Justitia: An Incentive Mechanism towards the Fairness of Cross-shard Transactions

Jian Zheng*, Huawei Huang*, Yinqiu Liu†, Taotao Li*, Hong-Ning Dai‡, Zibin Zheng*
* School of Software Engineering, Sun Yat-Sen University, China
† College of Computing and Data Science, Nanyang Technological University, Singapore
‡ Department of Computer Science, Hong Kong Baptist University, HK SAR, China
Corresponding author: Huawei Huang (huanghw28@mail.sysu.edu.cn)

***Abstract*— A cross-shard transaction (CTX) is parsed into two sub-transactions, which are then executed in the source and destination shards, respectively. However, the problem is that the client who submits the original transaction only pays one unit of the transaction fee. Thus, sub-transactions will experience much higher queueing delays than regular intra-shard transactions when they wait in shard transaction pools. This is *unfair* for those original transactions that will be parsed into sub-transactions from the perspective of a sharded blockchain. Therefore, how to ensure fairness for all CTXs while securing the atomicity of any pair of sub-transactions becomes a critical challenge. State-of-the-art solutions addressed the transaction atomicity challenge, but the literature still lacks a dedicated incentive mechanism to ensure the fairness of CTXs. To this end, we propose an incentive mechanism named `Justitia`, which aims to achieve fairness by motivating blockchain proposers to prioritize the CTXs queueing in transaction pools when they package transactions to generate a new block. We rigorously analyze that `Justitia` upholds the fundamental properties of a sharded blockchain, including security, atomicity, and fairness. We then implement a prototype of `Justitia` on an open-source sharding-enabled blockchain testbed. Our experiments using historical Ethereum transactions demonstrate that i) `Justitia` guarantees fairness while processing CTXs, ii) its token-issuance mechanism does not lead to unstable economic inflation, and iii) `Justitia` only yields 20%-80% of queueing latency for CTXs upon comparing with Monoxide protocol.**

***Index Terms*—Incentive, Blockchain, Fairness, Sharding.**

I. Introduction

In sharded blockchains, there are two types of transactions, i.e., intra-shard transactions (abbr. as ITX) and cross-shard transactions (abbr. as CTX). Suppose that a client submits a TX $\langle A \rightarrow B : x, f \rangle$ as shown in Fig. 1, where account A transfers a number x ($>$0) of tokens to account B, paying a transaction fee f ($>$0). This transaction is an ITX if account A and account B are located at the same shard. Otherwise, it is a CTX if accounts A and B are located at different shards. To ensure the *atomicity* of CTXs, researchers proposed the *relay-transaction* mechanism in Monoxide [1], which is a classic state-sharding protocol. A CTX is split into two CTXs (*e.g.,* `CTX` and `CTX'`, as shown in Fig. 1), which participate in individual consensus in the source and destination blockchain shards, respectively. The *queueing latency* of a CTX is the sum of its queueing duration in the transaction pools (TX pools) of both its source and destination shards. A CTX's queueing latency accounts for most of its overall makespan. Although the *relay-transaction* mechanism proposed by Monoxide [1] can secure the so-called *eventual atomicity* for CTXs [2], the queueing latency of CTXs is significantly higher than that of intra-shard transactions. Therefore, it is unfair for those CTXs from the perspective of a sharded blockchain. This unfairness in queueing latency can degrade the user experience for those who initiate a transaction, but this user does not know his/her transaction is going to be split into CTXs.

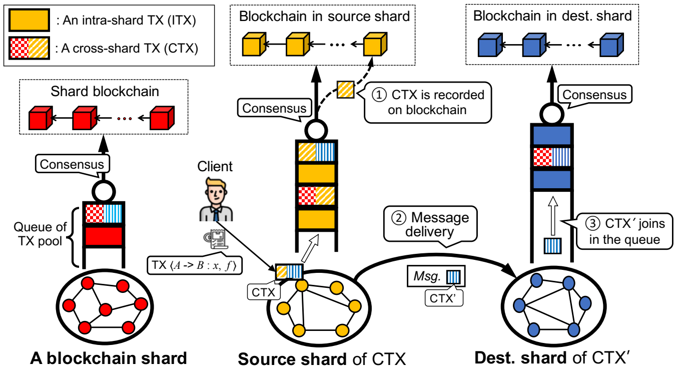
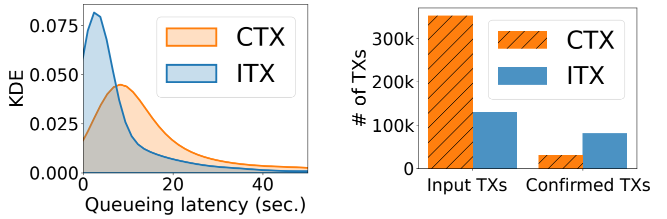


Fig. 1. The processing steps of a CTX in the sharded blockchain. Step ①: the first part of this CTX is recorded on the source shard's blockchain. Step ②: a *relay-transaction* message (the second part of this CTX, i.e., CTX', is enclosed) is delivered to the destination shard. Step ③: CTX' enters into the destination shard's TX pool.

Our hypothesis. Compared to ITXs, CTXs are suffering higher queueing latency in shards' TX pools because of the following two reasons: Firstly, from the client's perspective who submits a TX to the blockchain, the client only pays a regular transaction fee [3]. Clients don't know whether their TX will be processed as a CTX or an ITX. The transaction fee of a CTX is split into two portions, paid to the proposers located at the corresponding source and the destination shards, respectively. Secondly, in the *relay-transaction* mechanism, CTXs need to participate in the local consensus in both the source and the destination shard's TX pools. In contrast, ITXs only need to participate once. In summary, these two reasons induce that CTXs usually have much higher queueing latency in sharded blockchains. No clients desire unpredictable service



(a) Distribution of queueing latency (b) Input TXs *vs* Confirmed TXs

Fig. 2. Motivation experiments. (a): CTXs suffer from higher queueing latency than ITX, and (b) CTXs have a lower confirming ratio compared to ITXs. Here, KDE stands for Kernel Density Estimation.

time for their transactions. Thus, from the perspective of the sharded blockchain, both CTXs and ITXs must be treated *equally* and *fairly*.

To prove our hypothesis described above, we conducted a group of motivation experiments. The experimental observations illustrate the unfairness between CTXs and ITXs when they queue in the TX pools. The experiments are explained below.

Motivation experiments. The following experiments investigate the queueing latency of both CTXs and ITXs when adopting Monoxide's relay-transaction mechanism [1]. Our experiment replicates around 500k Ethereum TXs [4] across hundreds of consensus epochs. In each epoch, TXs are injected into four blockchain shards at a rate of 1000 TXs per second. The transaction fee of each CTX is shared on average by both its source and destination shards. Due to the fast transaction injection rate, proposers always have ITXs and CTXs available in the TX pools for selection when they generate blocks.

(a) Fig. 2(a) illustrates that CTXs generally experience higher queueing latency than ITXs.
(b) The left-hand side of Fig. 2(b) shows the input numbers of CTXs and ITXs queueing in TX pools (*i.e.,* 352k and 130k, respectively). The right-hand side shows the numbers of CTXs and ITXs that are confirmed in the blockchain at the end of the experiment (*i.e.,* 30k and 81k, respectively). CTXs have a lower confirming ratio than ITXs (*i.e.,* 30k/352k *vs* 81k/130k, respectively).

Motivated by those observations, we aim to achieve the following two goals in this paper: i) to reduce the unfairness in queueing latency between CTXs and ITXs. and ii) to guarantee the basic properties of a sharded blockchain [5]. To achieve our goals, we devise an incentive mechanism for the sharded blockchains using the theory of Shapley value [6], [7].

Our study includes the following **contributions**.

- **Originality.** We aim to address the challenge brought by the unbalanced transaction confirmation latency and queueing latency when blockchain shards package the CTXs and ITXs from transaction pools. To this end, we design an incentive mechanism to incentivize proposers to package CTXs from TX pools in a fair way. We also generalize our incentive to the *multi-input multi-output* case of CTXs.
- **Theory guaranteed.** We rigorously analyze its fundamental properties in the context of a sharded blockchain, *i.e.*, *security*, *atomicity*, and *fairness*.
- **Practicality.** Finally, we conduct experiments on an open-source testbed named BlockEmulator, using the historical Ethereum transactions. The evaluation results show that the proposed incentive mechanism can enable CTXs to have low queueing latency and assure security, atomicity, and fairness properties.

II. Related Work

Sharded Blockchains. In sharded blockchains, nodes are organized into shards for parallel operation, offering linear scalability [8], [9]. Elastico [10] implements the basic *network sharding*, with each shard processing distinct TXs while maintaining the full blockchain ledger. Omniledger [11] further presents *state sharding*, assigning a separate ledger to each shard. RapidChain [12] enhances Omniledger with high-throughput intra-shard consensus, lightweight shard reconfiguration via the Cuckoo rule, and fast cross-shard communication via the Kademlia protocol. Monoxide [1] features asynchronous sharding with Chu-ko-nu mining and eventual atomicity, improving system throughput and CTX processing efficiency. Furthermore, CoChain [13] solves the weak security of sharded blockchains facing corrupted shards with an inter-shard monitoring protocol.

Processing CTXs. Processing CTXs is crucial for *state sharding*. Existing sharded blockchains commonly use two strategies: 2-Phase Commit (2PC) protocols [11]–[15] and Asynchronous Methods (Asyn-M) [1]. Omniledger [11] pioneers 2PC for CTX processing, dividing each CTX into input and output operations. Clients collect acceptance proofs from input shards (Phase-1) and notify the output shard for payment (Phase-2) to prevent incomplete operations. RapidChain [12] adapts 2PC, assigning coordination to output shard leaders to address client reliability concerns. Dang *et al.* [14] combine Practical Byzantine Fault Tolerance (PBFT) and 2PC in a distributed protocol for CTX processing, ensuring protocol liveness through a reference committee. Monoxide [1] introduces Asyn-M, using anonymous, lock-free CTX processing where transaction proofs integrate into input shards' blocks and relay to output shards for eventual atomicity. Asyn-M circumvents lock/unlock costs but cannot fully meet real-time CTX processing requirements (e.g., security and fairness). Unlike state-of-the-art studies, we investigate how to guarantee fairness between CTXs and ITXs.

Note that some recent research aims to address the concern of CTX processing by reducing the CTX number. For example, OptChain [16] and SPRING [17] use graph theory and deep reinforcement learning, respectively, to optimize TX placement. Similarly, Pyramid [18] and BrokerChain [19] introduce bridge shards and broker accounts to convert CTXs into ITXs. Nonetheless, given that the proportion of CTX in practical sharded blockchains exceeds 90% [20], the frequent executions of these mechanisms complicate the sharding workflow and may disrupt decentralization. In contrast, our proposal encourages mining nodes to process CTXs via incentives, incurring no additional cost.

III. Preliminaries and System Model

A. Preliminaries

1) Blockchain Nodes and Clients: Different from the previous work [18], [19], we do not introduce any additional centralized entities. Our system only involves two types of stakeholders, *i.e.,* clients and block proposers. Clients initiate transaction requests through their digital wallets, while proposers perform essential tasks such as participating in intra-shard consensus and processing client-submitted transaction requests.

According to their behaviors, proposers can be divided into two categories, *i.e.,honest nodes* and *Byzantine nodes*. Honest nodes are profit-driven, which means they will exploit the blockchain's consensus protocol and incentive mechanisms to maximize their profits. Differently, Byzantine nodes might launch malicious attacks. Note that a secure blockchain system can tolerate at most a specific percentage of Byzantine proposers, *e.g.*, 1/3 is such the security threshold in a PBFT-based blockchain. When the proportion of honest nodes exceeds the blockchain's security threshold, honest proposers will generate blocks through the consensus mechanism and obtain profits according to the incentive mechanism.

2) Transaction Model: Our system adopts the *account-based* TX model rather than the Unspent Transaction Output (UTXO) model. The payer's and payee's accounts of a CTX are at different shards.

3) Shard Formation and Network Model: We apply the most straightforward shard formation, *i.e.,* randomly distributing all proposers into multiple equal-size shards. Clients can connect to an arbitrary number of proposers. For the network model, we make the following two requirements: i) the intra-shard communication is synchronous, and ii) the cross-shard communication is partial-synchronous. Both requirements are also adopted by other sharded blockchains [10], [12].

B. System Model

The working flow of our system is divided into successive *epochs*. Each epoch includes the following major phases.

1) Randomness Generation and Node distribution: In this phase, the blockchain system distributes newly-arrived nodes into several shards. We adopt an established approach to reach this goal. More details can be found in [12].

2) Intra-shard Consensus: In our system, intra-shard consensus is a pluggable module. No matter what consensus is chosen, this step mainly completes three tasks: i) nominating a block proposer; ii) letting the proposer select TXs from the local TX pool and create a pending block (Step ① in Fig. 1); and iii) verifying the proposed block.

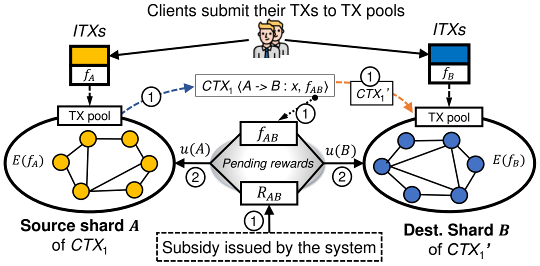


Fig. 3. `Source shard` A and `destination shard` B share the total reward R_{AB} + f_{AB} of a cross-shard TX $\texttt{CTX}_1$ $\langle A \rightarrow B : x, f_{AB}\rangle$, $(A, B \in [S])$. The transaction fee f_{AB} is offered by the client, while the subsidy R_{AB} is issued by the blockchain system. ①: $\texttt{CTX}_1$ is processed by both shards A and B. f_{AB} and R_{AB} accumulate as *Pending rewards*. ②: *Pending rewards* are divided into two portions (u_A and u_B) and allocated to the proposers in shards A and B, respectively.

3) CTX's Processing: Our system model considers only token-transfer TXs, which are not related to smart contracts. As shown in Fig. 1, a CTX originates from its source shard. We also represent each cross-shard TX as $\texttt{CTX}$ $\langle A \rightarrow B : x, f\rangle$, where $A \in [S]$ and $B \in [S]$ are the source and destination shards, respectively, and $[S]$ denotes the set of all blockchain shards.

When completing intra-shard consensus in the source shard, the proposer broadcasts a message (abbr. *Msg.*) to the entire network, *i.e.,* Step ② in Fig. 1. The message *Msg.* contains all the necessary information for the destination shard of `CTX`.

When the destination shard receives *Msg.*, the receiver nodes will verify all the contained CTXs. After verification, a new TX corresponding to the original `CTX` is then generated as shown in Step ③ of Fig. 1. We denote this new TX by `CTX'`. Next, `CTX'` joins the queue of the destination shard's TX pool and waits to participate in consensus.

IV. Design of Incentive Mechanism

In this section, we elaborate on the design of our incentive mechanism. To help readers better understand our design, we first introduce the basic symbols and definitions, then present a *strawman* and the proposed incentive mechanisms.

A. Basics of Incentive Mechanism

We consider a CTX, say $\texttt{CTX}_1$ $\langle A \rightarrow B : x, f_{AB}\rangle$ across shards A and B $(A, B \in [S])$, to facilitate the description of both the strawman and the proposed incentive mechanisms. Since proposers are profit-driven, a certain TX fee is required for both ITXs and CTXs [21]. As shown in Fig. 3, we let f_{AB}, $E(f_A)$, and $E(f_B)$ denote the TX fee for $\texttt{CTX}_1$, the expected average TX fee for ITXs in `source shard` A, and the expected average TX fee for ITXs in `destination shard` B, respectively. In practice, we can estimate the average ITX fees $E(f_A)$ and $E(f_B)$ by averaging all the ITXs contained in a few number of most recent blocks.

In fact, the underlying blockchain sharding is a black box to clients because clients cannot figure out whether their transaction is an ITX or a CTX. When clients submit a TX to the blockchain system, they pay a regular TX fee. This behavior will lead to the fact that the cross-shard fee f_{AB} approximates the intra-shard fee $E(f_A)$, *i.e.,* $E(f_{AB}) = E(f_A)$. This fact is an important condition to the proof of Theorem 4 in § V-C.

1) Introducing a subsidy parameter: Given a cross-shard TX $\texttt{CTX}_1$ $\langle A \rightarrow B : x, f_{AB} \rangle$, $(A, B \in [S])$, our system introduces a dedicated *subsidy* for the proposers in both shards A and B, to incentivize them to pick up CTXs from TX pools. We use R_{AB} ($R_{AB} \in \mathbb{R}^+$) to denote the subsidy issued by the blockchain system. For each CTX, say $\texttt{CTX}_1$, its transaction fee f_{AB} and the subsidy R_{AB} accumulate as a logical *Pending reward*. The accumulated reward f_{AB} + R_{AB} will be divided into two portions, *i.e.,* u_A and u_B, which are then reallocated to shards A and B, respectively.

The definition of R_{AB} is derived according to two observations. First, for $\texttt{CTX}_1$, it involves two CTXs that need to be processed in shards A and B. Second, in both A and B, $\texttt{CTX}_1$ needs to compete with other ITXs since profit-driven proposers only choose the TXs that can maximize their benefits. Therefore, R_{AB} is designed to consider both $E(f_A)$ and $E(f_B)$, *i.e.,*

$$R_{AB} = F(E(f_A), E(f_B)),\ \forall A, B \in [S], \tag{1}$$

where $F(\cdot)$ is a customized reward function. Eq. (1) indicates that R_{AB} is not affected by TX fee f_{AB}. Otherwise, clients would gain a great chance to disrupt our incentive mechanism by setting an arbitrary value of f_{AB}.

Note that introducing a subsidy to a blockchain system is **not straightforward** because the developers need to handle various issues such as the subsidy source, the inflation control, the configuration of the subsidy to contributors, etc.

2) The subsidy source: The subsidy is issued by the blockchain consensus to the proposer who wins the mining reward of each round of consensus, just as the same way as `coinbase` transaction works in Bitcoin. Clients do not need to pay such a subsidy for their transactions.

3) Inflation control: Similar to the `coinbase` award in Bitcoin, by setting an annual inflation rate [22], which is a real number within a specified range [$\Gamma_{\min}$, $\Gamma_{\max}$], the developer of a sharded blockchain system can control the inflation rate of issuing tokens. We analyze the economic inflation introduced by such subsidy-based incentive in § VI-C.

B. A Strawman Incentive Mechanism

Intuitively, the transaction fee f_{AB} should be equally allocated to shards A and B, because those two blockchain shards process individual CTXs, *i.e.,* $\texttt{CTX}_1$ and $\texttt{CTX'}_1$, respectively. Thus, as designed in Monoxide [1], a *strawman* incentive mechanism is designed to split f_{AB} equally for rewarding A and B, respectively. Therefore, the final reward received by A and B for processing a CTX is $f_{AB}/2$ for each. As a result, the equally-split reward $f_{AB}/2$ obviously makes $\texttt{CTX}_1$ less competitive than most of ITXs in shard A, thereby bringing $\texttt{CTX}_1$ a long queueing latency in the TX pool of shard A.

Moreover, the strawman strategy ignores the workload imbalance among shards. We consider a simple example as a case study. Suppose that $E(f_A)$ rises to 20 since A is extremely busy, thus clients have to offer higher TX fees to mitigate the possible high queueing latency of their TXs. In contrast, B is currently idle, and $E(f_B)$ drops to only 5. If f_{AB} is 30, both A and B can acquire 15 when the reward is split equally. Since such a reward is less than $E(f_A)$, the *equal allocation* cannot encourage the proposer in A to package $\texttt{CTX}_1$.

TABLE I
THE DISTRIBUTION OF MARGINAL CONTRIBUTION.

Prob.	Order	A's marginal contribution	B's marginal contribution
1/2	$A \rightarrow B$	$E(f_A)$	$R_{AB} + f_{AB} - E(f_A)$
1/2	$B \rightarrow A$	$R_{AB} + f_{AB} - E(f_B)$	$E(f_B)$

C. The Proposed Incentive Mechanism

Next, we elaborate on how to allocate the total reward R_{AB} + f_{AB} to the proposers in shards A and B.

Inspired by the 2-party Shapley value [6], [7], we present the idea of a new reward-allocation strategy. Firstly, the processing of $\texttt{CTX}_1$ can be viewed as a cooperative game G, whose potential participants are A and B. Then, we derive the marginal contribution as shown in Table I. If not participating in G, proposers in shard A can use the reserved position to package one ITX; the same rule applies for B. Thus, when working individually, shards A and B can make the contributions denoted by $E(f_A)$ and $E(f_B)$, respectively. As for the arrival sequence, since processing a CTX is symmetric, there are two orders: i) *first* A*, then* B; and ii) *first* B*, then* A. If following the former, the marginal contribution of A is $E(f_A)$. Consequently, B's marginal contribution equals the total cooperative reward from G subtracted by the contribution made by A when working individually, *i.e.,* $R_{AB} + f_{AB} - E(f_A)$. Similarly, the marginal contribution of A and B following the second order can also be calculated. In our design, the allocated rewards for the proposers in A and B, denoted by $u(A)$ and $u(B)$, are perfectly equal to their Shapley values [6], [7]:

$$u(A) = \frac{f_{AB} + R_{AB} + E(f_A) - E(f_B)}{2},\ A, B \in [S]; \tag{2}$$

$$u(B) = \frac{f_{AB} + R_{AB} + E(f_B) - E(f_A)}{2},\ A, B \in [S]. \tag{3}$$

As shown in Fig. 3, for $\texttt{CTX}_1$ $\langle A \rightarrow B : x, f_{AB} \rangle$, Eqs. (2) and (3) represent how `source shard` A and `destination` B share the total reward, *i.e.,* R_{AB} + f_{AB}. As for an intra-shard transaction, the proposer packaging an ITX obtains all its transaction fees. Differently, our reward-allocation strategy has the following three features. *Feature i*), $u(A)$ + $u(B)$ = f_{AB} + R_{AB}. *Feature ii*), the workload imbalance among shards is considered, because a higher workload in a shard indicates a higher subsidy. Recalling the previous example, if R_{AB} is 5 and the total reward f_{AB} + R_{AB} is allocated according to Eqs. (2) and (3), $u(A)$ and $u(B)$ will be 25 and 10, respectively. Since $E(f_A)$ is 20 and $E(f_B)$ is 5, both shards A and B, in this case, are willing to participate in CTX's packaging. *Feature iii*), the bidirectional processing of CTXs is well-preserved because Eqs. (2) and (3) also determine the reward allocation for those CTXs transferring tokens from B to A.

D. Proposers' Choices and Profits

Based on the proposed incentive mechanism, we then discuss what choices proposers have when they intend to maximize their profits. Since proposers' profits all come from the TX fees and the subsidies of CTXs, they need to figure out which ITXs and CTXs are worth being packaged. Intuitively, proposers could sort all queueing TXs according to their rewards and select the most valuable ones to fully package a block. However, in practice, the actual rewards for packaging CTXs are unpredictable. For example, the reward for $\mathtt{CTX}_1$ will be issued only when it is completely processed and confirmed on chain. If the withdrawal operation of $\mathtt{CTX}_1$ is executed by a node in shard A while the payment is blocked in B, A has the risk of wasting a position to package a TX and thus losing a certain TX fee. Based on Eqs. (2) and (3), we can acquire $u(B) = u(A) - E(f_A) + E(f_B)$. Next, we classify all the choices of whether a proposer should package $\mathtt{CTX}_1$ into three cases for discussion.

- **Case #1:** When $u(A) \geq E(f_A)$, our reward-allocation strategy guarantees that $u(B) \geq E(f_B)$. Hence, the proposer in A tends to package $\mathtt{CTX}_1$ since $\mathtt{CTX'}_1$ has a great chance to be picked by another proposer in B shortly.
- **Case #2:** When $u(A) \leq E(f_A) - E(f_B)$, we find $u(B) \leq 0$. In this case, the proposer in A tends not to package $\mathtt{CTX}_1$ since another proposer in B has no chance to package $\mathtt{CTX'}_1$.
- **Case #3:** When $E(f_A) - E(f_B) < u(A) < E(f_B)$, both the proposers in A and B have some probabilities to package $\mathtt{CTX}_1$ and $\mathtt{CTX'}_1$, respectively, according to the current congestion situations of their TX pools.

Remark 1. *Proposers sort all TXs and select the ITXs whose TX fee is no less than $E(f_A)$, and the CTXs meet **Case #1**. If there is still available space in the block, proposers will choose other ITXs and the CTXs meet **Case #3**.*

E. How to Set the Subsidy Parameter

We next discuss the range of R_{AB} and how to set R_{AB} in practice. Firstly, the lower bound of R_{AB} is 0, implying that our incentive mechanism does not give CTXs any subsidy. In this case, only a few numbers of CTXs satisfying $f_{AB} \geq E(f_A) + E(f_B)$ have a high probability of being picked up from the TX pool. With the increasing value of R_{AB}, CTXs become more and more competitive than ITXs. When R_{AB} reaches $E(f_A) + E(f_B)$, we can ensure the results $u(A) \geq E(f_A)$ and $u(B) \geq E(f_B)$, as long as $f_{AB} \geq 0$. In other words, even though a CTX satisfies $f_{AB} = 0$, it will not be ignored by both shards A and B. This is because current $u(A)$ and $u(B)$ are equal to $E(f_A)$ and $E(f_B)$, respectively. Therefore, the upper bound of R_{AB} is $E(f_A) + E(f_B)$.

To support different sharding architectures and applications, our incentive mechanism enables system designers to set a sophisticated value of R_{AB} by defining the reward function $F(\cdot)$. In this paper, we mainly consider three cases, *i.e.,* R_{AB} = 0, $R_{AB} = E(f_A) + E(f_B)$, and $R_{AB} = E(f_B)$. This is because the first two cases are the lower bound and the upper bound of R_{AB}, respectively. The third case $R_{AB} = E(f_B)$ can ensure the *proposer's fairness*, which is analyzed in the next section.

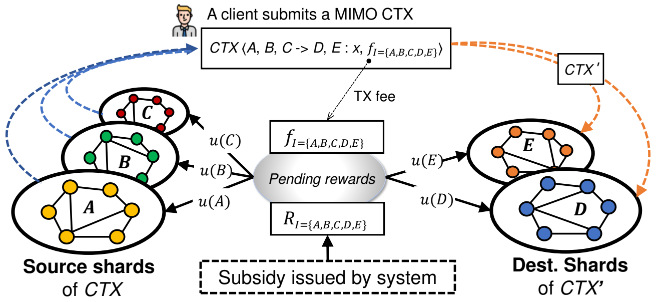


Fig. 4. An example of a *3-input & 2-output* CTX $\langle (A, B, C) \rightarrow (D, E) : x, f_{I=\{A,B,C,D,E\}} \rangle$, in which f_I is the fee of this CTX.

F. Generalizing to Multi-Input Multi-Output (MIMO) CTXs

In practice, MIMO CTXs exist in some blockchains that support complicated token transfers. As shown in Fig. 4, a MIMO CTX has multiple input accounts and/or multiple output accounts. Our incentive mechanism can be further extended to the MIMO case. We let m ($m \in \mathbb{N}_+$) and n ($n \in \mathbb{N}_+$) represent the number of input accounts and output accounts of a MIMO CTX, respectively. The set of shards where all the input and output accounts of a MIMO CTX locate is represented by I ($I \subseteq [S]$). We then define R_I ($I \subseteq [S]$) to denote the subsidy parameter for all the proposers who locate at shard i ($\forall i \in I$). Thus, the MIMO CTXs having the same set I share the same subsidy R_I. Next, f_I ($I \subseteq [S]$) is defined to represent the TX fee of the MIMO CTX whose involved accounts are in shard i ($\forall i \in I$). In addition, $E(f_i)$ ($i \in I$) is the expected average TX fee of all the ITXs in the pool of shard i ($i \in I$).

To show a generalized formulation, we consider the case where each input account and output account belong to different shards, just as the CTX shown in Fig. 4. Similarly, we still adopt the Shapley value [6], [7] to derive the following reward-allocation function $u(i)$ ($\forall i \in I$), which is shared by all the proposers located at shard i ($\forall i \in I$).

$$u(i) = E(f_i) + \frac{f_I + R_I - \sum_{i' \in I} E(f_{i'})}{m+n}, \ \forall i \in I. \tag{4}$$

Proof. Referring to the Shapley value [6], [7], the allocated reward $u(i)$ ($i \in I$) stems from the following two scenarios. Scenario i): When not all the other shards in I participate in this MIMO CTX before shard i, shard i brings a contribution $E(f_i)$. Scenario ii): When all the other shards in I participate in this MIMO CTX before shard i, shard i's participation can help this MIMO CTX be packaged successfully. Thus, shard

i brings a contribution $f_I + R_I - \sum_{i' \in I, i' \neq i} E(f_{i'})$. Then, we have

$$\begin{aligned} u(i) &= \frac{1}{(m+n)!} \sum \{\text{the contributions of } i\text{'s participation}\} \\ &= \frac{((m+n-1)! \cdot (f_I + R_I - \sum_{i' \in I, i' \neq i} E(f_{i'}))}{(m+n)!} + \\ &\quad \frac{((m+n)! - (m+n-1)!) \cdot E(f_i)}{(m+n)!} \\ &= \frac{(m+n-1)! \cdot (f_I + R_I)}{(m+n)!} - \\ &\quad \frac{(m+n-1)! \cdot \sum_{i' \in I} E(f_{i'})}{(m+n)!} + \frac{(m+n)! \cdot E(f_i)}{(m+n)!} \\ &= E(f_i) + \frac{f_I + R_I - \sum_{i' \in I} E(f_{i'})}{m+n}, \ \forall i \in I. \end{aligned}$$

Therefore, Eq. (4) holds. □

Discussion: For MIMO CTXs, our incentive mechanism has three merits similar to single-input single-output CTXs. Firstly, we have $\sum_{i \in I} u(i) = f_I + R_I$. Secondly, the workload imbalance among shards is considered, and a higher workload indicates a higher subsidy. Thus, the short queueing latency of CTXs can be guaranteed. Finally, the bidirectional processing of CTXs is well-preserved for MIMO CTXs.

V. Analysis of the Proposed Mechanism

This section first analyzes three types of threats of Justitia and then proves Justitia achieves the desired properties of atomicity and fairness.

A. Threat Analysis of Justitia

Before analyzing the threats of Justitia, we present the setting of sharded blockchains served by Justitia. In sharded blockchains, nodes have the following four straightforward types of facts: i) *Fact I*, the node is awarded by the sharded blockchain protocol while becoming a block proposer; ii) *Fact II*, the nodes within an epoch of consensus is randomly selected; iii) *Fact III*, honest nodes are profit-driven and always pursue the maximum reward; iv) *Fact IV*, The proportion of profit-driven honest nodes on each shard exceeds the security threshold of the sharded blockchain. In other words, even Byzantine nodes are unable to generate blocks that violate the incentive mechanism of the blockchain.

In Justitia, there exist the following three types of threats.

- **Threat #1 (Token Inflation):** The economic inflation of a subsidy-rewarded blockchain could be possible. This is because the subsidy R_{AB} is the extra tokens issued by the sharded blockchain. Thus, the total amount of tokens dynamically increases.
- **Threat #2 (Selfish Packaging):** Proposers can enforce the system to increase the subsidy R_{AB}. This is because proposers always prefer packaging the TXs with high fees. Thus, malicious proposers may create numerous high-fee TXs, thereby making a shard's average fee increase by force.
- **Threat #3 (CTX Flood):** Proposers can create and submit numerous CTXs to the TX pools of blockchain shards by colluding with other peers through inter-shard collusion. Therefore, those proposers can acquire a large portion of subsidies R_{AB} by packing these high-fee TXs.

Next, we analyze how Justitia tackles those threats. Corresponding to **Threat #1**, the blockchain developers can eliminate economic inflation by setting an annual inflation rate [22], which is within a specified range $[\Gamma_{\min}, \Gamma_{\max}]$.

Denoting the total expected number of blocks by N_0, the total amount of subsidies of each single block, denoted by $\sum_{\forall AB} R_{AB}$, should satisfy $N_0 \cdot \sum_{\forall AB} R_{AB} \in [\Gamma_{\min}, \Gamma_{\max}]$. In this way, blockchain developers get the boundaries of $\sum_{\forall AB} R_{AB}$. Once the total subsidies of every block exceed $\max(\sum_{\forall AB} R_{AB})$, the proposer in a round of consensus will not receive extra rewards, even if it packages extra CTXs. On the other hand, the blockchain system will issue $\min(\sum_{\forall AB} R_{AB})$ to the proposer, when the total amount of subsidies is lower than the minimum value. Through this manner to choose the right annual inflation rate, **Threat #1** can be avoided.

Corresponding to **Threat #2**, we have Theorem 1 as follows.

Theorem 1. *Given a shared blockchain wherein nodes exhibit four types of facts I-IV mentioned above,* ***Threat #2*** *does not hold.*

Proof. From $E(f_s)$, $(s \in [S])$, we know that the collusive proposers need to improve the average fee of the most recent few blocks, if they intend to improve the average fee of the current epoch of consensus in a shard. To achieve this goal, the proposers of the most recent few epochs of consensus must choose to package only a few number of high-fee TXs and ignore the low-fee TXs by all means. Then, we discuss two cases contradicted each other:

- If those proposers do not collude, *Fact I* indicates that those proposers will lose benefits because of ignoring those low-fee TXs even if they could have packaged them. Thus, this result contradicts *Fact III*.
- If those proposers collude with each other, *Fact II* indicates that those proposers cannot predict the new proposer in the next epoch of consensus. Thus, a successful collusion in a shard requires all shard nodes to participate in the collusion. This result contradicts *Fact IV*.

Thus, it is not possible for malicious proposers to improve the average fee by colluding with each other in a shard. **Theorem 1** holds. □

Finally, we analyze **Threat #3**. From the conclusion of **Theorem 1**, the average fee in a shard cannot be improved by force. The proposers will lose benefits if they only package the CTXs created by those collusive proposers themselves. This result contradicts *Fact III*. Thus, proposers in any shard will not choose to create CTXs by themselves. In summary, **Threat #3** does not exist when *Fact I-IV* hold.

B. Atomicity Analysis

Definition 1. *(Latency-guaranteed atomicity) We call a cross-shard TX has a time(t)-guaranteed atomicity if its relay transaction completes consensus in the destination shard and it is stored in a block within time t ($t \in \mathbb{R}^+$).*

Knowing the definition of *latency-guaranteed atomicity*, we then analyze the latency-guaranteed atomicity of CTXs under the proposed incentive mechanism. We consider two extreme cases: *hot shards* and *cold shards*, in which blockchain shards are congested by an overwhelming number of TXs or filled by almost none TXs, respectively. When a shard is hot, clients have to improve their TX fees in order to ensure their TXs can be packaged by proposers within a short queueing latency. Thus, $E(f_s)$ ($s \in [S]$) could be very large in hot shards. On the contrary, in a cold shard, blocks are not packaged with the full size. Thus, proposers are willing to package any transactions from the shard's TX pool. In a cold shard $s \in [S]$, we consider that $E(f_s) \to 0$. With those two extreme cases in mind, we have the following lemmas.

Lemma 1. *Given a CTX $\langle A \to B : x, f_{AB} \rangle$, ($A, B \in [S]$), if B is a cold shard and satisfies $u(B) > 0$, this CTX has a latency-guaranteed atomicity.*

Proof. A CTX satisfying $u(B) > 0$ in a cold shard will be packaged immediately. In other words, since this CTX enters into the TX pool of a shard, there exists a time t_0 ($t_0 \in \mathbb{R}^+$), in which this CTX completes the consensus and is recorded on the blockchain. Referring to Definition 1, we say this CTX has a t_0-guaranteed atomicity. □

Lemma 2. *Given a CTX $\langle A \to B : x, f_{AB} \rangle$, ($A, B \in [S]$), if B is a hot shard and satisfies $u(B) > E(f_B)$, this CTX has a latency-guaranteed atomicity.*

Proof. When a CTX satisfying $u(B) > E(f_B)$ in a hot shard, a proposer will choose to package this CTX by replacing a low-fee ITX from its block. That is, since this CTX enters the TX pool of a shard, there exists a time $t_1 (\in \mathbb{R}^+)$, in which this CTX can complete the consensus and is recorded on the blockchain. Referring to Definition 1, we say this CTX has a t_1-guaranteed atomicity. □

Theorem 2. *Given a CTX $\langle A \to B : x, f_{AB} \rangle$, ($A, B \in [S]$), if $u(B) > E(f_B)$, this CTX has a latency-guaranteed atomicity.*

Proof. On the one hand, when B is a cold shard, the condition $u(B) > E(f_B) > 0$ holds. On the other hand, when B is a hot shard, the condition $u(B) > E(f_B)$ holds. Referring to Lemma 1 and Lemma 2, this CTX has a latency-guaranteed atomicity. The conclusion of **Theorem** 2 holds. □

Next, we prove that latency-guaranteed atomicity also holds in three additional cases depending on the hot/cold conditions of both the source and destination shards of a CTX.

Corollary 1. *When the source shard of a CTX $\langle A \to B : x, f_{AB} \rangle$ ($A, B \in [S]$), i.e., shard A, is a hot one and the destination shard B is a cold one, proposers in A only package the high-fee CTXs, and proposers in B are willing to package any CTXs submitted to the TX pool immediately. This CTX has a latency-guaranteed atomicity.*

Proof. When the source shard A is a hot one, it only packages the TXs satisfying $u(A) > E(f_A)$. When offering the subsidy R_{AB} to the proposers in the hot A, proposers tend to package those high-fee CTXs if those CTXs satisfy $f_{AB} + R_{AB} > E(f_A)$, due to *Fact III*. On the other hand, the subsidy R_{AB} helps a cold shard B improve the reward of proposers. Thus, B has $u(B) > (E(f_B) = 0)$. Referring to Lemma 1, this CTX has a *latency-guaranteed atomicity*. □

Corollary 2. *When the source shard of a CTX $\langle A \to B : x, f_{AB} \rangle$ ($A, B \in [S]$) is a cold one and the destination shard B is a hot one, proposers in shard A only package the CTXs that meet the latency-guaranteed atomicity in shard B. This CTX has a latency-guaranteed atomicity.*

Proof. The cold source shard indicates that $u(A) > E(f_A)$, and the destination shard only packages the CTXs satisfying $f_{AB} + R_{AB} > E(f_B)$. Thus, we have $u(B) = f_{AB} + R_{AB} > E(f_B)$. Referring to Lemma 2, this CTX has a *latency-guaranteed atomicity*. □

Corollary 3. *When the source and destination shards of a CTX $\langle A \to B : x, f_{AB} \rangle$ ($A, B \in [S]$) are both cold ones, proposers in both A and B are willing to package this CTX, and the CTX has a latency-guaranteed atomicity.*

Proof. The cold source and destination shards indicate that $E(f_A) \to 0$ and $E(f_B) \to 0$. We then have $u(A) = (f_{AB} + R_{AB})/2 > 0$ and $u(B) = (f_{AB} + R_{AB})/2 > 0$. Thus, referring to Lemma 1, this CTX has a *latency-guaranteed atomicity*. □

TABLE II
ATOMICITY COMPARISON.

CTX$_1$ $\langle A \to B : x, f_{AB} \rangle$		Atomicity guarantee for CTX$_1$	
Shard A	*Shard B*	Monoxide [1]	Ours
Hot	Hot	✗	✓
Cold	Hot	✗	✓
Hot	Cold	✓	✓
Cold	Cold	✓	✓

As shown in Table II, our incentive mechanism maintains good atomicity of CTXs in different cases of shards. In contrast, monoxide's solution [1] can guarantee CTX's atomicity only when the destination shard is cold.

C. Analysis of Significant Fairness

To disclose more insights into the proposed incentive mechanism in the context of a sharded blockchain system, we also need to analyze whether it can guarantee two types of significant fairness. i.e., *System Fairness* and *Proposer's Fairness*. First, we give their definitions as follows.

Definition 2. *(System Fairness) For all transactions, whether an ITX or a CTX, transactions with higher TX fee get the lower expectation of queueing latency.*

Definition 3. *(Proposer's Fairness) For each block's proposer in both shards A and B, packaging ITXs and CTXs brings the same benefits, i.e., $E(u(A)) = E(f_A)$ and $E(u(B)) = E(f_B)$.*

Then, we analyze whether the proposed incentive mechanism can ensure these two types of fairness.

Theorem 3. *Given any R_{AB} ($R_{AB} \in \mathbb{R}^+$), our incentive mechanism can ensure the system fairness.*

Proof. According to the proposed incentive mechanism, each shard's proposer sorts all the TXs by their potential rewards, *i.e.*,, the higher the TX fee, this TX has the higher priority in the queue of the TX pool. In consequence, the TXs with higher fees will be packaged by proposers more easily, leading to lower queueing latency in the TX pool. Therefore, the TX fee and queueing latency are inversely correlated, thereby guaranteeing the system fairness. □

Remark 2. *Although the system fairness might look trivial, it tells us that the subsidy for the transaction with a higher TX fee should be no less than the subsidy for the transaction with a lower TX fee in order to satisfy such system fairness.*

Theorem 4. *When $R_{AB} = E(f_B)$ ($R_{AB} \in \mathbb{R}^+$), our incentive mechanism can ensure the proposer's fairness.*

Proof. We have $u(A) = \frac{f_{AB}+R_{AB}+E(f_A)-E(f_B)}{2}$ and $u(B) = \frac{f_{AB}+R_{AB}+E(f_B)-E(f_A)}{2}$. When $R_{AB} = E(f_B)$, $u(A) = \frac{f_{AB}+E(f_A)}{2}$, $u(B) = \frac{f_{AB}+2\cdot E(f_B)-E(f_A)}{2}$. Recall that $E(f_{AB}) = E(f_A)$ mentioned in § IV-A, we get $E(u(A)) = E(\frac{f_{AB}+E(f_A)}{2}) = \frac{E(f_{AB})+E(f_A)}{2} = E(f_A)$, $E(u(B)) = E(\frac{f_{AB}+2\cdot E(f_B)-E(f_A)}{2}) = \frac{E(f_{AB})+2\cdot E(f_B)-E(f_A)}{2} = E(f_B)$. Referring to Definition 3, when $R_{AB} = E(f_B)$ ($R_{AB} \in \mathbb{R}^+$), the conclusion holds. □

Additionally, we also have the following corollary.

Corollary 4. *When the proposer's fairness is satisfied, we have $E(f_{AB}) + R_{AB} = E(f_A) + E(f_B)$.*

Proof. $E(f_{AB} + R_{AB}) = E(f_{AB} + E(f_B)) = E(f_{AB}) + E(f_B)$. Recall that $E(f_{AB}) = E(f_A)$. Therefore, $E(f_{AB}) + R_{AB} = E(f_{AB}) + E(f_B) = E(f_A) + E(f_B)$. □

Remark 3. *Corollary 4 tells us when the equation $E(f_{AB}) + R_{AB} = E(f_A) + E(f_B)$ holds, the proposer's profit has no difference no matter it packages either a CTX or an ITX.*

VI. Implementation and Evaluation

A. Experiment Settings

Prototype Implementation. We deployed the proposed incentive mechanism on BlockEmulator [23] and evaluated multiple blockchain metrics. To support all experiments, we customize the PBFT as the intra-shard consensus protocol. All blockchain shards asynchronously process TXs in parallel. We also implement the relay-transaction mechanism proposed by Monoxide [1] to enable the CTXs.

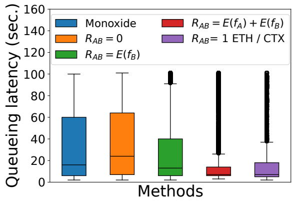


Fig. 5. Queueing latency of CTXs under various subsidy solutions.

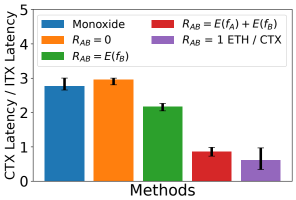


Fig. 6. Queueing latency declines as subsidy R_{AB} increases.

Datasets. We downloaded Ethereum [4] transaction dataset, which contains more than one million real-world TXs from the blocks whose heights range from 14920000 to 14937632. And we use only token-transfer TXs in the dataset.

Transaction processing. Each shard block yielded by *BlockEmulator* can include up to 400 TXs. Each shard node queues the arrived TXs in the local TX pool and sorts them according to the strategy elaborated in § IV-D.

Other parameters. The number of blockchain shards is set to 4 by default. Our blockchain prototype injects 1,000 TXs into all shards per second.

Baselines. We compare the proposed incentive mechanism mainly with *Monoxide* [1], which equally splits f_{AB} as the reward for shards A and B, without subsidy for CTX's processing. We also implement various versions of incentives by setting R_{AB} to different values as follows.

- $R_{AB} = 0$. No subsidy when packaging any CTX.
- $R_{AB} = E(f_B)$. Fix the subsidy of a CTX to the average fee of its destination shard.
- $R_{AB} = E(f_A) + E(f_B)$. Fix the subsidy of a CTX to the summed average fees of both its source and destination shards.
- R_{AB} = 1 ETH / CTX. Fix the subsidy of packaging a CTX to an extremely large value, *i.e.*, 1 ETH per CTX.

B. The Effect of Varying Subsidy Parameter

We conduct the first group of experiments to evaluate how queueing latency varies between the CTXs and ITXs while changing the subsidy R_{AB}. Taking the first 500 thousand TXs from the original TX dataset, we measure the queueing latency of all CTXs under various incentive settings. Fig. 5 shows the following observations. Under Monoxide's case and $R_{AB} = 0$, the blockchain system does not issue any subsidy to CTX's processing. Thus, the queueing latency in these two cases is much higher than that in the other three cases, as shown in the right-hand side of Fig. 5. We also see that the subsidy solution $R_{AB} = E(f_A) + E(f_B)$ leads to the lowest queueing latency of CTXs. This observation validates that this subsidy solution can help the sharded blockchain quickly package CTXs into new blocks. However, CTX's subsidy does not necessarily reach an extremely large value, such as 1 ETH/CTX, because this large subsidy does not further reduce CTX's queueing latency.

To highlight the effectiveness of reducing the CTX's queueing latency, we present Fig. 6 to illustrate the proportion of the average queueing latency of CTXs to that of ITXs. Such

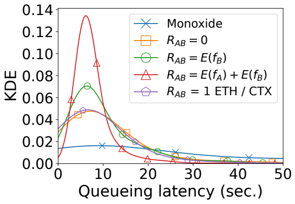


Fig. 7. The queueing latency distribution of confirmed CTXs under various subsidy parameters R_{AB}, measured by KDE.

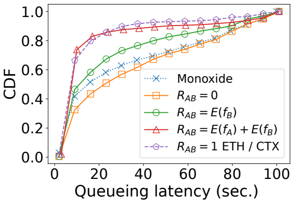


Fig. 8. Cumulative Distribution Function(CDF) of the queueing latency of confirmed CTXs, under various subsidy parameter R_{AB}.

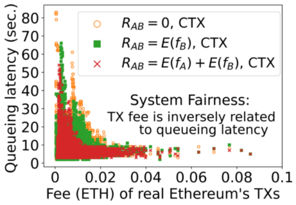


Fig. 11. The proposed incentive mechanism satisfies the *system's fairness* (*i.e.*, Definition 2) under different subsidy R_{AB}.

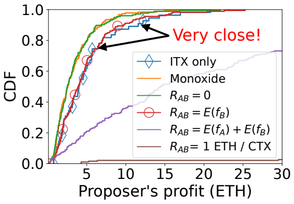


Fig. 12. With the specific subsidy $R_{AB} = E(f_B)$, the proposer's profit closely mirrors that of *ITX only*, ensuring *proposer fairness*.

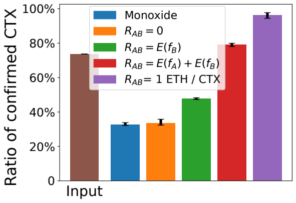


Fig. 9. The ratio of CTXs out of all TXs in packaged blocks.

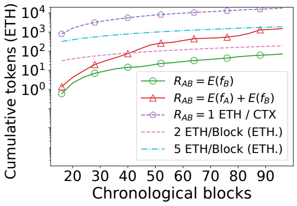


Fig. 10. The cumulative tokens issued.

proportion decreases with the CTX's increased subsidy R_{AB}. This result proves that the proposed incentive mechanism is capable of ensuring CTXs' higher priority than ITXs when proposers are picking up TXs from the TX pool.

Fig. 7 presents a Kernel Density Estimation (KDE) plot illustrating the distribution of transaction queueing latency in seconds.We observe that the proposed incentive mechanism effectively reduces CTXs' queueing latency.Especially, even without additional subsidy for cross-shard transactions (i.e., when $R_{AB} = 0$), the proposed incentive mechanism still keeps the queueing latency of CTXs concentrated at lower values. This is because the proposed incentive mechanism mitigates the impact of unbalance transaction workloads across the blockchain shards.

Fig. 8 shows the cumulative distribution function (CDF) of CTXs' queueing latency under various subsidy solutions. We observe that the zero-subsidy solutions, such as Monoxide and $R_{AB} = 0$, induce large queueing latency of CTXs. In contrast, a higher subsidy brings much lower queueing latency. Again, the extremely large subsidy solution, $R_{AB} = 1$ ETH/CTX shows a saturated effect on the reduction of queueing latency (*i.e.*, a close latency to $R_{AB} = E(f_A) + E(f_B)$).

In Fig. 9, we compare the ratios of CTXs between the original *input* of all CTXs and the CTXs *stored in packaged blocks*. Observing from the five bars on the right-hand side, we see that a higher subsidy leads to a higher CTX ratio.

C. Security of Economic Inflation

Corresponding to **Threat #1** as depicted in § V-A, we investigate the cumulative tokens issued by the sharded blockchain under the proposed incentive mechanism. To provide comparison baselines, we also study two token-offering policies under Ethereum's two real cases. The first case is from the early-stage Ethereum before the `Byzantine forking` [24]. The token's offering policy at that time was 5 ETH/block. The second case refers to the Ethereum since `Constantinople forking` at 2019 [25]. Since then, Ethereum's token-issuing speed has reduced to 2 ETH/block.

Fig. 10 shows the results under three subsidy settings and Ethereum's two cases. We have the following observations.

i) Although each subsidy solution of R_{AB} dynamically changes with the increased number of CTXs in new blocks and the average TX fees in shards, the growing trend of new tokens issued is stable. This trend is similar to that of Ethereum's cases.
ii) Comparing the two Ethereum cases, we see that the number of cumulative tokens issued is positively related to the block-proposing reward, *i.e.*, 2 ETH/block or 5 ETH/block. The same observations apply to the three subsidy-based solutions. For example, $R_{AB} = E(f_B)$ shows the lowest amount of tokens issued. The solution $R_{AB} = E(f_A) + E(f_B)$ issues fewer tokens than Ethereum's 2 ETH/block before the first 40 chronological blocks, and then approaches Ethereum's 5 ETH/block policy. The extreme subsidy $R_{AB} = 1$ ETH/CTX induces the largest amount of issued tokens.

In summary, under various settings of subsidy R_{AB}, the economic inflation of the total amount of tokens issued in the sharded blockchain is approximated to that of Ethereum.

D. Verifying the Fairness Properties

To study the *system fairness* (*i.e.,* Definition 2) of our proposed incentive mechanism, we replay 500 thousands of real Ethereum TXs in our BlockEmulator, following three subsidy solutions, *i.e.*, $R_{AB} = 0$, $R_{AB} = E(f_B)$, and $R_{AB} = E(f_A) + E(f_B)$, respectively. Fig. 11 illustrates the natural distribution of CTX's queueing latency corresponding to their real TX fees. We have the following two observations. i) When the subsidy value for CTXs decreases from the highest $R_{AB} = E(f_A) + E(f_B)$ to the lowest $R_{AB} = 0$, the queueing latency of CTXs grows. ii) The CTXs having higher subsidies (under $R_{AB} = E(f_B)$ and $R_{AB} = E(f_A) + E(f_B)$) show lower queueing latency, even when the TX fees are low. Those observations show that the proposed incentive mechanism guarantees the fairness of the blockchain system.

Finally, we verify the *proposer's fairness* (*i.e.,* Definition 3). To prove the *proposer's fairness*, we design a special experiment with *ITXs only*, in which ITXs constitute the sole input to the blockchain system. We observe from Fig. 12 that a higher subsidy indicates the proposer's higher profit. Specifically, the proposer's profits under the subsidy $R_{AB} = E(f_B)$ are very close to those observed in the *ITX only* scenario. This indicates that when our incentive mechanism satisfies $R_{AB} = E(f_B)$, it can ensure the *proposer's fairness*.

VII. Conclusions

To ensure the fairness of CTXs in a sharded blockchain, we proposed an incentive mechanism named `Justitia`. We also consider multi-input multi-output CTXs. The security, atomicity, and fairness properties of `Justitia` are rigorously analyzed. To evaluate `Justitia`, we implemented a prototype on top of the open-source BlockEmulator to conduct experiments. Experimental results demonstrate that `Justitia` effectively guarantees fairness in terms of queueing latency between CTXs and intra-shard transactions. Our findings also provide insights into how the subsidy parameter affects queueing latency and economic inflation from token issuance in the sharded blockchain.

References

[1] J. Wang and H. Wang, "Monoxide: Scale out blockchains with asynchronous consensus zones," in *Proc. of 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI'19)*, 2019, pp. 95–112.

[2] W. Vogels, "Eventually consistent," *Communications of the ACM*, vol. 52, no. 1, pp. 40–44, 2009.

[3] Y. Liu, Y. Lu, K. Nayak, F. Zhang, L. Zhang, and Y. Zhao, "Empirical analysis of EIP-1559: transaction fees, waiting times, and consensus security," in *Proc. of ACM SIGSAC Conference on Computer and Communications Security (CCS'22)*, H. Yin, A. Stavrou, C. Cremers, and E. Shi, Eds. ACM, 2022, pp. 2099–2113.

[4] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[5] M. Saad, A. Anwar, S. Ravi, and D. Mohaisen, "Revisiting nakamoto consensus in asynchronous networks: A comprehensive analysis of bitcoin safety and chainquality," in *Proc. of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS'21)*, 2021, pp. 988–1005.

[6] L. S. Shapley, "Notes on the n-Person Game—II: The Value of an n-Person Game," *US. Air Force Project*, 1951.

[7] I. Covert and S.-I. Lee, "Improving kernelshap: Practical shapley value estimation using linear regression," in *Proc. of International Conference on Artificial Intelligence and Statistics*, 2021, pp. 3457–3465.

[8] B. David, B. Magri, C. Matt, J. B. Nielsen, and D. Tschudi, "Gearbox: Optimal-size shard committees by leveraging the safety-liveness dichotomy," in *Proc. of ACM SIGSAC Conference on Computer and Communications Security (CCS'22)*, 2022, pp. 683–696.

[9] X. Qi, "S-store: A scalable data store towards permissioned blockchain sharding," in *Proc. of IEEE Conference on Computer Communications (INFOCOM'22)*. IEEE, 2022, pp. 1978–1987.

[10] L. Luu, V. Narayanan, C. Zheng, K. Baweja *et al.*, "A secure sharding protocol for open blockchains," in *Proc. of ACM SIGSAC Conference on Computer and Communications Security (CCS'16)*, 2016, pp. 17–30.

[11] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "Omniledger: A secure, scale-out, decentralized ledger via sharding," in *Proc. of IEEE Symposium on Security and Privacy (SP'18)*, 2018, pp. 583–598.

[12] M. Zamani, M. Movahedi, and M. Raykova, "Rapidchain: Scaling blockchain via full sharding," in *Proc. of ACM SIGSAC Conf. on Computer and Communications Security (CCS'18)*, 2018, pp. 931–948.

[13] M. Li, Y. Lin, J. Zhang, and W. Wang, "Cochain: High concurrency blockchain sharding via consensus on consensus," in *Proc. of IEEE Conference on Computer Communications (INFOCOM'23)*, 2023, pp. 1–10.

[14] H. Dang, T. T. A. Dinh, D. Loghin, E.-C. Chang, Q. Lin, and B. C. Ooi, "Towards scaling blockchain systems via sharding," in *Proc. of the International Conference on Management of Data (SIGMOD'19)*, 2019, pp. 123–140.

[15] Z. Hong, S. Guo, E. Zhou1, J. Zhang, W. Chen, J. Liang, J. Zhang, and A. Zomaya, "Prophet: Conflict-free sharding blockchain via byzantine-tolerant deterministic ordering," in *Proc. of IEEE Conference on Computer Communications (INFOCOM'23)*, 2023, pp. 1–10.

[16] L. N. Nguyen, T. D. Nguyen, T. N. Dinh, and M. T. Thai, "Optchain: optimal transactions placement for scalable blockchain sharding," in *Proc. of IEEE 39th International Conference on Distributed Computing Systems (ICDCS'19)*, 2019, pp. 525–535.

[17] P. Li, M. Song, M. Xing, Z. Xiao, Q. Ding, S. Guan, and J. Long, "Spring: Improving the throughput of sharding blockchain via deep reinforcement learning based state placement," in *Proceedings of the ACM on Web Conference 2024*, 2024, p. 2836–2846.

[18] Z. Hong, S. Guo, P. Li, and W. Chen, "Pyramid: A layered sharding blockchain system," in *Proc. of IEEE Conference on Computer Communications (INFOCOM'21)*, 2021, pp. 1–10.

[19] H. Huang, X. Peng, J. Zhan, S. Zhang, Y. Lin, Z. Zheng, and S. Guo, "Brokerchain: A cross-shard blockchain protocol for account/balance-based state sharding," in *Proc. of IEEE Conference on Computer Communications (INFOCOM'22)*, 2022, pp. 1–10.

[20] X. Qi and Y. Li, "Lightcross: Sharding with lightweight cross-shard execution for smart contracts," in *Proc. of IEEE Conference on Computer Communications (INFOCOM'24)*, 2024, pp. 1–10.

[21] Y. Huang, J. Tang, Q. Cong, A. Lim, and J. Xu, "Do the rich get richer? fairness analysis for blockchain incentives," in *Proc. of the 2021 International Conference on Management of Data (SIGMOD'21)*, 2021, pp. 790–803.

[22] Y. Cai, F. Long, A. Park, and A. Veneris, "Engineering economics in the conflux network," in *Proc. of 2nd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS'20)*, 2020, pp. 160–167.

[23] H. Huang, G. Ye, Q. Chen, Z. Yin, X. Luo, J. Lin, T. Li, Q. Yang, and Z. Zheng, "Blockemulator: An emulator enabling to test blockchain sharding protocols," *arXiv preprint arXiv:2311.03612*, 2023.

[24] Ethereum, *The details of Ethereum Byzantine Update.*, 2017, https://eips.ethereum.org/EIPS/eip-649.

[25] ——, *The details of Ethereum Constantinople Update.*, 2018, https://eips.ethereum.org/EIPS/eip-1234.