Broker2Earn: Towards Maximizing Broker Revenue and System Liquidity for Sharded Blockchains

Qinde Chen, Huawei Huang*, Zhaokang Yin, Guang Ye, Qinglin Yang
School of Software Engineering, Sun Yat-Sen University, China
*Corresponding author: Huawei Huang (huanghw28@mail.sysu.edu.cn)

IEEE INFOCOM 2024 - IEEE Conference on Computer Communications | 979-8-3503-8350-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/INFOCOM52122.2024.10621431

Abstract*—Cross-shard Transactions (CTXs) widely exist in sharded blockchains. CTXs have to endure large confirmation latency because they need to participate in consensus in both their source and destination shards. To diminish CTXs, plenty of state-of-the-art blockchain protocols have been proposed. For example, in BrokerChain [1], some intermediary broker accounts can help turn CTXs into intra-shard transactions through their voluntary liquidity services. Thereby, the original CTXs can be confirmed in blockchain shards quickly. However, we found that BrokerChain is impractical for a sharded blockchain because it does not consider how to recruit a sufficient number of broker accounts. Thus, blockchain clients do not have the motivation to provide token liquidity for others. To address this challenge, we design *Broker2Earn*, which is essentially a decentralized finance (DeFi) protocol that works as an incentive mechanism for blockchain users who choose to become brokers. Via participating in Broker2Earn, brokers can earn native revenues when they collateralize their tokens to the protocol. Furthermore, Broker2Earn can also benefit the sharded blockchain since it can efficiently spend each staked liquidity provided by brokers on diminishing CTXs. We formulate the core module of Broker2Earn into a revenue-maximization problem, which is proven NP-hard. To solve this problem, we design an online approximation algorithm using the relax-and-rounding technique. We also rigorously analyze the approximation ratio of our online algorithm. Finally, we conduct extensive experiments using real-world Ethereum transactions on both a transaction-driven simulator and an open-source blockchain testbed. The evaluation results show that the proposed Broker2Earn protocol demonstrates a near-optimal performance that outperforms other baselines, in terms of broker revenues and the usage of system liquidity.

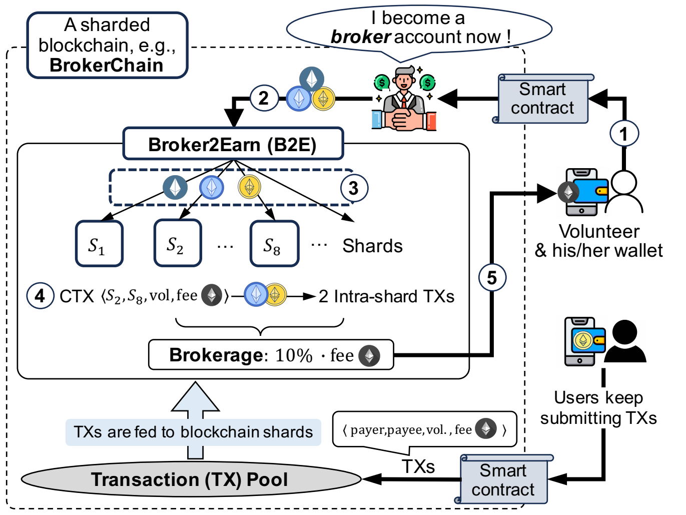

A sharded blockchain, e.g., BrokerChain
I become a *broker* account now !
2
Smart contract
1
Broker2Earn (B2E)
3
S_1
S_2
S_8
Shards
Volunteer & his/her wallet
4 CTX ⟨S_2,S_8,vol,fee⟩
2 Intra-shard TXs
5
Brokerage: 10% · fee
Users keep submitting TXs
TXs are fed to blockchain shards
⟨ payer,payee,vol., fee ⟩
TXs
Transaction (TX) Pool
Smart contract

Fig. 1. Overview of how Broker2Earn (B2E) protocol collaborates with a sharded blockchain like BrokerChain [1]. The digital wallet is a software application running on users' end devices. The five steps of B2E are explained as follows. ① : A volunteer chooses to become a broker account. ② : Broker collateralizes tokens to B2E protocol. ③ : B2E allocates tokens to blockchain shards. ④ : Broker's tokens turn a CTX into 2 Intra-shard TXs. Here $\langle S_2, S_8, \texttt{vol}, \texttt{fee} \rangle$ indicates that a transaction is issued from a payer account located in shard S_2 to another payee account located in shard S_8 with a volume $\texttt{vol}$ while paying $\texttt{fee}$ as TX fee. ⑤ : Broker receives a brokerage $10\% \cdot \texttt{fee}$ for his/her *liquidity service*. Note that, the commission rate 10% is an example of the tunable brokerage returned to brokers.

I. Introduction

Blockchain sharding is one of the mainstream technical routines that are promising to scale out the throughput of blockchains. Several state-of-the-art blockchain sharding protocols have been proposed [1]–[5]. In those solutions, BrokerChain [1] is a well-designed blockchain sharding protocol that secures a low number of *cross-shard transactions* (CTXs) and achieves the balance of transaction workloads across blockchain shards. In BrokerChain [1], the authors have proposed a transaction handling framework by exploiting the so-called *broker account*, which can provide voluntary intermediary transactions to turn CTXs into intra-shard transactions (as shown in step ④ of Fig. 1). In this way, the original CTXs will be quickly confirmed by intra-shard consensus. However, the problem is that BrokerChain did not provide an incentive mechanism for recruiting broker accounts. Thus, it is doubted whether BrokerChain can be established as a practical real-world blockchain system.

In a broker-based blockchain sharding system (e.g., Fig. 1), broker accounts are difficult to recruit from normal blockchain users. This is because users do not have the motivation to serve as brokers for other users. The only way is to introduce an incentive mechanism that can stimulate users to participate in the ecosystem of the broker-based blockchain system.

In the era of Web3, products that play as incentive mechanisms for blockchain users include decentralized finance (DeFi) protocols such as Polygon protocol [6], and digital wallet applications such as Argent wallet [7]. DeFi is a series of financial products and services drawing people's attention globally. It is built upon blockchain technologies. A digital wallet App is software that runs on users' end devices and enables users to trade crypto tokens and manage their private identity information. In those DeFi protocols or wallet Apps, users collateralize their crypto tokens to a third party like Polygon [6], which guarantees a certain rate of returned revenues for their clients. However, those DeFi protocols and digital wallets are essentially centralized commercial projects that run on top of a public blockchain such as Ethereum [8]. Thus, they cannot be applied directly in a sharded blockchain as an incentive mechanism. In particular, some new blockchains such as BrokerChain [1], do not have the foundation of large-scale users. Those new blockchains have to endure a long period of *cold start* when they compete with popular blockchains like Ethereum. Therefore, how to attract a large number of users at the very initial bootstrapping stage becomes a challenge for a new blockchain system.

To make a sharded blockchain (e.g., BrokerChain [1]) more competitive, our idea is to design a dedicated DeFi protocol that is distinguished from conventional DeFi products. To this end, we propose Broker2Earn (abbr. as B2E) protocol, which serves as an incentive mechanism for the sharded blockchains. The principle of B2E is shown in Fig. 1. B2E can encourage normal blockchain users to become broker accounts in the protocol layer of a sharded blockchain. Working as a core module in a sharded blockchain, the B2E protocol recruits volunteers by invoking a sophisticated revenue-maximization economic model. To attract as many blockchain users to participate in our system, the B2E protocol guarantees a tunable brokerage rate (say 10% in Fig. 1) as the service commission.

We take Fig. 1 as an example to illustrate how the proposed B2E protocol works in a sharded blockchain like BrokerChain [1]. Each blockchain user accesses B2E through their digital wallet. In step ①, a volunteer signs in the B2E protocol via a dedicated smart contract. The volunteer then becomes a broker account. After that, in step ②, this broker can collateralize his/her crypto tokens to the B2E protocol. As shown in step ③, those collateralized tokens are then utilized to diminish CTXs in the consensus layer of the blockchain system. As shown in step ④, an original CTX, say $\langle S_2, S_8, \texttt{vol}, \texttt{fee}\rangle$, is turned into 2 *intra-shard transactions* (abbr. as ITXs) using the collateralized tokens from the broker. Here, we call the operation ($\langle S_2, S_8, \texttt{vol}, \texttt{fee}\rangle \rightarrow$ 2 ITXs) shown in step ④ the *liquidity service*, the principle of which is referred to BrokerChain [1]. Then, these two ITXs are then executed easily in shards S_2 and S_8, respectively. Finally, in step ⑤, this broker receives a brokerage $10\% \cdot \texttt{fee}$ for his/her liquidity service. Through the overview shown in Fig. 1, we see that volunteer accounts are able to earn revenues if they choose to become broker accounts and collateralize tokens to the proposed B2E protocol. More importantly, the sharded blockchain system gains crucial liquidity, which determines whether a public blockchain can survive or not during its initial launching stage.

Our study includes the following contributions.

- **Originality.** We study the incentive mechanism for sharded blockchains. The proposed Broker2Earn protocol plays as a DeFi module for the sharded blockchain because it can provide crucial liquidity for the ecosystem.
- **Methodology.** We formulate the core module of the Broker2Earn protocol as a revenue-maximization problem, which is proven NP-hard. To solve this problem, we design an online approximation algorithm using the *randomized rounding* technique, such that it can offer near-optimal solutions for the broker-based blockchain system. We also rigorously analyze the approximation ratio of the proposed algorithm.
- **Usefulness.** Finally, we implement our Broker2Earn protocol in an open-source testbed BlockEmulator and evaluate its performance extensively using real-world historical Ethereum transactions. The experimental results demonstrate the efficacy of B2E protocol. For example, the core module of B2E (i.e., the online B2E algorithm) shows near-optimal performance that outperforms other baselines in terms of broker revenues and the usage of system liquidity.

II. Background and Related work

A. Classic Sharding Protocols

Along with Directed Acyclic Graph (DAG) [9], sidechains [10], payment channel networks [11], [12], cross-chain technologies [13], and rollup solutions [14], [15], blockchain sharding is one of the most promising solutions that can improve the scalability of a blockchain system. Recently, several state-of-the-art blockchain sharding protocols [1]–[5], [16], [17] have been proposed. The commonly-known first sharding protocol Elastico [2] was proposed in 2016. In Elastico, disjoint sets of transactions are distributed into different shards and processed by intra-shard consensus. Then, in Monoxide [5], the blockchain system divides accounts' states into different state shards, in each of which consensus nodes only maintain the corresponding states of their local accounts. The account distribution rule adopted in Monoxide [5] is called *static sharding*. This type of sharding leads to low TPS (transactions per second) when the workloads across shards are imbalanced. To achieve workload balance, Li *et al.* [16] proposed CLPA algorithm, which updates the transaction distribution dynamically.

In another work, Huang *et al.* [1] introduce *broker* accounts in their BrokerChain protocol, aiming to process CTXs. Instead of using the transaction relaying mechanism [5], BrokerChain offers an opportunity for volunteer accounts who play the role of *broker*. From the perspective of a sharded blockchain, developers can create intermediary transactions in the protocol layer using the staked tokens from broker accounts to *bridge* the payer and payee accounts of a CTX. In this way, a CTX can be turned into two intra-shard transactions located at different shards. As a result, the number of CTXs can be reduced and the global consensus of the sharded blockchain can be accelerated because the system eliminates the atomicity issue [5] of CTXs.

BrokerChain [1], however, does not consider how to recruit broker accounts. When a sharded blockchain does not have sufficient broker accounts, BrokerChain cannot be implemented in a real-world sharded blockchain. That is why we conduct this study in this paper. Inspired by the broker mechanism, we aim to design an incentive mechanism for such a sharded blockchain in which some intermediary accounts can potentially contribute their idle tokens to offer liquidity for blockchain shards. In summary, our proposed *Broker2Earn* solution is essentially a DeFi protocol, in which any blockchain users can choose to collateralize their tokens to the proposed *Broker2Earn* protocol and earn revenue.

B. Wallet Apps and DeFi Protocols

A number of digital wallet Apps, e.g., Metamask [18] and Argent [7], have been designed for blockchain cryptocurrency users to enable them to trade crypto tokens and manage private keys. Thus, wallet Apps play an essential role in the DeFi ecosystem. Upon joining Ethereum's Layer2 network, most existing wallet Apps not only provide basic wallet functions but also allow their users to earn extra revenue with their staked crypto tokens. For instance, users of Argent wallet [7] only need to pay low fees when issuing a transaction, because Ethereum's Layer2 DeFi protocols are supported in Argent wallet. The Layer2 feature can reduce transaction fees and increase Ethereum's system liquidity.

Our Broker2Earn also needs to rely on a wallet App to enable crypto token holders to access the sharded blockchain. For example in Fig.1, a volunteer collateralizes his/her tokens to B2E protocol via a dedicated smart contract. The brokerage also returns to the broker's wallet when the staked tokens provide liquidity for the sharded blockchain.

DeFi is built in decentralized systems without centralized institutions, for which plenty of DeFi protocols [6], [19]–[21] have been proposed. In most instances, DeFi protocols will help users earn revenues. For example, the *Yield Farming* mechanism adopted in Compound [20], gains financial liquidity such that Yield Farming can build a token pool between different blockchain systems by borrowing tokens from users. These users will earn COMP (native tokens issued in Compound) through borrowing operations.

Different from conventional DeFi protocols such as Yield Farming, the staked tokens from all brokers in our B2E protocol are only exploited to offer system liquidity within the sharded blockchain. On the other hand, the revenues earned by brokers originate from transaction fees. Therefore, B2E can provide more stable and native brokerage income for brokers.

III. System Model And Problem Statement

A. System Model

In a sharded blockchain, we call a round of global consensus across all shards an *epoch*. In each epoch, our system model recruits a set of broker accounts and has to handle a set of transactions submitted by blockchain users. The set of the broker accounts is written as $\mathcal{M} = \{1, 2, \dots, m, \dots\}$, the set of all blockchain shards is denoted by $\mathcal{J} = \{1, 2, \dots, j, \dots\}$, and the set of all *cross-shard transactions* (abbr. as CTXs) that are waiting to participate in the consensus algorithm is represented by $\mathcal{N} = \{1, 2, \cdots, n, \cdots\}$.

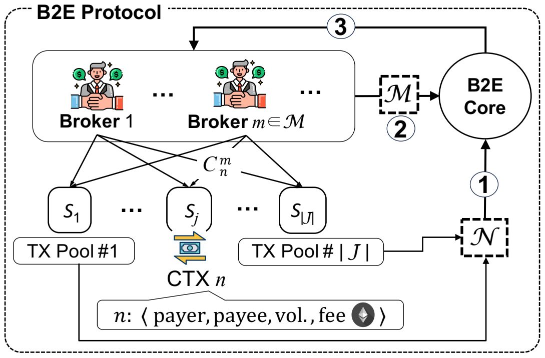


Fig. 2. System model of Broker2Earn (B2E) protocol in an epoch of consensus. ① : CTX set $\mathcal{N}$ is fed to **B2E-Core**. ② : $\mathcal{M}$ is fed to **B2E-Core**. Brokers then collateralize their tokens to B2E protocol. ③ : B2E determines the *liquidity service* for blockchain shards.

A transaction $\langle \texttt{payer}, \texttt{payee}, \texttt{vol}, \texttt{fee} \rangle$ implies that a volume $\texttt{vol} > 0$ of tokens is designated to transfer from the `payer` account's address to the `payee` account's address, while the user is paying an amount `fee` for transaction fees. When a blockchain user submits a transaction to a sharded blockchain, this transaction will be first put into the local TX pool of a specified shard according to its address prefix. Then, this transaction will be interpreted as either a CTX or an *intra-shard transaction* (abbr. as ITX), depending on the locations of its `payer` and `payee` accounts. ITXs participate in intra-shard consensus and will be confirmed quickly. In contrast, CTXs must be handled individually in both shards where the `payer` and `payee` accounts are located at by exploiting the *relay transaction* mechanism [1], [5].

At the beginning of an epoch of consensus, a bunch of transactions is submitted to the sharded blockchain, the system will assign each CTX into the local TX pool of a designated shard. To facilitate your easy understanding, we denote a CTX as $\langle j, j', K_{jj'}, f_{jj'} \rangle$, where $j, j' \in \mathcal{J}$, $K_{jj'} > 0$ is the `vol` of tokens transferred from the `payer` account in shard j to the `payee` account in shard j', and $f_{jj'} > 0$ is the transaction fee paid by the blockchain user who issues this transaction.

As shown in Fig. 2, to indicate whether a cross-shard transaction residing in a local TX pool of a blockchain shard is served by a broker account's intermediary service (also called *liquidity service*), we define a binary variable C_n^m $(n \in \mathcal{N}, m \in \mathcal{M})$:

$$C_n^m = \begin{cases} 1, & \text{CTX } n \text{ is served by broker } m; \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Referring to BrokerChain [1], if broker account $m \in \mathcal{M}$ offers *liquidity service* for $\texttt{CTX}\ n : \langle j, j', K_{jj'}, f_{jj'} \rangle$, $\texttt{CTX}\ n$ can be converted into 2 ITXs, say $\texttt{ITX}_1\ \langle j, \texttt{Account}_m, K_{jj'}, f_1 \rangle$ and $\texttt{ITX}_2\ \langle \texttt{Account}_m, j', K_{jj'}, f_2 \rangle$, where $f_1 + f_2 = f_{jj'}$. Thus, for $\texttt{ITX}_1$ and $\texttt{ITX}_2$, broker account m only needs to pay an amount $K_{jj'}$ of tokens to the payee account of $\texttt{ITX}_2$ located at shard j' for the liquidity service. Thus, given a variable C_n^m, all the related information of this transaction n such as $K_{jj'}$, $f_{jj'}$, and both the host shards of payer and payee accounts (i.e., shards j and j') can be obtained immediately.

As shown in steps ① and ② of Fig. 2, the CTX set $\mathcal{N}$ and broker set $\mathcal{M}$ are fed to B2E protocol. In step ③ of Fig. 2, B2E determines the *liquidity service* for as many CTXs as possible using the collateralized tokens of broker accounts.

Let $\text{Num}(m)$ denote the number of the CTXs served by broker account $m \in \mathcal{M}$, we have:

$$\text{Num}(m) = \sum_{n \in \mathcal{N}} C_n^m,\ \forall m \in \mathcal{M}. \tag{2}$$

Since each broker $m \in \mathcal{M}$ has a limited budget of his/her collateralized tokens denoted by D_m, we then have the following constraint describing the budget of each broker account.

$$\sum_{n \in \mathcal{N}} C_n^m \cdot K_n \leq D_m,\ \forall m \in \mathcal{M}, \tag{3}$$

where K_n is the volume of tokens transferred in CTX n.

Denote by f_n the transaction fee of CTX n, the expected revenue earned by broker account $m \in \mathcal{M}$ is written as:

$$\text{Revenue}(m) = \sum_{n \in \mathcal{N}} C_n^m \cdot f_n \cdot \sigma(D_m),\ \forall m \in \mathcal{M}, \tag{4}$$

where $\sigma(D_m) \in (0, 1]$ is a customized DeFi function that describes the tunable brokerage rate to measure this broker's revenue. For example in Fig. 1, $\sigma(D_m)$ = 10% for a broker account m. In practice, $\sigma(D_m)$ can be designed as several sliding gears by protocol developers according to the total volume of collateralized tokens from the broker account $m \in \mathcal{M}$.

B. Problem Formulation

The goal of **Broker2Earn** protocol is to provide an incentive mechanism for the candidate accounts who could become broker accounts in a sharded blockchain like BrokerChain [1]. On the one hand, the proposed **Broker2Earn** protocol should ensure that broker accounts receive as much revenue as possible when they provide *liquidity services* for handling CTXs. On the other hand, **Broker2Earn** also needs to ensure broker accounts could offer the maximum service capabilities for processing as many CTXs as possible for the sharded blockchain system.

We now formulate the incentive mechanism of **Broker2Earn** as the following linear integer-programming maximization problem. To any broker account $m \in \mathcal{M}$, the objective function denoted by $U_m = \alpha \cdot \text{Revenue}(m) + \text{Num}(m)$ includes two terms, i.e., the first term $\alpha \cdot \text{Revenue}(m)$ represents the brokerage revenue received by broker account m, and the second term $\text{Num}(m)$ is the total number of CTXs that the broker m can serve for *liquidity services*. Here, α (>0) is a coefficient measuring the weights between these two terms. Thus, we have

$$\begin{aligned} U_m &= \alpha \cdot \text{ Revenue}(m) + \text{ Num}(m) \\ &= \sum_{n \in \mathcal{N}} C_n^m [\alpha \cdot \sigma(D_m) f_n + 1],\ \forall m \in \mathcal{M}. \end{aligned} \tag{5}$$

Now we have the revenue-maximization problem (denoted by **B2E-Core**) from the perspective of the project founder of **Broker2Earn** protocol.

$$\textbf{B2E-Core:} \quad \max \sum_{m \in \mathcal{M}} U_m \tag{6}$$

$$\text{s.t.} \quad \sum_{n \in \mathcal{N}} C_n^m \cdot K_n \leq D_m, \forall m \in \mathcal{M}. \tag{7}$$

$$\sum_{m \in \mathcal{M}} C_n^m \leq 1, \forall n \in \mathcal{N}. \tag{8}$$

$$\text{Variables: } C_n^m \in \{0, 1\},\ \forall n \in \mathcal{N}, \forall m \in \mathcal{M}. \tag{9}$$

C. NP-hardness Proof

Theorem 1. *The **B2E-Core** problem is NP-hard.*

Proof. To prove **B2E-Core** problem is an NP-hard problem, we reduce a classic Multiple Knapsack Problem (MKP) to **B2E-Core** (Eq. (6) - Eq. (9)).

We now construct a new MKP problem. Given a set of items $\mathcal{N}$, each item $n \in \mathcal{N}$ has an associated weight w_n (>0) and a value g_n (>0). Suppose that we have a set of knapsacks $\mathcal{M}$, each of which $m \in \mathcal{M}$ is with a capacity $\hat{D}_m$. The goal of MKP is to select as many disjoint subsets of items as possible, such that the total value of the selected items is maximized, while all the items packed into knapsack m fit capacity $\hat{D}_m$. Denoted by x_n^m a binary variable representing whether the item $n \in \mathcal{N}$ is selected to pack into knapsack $m \in \mathcal{M}$, we have the **MKP** problem as follows.

$$\textbf{MKP:} \ \max \sum_{m \in \mathcal{M}} \sum_{n \in \mathcal{N}} g_n \cdot x_n^m \tag{10}$$

$$\text{s.t.} \quad \sum_{n \in \mathcal{N}} w_n \cdot x_n^m \leq \hat{D}_m, \forall m \in \mathcal{M}. \tag{11}$$

$$\sum_{m \in \mathcal{M}} x_n^m \leq 1, \forall n \in \mathcal{N}. \tag{12}$$

$$\text{Variables: } x_n^m \in \{0, 1\}, n \in \mathcal{N}, m \in \mathcal{M}.$$

We then construct a new problem based on the original MKP. First, we construct the following new parameters: $g_n = \alpha \cdot \sigma f_n + 1, n \in \mathcal{N}$; $w_n = K_n, n \in \mathcal{N}$; $x_n^m = C_n^m, n \in \mathcal{N}, m \in \mathcal{M}$; $\hat{D}_m = D_m, m \in \mathcal{M}$. Then, we have the following new problem called **MKP-New**.

$$\textbf{MKP-New:} \ \max \sum_{m \in \mathcal{M}} \sum_{n \in \mathcal{N}} [\alpha \cdot \sigma f_n + 1] \cdot C_n^m \tag{13}$$

$$\text{s.t.} \quad \sum_{n \in \mathcal{N}} K_n \cdot C_n^m \leq D_m, \forall m \in \mathcal{M}. \tag{14}$$

$$\sum_{m \in \mathcal{M}} C_n^m \leq 1, \forall n \in \mathcal{N}. \tag{15}$$

$$\text{Variables: } C_n^m \in \{0, 1\}, n \in \mathcal{N}, m \in \mathcal{M}.$$

Through the reduction of the original MKP, we obtain a simplified instance of **B2E-Core**. This reduced **MKP-New** problem is completed in a polynomial time. Because of the NP-hardness of the MKP, we can conclude via this reduction that the **B2E-Core** problem is NP-hard, too. □

IV. Online Approximation Algorithm to Solving Broker2Earn

Due to the NP-hardness of the **B2E-Core** problem, it is intractable to obtain the optimal solution in polynomial time. To cope with the challenge of the computation complexity, we propose a *Randomized Rounding Algorithm*, named *B2E Alg.*. The core principle of *B2E Alg.* is to leverage a probabilistic approach to transform the solution of the relaxed problem into an approximately sub-optimal solution of the original **B2E-Core** problem.

A. Relaxed Problem and A Naive Algorithm

We first relax the **B2E-Core** problem defined in Eq. (6) - Eq. (9). By relaxing the integer constraint in Eq. (9) to a fractional form $C_n^m \in [0,1], n \in \mathcal{N}, m \in \mathcal{M}$, the resulted relaxed problem, denoted by **B2E-Relax**, is written as follows.

$$\textbf{B2E-Relax:} \ \max \sum_{m\in\mathcal{M}} U_m \tag{16}$$

$$\text{s.t.} \ \sum_{n\in\mathcal{N}} C_n^m \cdot K_n \le D_m, \forall m \in \mathcal{M}. \tag{17}$$

$$\sum_{m\in\mathcal{M}} C_n^m \le 1, \forall n \in \mathcal{N}. \tag{18}$$

$$\text{Variables: } C_n^m \in [0,1], n \in \mathcal{N}, m \in \mathcal{M}. \tag{19}$$

To solve **B2E-Relax** problem, we first design a naive algorithm called *Unit Revenue First Algorithm (URFA)* inspired by [22], which is described as follows.

Algorithm *URFA*. Referring to parameters g_n shown in the proof of Theorem 1, we now define a new variable $p_n = \alpha \cdot \sigma f_n + 1, \forall n \in \mathcal{N}$, which represents the revenue of serving a single CTX with transaction volume $K_n, \forall n \in \mathcal{N}$. *URFA* orders all CTXs in descending order according to their unit revenues, which are calculated by $p_n/K_n, \forall n \in \mathcal{N}$. Brokers then serve those CTXs having higher unit revenues with higher priorities. Then we have the following Theorem 2.

Theorem 2. *Suppose that at least one broker account exists in the system, and the staked tokens can provide the liquidity service for at least one CTX, URFA yields an optimal solution for **B2E-Relax**.*

Proof. According to **B2E-Relax** problem (Eq. (16) - Eq. (19)), set $\{C_n^m \in [0,1], \forall n \in \mathcal{N}, \forall m \in \mathcal{M}\}$ denotes the solution yielded by *URFA* to B2E-Relax problem. We then define $r_n \in [0, K_n], n \in \mathcal{N}$ to represent the liquidity provided by all brokers for CTX $n \in \mathcal{N}$. Specifically, $r_n = \sum_{m\in\mathcal{M}} C_n^m \times K_n, n \in \mathcal{N}$. Moreover, the sets of liquidity provided by brokers for all CTXs $\forall n \in \mathcal{N}$ while applying $\{C_n^m, \forall n \in \mathcal{N}, \forall m \in \mathcal{M}\}$ can be written as $\{r_n \in [0, K_n], \forall n \in \mathcal{N}\}$. The total revenue of brokers is $\sum_{n\in\mathcal{N}} \frac{p_n}{K_n} r_n$ when the blockchain system executes the solution obtained from *URFA*. Following the principle of *URFA*, CTXs in $\mathcal{N}$ are ordered in descending order of unit revenues. Denote the ordered set as $\tilde{\mathcal{N}}$, if $n \succeq \tau, \ n, \tau \in \tilde{\mathcal{N}}$, we have $p_n/K_n \le p_\tau/K_\tau, \forall n, \tau \in \tilde{\mathcal{N}}$.

We utilize mathematical induction to prove that *URFA*'s solution is optimal to **B2E-Relax** problem. Firstly, we assume that the *URFA*'s solution is only able to serve one CTX. Since *URFA* chooses the CTX which has the smallest unit revenue, the total revenue available to brokers is $\frac{p_1}{K_1}\sum_{m\in\mathcal{M}} D_m$. Since CTXs in $\tilde{\mathcal{N}}$ are ordered by descending unit revenues, we have $\frac{p_1}{K_1}\sum_{m\in\mathcal{M}} D_m \ge \frac{p_n}{K_n}\sum_{m\in\mathcal{M}} D_m, n \in \tilde{\mathcal{N}}$, which implies that *URFA* yields the optimal solution.

Secondly, we assume that *URFA* achieves the optimal total revenue when serving the first k CTXs, $\forall k \in [1, |\tilde{\mathcal{N}}| - 1]$. We can then prove that *URFA* remains optimal when *URFA* is able to serve the first $k+1$ CTXs. According to *URFA*'s solution, the total revenues that the first $k+1$ CTXs can be served is $\sum_{n \preceq k+1, n\in\tilde{\mathcal{N}}} \frac{p_n}{K_n} r_n = \sum_{n\preceq k, n\in\tilde{\mathcal{N}}} \frac{p_n}{K_n} r_n + \frac{p_{k+1}}{K_{k+1}} r_{k+1}$. We already know the total revenues of brokers $\sum_{n\preceq k, n\in\tilde{\mathcal{N}}} \frac{p_n}{K_n} r_n$ is optimal for the first k CTXs. Thus, we just need to show that adding the $(k+1)$-th CTX can maintain optimality. With the CTX set $\tilde{\mathcal{N}}$, taking the first $k+1$ CTXs implies $\frac{p_{k+1}}{k_{k+1}} r_{k+1} \ge \sum_{n \succ k+1, n\in\tilde{\mathcal{N}}} \frac{p_n}{K_n} r_n, \sum_{n\succ k+1, n\in\tilde{\mathcal{N}}} r_n = r_{k+1}, k \in [1, |\tilde{\mathcal{N}}|]$.

In other words, when *URFA* can serve the first $k+1$ CTXs, the resulted total revenue is optimal. We can derive that *URFA*'s solution is optimal for the **B2E-Relax** problem. The proof of Theorem 2 concludes. □

B. Randomized Rounding Algorithm

Based on the relaxed solution yielded by *URFA*, we design a randomized rounding algorithm, named *B2E Algorithm*. The proposed algorithm B2E Alg. is presented in Algorithm 1.

B2E Alg. takes the following parameters as input: the set of CTXs $\mathcal{N}$, the set of brokers $\mathcal{M}$, the set $\{K_{n\in\mathcal{N}}\}$, and the set $\{D_{m\in\mathcal{M}}\}$. In line 1 of B2E Alg., the decision variables $x_n^m = 0, \forall n \in \mathcal{N}, \forall m \in \mathcal{M}$ is initialized. We then relax the binary variables $C_n^m, n \in \mathcal{N}, m \in \mathcal{M}$ by invoking the *URFA* to obtain the optimal real number-valued solutions $\tilde{x}_n^m, n \in \mathcal{N}, m \in \mathcal{M}$ (line 2). With the solution $\{\tilde{x}_n^m\}$, Alg. 1 keeps rounding the CTX $n \in \mathcal{N}$ served by broker $m \in \mathcal{M}$ by probability $\frac{\tilde{x}_n^m}{\sum_{n\in\mathcal{N}} \tilde{x}_n^m}$ until the remaining collateralized tokens of broker m become inadequate (line 4 - line 10). After serving the CTXs by a broker $m \in \mathcal{M}$, the decision variables X, the remaining collateralized tokens D_m, and the set of CTXs $\mathcal{N}$ will be updated (line 11 - line 13). Finally, the decision variables $\{x_n^m, \forall n \in \mathcal{N}, m \in \mathcal{M}\}$ will be returned.

C. Online B2E Algorithm

The goal of **Broker2Earn** protocol is to incentivize users to become brokers by maximizing their revenues. To achieve such a goal, Algorithm 1 must be executed in an online manner . We thus design Algorithm 2 named *online B2E Algorithm*, which invokes Algorithm 1 in the real-time execution of the sharded blockchain.

Let t denote each epoch of the blockchain system. At the beginning of epoch t, the smart contract constructs the set of CTXs $\mathcal{N}$ and the set of brokers $\mathcal{M}$ (line 2 - line 3). After constructing those sets, the volume of transferred tokens in CTX n (i.e., K_n in line 5) and the collateralized tokens from broker m (i.e., D_m in line 7) are read from $\mathcal{N}$ and $\mathcal{M}$,

Algorithm 1: Randomized Rounding (B2E Algorithm)

```
Input : $\mathcal{N}$, $\mathcal{M}$, $\{K_{n\in\mathcal{N}}\}$ ($K_n$ is the volume of
        transferred tokens in CTX $n$), $\{D_{m\in\mathcal{M}}\}$.
Output: The solution to **B2E-Core** (Eq. (6) - Eq. (9)).
1  $X := \{x_n^m \leftarrow 0, \forall n \in \mathcal{N}, \forall m \in \mathcal{M}\}$
2  $\{\tilde{x}_n^m\} \leftarrow$ URFA$(\mathcal{N}, \mathcal{M})$
3  for each $m \in \mathcal{M}$ do
4      $X_m := \{x_n^m \leftarrow 0, \forall n \in \mathcal{N}\}$
5      while each $n \in \mathcal{N}$ do
6          Set $x_n^m \leftarrow 1$ with probability $\frac{\tilde{x}_n^m}{\sum_{n\in\mathcal{N}} \tilde{x}_n^m}$
7          if $\sum_{n\in\mathcal{N}}(x_n^m \cdot K_n) \geq D_m$ then
8              Set $x_n^m \leftarrow 0$
9              Break
10         Set $\tilde{x}_n^m \leftarrow 0$ if $x_n^m = 1$
11     $X \leftarrow X \bigcup X_m$
12     $D_m \leftarrow D_m - \sum_{n\in\mathcal{N}} x_n^m \times K_n$
13     $\mathcal{N} \leftarrow \mathcal{N} \backslash \{n | (x_n^m == 1, \forall n \in \mathcal{N}\}$
14 Return $X$
```

Algorithm 2: Online B2E Algorithm

```
1  for at the beginning of epoch $t = 1, 2, \cdots$ do
2      $\mathcal{N} \leftarrow$ All CTXs observed by smart contract from
         TX pool
3      $\mathcal{M} \leftarrow$ All brokers accepted by smart contract
4      for $n \in \mathcal{N}$ do
5          $K_n \leftarrow$ the token volume transferred in CTX $n$
6      for $m \in \mathcal{M}$ do
7          $D_m \leftarrow$ collateralized tokens from broker $m$
8      $X_t \leftarrow$ **Algorithm 1**$(\mathcal{N}, \mathcal{M}, \{K_{n\in\mathcal{N}}\}, \{D_{m\in\mathcal{M}}\})$
9      The blockchain system executes according to $X_t$
```

respectively. In line 8, X_t is obtained by invoking Algorithm1. The sharded blockchain then interprets the liquidity services from X_t (line 9). The online B2E Algorithm 2 will terminate when the blockchain system is shut down.

D. Algorithm Analysis

Given $\mathcal{M}$ and $\mathcal{N}$, we denote $\mathbf{x} : \{\tilde{x}_n^m, \forall n \in \mathcal{N}, m \in \mathcal{M}\}$ as the solution obtained from *URFA*, where $\tilde{x}_n^m \in [0,1], n \in \mathcal{N}, m \in \mathcal{M}$ is a real-numbered variable indicating the fractional liquidity served by broker $m \in \mathcal{M}$ for CTX $n \in \mathcal{N}$. As depicted in line 6 of Algorithm 1, each CTX n is evaluated with a probability $\tilde{x}_n^m / \sum_{n\in\mathcal{N}} \tilde{x}_n^m$. Let $\lambda \in \mathbb{R}^+$ denote the total fractional number of served CTXs indicated by the optimal solution $\mathbf{x} : \{\tilde{x}_n^m, \forall n, m\}$, we have $\lambda = \sum_{m\in\mathcal{M}} \sum_{n\in\mathcal{N}} \tilde{x}_n^m$. Next, we have the following Theorem 3.

Theorem 3. *Given $\mathcal{M}$, $\mathcal{N}$ and λ, the B2E Algorithm (i.e., Algorithm 1) is a $\frac{(1-\frac{1}{\lambda})^{\lambda}(\lfloor\lambda\rfloor - |\mathcal{M}|)\cdot e}{\sqrt{\lambda(\lambda-1)}}$-approximation algorithm, where e is Euler's number.*

Proof. Given a broker $m \in \mathcal{M}$, we first define $\delta_m = \sum_{n\in\mathcal{N}} \tilde{x}_n^m, \forall m \in \mathcal{M}$ as the sum of the real number-valued number of CTXs served by broker m, after solving the B2E-Relax problem by *URFA*. Thus, $\delta = \sum_{m\in\mathcal{M}} \delta_m$ can represent the total number of CTXs served by all brokers' liquidity services. Note that, $\tilde{x}_n^m$ ranges continuously from 0 (not served by broker m at all) to 1 (served by broker m fully).

According to Alg. 1, to determine which CTX served by which broker's liquidity service, we denote $l = 1, 2, \cdots, \lfloor\delta_m\rfloor$ as the number of the served CTX by broker $m \in \mathcal{M}$. In the process of determining which CTX served by broker $m \in \mathcal{M}$, we define $\beta(l) \in \mathcal{N}, l = 1, 2, \cdots, \lfloor\delta_m\rfloor$ describing the CTX served by broker $m \in \mathcal{M}$ in the l-th rounding operation (line 6 in Alg. 1). For example, $\beta(l) = n, \forall\beta(l), n \in \mathcal{N}, l = 1, 2, \cdots, \lfloor\delta_m\rfloor$ when CTX $n \in \mathcal{N}$ is served in l-th rounding attempt. Thus, we use another new binary variable $y_n^{m,l} \in \{0,1\}$ to represent whether a CTX $n \in \mathcal{N}$ is served by broker $m \in \mathcal{M}$ for the l-th rounding operation.

We then define $P(\cdot)$ as the probability function for events. According to Eq.(16) - Eq.(19), we still adopt variables $C_n^m \in \{0,1\}, n \in \mathcal{N}, m \in \mathcal{M}$. Our goal is to derive $P(x_n^m = 1)$, which represents the probability that CTX $n \in \mathcal{N}$ is fully served by broker $m \in \mathcal{M}$.

In the following, we discuss two cases of l when $l = 1$ and $l \geq 2$. Let $l = 1$, we can calculate that the probability of CTX $n \in \mathcal{N}$ being served for the first time by broker $m \in \mathcal{M}$ indicated by the solution obtained from invoking B2E Algorithm. This probability is denoted by $P(y_{\beta(l)}^{m,1} = 1)$, and it is equivalent to the probability for the case $y_n^{m,1} = y_{\beta(l)}^{m,1} = 1, \beta(l) \in \mathcal{N}, m \in \mathcal{M}, l = 1, 2, \cdots, \lfloor\delta_m\rfloor$. We have

$$P(y_{\beta(l)}^{m,1} = 1) = \frac{\tilde{x}_{\beta(l)}^m}{\sum_{n\in\mathcal{N}} \tilde{x}_n^m} = \frac{\tilde{x}_{\beta(l)}^m}{\delta_m} = \frac{\tilde{x}_{\beta(1)}^m}{\delta_m}. \tag{20}$$

Let $l \geq 2$, we are able to get the probability that CTX $n \in \mathcal{N}$ is served by broker $m \in \mathcal{M}$ for the l-th rounding attempt. We denote $\vec{v} = \{\beta(l), \forall l = 1, 2, \cdots, |\mathcal{N}|\}$ (where $\beta(l) = n,\ n \in \mathcal{N}$) as the set depicting the CTX served in l-th rounding attempt by broker $m \in \mathcal{M}$. And we use $\vec{v}[q \preceq l] = \{\beta(q), \forall q \preceq l,\ q, l = 1, 2, \cdots, |\mathcal{N}|\}$ to represent the subset of CTXs that is served before the l-th rounding attempt. Thus, $P(y_{\beta(l)}^{m,l} = 1)$ can be expressed as a conditional probability given the occurrence of other CTXs being served:

$$P(y_{\beta(l)}^{m,l} = 1) = \sum_{\substack{\beta(l)\in\vec{v}[q\preceq l] \\ \forall l=2,3,\cdots,|\mathcal{N}|, \\ \forall q=1,2,\cdots,|\mathcal{N}|}} \left[\prod_{t=1}^{l-1} \frac{\tilde{x}_{\beta(t+1)}^m}{\delta_m - \sum_{i=1}^{t} \tilde{x}_{\beta(i)}^m}\right] \cdot \frac{\tilde{x}_{\beta(1)}^m}{\delta_m}.$$

Ultimately, we can derive the probability that CTX n is served by broker $m \in \mathcal{M}$, denoted by $P(C_n^m = 1)$, as follows:

$$\begin{aligned} P(C_n^m = 1) &= P(y_n^{m,1} = 1) + \sum_{l=2}^{|\mathcal{N}|} P(y_n^{m,l} = 1) \\ &= \frac{\tilde{x}_n^m}{\delta_m} + \sum_{\substack{\beta(l)\in\vec{v}[q\preceq l] \\ \forall l=2,3,\cdots,|\mathcal{N}|, \\ \forall q=1,2,\cdots,|\mathcal{N}|}} \left[\prod_{t=1}^{l-1} \frac{\tilde{x}_{\beta(t+1)}^m}{\delta_m - \sum_{i=1}^{t} \tilde{x}_{\beta(i)}^m}\right] \cdot \frac{\tilde{x}_{\beta(1)}^m}{\delta_m}. \end{aligned} \tag{21}$$

Now we get the utility expectation of B2E Algorithm, denoted by $\mathbb{E}(\text{B2E})$, by referring to p_n defined in *URFA*:

$$\mathbb{E}(\text{B2E}) = \sum_{m\in\mathcal{M}}\sum_{n\in\mathcal{N}} P(C_n^m = 1) \times p_n. \tag{22}$$

To calculate $P(C_n^m = 1)$, we need to discuss two cases depending on the situation of whether the broker-collateralized tokens are sufficient or not. In the first case, we assume the liquidity provided by broker $m \in \mathcal{M}$ is adequate to fulfill all CTXs according to the *URFA*'s solution. In the case where $D_m \geq \sum_{n\in\mathcal{N}} \tilde{x}_n^m K_n, m \in \mathcal{M}$, all CTXs provided by *URFA* can be served by brokers, meaning that $C_n^m = 1, \forall n \in \mathcal{N}, \forall m \in \mathcal{M}$. This is equivalent to $P(C_n^m = 1) = 1$, and is calculated as:

$$\begin{aligned} P(C_n^m = 1) &= P(y_n^{m,1} = 1) + \sum_{l=2}^{|\mathcal{N}|} P(y_n^{m,l} = 1) \\ &= \left[\frac{[\delta_m - (l-1)]!}{\delta_m!}\right] \times \frac{\lfloor\delta_m\rfloor!}{[\lfloor\delta_m\rfloor - (l-1)]!} = 1. \end{aligned} \tag{23}$$

In another case, we assume the liquidity provided by broker $m \in \mathcal{M}$ is inadequate and only sufficient to fulfill a subset of CTXs according to *URFA*'s solution, i.e., $D_m < \sum_{n\in\mathcal{N}} \tilde{x}_n^m \times K_n, m \in \mathcal{M}$. This implies that for a given CTX n served by broker m, the probability that CTX n being served by broker $m \in \mathcal{M}$ diminishes to zero once $l = 1, 2, \cdots, \lfloor\delta_m\rfloor$ exceeds a particular threshold.

Next, we let real-numbered variables $\lambda_m \in \mathbb{R}^+$ to represent the sum of the variables included in the optimal solution for broker $m \in \mathcal{M}$. Thus, we have $\lambda = \sum_{m\in\mathcal{M}} \lambda_m$. Given this, we can state that the CTX can't be served when $l > \lambda_m, m \in \mathcal{M}$. And we denote $\varphi \in \mathbb{N}^+$ as the number of the CTXs possibly served by all brokers, and $\varphi \in [\lfloor\lambda\rfloor - |\mathcal{M}|, \lfloor\lambda\rfloor]$. So we have the probability that CTX $n \in \mathcal{N}$ is served by brokers utilizing Stirling's approximation formula [23]:

$$\begin{aligned} &\sum_{m\in\mathcal{M}} P(C_n^m = 1) = \sum_{m\in\mathcal{M}} P(y_n^{m,l} | l < \lambda_m) \\ &= \frac{[\lambda - (l-1)]!}{(\lambda)!} \cdot \frac{(\lambda-1)!}{[\lambda - (l-1)]!} \cdot \varphi \\ &= \frac{\sqrt{2\pi(\lambda-1)}(\frac{\lambda-1}{e})^{\lambda-1}}{\sqrt{2\pi(\lambda)}(\frac{\lambda}{e})^{\lambda}} \cdot \varphi = \frac{(1-\frac{1}{\lambda})^{\lambda} \cdot e \cdot \varphi}{\sqrt{\lambda(\lambda-1)}}, \end{aligned} \tag{24}$$

where e is Euler's number. We denote A^* as the CTX set indicated by the theoretically optimal solution to the **B2E-Core** problem. Thereby, the expectation of B2E Alg. also can be calculated as:

$$\begin{aligned} \mathbb{E}(\text{B2E}) &= \sum_{n\in\mathcal{N}}\sum_{m\in\mathcal{M}} P(C_n^m = 1) \cdot p_n \\ &= \sum_{n\in\mathcal{N}} \frac{(1-\frac{1}{\lambda})^{\lambda} \cdot e \cdot \varphi}{\sqrt{\lambda(\lambda-1)}} \cdot p_n \geq \frac{(1-\frac{1}{\lambda})^{\lambda} \cdot e \cdot \varphi}{\sqrt{\lambda(\lambda-1)}} \sum_{\beta\in A^*} p_\beta \\ &\geq \frac{(1-\frac{1}{\lambda})^{\lambda}(\lfloor\lambda\rfloor - |\mathcal{M}|) \cdot e}{\sqrt{\lambda(\lambda-1)}} \cdot \texttt{OPT}, \end{aligned}$$

where $\texttt{OPT}$ denotes the theoretically optimal objective value of the **B2E-Core** problem. The proof concludes.

□

Discussion. Clearly, $(1-\frac{1}{\lambda})^{\lambda}$ is a monotonically increasing function in λ with $\lim_{\lambda\to+\infty}(1-\frac{1}{\lambda})^{\lambda} = e^{-1}$. Leveraging this, the previously derived approximation ratio can be further bounded as follows:

$$\begin{aligned} &\frac{(1-\frac{1}{\lambda})^{\lambda}(\lfloor\lambda\rfloor - |\mathcal{M}|) \cdot e}{\sqrt{\lambda(\lambda-1)}} \leq \frac{(\lfloor\lambda\rfloor - |\mathcal{M}|)}{\sqrt{\lambda(\lambda-1)}} \\ &= \sqrt{\frac{(\lfloor\lambda\rfloor - |\mathcal{M}|)}{\lambda} \cdot \frac{(\lfloor\lambda\rfloor - |\mathcal{M}|)}{\lambda - 1}} < 1, \end{aligned} \tag{25}$$

where $|\mathcal{M}| \geq 1$, indicating at least one broker serves others. This constraint arises because having fewer than one broker would be inadequate to serve any CTXs.

V. Experiments

A. Experiment Settings

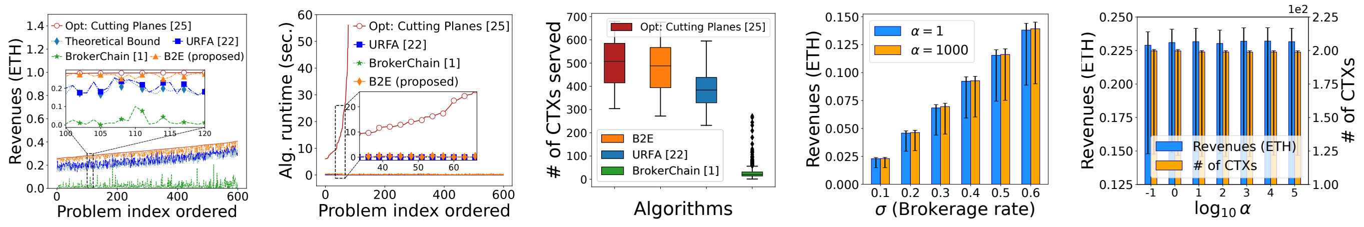
Experiment Tools: *TX-driven simulator* and *BlockEmulator testbed*. To evaluate the performance of our Broker2Earn protocol, we conduct experiments using two types of tools. Firstly, to verify the feasibility of our Broker2Earn, we develop a TX-driven simulator in Python. Secondly, for large-scale experiments, we adopt an open-sourced blockchain sharding testbed named BlockEmulator [24]. All experiments are carried out on 4 high-performance physical machines, each equipped with AMD 7950X CPU and 64GB RAM.

BlockEmulator settings. We deploy 16 blockchain shards in BlockEmulator. Each shard consists of 4 nodes. All nodes execute on our physical machines. We chose this scale because the performance of Broker2Earn protocol primarily depends on the input scale of historical transactions, not the number of blockchain shards. Thus, a 16-shard blockchain testbed is sufficient for evaluating our protocol.

Datasets. We collected 1.5 million Ethereum transactions from the blocks between heights 1000000 and 1999999. In each experiment, at the beginning of each epoch, the real-world historical Ethereum TXs are prepared in chronological order and replayed in either the TX-driven simulator or BlockEmulator with a specific arrival rate.

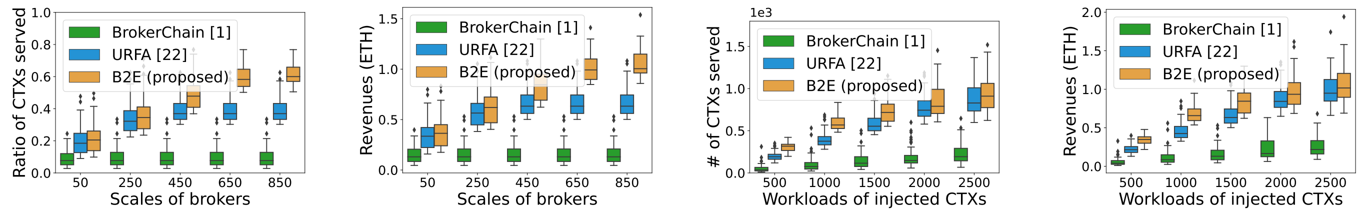
Baselines. We implement the following algorithms as baselines for our experiments.

- **Opt: Cutting Planes.** The optimal solution is obtained by solving **B2E-Core** using the mathematical solver Gurobi [25], which invokes the classical *Cutting Planes* method.
- **Theoretical Bound.** We also introduce this metric in a group of experiments. It represents the lower bound of B2E Algorithm's utility and is calculated by referring to Eq. (25) and the utility yielded by *Cutting Planes*.
- **URFA.** The principle of algorithm URFA is explained in Section IV-A.
- **BrokerChain.** Referring to [1], the solution of this algorithm is obtained by distributing the collateralized tokens of brokers into all shards equally.



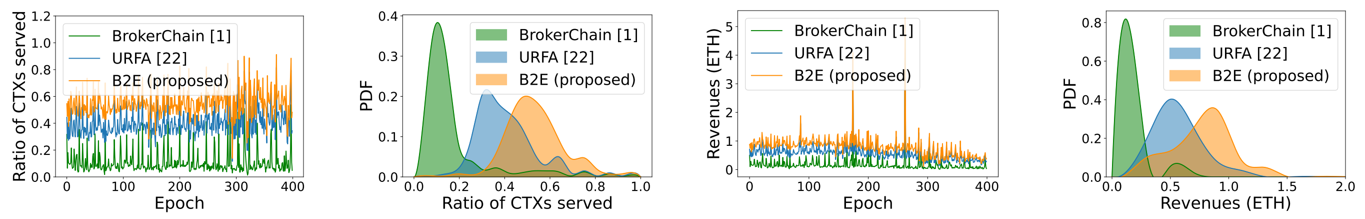
(a) Brokers' revenue (b) Algorithm runtime (c) # of CTXs served (d) Brokerage rate in B2E (e) Weight α (0.1-1e5) in B2E

Fig. 3. Performance while varying parameters. In (a) and (b), the problem indices in the x-axis are ordered by the absolute values of the y-axis metrics.



(a) Ratio of CTXs served (b) Revenues (c) # of CTXs served (d) Revenues

Fig. 4. (a) and (b) demonstrate the effect of broker scales, while varying broker # from 50 to 850, and keeping injecting 1500 CTXs per epoch. (c) and (d) show the effect of CTX workloads injected (varying from 500 to 2500), while fixing the scale of brokers at 450. Each broker collateralizes 2e18 wei.



(a) Ratio of CTXs served (b) PDF of ratio of CTXs served (c) Revenues (d) PDF of revenues

Fig. 5. Performance demonstration while invoking different algorithms in BlockEmulator. (b) and (d) are PDFs corresponding to (a) and (c), respectively.

Metrics. We evaluate our Broker2Earn protocol by measuring the following metrics. **i) Ratio/Number of CTXs served.** The Ratio/number of CTXs served by broker. These two metrics are used to assess the efficacy of the collateralized tokens from brokers. **ii) Revenues of brokers.** The total revenues received by brokers after serving CTXs because they offer system liquidity for the blockchain. **iii) Algorithm runtime.** The runtime of algorithms while getting solutions to **B2E-Core** problem. This metric is used to judge whether an algorithm is practical or not for a real blockchain system.

B. Performance Observed via TX-driven Simulator

In the experiments demonstrated in Fig. 3, we evaluate the performance of all algorithms in the TX-driven simulator. The number of blockchain shards is set to 64. The sharded blockchain simulation keeps running for 1000 epochs. In each epoch of consensus, 1500 transactions are injected into the simulator. The number of brokers linearly increases from 5 to 500 during the 1000 epochs of running. Each broker collateralizes 2 ETH (i.e., 2e18 wei) to the Broker2Earn protocol. Wei refers to the smallest denomination of ether (ETH), the currency used in Ethereum.

Fig. 3(a) illustrates the total revenues earned by all brokers in 600 epochs. Here, those results are shown following the problem index ordered by the y-axis metric (i.e., revenues). We observe that the proposed B2E algorithm shows a good performance that is very close to the optimal solution yielded by *Cutting Planes* and is higher than the theoretical bound. The broker revenues indicated by B2E algorithm are much higher than that of URFA and BrokerChain.

Fig. 3(b) shows the runtime of all algorithms in 600 out of 1000 instances of problem-solving. Although *Cutting Planes* yields the optimal solutions, its running time drastically increases after the 50th index, surpassing one minute. In contrast, other algorithms can execute so fast that their runtime is very close to 0 seconds when the number of brokers increases. This result implies that *Cutting Planes* is unacceptable in a large-scale system. However, the proposed online B2E algorithm is feasible for a dynamically scaling-out blockchain system.

Next, Fig. 3(c) investigates the number of CTXs served by all brokers. The result of each algorithm is obtained from executing 600 epochs. We see that the proposed B2E algorithm outperforms URFA and BrokerChain, and can diminish a large number of CTXs that is close to that of the optimal solution.

In Fig. 3(d), we study the cumulative brokers' revenues while varying the brokerage rate σ from 0.1 to 0.6, and fixing the weight coefficient α to 1 and 1000, respectively. The results show that the revenues linearly increase following σ, but α does not have essential effects on broker revenues.

To figure out the impact of α, we conduct another group of experiments shown in Fig. 3(e). We vary α from 0.1 to 1e5 and compare the two terms of the objective of **B2E-Core** problem (i.e., revenues and the # of CTXs served by brokers). To our surprise, we only find very slight changes on these two metrics even when α changes drastically. The results imply that the weight coefficient α indeed almost has no impact on the two terms of objective function (5). However, the settings do not contradict our design goal of the B2E protocol. This is because when any new broker joins the B2E protocol, both the two terms of objective function (5) are enlarged simultaneously.

C. Performance Observed via BlockEmulator Testbed

Using BlockEmulator testbed, we measure the performance of the proposed B2E algorithm with a large-scale number of brokers and large workloads of injected CTXs.

1) The effect of broker account's scale: While the number of broker accounts grows, the total staked tokens from those brokers also increase. That is, the system liquidity that can be exploited to handle CTXs improves. We are curious about how different scales of brokers joining the B2E protocol impact the ratio of CTXs served by liquidity services. To do so, we fix the TX injection rate at 1500 CTXs per epoch. Each broker collateralizes 2e18 wei to B2E protocol. The scales of brokers vary from 50 to 850 in 5 groups of experiments. In every epoch, we repeat 50 times for the system to run. Fig. 4(a) and 4(b) show the ratio of CTXs served by liquidity services and the total revenues earned by brokers versus different scales of brokers, respectively. We see that both metrics improve following the broker's scale injected into the B2E protocol. This is because growing system liquidity can serve more CTXs existing in the sharded blockchain. However, once the capacity of liquidity (i.e., the total staked tokens from all brokers) reaches a threshold such as 650 brokers for the proposed B2E algorithm, the service level becomes saturated because most of the CTXs existing in the transaction pool have been served. Only several ones with huge transaction volumes cannot be served for any reason. After all, we see that the proposed B2E algorithm achieves the best service level out of all algorithms. We observe similar performance in Fig. 4(b), and the insights behind those results are the same as that of Fig. 4(a). Thus, we omit its interpretation here.

2) The effect of CTX workloads: Applying similar settings to the experiments shown in Fig. 4(a) and 4(b), we then evaluate the effect of the workloads of CTXs arrived to the blockchain system, by varying the number of CTXs injected per epoch from 500 to 2500 and fixing the number of brokers to 450 per epoch. As shown in Fig. 4(c), Fig. 4(d), both the cumulative number of CTXs served by liquidity services and the total revenues of brokers increase for all three algorithms as the workloads of CTXs grow from 500 to 2500, but the pace of performance improvement becomes slow. These results can be attributed to the following reason. B2E protocol has a larger scheduling space while the number of CTXs increases, even with a fixed liquidity service level. Thus, both the number of CTXs served and the broker's revenues benefit from the enlarged scheduling space residing in the **B2E-Core** module.

3) Performance with a large-scale transaction workload: In the final group of experiments, we test the performance of the online version B2E algorithm (i.e., Alg. 2), while injecting a very large number of TXs into the system. By fixing the number of brokers to 450 and the CTX workload to 1500 per epoch, we run all algorithms for 400 epochs. The total number of CTXs dealt with in the system reaches 600 thousand. To emulate the real-world staking services in a DeFi protocol, for each epoch, the number of tokens collateralized by brokers is generated following a normal distribution with a mean of 2e18 wei and a standard deviation of 2e17. Fig. 5(a) and Fig.5(b) show the variation and the Probability Density Function (PDF) of the ratio of CTXs served by brokers, respectively. Fig. 5(c) and Fig. 5(d) show the variation and the PDF of the revenues earned by brokers from serving CTXs, respectively. We see that the proposed B2E yields the highest performance in all metrics demonstrated in this group of experiments. This result proves the efficacy of the B2E protocol when the sharded blockchain is working in a large-scale transaction workload.

VI. Conclusion

Broker2Earn (B2E) protocol can serve as an incentive mechanism for the economic model in a sharded blockchain such as BrokerChain. Through the B2E protocol, any voluntary account holders (called brokers) can join the protocol and stake their idle tokens to earn revenues. Those staked tokens provide crucial liquidity for the sharded blockchain system because brokers' tokens play as intermediary bridges when blockchain shards are handling cross-shard transactions. Thus, the proposed B2E protocol can benefit both the staking revenues of brokers and the system liquidity of a sharded blockchain. We formulate the core module of B2E protocol as a revenue-maximization problem, which is proven NP-hard. To solve this problem, we propose an online approximation algorithm by exploiting the relax-and-rounding technique. The approximation ratio of the proposed algorithm is analyzed rigorously. We implement our B2E protocol in an open-source blockchain testbed BlockEmulator. The evaluation results using the historical Ethereum transactions show that the B2E protocol is promising to provide a near-optimal performance in terms of system liquidity and broker revenues. We plan to market our B2E protocol as a DeFi product shortly.

Acknowledgment

This work was partially supported by the National Key R&D Program of China (No. 2022YFB2702304), NSFC (No. 62272496), General Universities Key Field Project of Guangdong Province (No.2023ZDZX1001), and Fundamental Research Funds for the Central Universities, Sun Yat-sen University (No. 23lgbj019).

References

[1] 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 *IEEE Conference on Computer Communications (INFOCOM'22)*, 2022, pp. 1968–1977.

[2] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A Secure Sharding Protocol For Open Blockchains," in *Proc. of ACM SIGSAC Conference on Computer and Communications Security (CCS'16)*, 2016, pp. 17–30.

[3] 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.

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

[5] 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)*. Boston, MA: USENIX Association, 2019, pp. 95–112.

[6] "Polygon," https://polygon.technology/solutions/polygon-pos, 2021.

[7] "Argent wallet," https://www.argent.xyz/, 2022.

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

[9] C. Li, P. Li, D. Zhou, Z. Yang, M. Wu, G. Yang, W. Xu, F. Long, and A. C.-C. Yao, "A decentralized blockchain with high throughput and fast confirmation," in *2020 USENIX Annual Technical Conference (ATC)*, 2020, pp. 515–528.

[10] J. Poon and V. Buterin, "Plasma: Scalable autonomous smart contracts," *White paper*, pp. 1–47, 2017.

[11] G. Malavolta, P. Moreno-Sanchez, A. Kate, M. Maffei, and S. Ravi, "Concurrency and privacy with payment-channel networks," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS'17)*, 2017, pp. 455–471.

[12] L. Aumayr, S. A. Thyagarajan, G. Malavolta, P. Moreno-Sanchez, and M. Maffei, "Sleepy channels: Bi-directional payment channels without watchtowers," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS'22)*, 2022, pp. 179–192.

[13] T. Xie, J. Zhang, Z. Cheng, F. Zhang, Y. Zhang, Y. Jia, D. Boneh, and D. Song, "zkbridge: Trustless cross-chain bridges made practical," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS'22)*, 2022, pp. 3003–3017.

[14] "Optimistic rollups," https://ethereum.org/en/developers/docs/scaling/optimistic-rollups, 2022.

[15] "Zero-Knowledge rollups," https://ethereum.org/en/developers/docs/scaling/zk-rollups/, 2022.

[16] C. Li, H. Huang, Y. Zhao, X. Peng, R. Yang, Z. Zheng, and S. Guo, "Achieving scalability and load balance across blockchain shards for state sharding," in *2022 41st International Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 2022, pp. 284–294.

[17] Z. Hong, S. Guo, E. Zhou, 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 42nd International Conference on Computer Communications (INFOCOM'23)*, 2023.

[18] "MetaMask," https://metamask.io/, 2022.

[19] "Lido," https://lido.fi/, 2022.

[20] "Compound," https://compound.finance/, 2022.

[21] "Uniswap," https://app.uniswap.org/, 2022.

[22] J. Leskovec, A. Krause, C. F. C. Guestrin, J. VanBriesen, and N. Glance, "Cost-effective outbreak detection in networks," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'07)*, 2007, p. 420–429.

[23] S. Athanassoulis, K. L. Kavousianos, and I. Nikolos, "Optimal, near-optimal and heuristic solutions for vehicle routing and related problems," *European Journal of Operational Research*, vol. 2020, no. 282, pp. 397–416, 2020.

[24] 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.

[25] "Gurobi," https://www.gurobi.com/, 2022.