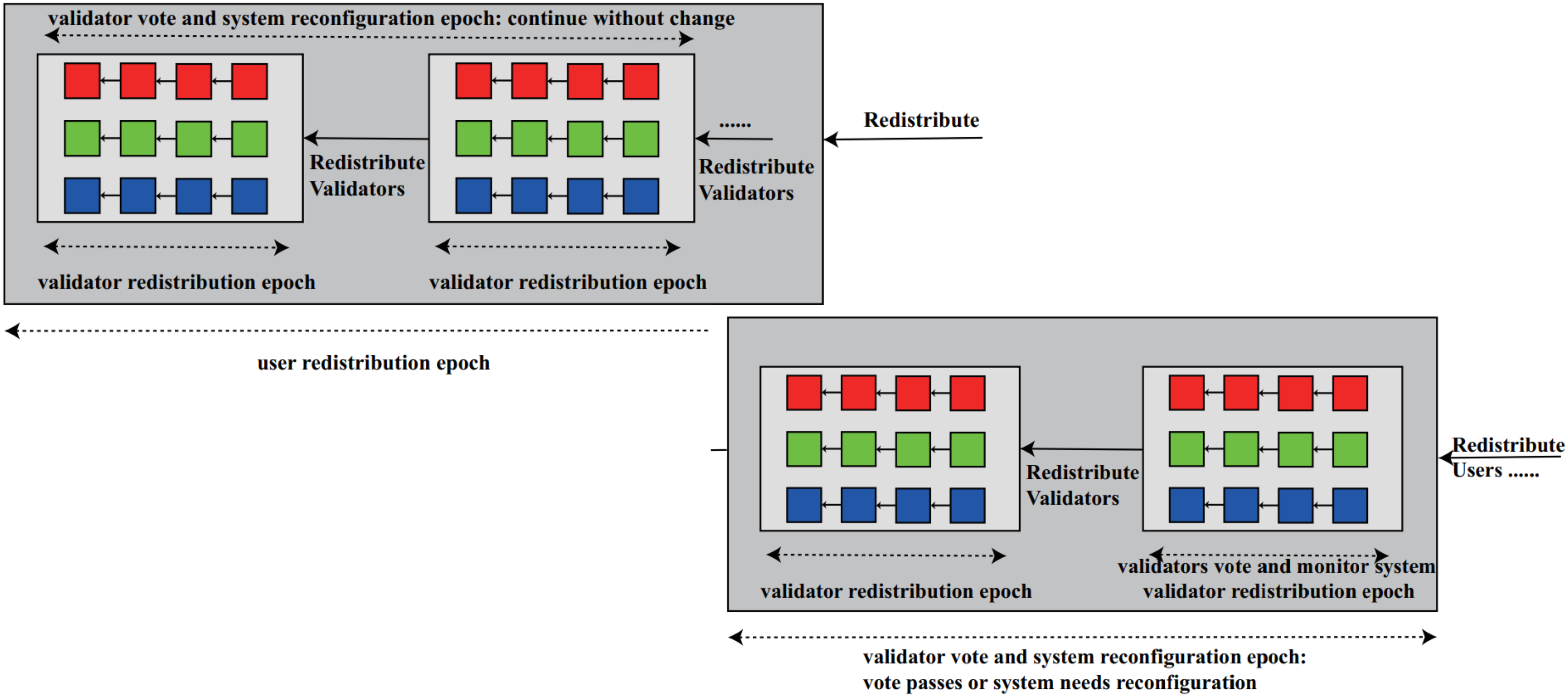
(c) TPS of different user distributions

Outline

Dynamic Sharding Protocol

- **Validator Redistribution Epoch**： validators will be randomly distributed to different shards.
- **User Redistribution Epoch:** users will be redistributed to different shards based on the proposed algorithm.
- **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.

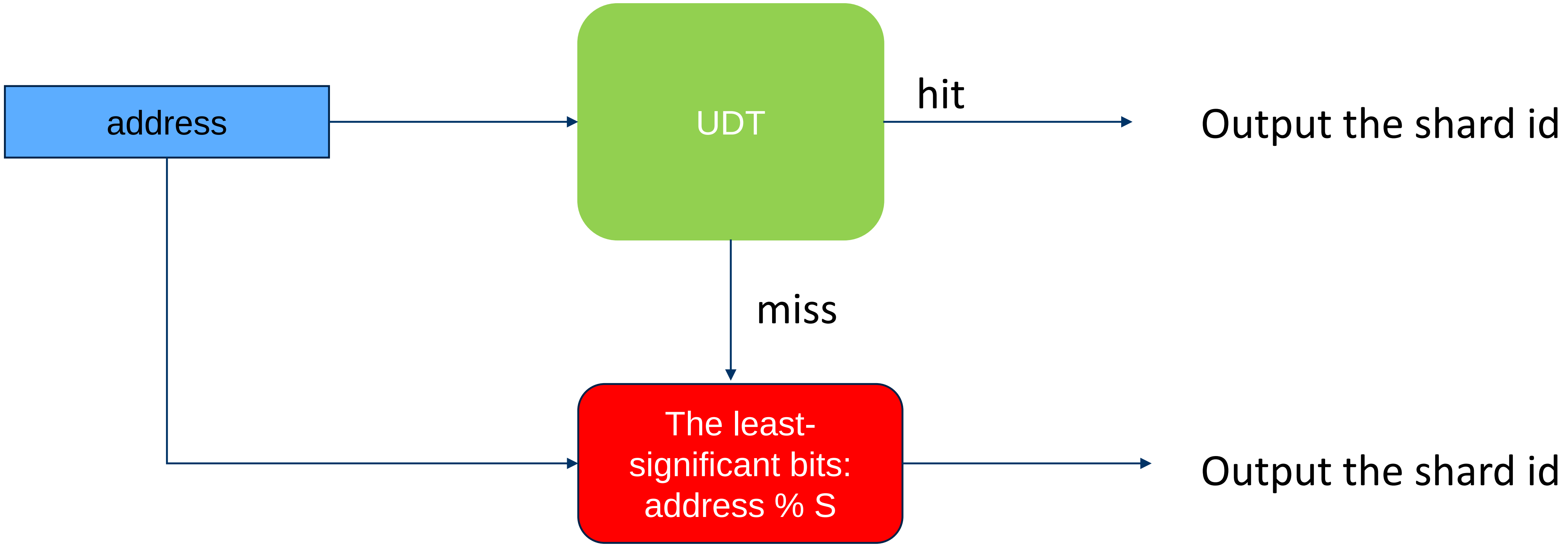
Dynamic Sharding Protocol



User Distribution Table (UDT)

- All the validators maintain a global user distribution table
- UDT maps each user address to his shard
- UDT only need to record the mapping information of the users that initiated or received at least b (b > 1) transactions in the previous user distribution epoch
- UDT is small enough to be stored in the memory cache
- 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.

User Distribution Table (UDT)



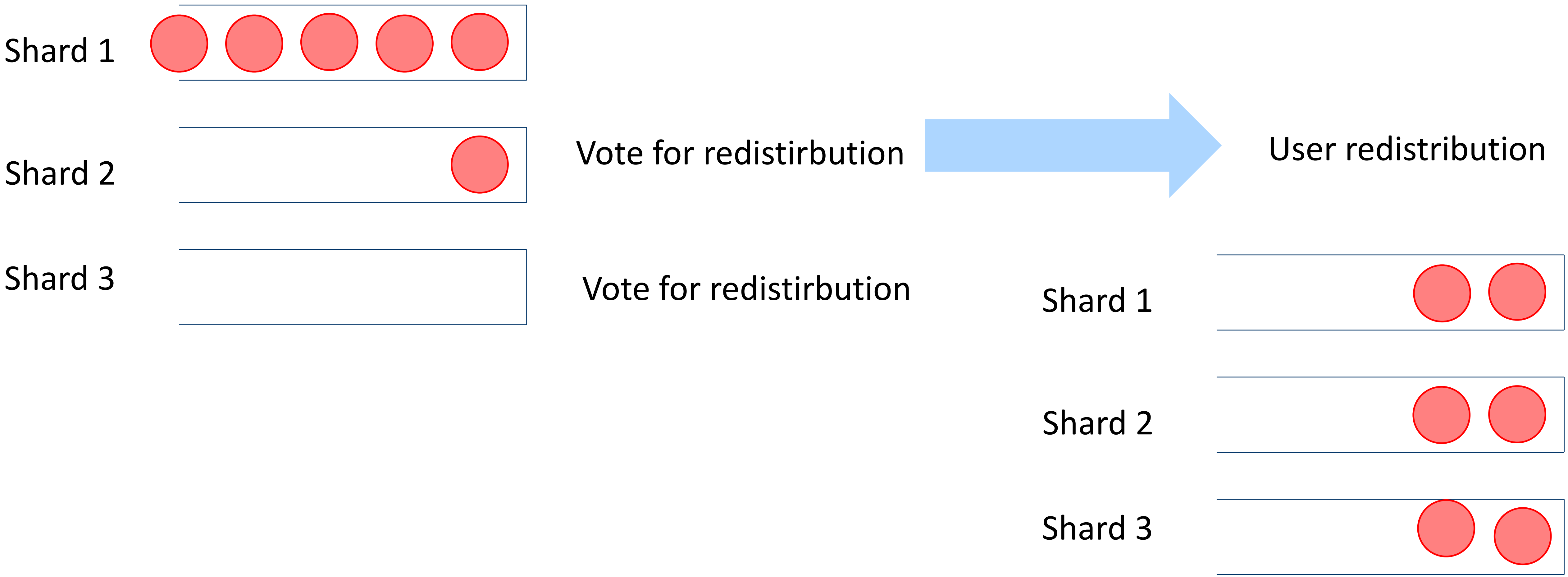
User Redistribution Process

- Validators in shard i generate a statistics block and broadcast it.
- Validators will construct a transaction network.
- All the validators run the community detection algorithm to classify users into different categories.
- All the validators run the proposed Algorithm to redistribute users to different shards and construct the user distribution table.
- Validators should broadcast the user distribution table to make it consistent globally.

Validator Vote Process

- Validators in shard i vote in the shard. If more than 50% validators vote for redistribution, the shard i will be marked as “**vote for redistribution**”
- If more than 2/3 of shards are marked as “**vote for redistribution**”, the user redistribution process will be conducted.

Validator Vote Process



System Reconfiguration Process

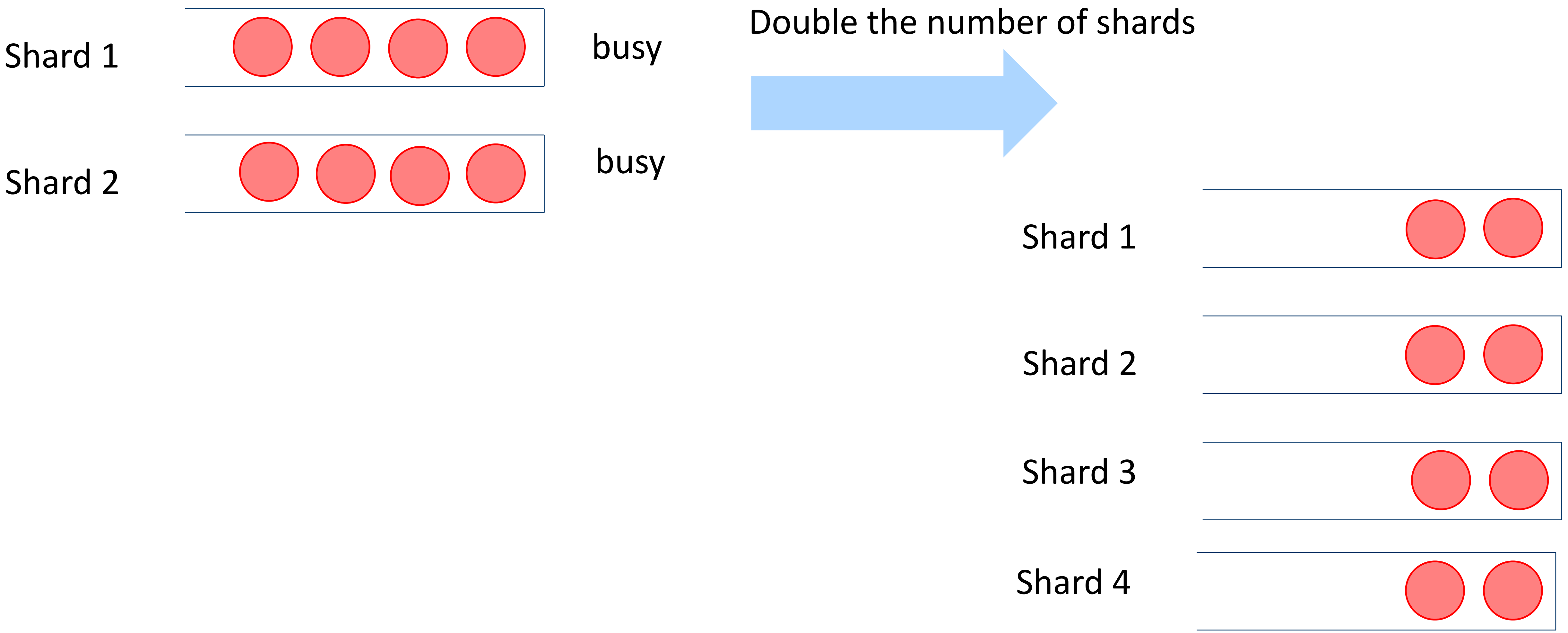
- Validators in shard i collect the historical block and transaction statistics from the last time of user redistribution, and broadcast it to other validators.
- If more than 80% blocks in shard i exceed 80% of the maximum block size, and the average transaction fee is at least 50% higher than that in the previous user redistribution epoch, shard i will be marked as "**busy shard**".
- 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**".

System Reconfiguration Process

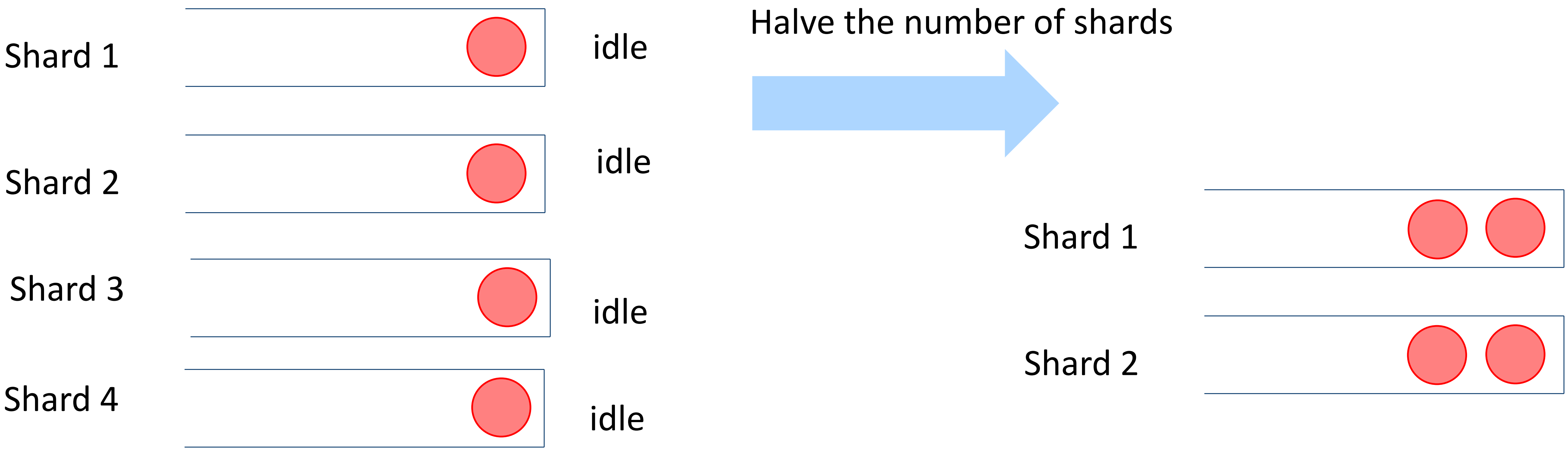
- If more than 80% shards are "**busy shard**", the system will double the number of shards and redistribute users.
- If more than 80% shards are "**idle shard**", the system will halve the number of shards and redistribute uses.

- Note: there exists a constraint between the number of shards and the number of validators.

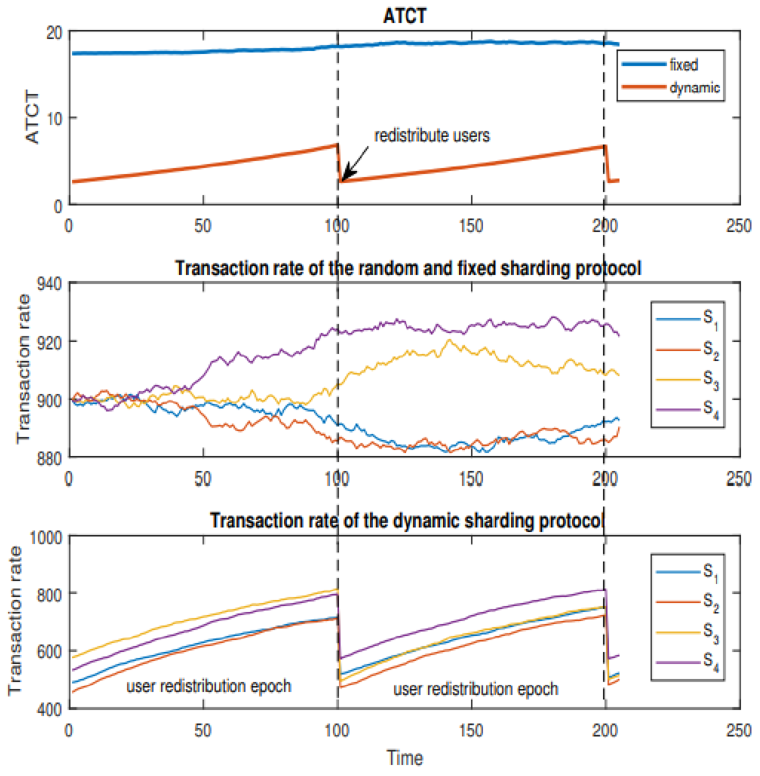
System Reconfiguration Process



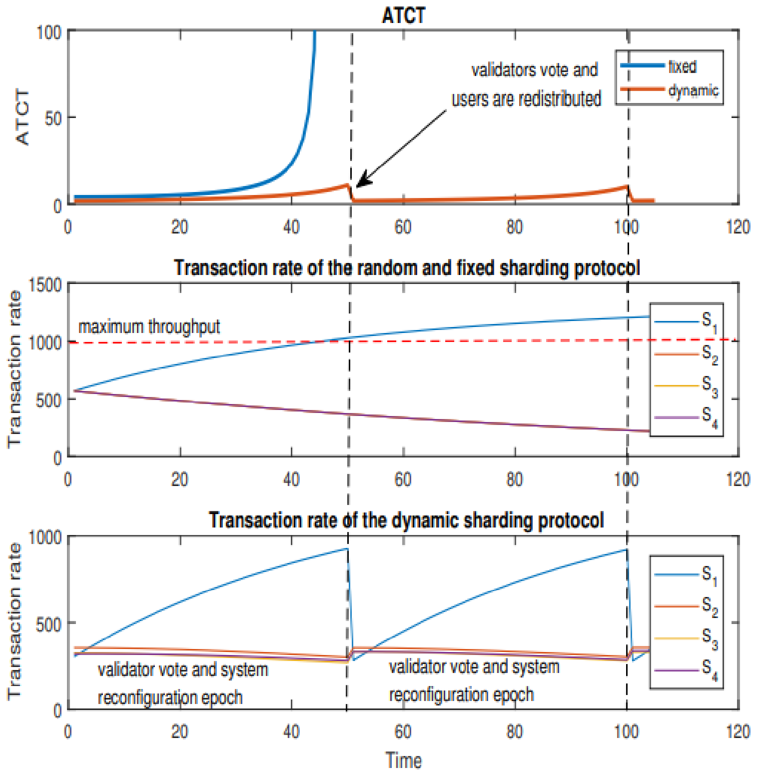
System Reconfiguration Process



Performance of the Dynamic Sharding Protocol



(a) User redistribution with the evolution of transaction pattern

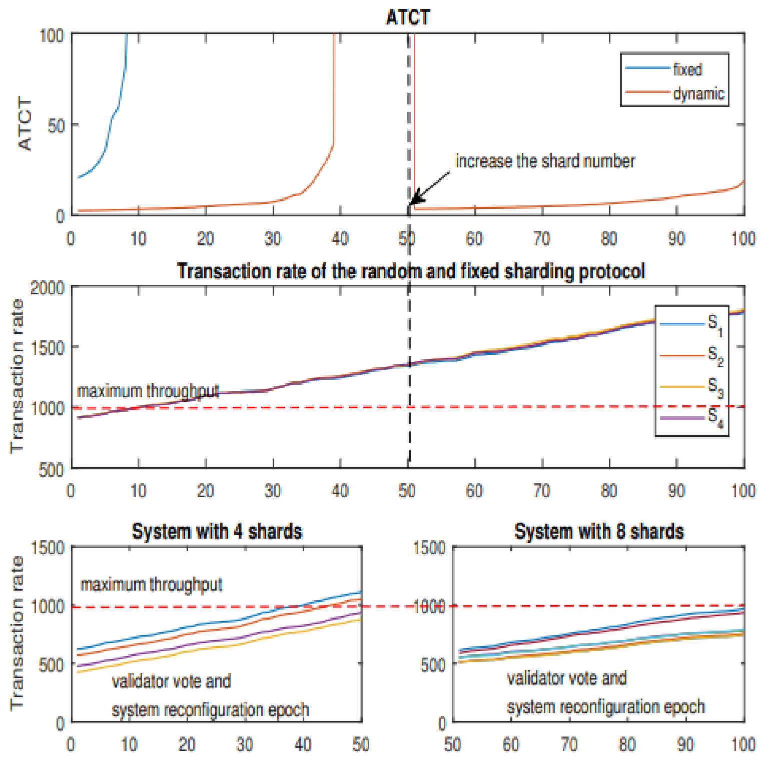


(b) Validator vote and user redistribution with the concentration of transactions

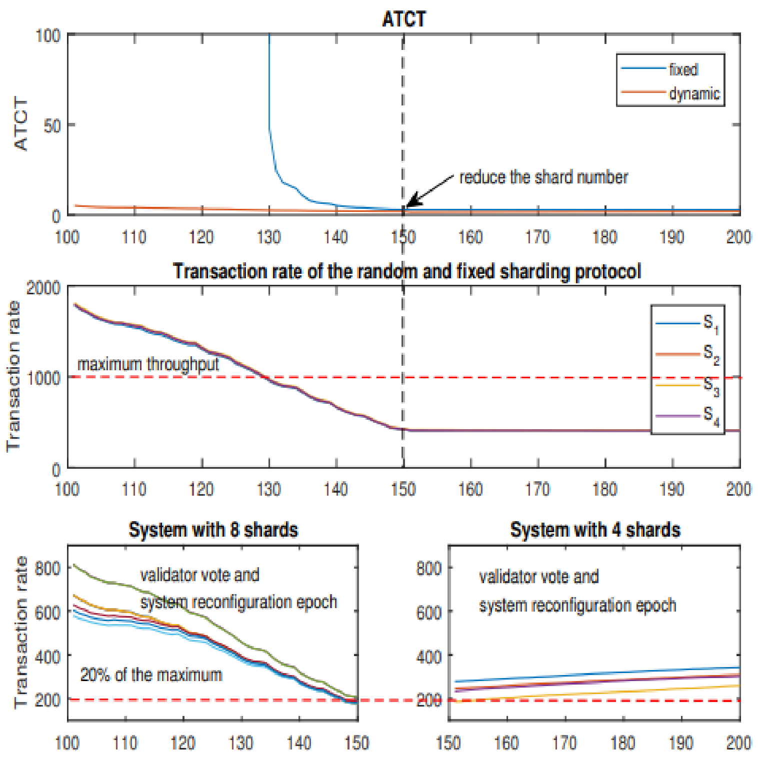
The dynamic sharding protocol
(a) Adapt to the dynamic change of the user transaction pattern.
(b) Rebalance the user distribution.

The property of equilibrium can maintain the system stability in long term.

Performance of the Dynamic Sharding Protocol



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



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

The dynamic sharding protocol
(c) Increase the number of shards, maintain system stability and boost the system performance.
(d) Decrease the number of shards, maintain system security, increase system utility and motivate validators.

Outline

Conclusion

- An open Jackson queueing network model to characterize the transaction dynamics of the shard-based blockchain.
- A novel shard-based blockchain game and explore various equilibria under different transaction patterns.
- An efficient equilibrium finding algorithm of low complexity, which can achieve an equilibrium with a near-optimal system performance.
- A novel sharding protocol with dynamic user distributions, which can maintain a good long-term performance in a dynamic environment.

Thanks