Performance Modeling of Relay Chain

Bo Li, *Student Member, IEEE,* TianTian Duan, Qinglin Zhao, *Senior Member, IEEE,* Zhaoxiong Song, Yi Guo, *Student Member, IEEE,* Hanwen Zhang, *Member, IEEE,* Zhongcheng Li,*Member, IEEE,* Yi Sun, *Member, IEEE*

Abstract—Demand for cross-chain interoperability has been increasing, and relay chain mode solutions are stat-of-the-art and mainstream nowadays. However, the relay chain mode solutions suffer from high end-to-end latency, which stems from its performance bottleneck – the relay chain. Therefore, configuring its system parameters appropriately to meet its performance requirements is vital. Currently, there is no specialized performance analysis model of the relay chain. Existing single-blockchain models are not applicable for relay chains because relay chains have several distinct features: transaction arrival in batches with uncertain sizes, support for on-chain simplified payment verification (SPV), Byzantine fault tolerance (BFT) type protocol, and specific packaging rules. This work first proposes an analytical framework for relay chain performance. It captures the mentioned features of relay chains by constructing a batch-arrival and bulk-service model. We give a concrete calculation of the relay chain with practical BFT (PBFT) consensus and develop a method to arrive at the computational forms of two essential performance descriptors of relay chains: system throughput and cross-chain transaction latency. Through the model in this work, we can judge accurately whether the relay chain is overloaded, and eliminate the overload state by tuning the parameters of the system; and we can evaluate the system performance under different traffic and design parameters. Finally, we verify the models through simulation. With our study, operators of relay chains can configure system parameters effectively to meet the requirements of practical use.

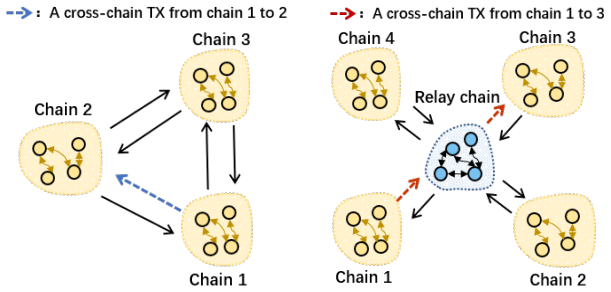
Index Terms—Blockchain, Performance analysis, Relay chain, Cross-chain transactions, Batch arrival and bulk service, Modeling

1 Introduction

1.1 Background

With the development of blockchain technologies, many blockchain systems have been emerging. Among them, cryptocurrency blockchains like Bitcoin [1], Ethereum [2], Litecoin [3], BNB [4], and supply chain platforms like [5–7] are notably representative. Meanwhile, demand for cross-chain interoperability is increasing, for example, cryptocurrency exchange among multiple blockchains and data sharing between supply chains. These use cases heavily rely on cross-chain technology, by which a blockchain verifies and executes the transaction (later referred to as TX) from another blockchain to realize cross-chain data circulation.

From a topology perspective and considering whether a third-party blockchain is required, there are two primary categories of solutions to cross-chain interoperation: direct mode and relay chain mode. The direct mode is shown in Fig. 1(a), where interoperating blockchains connect to and communicate with each other in pairs, e.g., BTC Relay [8], WeCross [9], zkRelay [10], zkBridge [11]. The connection here means deploying the TX verification mechanism of other chains on the current chain. In the direct mode, a cross-chain TX, after its consensus finality on the source chain (e.g., chain 1 in the blue arrow), is sent directly to the target chain (e.g., chain 2 in the blue arrow), where it is then verified and executed. The direct mode has low latency, but when multiple chains interoperate with each other, each chain needs $O(n)$ connections, that is, deploying TX verification mechanisms of all other chains, which results in poor scalability. The relay chain mode is shown in Fig.1 (b), where a specialized chain (called relay chain) is responsible for connecting all interoperating chains. The relay chain deploys all the TX verification mechanisms needed, and a chain only needs to connect to it to interoperate with all other connected chains. In the relay chain mode, a cross-chain TX, after its appearance on the source chain (e.g., chain 1), is firstly sent to the relay chain for verification and then to the target chain (e.g., chain 3). Compared with the direct mode, the relay chain mode decreases the connection complexity for each chain to $O(1)$. It is more suitable to use cases involving multiple chains. Many well-known cross-chain projects like Cosmos [12], BitXHub [13], and Polkadot [14] employ the relay chain mode. However, the relay chain mode has its apparent weakness: the high end-to-end latency of cross-chain TXs.



(a) Derect mode (b) Relay chain mode

Fig. 1: Two kinds of cross-chain solutions.

- *B. Li, T. Duan, Z. Song, Y. Guo, H. Zhang, Z. Li, Y. Sun are with the Institute of Computing Technology, Chinese Academy of Sciences, China. (e-mail: {libo18z,duantiantian,songzhaoxiong,guoyi19g,hwzhang,zcli, sunyi}@ict.ac.cn)*
- *B. Li, T. Duan, Z. Song, Y. Guo, H. Zhang, Z. Li, Y. Sun are with the University of Chinese Academy of Sciences, China.*
- *Q. Zhao is with the School of Computer Science and Engineering, Macau University of Science and Technology, Avenida Wei Long, Taipa, Macau 999078, China. (e-mail: qlzhao@must.edu.mo)*
- *Y. Sun is with Shandong Key Laboratory of Blockchain Finance, China.*

The end-to-end latency of cross-chain TXs in the relay chain mode is primarily decided by the performance of the relay chain. For cross-chain TXs, the latency caused by the source chain and target chain is unavoidable and does not differ notably in both direct and relay chain modes. However, in the relay chain mode, the new hop for cross-chain TXs - relay chain will significantly increase the latency. On the one hand, the relay chain needs to process the cross-chain TXs, which apparently introduces latency. The relay chain packages cross-chain TXs into blocks by packaging rules; these blocks finish consensus in the network, and the cross-chain TXs go through the TX verification mechanism simultaneously. On the other hand, in the relay chain mode, multiple chains are connected to the relay chain, and all the cross-chain TXs need to be processed by the relay chain, which causes the relay chain to queue. Relay chain queuing greatly adds to the end-to-end latency.

1.2 Motivation

Since a proper configuration can help relay chains satisfy performance requirements, a performance analysis model to guide this configuration has become increasingly vital. However, a specialized and rigorous relay chain model is still absent. Moreover, existing models on single chains do not apply to relay chains because they have many different features. Here we compare the relay chain with two of the most well-known single chains: Bitcoin [1] (represents public chains) and Hyperledger Fabric [15] (represents consortium chains, later referred to as Fabric). Since the relay chain receives TXs from connected chains, and it needs to verify all the incoming cross-chain TXs by TX verification mechanisms, the relay chain has some unique features on TX arrival and TX processing, as shown in Table 1:

1) The TXs on the relay chain come in batches with uncertain sizes and contain the block headers. In a typical workflow, when a connected chain generates a new block, a batch including the cross-chain TXs from this block will be constructed and then sent to the relay chain. Since the number of cross-chain TXs in a block of the connected chain is uncertain, so is the batch size. Moreover, the block headers of connected chain are needed: when a batch is being constructed, the block header should also be included. SPV [1] is the most popular TX verification mechanism that relay chains employ, which only requires block headers instead of whole blocks to verify cross-chain TXs. The block headers should also be delivered since the relay chain verifies cross-chain TXs by SPV. However, on single chains like Bitcoin and Hyperledger Fabric, TXs are triggered by users randomly and separately, and no block header needs to be included.
2) Regarding processing TXs, relay chains are featured by packaging relay chain blocks with some specific packaging rules, widely adopting BFT-type protocols, and disciplines about SPV. In a typical SPV workflow, for different batches, the relay chain verifies cross-chain TXs and block headers in FCFS (i.e., FIFO) discipline; in a single batch, since the block header is needed for verifying cross-chain TXs, the relay chain verifies the block header prior to cross-chain TXs. This specific sequence stems from SPV on relay chains, which does not apply to single chains. Besides, in packaging rules, relay chains differ from Bitcoin and Fabric, as depicted in Table 1. Taking Cosmos [12] for instance, we propose a set of packaging rules for modeling relay chains. In terms of consensus protocols, relay chains ([12, 14]) widely adopt BFT [26] type protocols. In contrast, Bitcoin adopts Proof-of-Work (PoW) [1], and Fabric adopts crash fault tolerance (CFT) [27] type protocols.

Since single chains represented by Bitcoin and Fabric differ significantly from relay chains, the performance analysis models constructed for them cannot be applied to relay chains. Performance analysis models for relay chains are still absent, which motivates us to do the work in this paper. We leave a further introduction to the existing single-chain models in section 2.

1.3 Contributions

This paper is the first to propose a performance analysis model for relay chains, which is meaningful for guiding the configuration of relay chains. Our main contributions are as follows:

1) We propose a performance analysis framework for the relay chain for the first time. We use a batch arrival and bulk service ($M^X/G^B/1$) model in queuing theory to model critical features of relay chains. To model the arrival of TXs on relay chains, we introduce four traffic parameters: the number of connected chains, batch intervals on the relay chain, the proportion of cross-chain TXs, and block sizes of connected chains. To model TX processing on relay chains, we introduce two design parameters: the block size of the relay chain and consensus duration, and adopt the supplementary variable technique to accurately describe the packaging rules. We deduct the recursive relationship of the variable to depict queuing dynamics. Furthermore, the uncertain-size batches, the main feature of relay chain TX arrival, make the modeling of this variable very complicated. Lastly, the fact that SPV demands a specific TX processing sequence adds to the complexity of modeling. To solve this, we introduce a simplification method that successfully depicts the sequence and simplifies the problem without introducing priority. Through the above modeling work, we give two necessary performance measurements of relay chains: system throughput and cross-chain TX delay.
2) This paper presents the calculation of the model when the relay chain adopts PBFT consensus, giving a concrete computational form of the system throughput and cross-chain TX delay. We model the duration of PBFT as a kind of phase-type distribution, which makes the service process not a Poisson process anymore. This gives a more accurate model but makes the calculation more complicated. To cope with this complexity, we develop a method based on Residue Theorem and Rouche's Theorem in complex analysis and successfully calculate the concrete forms of system throughput and cross-chain TX delay.
3) We verify our model by simulations and give an analysis to guide the relay chain configuration. With our

Blockchains	Features				Model
	TX arrival	On-chain SPV	Packaging rules $\langle \mathscr{A}, \mathscr{B}, \mathscr{C}, \mathscr{D} \rangle$	Consensus protocol	
Single chain (Bitcoin)	One by one	Not in need	$\langle \checkmark, \checkmark, \times, \checkmark \rangle$	PoW	e.g., [16–21]
Single chain (Fabric)	One by one	Not in need	$\langle \times, \times, \checkmark, \times \rangle$	CFT type	e.g., [22–25]
Relay chains	In uncertain-size batches	In need	$\langle \times, \checkmark, \times, \times \rangle$	BFT type	**No model**

$\mathscr{A}$: Whether to package a new block when the TX pool is empty. $\mathscr{B}$: Whether the packaging is working-conserving.
$\mathscr{C}$: Whether to package blocks in pipeline. $\mathscr{D}$: Whether the packaged blocks are accessible to newly-arrived TXs.

TABLE 1: Features of relay chains compared with single chains.

model, we can eliminate the overload state by decreasing the number of connected chains or enlarging the relay chain block when the relay chain is overloaded. When the relay chain is in a steady state, we can decrease the cross-chain TX delay to a designated level or connect to more chains with acceptable performance loss.

The remainder of this paper is organized as follows. Section 2 discusses related work. Section 3 outlines the relay chain system and introduces three phases of the workflow of a relay chain. Section 4 presents and calculates the batch arrival and bulk service model while employing PBFT protocol. Section 5 verifies the models through simulations and gives a simple analysis of the relay chain configuration. Finally, Section 6 summarizes the paper.

2 RELATED WORK

Regarding the relay chain performance analysis, we are unaware of any published theoretical modeling. Therefore, this section summarizes the current work on single-chain performance models. Bitcoin [1] and Fabric [15] are famous blockchains on which most performance modeling work cast effort. Therefore, the subsequent illustration will be carried out in three categories: performance models of Bitcoin [1], Fabric, and other blockchain protocols. All existing single-chain models we find do not meet the features of batch-arrival and SPV, which are significant in the relay chain scenario. Besides, most models focus on Bitcoin category and Fabric category, which are hardly to be used in the relay chain. The packaging rules in PBFT models can neither satisfy the relay chain scenario.

Bitcoin models: Kasahara *et al.* use the queuing theory to establish a $M/G^B/1$ model with priority, which analyzes the effect of TX fees on TX confirmation delays in the Bitcoin network in [28] and [17]. The packaging rules in these two works differ. In [28], a partially filled block that is being mined is accessible to newly-arrived TXs, while it is non-accessible in [17]. However, both the calculation results do not match the experiment results well. They believe that it may clog the simulation experiment system when the block is small. In addition, the calculation in them is partially incomplete and some results are hard to obtain directly. In [16], an accessible model is adopted and a new framework is developed. The generation of blocks is divided into two service processes: mining and broadcasting, with a two-stage service model in queuing theory to describe. However, Li *et al.* claim in [16] that only when the broadcasting stage of the former block ends, the mining stage on a new block begins. But in the actual Bitcoin network, they are performed in parallel. In [18], the mathematical innovation in the modeling of blockchains using queuing theory is emphasized. This paper presents a new calculation method for the TX confirmation delay, based on [16]. In [21], the models, $M_{NET+BC}/Pa/v$ and $M_{NET+BC}/Pa/1$, are established to analyze blockchain's workflow in a real network environment for Bitcoin and Ethereum. However, in [21], numerous parameters are derived from measurements, e.g., the waiting delay of TXs in the TX pool, and the TX confirmation delay. The model just combines the measurement outcomes, which makes it hard to predict and guide system design without abundant historical data. Mišić *et al.* present a model to fix this problem and analyze Bitcoin delivery network's delay in [19]. In [20], a model for analyzing the TX fees and security of Bitcoin and Ethereum is presented. The model focuses on the miner's strategies, which has little relationship with the system's performance.

Fabric models: Fabric contains several parts, execution, sorting, and verification part. Different from the relay chain, they are usually pipelined. Their models on each part are usually simplified, without consideration on the bulk-service queuing. Therefore, how to package blocks is no longer described precisely. Xu *et al.* [25] model the three parts sequentially. During the sorting part, they discuss PBFT and solo mode, with a simple $M/M/1$ to represent the first stage of PBFT and a constant delay to approximate the remaining stages. The modeling is relatively crude and experiments are conducted using open-source code for Fabric. Nevertheless, PBFT consensus is no longer supported in Fabric, of which the major version was updated many years ago. In this condition, Wu *et al.* [23] provides a model for a newer version of Fabric, Fabric 2.0, and verifies it. Meng *et al.* [22] claim they are the first to develop a theoretical framework of consortium blockchains. Taking Fabric for instance, Meng *et al.* study Fabric's theoretical performance model and conclude that it is inaccurate to characterize all stages with Poisson processes. In this paper, they model the execution, sorting, and verification phases with $M/H_2/1$, $M/M/1$, and $M/E_r/1$ models. However, the model developed for the ordering phase of Fabric is only suitable for solo and raft consensus, which is rarely used for other protocols.

Other models: Besides Bitcoin and Fabric, some models focus on blockchains of PBFT protocol and PoS protocol, and

1. Before August 2022, Ethereum [2] also used PoW consensus, so some models in the Bitcoin category also focus on Ethereum

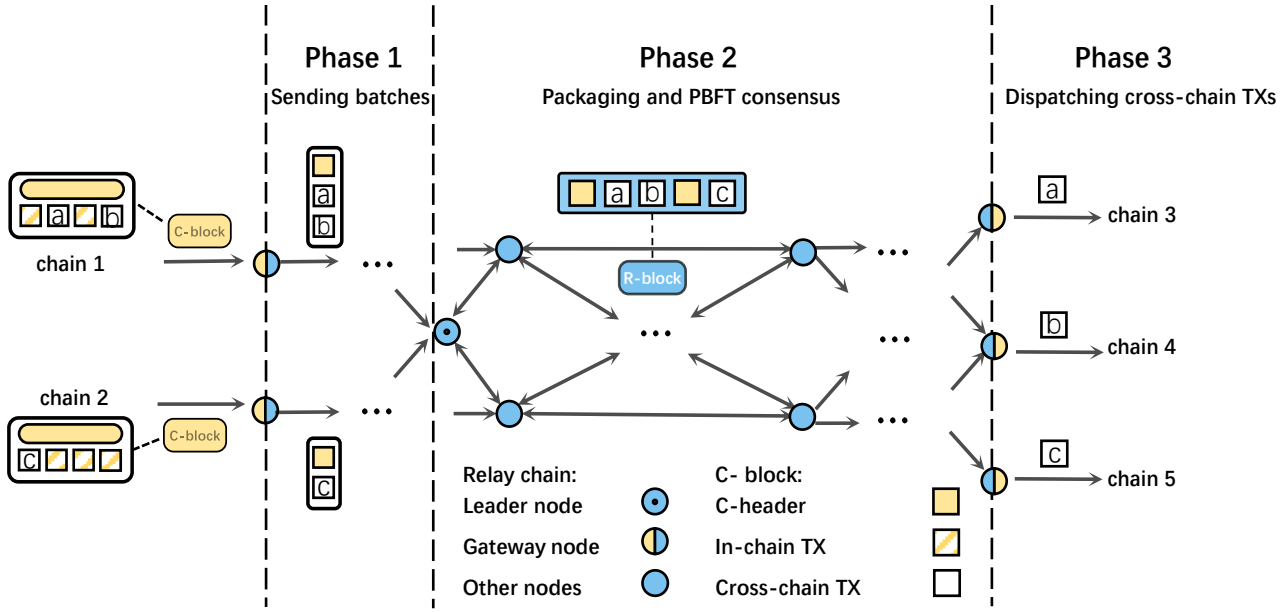


Fig. 2: The whole architecture of the relay chain system.

some develop general models. [29] goes into details about the PBFT consensus. The model provides the theoretical relationship between the time-cost of PBFT and the number of nodes size using a QBD process, without consideration of packaging blocks. Besides, though this paper claims the total time cost of the three stages of PBFT should be a sub-exponential distribution rather than a Poisson process, they fail to adopt the statement because of its complexity and simplify it as a Poisson process. Ma *et al.* [30] propose a new consensus mechanism that combines DPoS and PBFT together, and then analyze it based on [29]. Similar to [29], the same statement and simplification also occur in [31]. In [31], a complex PBFT-based blockchain is studied with dynamic nodes and the rolling-back protocol. However, the PBFT in [31] is pipelined and non-work-conserving. [25] also involves modeling of PBFT, but it is simple and inaccurate with an $M/M/1$. [32] is a general performance model for blockchains, which emphasizes the broadcasting of blocks on wireless networks. In [33], Geissler *et al.* build a discrete-time model for blockchains with Proof-of-Authority (PoA) protocol.

In conclusion, the guidance of existing work on our research is limited, adding to the difficulty and significance of this paper. We build a model in this paper to capture the relay chain's features. Besides, we adopt the statement stems from [29, 31] which makes the model of PBFT more precise.

3 The Relay Chain

In this section, we first give an overview of the entire system. Moreover, we explain it in detail to illustrate how the relay chain works.

The relay chain system is a cross-chain system that delivers and verifies TXs among connected chains. Taking Fig 2 for instance, two connected chains have requirements to deliver TXs to three other connected chains. Currently, chains 1 and 2 are source chains, while chains 3, 4 and 5 are target chains. In a block of source-connected chains (C-block), there is a block header (C-header), cross-chain TXs that should be sent to target chains, and in-chain TXs that do not involve other chains. The relay chain, the core facility of this system, helps source chains to deliver cross-chain TXs to target chains and verify them by SPV during PBFT in relay chain blocks (R-blocks). On the relay chain, there are some gateway nodes, which also have accounts on some connected chains. They keep monitoring the chains they connect to and deliver TXs.

The whole workflow can be divided into three phases. In phase 1, when detecting a new C-block on a source chain, the gateway nodes send cross-chain TXs and the C-header of it as a batch to the relay chain nodes. In phase 2, after receiving cross-chain TXs in batches, the leader of the relay chain nodes is responsible for packaging them in blocks, starting PBFT protocol, and verifying them via SPV. When the consensus ends, the leader moves to package a new block. And the gateway node forwards the cross-chain TX to the target chain in phase 3. Finally, the cross-chain TXs will be executed in the target chain.

3.1 Phase 1: Sending batches

Here we clarify how and what the relay chain gets from the TX batch after the gateway node detects a new block. In view of relay chain, it is a batch arrival process with the following two features.

Feature 1: The batch size is uncertain. Each time the gateway node notices a C-block, it selects cross-chain TXs in it. After that, it forwards the cross-chain TXs as a batch to the relay chain. Because the quantity of cross-chain TX in each block is uncertain, the cross-chain TX are delivered to the relay chain in uncertain-size batches.

Feature 2: The header is placed in the front of the batch, even though cross-TXs don't exist. Since the relay chain is unified for verifying cross-chain TXs' existence in source chains via SPV, the C-header of each C-block, and cross-chain TXs together with their proofs, should be delivered

in the batch. For the verification of the relay chain, the C-header will be placed before cross-chan TXs in their batch. Moreover, the C-header should still be delivered if no cross-chan TX exists in a block. It will be used to check whether the consequent headers can link to it. Besides, a cross-chain TX with its proof will be collectively regarded as one cross-chain TX in the relay chain. We will not specifically mention the proof in the subsequent discussion

For instance, in Fig.2, if two cross-chan TXs, a and b, exist in the newest C-block of chain 1, this TX batch contains one C-header at the front and two cross-chain TXs at the back. For the block from chain 2, its batch includes one header and one cross-chan TX, c. The relay chain receives cross-chain TXs and block headers from each chain in their batches.

Notably, feature 2 shows our simplification to describe the specific sequence SPV needs. When assembling a batch, there are usually two approaches. One is that the gateway will not deliberately put the header in the front of the batch, but because of the header's higher priority, the relay chain leader selects the header and places it before its cross-chain TXs in R-blocks. The other is that the gateway node has already put the header in the front when assembling the batch, and the leader only needs to package the TXs in order. These two approaches will not affect the modeling result when we ignore the time cost of sorting TXs. For simplifying modeling, we use the second for description. Currently, the first TX in each batch is always the block header, and there is also FCFS discipline in each batch. Therefore, we don't need to introduce priority additionally for the subsequent modeling.

After assembling the TX batch, gateway nodes deliver it to the relay chain nodes according to their transfer strategies. We move to introduce how the relay chain nodes handle the TX batch.

3.2 Phase 2: Packaging and PBFT consensus

In this phase, the leader of the relay chain nodes packages them as blocks and verifies them by SPV during PBFT consensus. In this section, we introduce the entire procedure first and then illustrate the two essential steps in-depth, which determine the process as a bulk-service process from the perspective of the relay chain.

The entire workflow of the relay chain is as follows. After receiving batches of cross-chain TXs and headers, the relay chain nodes store them in the pool. The leader of the nodes has to package a block from the TX pool and start a round of PBFT among all relay chain nodes to verify it. When a node realizes PBFT ends, it updates its local state according to the block, e.g., storing the C-headers of source chains. Then the leader moves to start a new round of PBFT, and some gateway nodes need to transfer the cross-chain TX to the target chain to start phase 3. However, the time spent storing batches and updating the local state are generally ignored. To summarize, two essential steps in this procedure seriously affect the relay chain's performance, 1) packaging TXs into blocks and 2) PBFT consensus. The packaging step determines how the leader packages TXs to become R-blocks. Moreover, the consensus step describes how the relay chain nodes verify the TXs in blocks via SPV.

Step 1: Packaging TXs into blocks. According to [12], the relay chain leader obeys the following rules while packaging:

The Serial Rule:

Only after the previous block has completed its PBFT consensus, the new block begins to be packaged. In other words, the classic PBFT referred to Cosmos [12] is serial and non-pipelined.

The Fetching Rules:

1. The leader packs TXs from the TX pool according to the FIFO (FCFS) discipline.

2. When the leader begins to package a R-block, it inspects the TX pool. If there are TXs more than the R-block size, it fills up a block; if the Txs are less than the R-block size but not empty, it packages all existing TXs without waiting for new batches; for the exception, if the pool is empty, the leader will wait until a new batch arrives, which means no empty R-block will generate[2].

The Work-conserving Rule:

The leader, together with other nodes, keeps working when TXs exist in the pool. To be speific, when the relay chain is idle, the leader returns to package a new R-block and starts PBFT as soon as a new batch arrives.

The Non-accessible Service Rule:

The ongoing consensus block is non-accessible for TX batches. The TX batch that enters the TX pool during the current consensus round can only be packaged in subsequent R-blocks, even though the current block is not full.

Step 2: PBFT consensus. In this step, the leader broadcasts the block to other nodes to start PBFT consensus. For TXs, this is a bulk-service process. Generally speaking, PBFT contains three stages, as shown in Fig.3, and includes complex three-stage communication and verification.

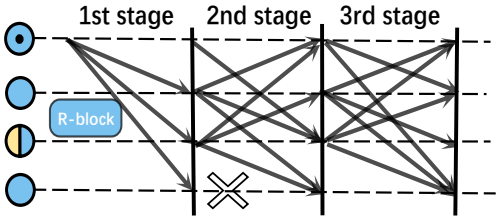


Fig. 3: PBFT consensus on the relay chain.

The 1st stage: The leader broadcasts the R-block to all other relay chain nodes to verify it. According to our simplification in Feature 2 of phase 1, the nodes only need to verify the cross-chain TXs and C-headers in turn by SPV.

The 2nd stage: If a node considers the block qualified, it broadcasts its vote to all other nodes. When a node collects 2/3 of votes from all nodes, it enters the third stage.

The 3rd stage: The nodes have to accomplish a similar vote procedure as the second stage. The block finally

2. This is a significant difference compared with other famous consensuses in blockchains, e.g., PoW of Bitcoin. Bitcoin will generate blocks even the pool is empty because of block generation rewards.

passes all three stages if a node collects 2/3 of the votes in this stage.

It should be noted that either a C-header or a cross-chain TX is a common TX that should be verified and packaged by the relay chain. For C-headers, the node verifies their compliance with the generation rules and then links them to former C-headers stored in the smart contract, if applicable. For cross-chain TXs, the node verifies whether they belong to the C-block containing the front C-header. Taking the R-block in Fig.2 for instance, the nodes first check whether the C-header can link to other C-headers in its contract. After that, they verify cross-chain TX, a, b, according to their header. For the latter header and TX c, they repeat the verification.

Whenever a node detects that the block passes all three stages, the consensus finishes, and the node updates its local state according to the block. Especially, for the leader node, the workflow on this block ends, and a new block will be packaged; for a gateway node, it may start phase 3.

3.3 Phase 3: Dispatching cross-chain TXs

In phase 3, cross-chain TXs are dispatched to their target chains and processed. This phase has little relationship with the relay chain. Therefore, it is only illustrated briefly here. When a gateway node inspects the PBFT on an R-block finish, it selects cross-chain TXs from the block and analyzes the target chain. For the cross-chain TXs, which aim to target chains it has accounts on, it dispatches them according to its strategy. After being delivered to the target chain, cross-chain TXs are processed as normal in-chain TXs in the target chain.

4 Batch-arrival and bulk-service model

In this section, we focus on building a batch arrival and bulk service model for the relay chain as above and then supplement the approach in [34] to solve the model. Consequently, the computation expressions of two measures are displayed.

To be specific, in sections 4.1 and 4.2, we model how the relay chain receives TXs in batches and then handles them in bulks (R-blocks). Section 4.3 supplements a significant variable, the queue length when a round of PBFT ends. Based on it, section 4.4 gives the waiting time TX batches spend in the pool. In sections 4.5 and 4.6, expressions of the metrics, the relay chain's average throughput, and the cross-chain TX' average delay are illustrated respectively. The following table, Table 2, provides a summary of notations used throughout this paper.

4.1 Batch arrival

Batch arrival refers to the arrival of TXs in batches at leader nodes' TX pool on relay chain. To describe the batch arrival process mathematically, two aspects should be modeled: 1) the number of TXs in a batch and 2) the arrival of a TX batch. From now on, TXs refer to relay chain TXs that carry any block header or cross-chain TX from C-blocks. Each TX carries one block header or one cross-chain TX.

1) the number of TXs in the batch:

Input parameters	
α	The average interval of batches from a connected chain.
N	The number of connected chains.
θ	The proportion of cross-chain TXs
M_1	The C-block size.
M_2	The R-block size.
$\beta = \beta_1 + \beta_2 + \beta_3$	The mean duration of the three-stage PBFT.
Random variables and other notations	
ρ	The load intensity of the relay chain.
X	The number of TXs in a batch.
H_t	The number of batches that arrive in $[0, t]$.
Y	The number of TXs an R-block contains.
T	The duration of a round of PBFT
K_n	Number of TXs arrive during the nth round of PBFT.
Z_{n+1}	Number of TXs left in the pool when the nth R-block leaves.
W	The waiting time of a batch in the pool.
Q_j	The consensus time for j TXs in the pool before packaged.
S	The residue time of the ongoing PBFT.
ϵ	The probability that the relay chain is idle.
π_j	The probability that j TXs wait in the pool.
R	The elapsed time of the ongoing PBFT.
τ_j^R	The probability of j TXs waiting when R happens.
$\eta(\sigma, t)$	The probability the ongoing PBFT lasting for σ ends in t.
κ_j	The number of R-blocks that j TXs in the pool will use.
β'	The duration of PBFT including possible idle gaps.
W_q	The average waiting time of a batch before packaged
W_s	The total waiting time of a C-header in the relay chain.
L_q	Average number of TXs in the pool.
L_s	Average number of TXs in the relay chain.
L_b	Average number of TXs in the R-block when it exists.
TPS_{max}	The maximum theoretical throughput of the relay chain.
$f_X, F_X, G_X, \widehat{F}_X$	PDF, CPF, GF and LST of the variable, X.
Output measures	
W_{C-CTX}	The average delay of a cross-chain TX in the relay chain.
TPS_{RC}	The average throughput of the relay chain.

TABLE 2: Summary of notations

Let X denote the amount of TXs in an arrival batch. According to Feature 2 in section 2.1, we have,

$$X \in \{1, 2, ..., M_1 + 1\},$$

where M_1 denotes the C-block size. We have the following probability massing function (PMF) according to Feature 1 for $i = 1, 2, ..., M_1 + 1$,

$$Pr(X = i) = C_{i-1}^{M_1} \theta^{i-1} (1 - \theta)^{M_1 - i + 1}. \quad (1)$$

θ represents the proportion of cross-chain TXs in C-blocks. The exponent of θ is $i - 1$ because the i Txs in a batch must include one C-header and all the others are cross-chain TXs. Furthermore, we can derive generating function (GF) of X,

$$G_X(p) = \sum_{j=1}^{M_1+1} Pr(X = i) p^i, \quad for \quad |p| \leq 1.$$

2) The arrival of the TX batch:

Assume that there are N connected chains in the system, and TX batches from each chain obey Poisson arrival with $\frac{1}{\alpha}$.

Let H_t denote a random variable representing the number of batches arriving in a period $[0, t]$, and it is obvious that,

$$H_t \in \{0, 1, 2, ...\}.$$

We have the PMF of H_t,

$$Pr(H_t = i) = \frac{(Nt/\alpha)^i}{i!} e^{-Nt/\alpha}, i = 0, 1, 2..... \quad (2)$$

We adopt the common assumption used in blockchain modeling that batches arrive with a Poisson distribution. The block generation rules of different connected chains and the forwarding strategies of different gateway nodes vary. It is challenging to give a unifying distribution to replace the Poisson distribution.

4.2 Bulk service

The bulk service reflects the PBFT step in Phase 2. The leader starts PBFT consensus to process the TXs in blocks, which represents a batch serive process in queueing theory. This section models two aspects, the service capacity and the service time.

The service capacity is represented by the block size of the relay chain. Let us assume that all R-blocks have the same size, M_2. For the reason that the generating function form is needed for our further computation, we introduce a random variable, $Y = M_2$, with a constant value. The generating function of Y is as follows,

$$G_Y(p) = p^{M_2}, \quad for \quad |p| \leq 1$$

The service time is represented by the time each round of PBFT consensus takes. According to [22, 25], in an ideal model for PBFT consensus, each stage in it should be a negative exponential distribution. Here we adopt this assumption.

Let us introduce a random variable T to represent the total time spent on a round of PBFT,

$$T = T_1 + T_2 + T_3,$$

where T_1, T_2, T_3 are random variables to denote the time spent on each stage of PBFT, and they are all negative exponential distributions. Therefore, T is a kind of phase-type distribution, the three-phase sub-exponential distribution. Let us define its probability density function (PDF) to be $f_T(t)$, $t \geq 0$,

$$f_T(t) = \frac{e^{-t\frac{1}{\beta_3}} \frac{1}{\beta_1}\frac{1}{\beta_2}\frac{1}{\beta_3}}{(\frac{1}{\beta_1} - \frac{1}{\beta_3})(\frac{1}{\beta_2} - \frac{1}{\beta_3})} + \frac{e^{-t\frac{1}{\beta_1}} \frac{1}{\beta_1}\frac{1}{\beta_2}\frac{1}{\beta_3}}{(\frac{1}{\beta_2} - \frac{1}{\beta_1})(\frac{1}{\beta_3} - \frac{1}{\beta_1})} + \frac{e^{-t\frac{1}{\beta_2}} \frac{1}{\beta_1}\frac{1}{\beta_2}\frac{1}{\beta_3}}{(\frac{1}{\beta_1} - \frac{1}{\beta_2})(\frac{1}{\beta_3} - \frac{1}{\beta_2})}.$$

Therefore, we can derive its cumulative distribution function (CDF), $F_T(t)$ for $t \geq 0$,

$$F_T(t) = P(T \leq t) = \int_0^t f_T(x)\mathrm{d}x.$$

Assume that means of T_1, T_2, T_3 are $\beta_1, \beta_2, \beta_3$, we are able to get the mean of T,

$$\beta = \beta_1 + \beta_2 + \beta_3.$$

The Laplace-Stieltjes transform (LST) of $f_T(t)$ is necessary for the consequent modeling, defined as $\widehat{F_T}(s)$,

$$\widehat{F}_T(s) = \int_0^\infty e^{-st}\mathrm{d}F_T(t) = \frac{\beta_3^2}{(\beta_3 - \beta_1)(\beta_3 - \beta_2)} \frac{1}{\beta_3 s + 1} + \frac{\beta_2^2}{(\beta_2 - \beta_1)(\beta_2 - \beta_3)} \frac{1}{\beta_2 s + 1} + \frac{\beta_1^2}{(\beta_1 - \beta_3)(\beta_1 - \beta_2)} \frac{1}{\beta_1 s + 1}, \quad (3)$$

$$\widehat{F_T}(0) = 1, \widehat{F_T}(0) = -\beta.$$

Notably, though [25, 31] claim that the service time in PBFT consensus should be a sub-exponential distribution, they fail to adopt the assumption but simplify it to be a simple negative exponential distribution. We will introduce a way to model it without simplication in our consequent analysis.

Besides, the three-phase sub-exponential distribution utilized in this paper is highly interchangeable. For instance, in a particular scenario, the BFT-type protocol used by the relay chain may take a long time to update local state or select a new leader, which cannot be disregarded compared to PBFT in [12]. Therefore, a four or five-phase model is more suitable, and it can replace the three-phase sub-exponential model employed in this study. Likewise, the calculation's complexity will increase proportionally. It's the same thing when the relay chain needs a two-stage BFT-type consensus with a two-phase model.

4.3 Queue length at the post-bulk-service completion

The queue length at the post-bulk-service completion is a radom variable refers to the number of TXs left in the leader's TXs pool at the moment a PBFT round ends. It is essential for us to compute the distribution of the consensus's elapsed time with various queue lengths in order to derive the delay in section 3.6.

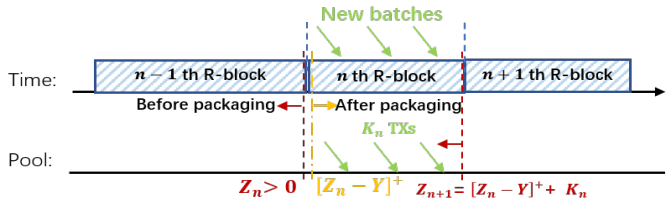
Let us introduce a random variable, $Z_{n+1}, n = 1, 2, ...,$ to represent this queue length when the nth round of PBFT ends. According to the packaging rules we introduce in section 2.2, we have the following expression,

$$Z_{n+1} = \begin{cases} [Z_n - Y]^+ + K_n, & if \quad Z_n > 0, \\ [X - Y]^+ + K_n, & if \quad Z_n = 0, \end{cases} \quad (4)$$

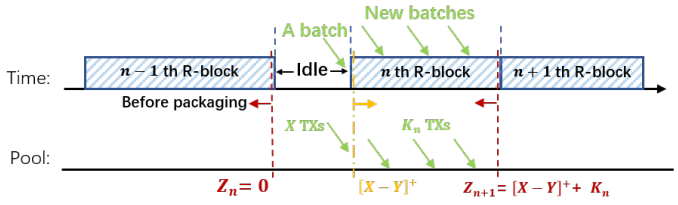
where K_n denotes the total number of TXs arriving during the nth round of consensus. Let us assume that $M_1 < M_2$, the Equation 4 still holds for $n = 1$ if we let $Z_1 = 0$. We will first introduce how the packaging rules in section 2.2 lead to Equation 4, and then calculate K_n.

According to the Serial Rule, Z_{n+1} can be determined by Z_n.Then we have the following two conditions, as depicted in Fig.4. The line above is the timeline, while the line blow describes the queue length in the leader's TX pool.

- If $Z_n > 0$, $[Z_n - Y]^+$ represents the number of TXs in the pool at the moment the nth R-block begins to be packaged. Specifically, if $z_n \geq Y$, at this moment, the remaining TXs in the pool are more than what the nth R-block can pack. According to the Fetching Rules, $z_n - g_{2,n}$ TXs remain in the pool after packaging. On the contrary, when $Z_n < Y$, the nth block packages all TXs and none of them is left. By now what



(a) When $Z_n > 0$.



(b) When $Z_n = 0$.

Fig. 4: The length of the pool when a round of PBFT ends.

$[z_n - g_{2,n}]^+$ means have been illustrated. Meanwhile, during the consensus process of the nth R-block, K_n TXs, arrive. Due to the Non-accessible Service Rule, K_n can only be packed in subsequent blocks. Therefore, according to the Work-conserving Rule, we have $Z_{n+1} = [Z_n - Y]^+ + K_n$ for $Z_n > 0$.

- If $Z_n = 0$, the pool is empty, which means no TXs are waiting to be packaged. The relay chain is idle. According to the fetching rule that no-empty-block generates, when a new TX batch arrives, the leader will package the nth block, which means at the moment the nth R-block is packaged, $[X - Y]^+$ TXs remain. For the same reason, because of the Non-accessible Service Rule, K_n is added. In a word, we have $Z_{n+1} = [X - Y]^+ + K_n$ for $Z_n = 0$.

Define the generating function (GF) of Z_{n+1} as $G_Z(p)$ when $n \to \infty$ and $Z_1 = 0$, we have

$$G_Z(p) = \lim_{n\to\infty} \sum_{j=0}^{\infty} P(z_{n+1} = j | z_1 = 0) p^j, j = 0, 1, 2, ...,$$

The concrete form of $G_Z(p)$ is introduced later (Theorem 1).

In the nth round of PBFT with a duration of t, H_t batches arrive and each batch includes X TXs. Let us assume that each batch includes $X_1, X_2, ..., X_{H_t}$ TXs, we have,

$$K_n = \sum_{i=1}^{H_t} X_i.$$

According to Equation 1 and 2, for $j = 0, 1, ..., K_n$'s PMF is,

$$Pr(K_n = j) = \int_0^{\infty} \sum_{i=0}^{\infty} Pr(H_t = i) \left[Pr\left(\sum_{l=1}^{H_t} X_l = j \right) \right] \mathrm{d}F_T(t).$$

According to Equation 3, we have the GF of K_n, for $|p| \leq 1$,

$$\begin{aligned} G_{K_n}(p) &= \sum_{j=0}^{\infty} Pr(K_n = j) p^j \\ &= N\widehat{F}_T(1 - G_X(p))/\alpha \\ &= \frac{\beta_3^2}{(\beta_3 - \beta_1)(\beta_3 - \beta_2)} \frac{1}{N\beta_3(1 - G_X(p))/\alpha + 1} \\ &+ \frac{\beta_2^2}{(\beta_2 - \beta_1)(\beta_2 - \beta_3)} \frac{1}{N\beta_2(1 - G_X(p))/\alpha + 1} \\ &+ \frac{\beta_1^2}{(\beta_1 - \beta_3)(\beta_1 - \beta_2)} \frac{1}{N\beta_1(1 - G_X(p))/\alpha + 1}. \end{aligned}$$

Moreover, when $n \to \infty$, we give $G_K(p) = G_{K_n}(p)$.

4.4 The waiting time of a batch in the TX pool

The waiting time of the batch in the TX pool refers to the interval between the moment of its arrival at the TX pool, and the moment at which its C-header begins to be packaged (notice that C-header is at the head of the batch).As shown in Fig.5, the ongoing R-block when the batch arrives is denoted as 0th and its C-header will be packaged in the $n + 1$th. In this section, we model this waiting time, and calculate its mean. The analysis in this section holds when the steady state exists.

4.4.1 *The random variable of the waiting time*

Let us introduce a random variable, W, to denote this waiting time.

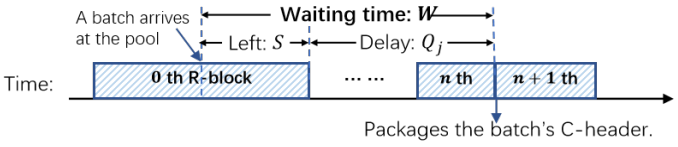


Fig. 5: The waiting time includes S and Q_j.

Let S denote the random variable that describes the consensus time residue for the ongoing consensus, and Q_j denote the time j TXs in the pool will take for consensus, before the newly-arriving batch's C-header gets packaged. As shown in Fig.5, W is the sum of Q_j and S. Assume that $\pi_j, j = 0, 1, 2, ...$ denote the steady-state distribution of waiting time of the j TXs in the pool, we have the equation,

$$W = \begin{cases} S + Q_0, & w.p. \quad \pi_0, \\ S + Q_1, & w.p. \quad \pi_1, \\ ... & ... \quad ... \\ S + Q_j, & w.p. \quad \pi_j, \\ ... & ... \quad ... \end{cases}$$

We need the LST of W to compute its mean. It has the following expression,

$$\widehat{F}_W(s) = \int_{0-}^{\infty} e^{-st} \mathrm{d}F_W(t), \mathrm{Re}\, s > 0,$$

$$P(W \leq t) = F_W(t) = \epsilon + (1 - \epsilon) \sum_{j=0}^{\infty} \left[(\pi_j F_S) * F_{Q_j} \right](t), \quad (5)$$

where ϵ refers to the steady-state probability that the TX pool is empty, f_s and f_{Q_j} are PDFs of S and Q_j, and the $*$ means the convolution of them.

Expression of ϵ:

According to [34], ϵ has a much more complicated form than usual in an $M^X/G^B/1$ model, which can be expressed as follows,

$$\epsilon = \frac{1}{1+N\beta E(m)/\alpha},$$
$$m = min(n : Z_{n+1} = 0, n = 1, 2, ...), for \quad Z_1 = 0,$$

where $E(m)$ is the mean of m and [34] gives its proof. The concrete form of ϵ is introduced later (Theorem 2).

Expression of $\pi_j F_S$:

Let us define a random variable, $R, R \geq 0$, to be the elapsed consensus time of the ongoing consensus, we have the following relationship for $R \leq T$,

$$S = T - R.$$

$Pr(j|\sigma)$ represents the probability that j TXs are waiting in the pool when the ongoing PBFT has elapsed σ. Then $\pi_j F_S$ can be expressed as follows,

$$\pi_j F_S(t) = \int_0^\infty f_R(\sigma) \cdot Pr(j|\sigma) \cdot Pr(T - R \leq t | R \leq T) \mathrm{d}\sigma. \quad (6)$$

f_R is the PDF of R,

$$f_R(\sigma) = \frac{1}{E(T)} \cdot (1 - F_T(\sigma)) = (1 - F_T(\sigma))/\beta.$$

$Pr(j|\sigma)$ is based on Z_n, and its expression is also similar to Z_{n+1},

$$Pr(j|\sigma) = \lim_{n\to\infty} Pr([Z_n - Y]^+ + K_\sigma = j, Z_n > 0) + Pr([X - Y]^+ + K_\sigma = j, Z_n = 0).$$

We do not repeat the proof which is the same as Equation 4. To be specific, K_σ refers to the number of TXs arriving during $[0, \sigma]$, and its expression and CDF are also similar to K_n,

$$K_\sigma = \sum_{i=1}^{H_\sigma} X_i.$$

$$Pr(K_\sigma = j) = \int_0^R \sum_{i=0}^{\infty} Pr(H_\sigma = i) \cdot Pr\left(\sum_{l=1}^{H_\sigma} X_l = j\right) \mathrm{d}F_T(t).$$

Therefore, the GF of K_σ is,

$$G_{K_\sigma} = \exp\left\{[G_X(p) - 1] \cdot \sigma/\alpha\right\}, |p| \leq 1.$$

According to Equation 4, we have,

$$\lim_{n\to\infty} \sum_{i=0}^{\infty} p^i Pr([Z_n - Y]^+ = i) + Pr([X - Y]^+ = i) = \frac{G_Z(p)}{G_K(p)}.$$

Therefore, according to [34], the sum of $Pr(j|\sigma)$ can be derived directly, which is useful in Equation 5,

$$\sum_{j=0}^{\infty} Pr(j|\sigma) = \frac{G_Z(1) \cdot \exp\left\{[G_X(1) - 1]\, \sigma/\alpha\right\}}{G_K(1)}. \quad (7)$$

$Pr(T - R \leq t | R \leq T)$ is the steady-state probability that an ongoing consensus that has lasted for R will end in t. It has the form,

$$Pr(T - R \leq t | R \leq T) = \frac{F_T(t + R) - F_T(R)}{1 - F_T(R)}. \quad (8)$$

Expression of F_{Q_j}:

Let us assume that $\kappa_j, j = 0, 1, 2, ...$ is the number of rounds that the c-header of the newly-arriving batch needs to wait before getting packaged when j TXs are waiting in the pool. Q_j is determined by the total time of κ_j rounds of PBFT. We have the CDF of Q_j,

$$P(Q_j \leq t) = F_{Q_j}(t) = F_T^{\kappa_j *}(t), \quad (9)$$

where $F_T^{\kappa_j *}$ is the κ_j-fold convolution of F_T itself, and

$$\kappa_j = \lfloor \frac{j}{Y} \rfloor, \quad (10)$$

$\lfloor \frac{j}{Y} \rfloor$ means the floor of $\frac{j}{Y}$. In other words, the C-header of the newly-arriving batch will be packaged in the $\lfloor \frac{j}{Y} \rfloor + 1$th round of PBFT.

Expression of $\widehat{F}_W(s)$:

According to [34], we are able to derive the expression of $\widehat{F}_W(s)$ combining Equations 6, 7, 9, 10 and 8,

$$\widehat{F}_W(s) = \epsilon + \frac{1-\epsilon}{2\pi i} \int_D \frac{G_Z(1/\xi)}{\xi(1-\xi)} \frac{1 - G_Y(\xi)}{1 - \widehat{F}_T(s) G_Y(\xi)} \frac{\widehat{F}_T(s) - G_K(1/\xi)}{(1 - G_X(1/\xi))\beta/\alpha - \beta s} \frac{\mathrm{d}\xi}{G_K(1/\xi)}, \quad (11)$$

where D is the unit circle at the origin and the integration is counterclockwise.

4.4.2 The mean of W

Let us define the mean of W to be W_q, which can be expressed as,

$$W_q = \int_0^\infty t \mathrm{d}F_W = \frac{\mathrm{d}\widehat{F}_W(s)}{\mathrm{d}s}\Big|_{s=0}.$$

However, though we derive the expression of $\widehat{F}_W(s)$ now, it is challenging to calculate W_q for the following two reasons. 1) Some functions in the expression of $\widehat{F}_W(s)$ lack expressions, e.g., G_Z and ϵ. Only the definitions are introduced. 2) The integral on the complex field cannot be directly computed, in other words, we can not derive the numerical result of $\widehat{F}_W(s)$ now. To fix the problems and obtain W_q, we give Lemma 1 and three following Theorems which are based on it. Theorems 1 and 2 give the numerical results of G_Z and ϵ, and Theorem 3 gives the approach to derive W_q. Proofs of the Lemma and Theorems are in the appendix of the supplementary file.

Lemma 1. Suppose $p \in \mathbb{C}$ and $|r| \leq 1$, the equation

$$G_Y(p) - rG_K(p) = 0$$

has M_2 zeros inside the unit circle, $\mu_i(r), j = 1, 2, ..., M_2$, as well as $3(M_1 + 1)$ zeros outside the unit circle, $\mu_i(r), i = M_2 + 1, M_2 + 2, ..., 3(M_1 + 1) + M_2$.

Theorem 1. For $p \in \mathbb{C}$, $G_Z(p)$ has the following relationship with $\mu_i(r), i = 1, 2, ..., 3M_1 + M_2$,

$$G_Z(p) = \Gamma(G_X(p))G_K(p)\Pi_{i=M_2+1}^{3(M_1+1)+M_2}\frac{1-\mu_i(r)}{p-\mu_i(r)},$$

where for $l = 1, 2, 3$,

$$\begin{aligned}\Gamma(G_X(p)) =&(b_1(1 - G_X((p)) + 1)\cdot \\ &(b_2(1 - G_X((p)) + 1) \cdot (b_3(1 - G_X((p)) + 1),\end{aligned}$$

$$a_l = \frac{\beta_l^2}{(\beta_l - \beta_{l+1})(\beta_l - \beta_{l+2})}, b_l = N\beta_l/\alpha, \beta_4 = \beta_1, \beta_5 = \beta_2.$$

Theorem 2. Suppose $\mathrm{Re}\, s > 0$,, ϵ has the expression,

$$\epsilon = \frac{\alpha\Gamma(\Phi_X(\infty))\Phi_K(\infty)\Pi_{i=M_2+1}^{3(M_1+1)+M_2}(\mu_i(1)-1)/\mu_i(1)}{\alpha\Gamma(\Phi_X(\infty))\Phi_K(\infty)\Pi_{j=M_2+1}^{3(M_1+1)+M_2}(\mu_i(1)-1)/\mu_i(1) + \beta N},$$

where

$$\Phi_X(s) = G_X(e^{-s}), \Phi_K(s) = G_K(e^{-s}).$$

Theorem 3. Define $\xi_i = \mu_i(r)^{-1}, i = M_2 + 1, ..., 3(M_1 + 1) + M_2$,

$$W_q = \left[\epsilon + (1-\epsilon)\sum_{i=M_2+1}^{3(M_1+1)+M_2} I(D, \xi_i)Res(\Omega(s, \xi), \xi_i)\right]_{s=0}^{'}, \tag{12}$$

where $I(D, \xi_i)$ and $Res(\Omega(s, \xi)$ are common notations in the Residue Theorem, and $\Omega(s, \xi)$ is,

$$\Omega(s,\xi) = \frac{1}{2\pi i}\frac{G_Z(1/\xi)}{\xi(1-\xi)}\frac{1-G_Y(\xi)}{1-\widehat{F}_T(s)G_Y(\xi)}\frac{\widehat{F}_T(s)-G_K(1/\xi)}{(1-G_X(1/\xi))\beta/\alpha-\beta s}\frac{1}{G_K(1/\xi)}. \tag{13}$$

Combining Theorem 1, 2 and 3 together, we are able to derive the numerical expression W_q.

However, as we have mentioned, W_q exists only when the steady state exists. Let us define the load intensity of the relay chain, ρ, to judge whether the steady state exists,

$$\rho = \frac{N\beta \cdot (\theta M_1 + 1)}{\alpha M_2}. \tag{14}$$

When $\rho > 1$ happens, the relay chain is overloaded and the steady state doesn't exist, $W_q \to \infty$. W_q can be computed if and only if $\rho \leq 1$ happens.

4.5 System throughput

The system throughput refers to the number of TXs (including C-headers and cross-chain TXs) the relay chain processes per unit of time. We will derive the mean of the throughput in a steady state via W_q.

Let us define the average throughput of the relay chain as TPS_{RC}, it has the following expression,

$$TPS_{RC} = \frac{L_b}{\beta'}. \tag{15}$$

L_b (Equation 17) denotes the average number of TXs contained in the ongoing R-block when the relay chain is not idle. β' (Equation 16) denotes the time of a round of consensus including the possible idle gap .

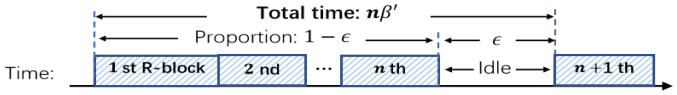


Fig. 6: Numerical relations between blocks and their gaps.

Expression of β':

As depicted in 6, when the idle gap exists, no TX exists in the pool and the relay chain is idle. Therefore, In a β'-long round, the consensus takes up $1 - \epsilon$, while the gap occupies the left ϵ. In a word, β' is,

$$\beta' = \frac{n \cdot \beta}{1-\epsilon} \cdot \frac{1}{n}. \tag{16}$$

Expression of L_b:

Let us assume that when a batch arrives at the pool, the average number of TXs waiting in the pool is L_q, and the total number of TXs on the relay chain (which include TXs both in the pool and in the current block being processed) is L_s. We have the expression,

$$L_s - L_q = \epsilon \cdot 0 + (1-\epsilon) \cdot L_b$$

Due to the Little's Law, L_q and L_s can be easily derived from W_q,

$$L_q = W_q \cdot \frac{N(\theta M_1 + 1)}{\alpha}.$$

$$L_s = (W_q + \beta) \cdot \frac{N(\theta M_1 + 1)}{\alpha}).$$

The queue length of the pool is related to the waiting time in the pool, while the total number of TXs is related to the service time.

Therefore, we have,

$$L_b = \frac{L_s - L_q}{1-\epsilon} = \frac{\beta N(\theta M_1 + 1)}{\alpha(1-\epsilon)}. \tag{17}$$

With Equation 15, TPS_{RC} has the final expression,

$$TPS_{RC} = \frac{N(\theta M_1 + 1)}{\alpha}. \tag{18}$$

Notably, the final expression of TPS_{RC} is only related to our input parameters, N, θ, M_1, α. In other words, when the steady state is lost and W_q does not exist, TPS_{RC} still has a result. However, its meaning varies. It turns to expressing the throughput ability the relay chain needs to reach in order to recover the relay chain to a steady state.

Let us define the theoretical maximum throughput of the relay chain to be TPS_{max}. TPS_{max} occurs when the R-blocks are full, and the relay chain is never idle, which means $\epsilon = 0$,

$$TPS_{max} = \frac{M_2}{\beta}. \tag{19}$$

According to Equations 14, 18 and 19, when $\rho > 1$, it is obvious that $TPS_{RC} > TPS_{max}$, which means the relay chain's theoretical actual throughput exceeds the relay chain's maximum throughput. It is impossible and the relay chain is overloaded. We need to adjust the relay chain parameters to make $TPS_{RC} \leq TPS_{max}$, and then $\rho \leq 1$ is achieved. That's because the relay chain's actual throughput covers the ability it needs.

4.6 Cross-chain TX delay

A cross-chain TX delay refers to the interval between the cross-chain TX arrival into the TX pool in its batch and the moment the block containing it finishes consensus, as shown in Fig.7. We drive the mean of the cross-chain TXs' delay in a steady state via W_q, with an approximate method.

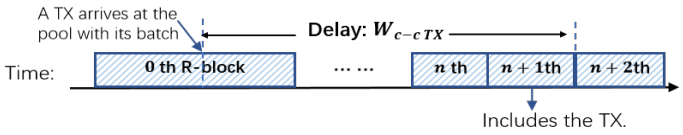


Fig. 7: A cross-chain TX's delay.

Let us define W_{C-CTX} to represent the mean of the delay. For the reason that $M_2 > M_1$, the TX batch containing the cross-chain TX which we focus on will be packaged into at most two continuous R-blocks. We calculate W_{C-CTX} by estimating the probabilities of which R-block the cross-chain TX will be packaged in. We have the following result,

$$W_{C-CTX} \approx \begin{cases} W_q + \beta, & if\ \zeta_1 = \zeta_2, \\ \lambda_1 \cdot (W_q + \beta) + \lambda_2 \cdot (W_q + 2\beta), & if\ \zeta_1 < \zeta_2. \end{cases} \tag{20}$$

λ_1, λ_2 are defined in Equation 22. ζ_1 means that the C-header of the TX batch, which contains the cross-chain TX we focus on, will be packaged in the ζ_1th block after the ongoing block being processed, and the last TX in the batch will be included in the ζ_2th block. As shown in Fig.8 and Fig.9, we have,

$$\zeta_1 = \lceil \frac{1 + L_q}{M_2} \rceil, \quad \zeta_2 = \lceil \frac{\theta M_1 + 1 + L_q}{M_2} \rceil. \tag{21}$$

If $\zeta_1 = \zeta_2$: the entire TX batch is packaged in one R-block.

As Fig.8 shows, $\zeta_1 = \zeta_2$ means every TX in the batch can be packaged in one R-block on average. In other words, no matter where a cross-chain TX locates in the batch, it will wait for W_q to be packed and wait for $W_{C-CTX} \approx W_q + \beta$ to finish PBFT consensus.

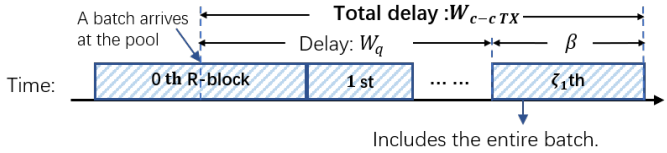


Fig. 8: The entire batch is contained in one block.

If $\zeta_1 < \zeta_2$: the entire TX batch is packaged in two R-blocks.

In Fig.9, when $\zeta_1 < \zeta_2$, it is estimated some TXs at the head of the batch and the other TXs will be packaged in two R-blocks. Besides, according to $M_1 < M_2$, we have $\zeta_2 = \zeta_1 + 1$.

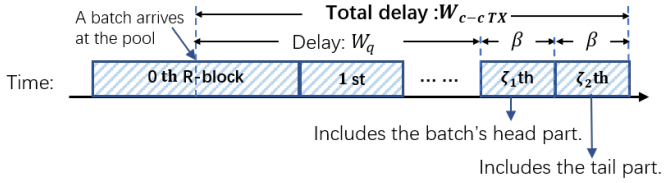


Fig. 9: The entire batch is contained in two blocks.

Let us define λ_1, λ_2 to be the probabilities that the cross-chain TX lies in the head part, and the tail part of the TX batch. We estimate it as follows,

$$\begin{aligned} \lambda_1 &= \frac{M_2 - [(L_q + 1) mod M_2]}{\theta \cdot M_1}, \\ \lambda_2 &= 1 - \lambda_1. \end{aligned} \tag{22}$$

The head part is processed in the ζ_1 R-block, which waits for W_q to be packaged and $W_q + \beta$ to finish PBFT; the tail part is packaged in the next (ζ_2) R-block, which waits for $W_q + \beta$ to be packaged and $W_q + 2\beta$ to finish PBFT. Therefore, we have the equation in Equation 20 when $\zeta_1 < \zeta_2$.

And Equation 20 has the final expression,

$$W_{C-CTX} \approx \begin{cases} W_q + \beta, & \lceil \frac{1+L_q}{M_2} \rceil = \lceil \frac{\theta M_1 + 1 + L_q}{M_2} \rceil, \\ W_q + 2\beta - \frac{M_2 - (L_q+1) mod M_2}{\theta M_1}\beta, & \lceil \frac{1+L_q}{M_2} \rceil < \lceil \frac{\theta M_1 + 1 + L_q}{M_2} \rceil. \end{cases}$$

Although we get the exact form of W_q, the expression of W_{C-CTX} is estimated. However, in the following numerical simulation experiments, we will show that it is accurate enough.

Notably, when the steady state is lost, W_{C-CTX} does not exist at all. With the relay chain's continuous operation, the delay will theoretically increase infinitely. The approach to the numerical result of W_q only returns NaN, which differs from our approach to TPS_{RC}.

5 Model Verification and Analysis

In this section, we perform a numerical simulation of the relay chain with PBFT. After comparing the results from the model and simulation under the same parameters, there are two advantages. On the one hand, the model solution's accuracy can be verified. On the other hand, similar to [16, 18, 29], by observing the numerical results of the curve, we can perform a preliminary analysis and guide the relay chain's configuration.

Based on Python, we employ stochastic processes to perform the relay chain's features of TX batches' arrival and bulk service described in section III. As mentioned previously, the cross-chain TXs' delay exists when the system is in a steady state ($\rho \leq 1$). Therefore, this section will first demonstrate briefly through numerical simulation that when $\rho > 1$, the system is not in a steady state (section 5.1), so there is no theoretical delay of cross-chain TXs, and the relay chain operates at nearly full capacity. We send an average of 10,000 TX batches to the relay chain, then compute the average cross-chain TX delay and system throughput in units for every 100 batches. We repeat this process ten times and display their results mean. Secondly, this section performs six experiments, and one parameter varies within a reasonable range in each set of experimentsge. Two design parameters vary in sections 5.2 and 5.3: the consensus time-cost of the relay chain, the R-block size, and four traffic parameters vary in sections 5.4-5.7: the C-block size, the cross-chain TX proportion, the number of connected chains, and the interval between TX batches' arrival. To make the simulation experiments reflect the steady state as realistic as possible, we send between 10,000 and 15,000 TX batches to the relay chain on average. Then, we compute the final 20% cross-chain TXs' delay, as well as the actual throughput of the relay chain during this period. We perform ten times for each set of parameters and then display their mean.

5.1 Simulation on whether the steady state exists

In this section, we make the number of connected chains vary and other parameters fixed. We use the following parameters: setting the average interval of TX batches to 15s, the cross-chain TX proportion to 5%, and the three stages of PBFT to a sub-exponential distribution with values of 0.3s, 0.7s, and 1.5s, respectively. An R-block size is 50 TXs, while a C-block contains 30. In the meantime, the number of chains connected to the system varies, which determines whether the relay chain system is steady. The number of connected chains is set to 50, 100, 150, or 200, and the corresponding ρ values are 0.417, 0.833, 1.25, and 1.67.

Verification: As depicted in Fig.10, the model can accurately tell whether the relay chain is steady. When $\rho = 0.417, 0.833$, the delay experienced by the cross-chain TX in the relay chain is stable and always at a low level because the system has a steady state. After multiple batches, the delay gradually converges to a fixed value instead of increasing erratically. However, when $\rho = 1.25, 1.67$, although the relay chain has already accepted 10,000 batches, the delay gradually increases to 70s and 100s without converging. As the system continues to operate, the delay will grow indefinitely. The system's state is not steady. Regarding the throughput, TPS_{max} is always 20 in these four conditions. If $\rho = 0.417, 0.833$, the relay chain does not reach the maximum throughput most of the time. When $\rho = 1.25, 1.67$, the actual measured throughput is 18 TXs on average. Sometimes, the maximum throughput is reached or even exceeded. The relay chain is close to its maximum processing capacity.

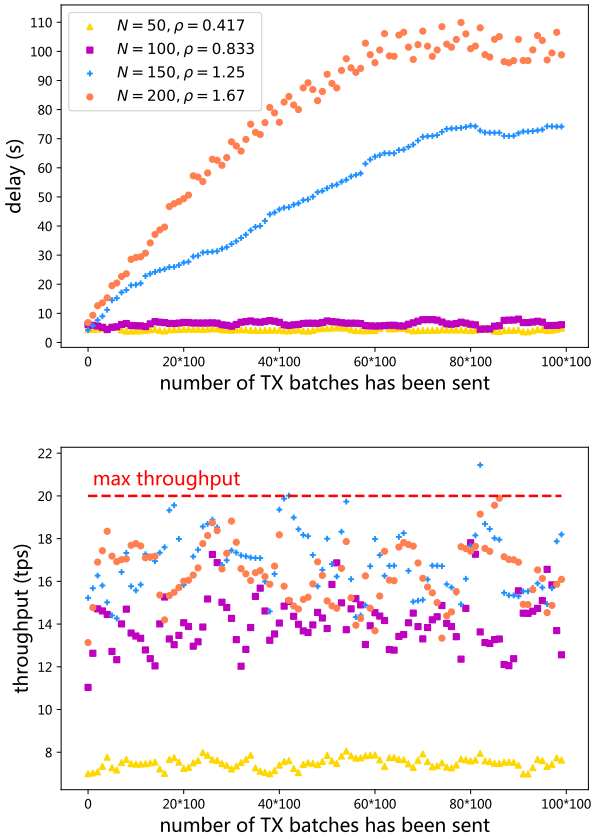


Fig. 10: Relay chain performance with different ρ s.

Analysis The model can be used to guide the configuration. For instance, losing a steady state is undoubtedly the first thing that should be avoided when setting up a relay chain. With the other parameters fixed, the number of connected chains should be limited to 100. However, $N = 50$ or 100 has little influence on the delay, only affecting R-blocks utilization.

5.2 Only the duration of consensus varies

In general, the duration of PBFT consensus is primarily influenced by the number of nodes and the behavior of malicious nodes. Predicting the behavior of malicious nodes is difficult. Consequently, people typically control the PBFT duration by adjusting the number of nodes. How to adjust the number of nodes to change the time-spent on PBFT can be found in [29, 31]. According to the PBFT duration discussed in [35], we range β from 0.5 to 5s. Other parameters are shown in Table 3.

Verification: From Fig.11 to 16, the abscissa represents how the variable parameter changes while the main coordinate axes of the two subfigures represent the average delay of cross-chain TXs and the system's throughput under

M_1	N	α	θ	M_2	$(\beta_1, \beta_2, \beta_3)$
30	20	15	25%	50	$variable$

$(\beta_1, \beta_2, \beta_3)$				
(0.06,0.14,0.3)	(0.12,0.28,0.6)	(0.18,0.42,0.9)	(0.24,0.56,1.2)	(0.3,0.7,1.5)
(0.36,0.84,1.8)	(0.42,0.98,2.1)	(0.48,1.12,2.4)	(0.54,1.26,2.7)	(0.6,1.4,3)

TABLE 3: Parameters when the duration of consensus varies.

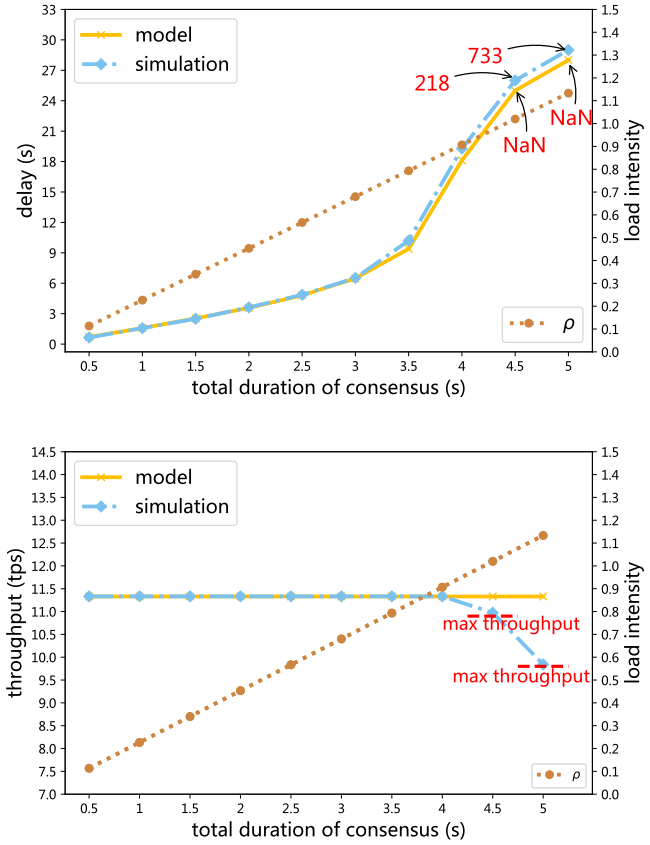


Fig. 11: Relay chain performance when the duration of consensus varies.

the aforementioned parameters in their tables. Model curve and simulation curves represent the results of the model and numerical simulations. The two subfigures have the same meaning for the coordinate axis, which is the values of the load intensity ρ under the range of the abscissa, as depicted by the curve ρ. In this section, when the consensus duration varies, as indicated in Table 3, the model and the numerical simulation results results are highly congruent, demonstrating the model's accuracy. After $\beta = 4.5$ and 5, the relay chain has lost its steady state. Although the total time-consuming consensus only increases by 0.5s and 1s compared with $\beta = 4$, the delay measured by the simulation rapidly increases from 18s to 218s and 733s, and the throughput from the simulation reaches its max throughput (denoted as red "max throughput" in Fig.11). In the simulation, only 10,000 batches were transmitted. If we continuously send batches, the delay will increase indefinitely without a theoretical expectation, which is denoted by NaN.

Analysis: Here we consider the numerical results of the curve. When β increases but is smaller than 3s, the delay of cross-chain TXs increases slightly, and the throughput maintains. Increasing the number of relay chain nodes will not increase the delay significantly. When $\beta \geq 3.5$, the delay increases rapidly until $\rho > 1$. Then the relay chain loses its steady state. To restore the steady state and decrease the delay, we need to make the max throughput exceed the model curve for $\beta = 4.5$ and 5, e.g., enlarging the R-block size to at least 51 and 57. Then we have $TPS_{RC} < TPS_{max}$ and $\rho \leq 1$.

5.3 Only the R-block size varies

M_1	N	α	θ	M_2	$(\beta_1, \beta_2, \beta_3)$
30	20	15	25%	$variable$	(0.3,0.7,1.5)

M_2										
50	55	60	65	70	75	80	85	90	95	100

TABLE 4: Parameters when the R-block size varies.

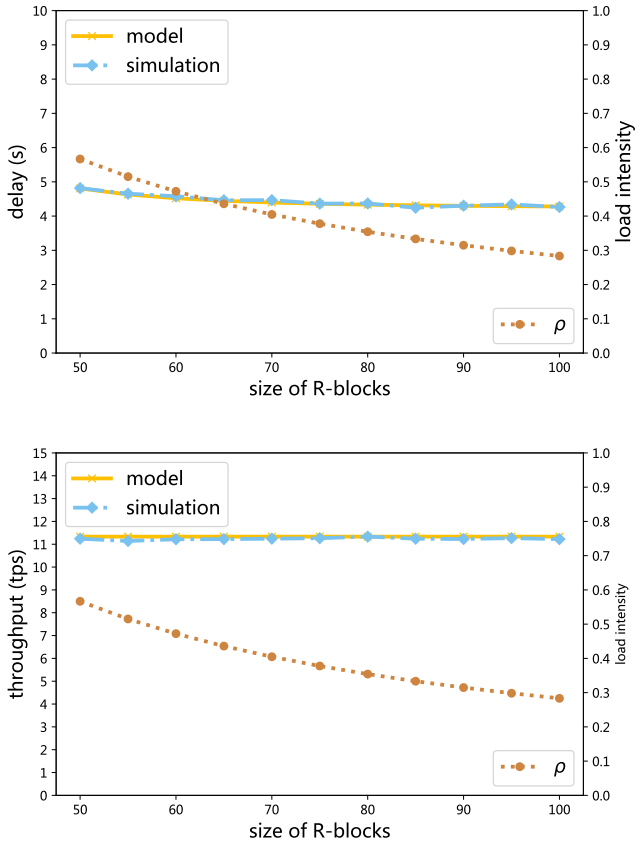


Fig. 12: Relay chain performance when the R-block size varies.

Verification: The R-block size plays a crucial role in the relay chain's performance. Abscissa axes in Fig.12 and Table 4

represent the R-block size, which ranges from 50 to 100, and the model calculation results closely match the simulation results.

Analysis: Now, we concentrate on the delay and throughput. The delay of cross-chain TXs in the relay chain decreases rapidly when the relay chain's load is high. At this time, increasing the R-block size can significantly decrease the delay. However, the decline becomes slight when the overall load intensity is low. Moreover, by observing the throughput of the relay chain, we can determine that its steady-state throughput is lower than its maximum throughput. Adjusting the R-block size does not impact the actual throughput currently.

5.4 Only the cross-chain TX proportion varies

Verification: In this group, we discussed the model results and numerical simulations conducted when the proportion of cross-chain TXs varies in the range of Table 5. The abscissa in Fig.13 is the logarithmic coordinate of the proportion of cross-chain TXs, and the calculated results correspond well to the simulation measurement outcomes.

M_1	N	α	θ	M_2	$(\beta_1, \beta_2, \beta_3)$
30	10	15	$variable$	50	(0.3,0.7,1.5)

θ							
0.1%	1%	5%	10%	25%	50%	75%	100%

TABLE 5: Parameters when the cross-chain proportion varies.

Analysis: When the cross-chain TX proportion is deficient, ρ and the relay chain load are extremely low. Even if the proportion of cross-chain TXs has an exponential increment, the delay and the throughput will not be affected significantly. We do not need to adjust the configuration, and most TXs in the R-blocks and the headers from connected chains. Nonetheless, as the proportion increases, the relay chain's load intensity increases significantly and gradually loses its steady state. When θ reaches 100%, the cross-chain TX delay from the model does not have a limit, marked by NaN. In contrast, the simulation result reaches 261s, increased by two orders of magnitude when there is only a 25% increase in the proportion compared to $\theta = 75\%$. This is the consequence of the demise of the steady state. In terms of the throughput, when $\theta = 100\%$, the throughput measured by the numerical simulation reaches the theoretical maximum, and the throughput of the model solution slightly exceeds TPS_{max}. We can limit the duration of PBFT to at most 2.42s or enlarge the R-block size to at least 52 to restore the steady state. Besides, reducing the number of connected chains is also available.

5.5 Only the C-block size varies

Verification: As depicted in Fig.14 and Table 6, when the C-block size varies, the model results are close to the numerical simulation results, proving that the calculation results are sufficiently accurate to describe the delay and throughput.

Analysis: Although the two measurements will increase as the C-block size increases, because ρ is currently small,

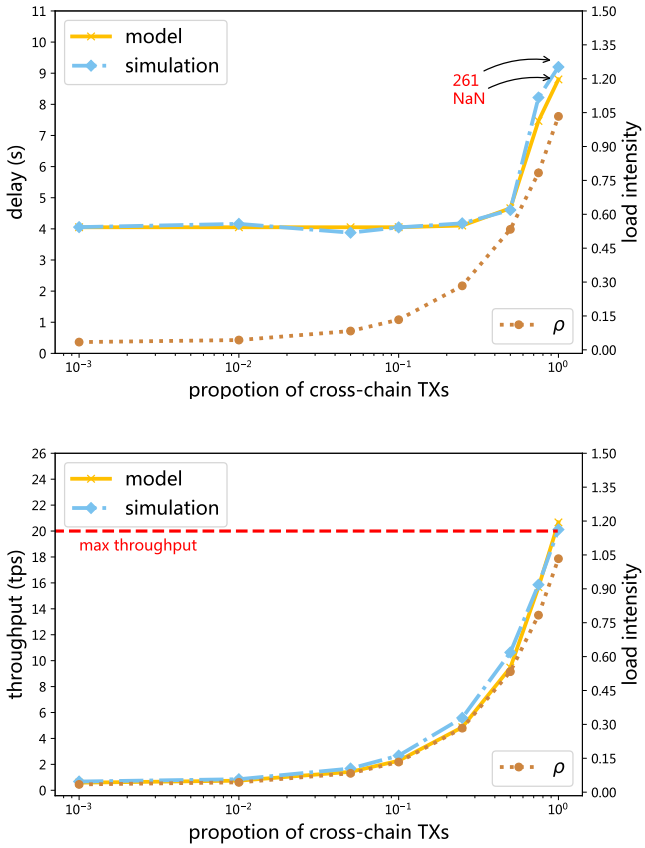


Fig. 13: Relay chain performance when the cross-chain TX proportion varies.

M_1	N	α	θ	M_2	$(\beta_1, \beta_2, \beta_3)$
$variable$	20	15	25%	50	(0.3,0.7,1.5)

M_1								
10	15	20	25	30	35	40	45	50

TABLE 6: Parameters when the C-block size varies.

the average delay of cross-chain TXs will increase only marginally. As ρ approaches 1, the throughput gradually approaches the maximum throughput $\phi_{max} = \frac{M_2}{\beta} = 20tps$. The delay increases significantly. To decrease the delay, when C-block size is more than 40, we had better enlarge the R-block size or cut down the number of connected chains.

5.6 Only the number of connected chains varies

M_1	N	α	θ	M_2	$(\beta_1, \beta_2, \beta_3)$
30	$variable$	15	5%	50	(0.3,0.7,1.5)

N								
2	5	10	20	50	75	100	150	200

TABLE 7: Parameters when the number of connected chains varies.

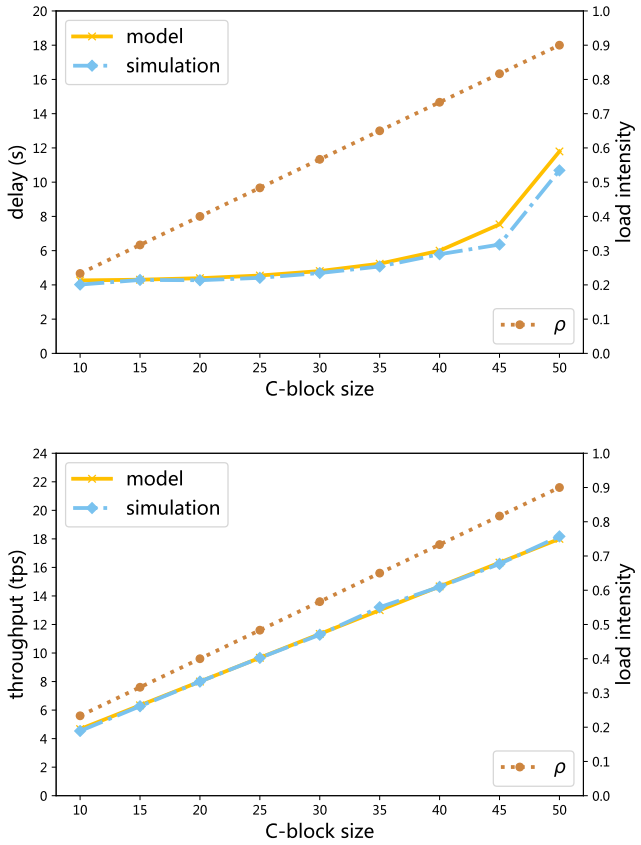


Fig. 14: Relay chain performance when the C-block size varies.

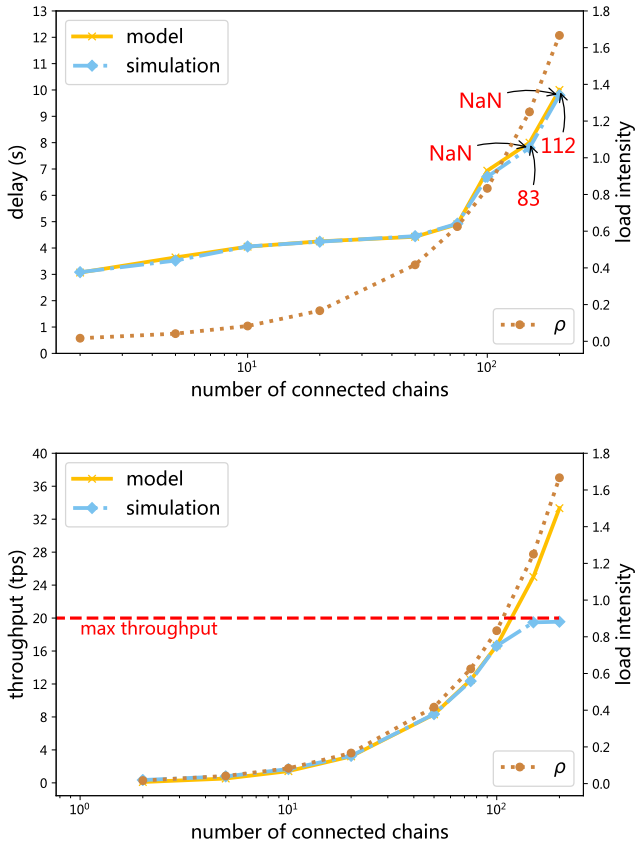


Fig. 15: Relay chain performance when the number of connected chains varies.

Verification: In this series of experiments, the number of chains connected to the relay chain is varied from the minimum of two required for the cross-chain system to a maximum of two hundred, as shown in Table 7. Fig.15's abscissa axis depicts the number of chains in the logarithmic coordinate, and the model calculation results are in excellent agreement with the simulation program.

Analysis: Observing the curve, as the number of connected chains increases, the steady state of the relay chain is gradually destroyed. The cross-chain TX delay and the throughput of the relay chain increase rapidly until they lose their steady state. When the theoretical delay does not exist, as indicated by NaN, simulation results rapidly increase from 6.71s to 83s and 112s. The throughput measured by numerical simulation is close to the theoretical maximum throughput, whereas the throughput of the model calculation significantly exceeds the maximum throughput. Under the current configuration, the relay chain's maximum throughput is significantly lower than the throughput necessary to restore its steady state. We need to increase the throughput to 25 and 34 by adjusting parameters.

5.7 Only the interval of TX batches varies

Verification: In this group, we assume that the arrival interval of TX batches varies in the range of Table 8. When we assume that the gateway node forwards TX batches immediately, the interval between batches' arrival can be regarded as the block generation interval of the connected chains. Therefore, referring to the actual block interval of some famous blockchains, α ranges from 5s for Tendermint [12], 15s for Ethereum [2] to 600s for Bitcoin [1].

M_1	N	α	θ	M_2	$(\beta_1, \beta_2, \beta_3)$
30	20	$variable$	25%	50	(0.3,0.7,1.5)

α							
5	10	15	30	60	120	300	600

TABLE 8: Parameters when the interval of TX batches varies.

Analysis: As depicted in Fig.16, as the interval between TX batches gradually increases, the load on the relay chain gradually decreases, and the steady state gradually returns to a small ρ. When the load intensity of the relay chain is high, and the steady state does not exist, the expectation of cross-chain TX delay does not exist either, marked as NaN. The simulation measures a ridiculously long delay of 731s. The actual throughput measured by the simulation reaches and even exceeds the maximum theoretical throughput by 1tps. The model predicts a throughput of 34tps, significantly higher than the current maximum throughput of 20tps. To restore the steady state, we should either increase the block size or decrease the time required for each consensus round on the relay chain.

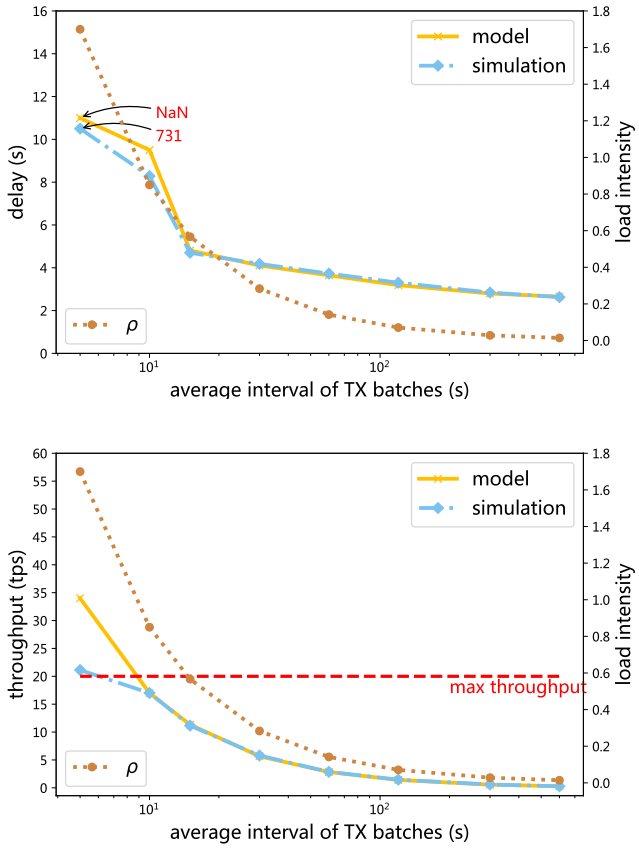


Fig. 16: Relay chain performance when the interval of TX batches varies.

6 Conclusion and Future Work

This paper develops a performance analysis framework for the relay chain and has a concrete calculation when the relay chain adopts PBFT consensus. This paper models the batch arrival feature and the bulk service feature, considering four traffic parameters: the interval of TX batches, the number of connected chains, the size of connected chain blocks, the percentage of cross-chain TXs; and two design features: the size of R-blocks, and the relay chain's consensus time-duration. This paper constructs a $M[x]/G^B/1$ model in queuing theory to evaluate the performance of the relay chain. Here, we express two measures: 1) the average delay of cross-chain TXs from arriving at the leader of the relay chain to achieving agreement on the relay chain; and 2) the throughput of the relay chain. This paper then establishes a numerical simulation experiment using Python to verify the accuracy of the calculation results. Lastly, based on the shown numerical calculation results, this paper analyzes and demonstrates how the model can guide the configuration briefly.

Nonetheless, the performance analysis of the relay chain system has many unexplored avenues, and we intend to conduct additional research in the future.

- Although as many variables as possible are considered in this research, some simplifications are made. For instance, we simplify each C-block has the same size and follows the same Poisson distribution to join the relay chain TX pool. This is not always the case, however, in practical applications. Future research may concentrate on developing a mathematical model to describe the arrival of TX batches at the relay chain in the presence of multiple distinct connected chains and forwarding strategies.
- This model analyzes only the relay chain performance instead of the end-to-end delay of the whole relay chain mode. As mentioned in the introduction, the performance of the relay chain is analyzed because it is the most important factor in determining the relay chain system's performance. Therefore, it is interesting to apply the research results of this paper to the delay and throughput analysis of the entire relay chain mode.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (2021YFB2700404); National Natural Science Foundation of China (U22B2032, 61972382); Science and Technology Development Fund, Macau SAR (File No. 0076/2022/A2); Natural Science Foundation of Inner Mongolia (2020MS06017); ICT-SSC Blockchain Joint Lab; Hainan Zhongke Computing Blockchain Innovation Academy.

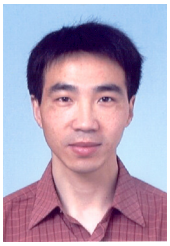
References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Business Review*, p. 21260, 2008.
[2] "Welcome to ethereum," https://ethereum.org, 2023.
[3] "Litecoin," https://litecoin.com/en/, 2023.
[4] "Binance," https://www.binance.com/en, 2023.
[5] N. N. Ahamed, P. Karthikeyan, S. Anandaraj, and R. Vignesh, "Sea food supply chain management using blockchain," in *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2020, pp. 473–476.
[6] R. Wang, "Application of blockchain technology in supply chain finance in beibu gulf region," in *2021 International Wireless Communications and Mobile Computing (IWCMC)*, 2021, pp. 1860–1864.
[7] P. Dutta, T.-M. Choi, S. Somani, and R. Butala, "Blockchain technology in supply chain operations: Applications, challenges and research opportunities," *Transportation Research Part E: Logistics and Transportation Review*, vol. 142, p. 102067, 2020.
[8] "Btc relay," http://btcrelay.org/, 2023.
[9] "Wecross," https://wecross.readthedocs.io/, 2023.
[10] M. Westerkamp and J. Eberhardt, "zkrelay: Facilitating sidechains using zksnark-based chain-relays," Cryptology ePrint Archive, Paper 2020/433, 2020, https://eprint.iacr.org/2020/433. [Online]. Available: https://eprint.iacr.org/2020/433
[11] 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*, ser. CCS '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 3003–3017. [Online]. Available: https://doi.org/10.1145/3548606.3560652

[12] "Cosmos," https://cosmos.network/, 2023.
[13] "Bitxhub," https://bitxhub.cn/, 2023.
[14] "Polkadot," https://polkadot.network/, 2023.
[15] "Hyperledger foundation," https://www.hyperledger.org/, 2023.
[16] Q.-L. Li, J.-Y. Ma, and Y.-X. Chang, "Blockchain queue theory," in *Computational Data and Social Networks*, X. Chen, A. Sen, W. W. Li, and M. T. Thai, Eds. Cham: Springer International Publishing, 2018, pp. 25–40.
[17] Y. Kawase and S. Kasahara, "Transaction-confirmation time for bitcoin: A queueing analytical approach to blockchain mechanism," in *Queueing Theory and Network Applications*, W. Yue, Q.-L. Li, S. Jin, and Z. Ma, Eds. Cham: Springer International Publishing, 2017, pp. 75–88.
[18] Q. L. Li, J. Y. Ma, Y. X. Chang, F. Q. Ma, and H. B. Yu, "Markov processes in blockchain systems," *Computational Social Networks*, vol. 6, no. 1, pp. –, 2019.
[19] J. Mišić, V. B. Mišić, X. Chang, S. G. Motlagh, and M. Z. Ali, "Modeling of bitcoin's blockchain delivery network," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 3, pp. 1368–1381, 2020.
[20] J. He, G. Zhang, J. Zhang, and R. Q. Zhang, "An economic model of blockchain: The interplay between transaction fees and security," *Big Data & Innovative Financial Technologies Research Paper Series*, 2020.
[21] A. Vladyko, A. Spirkina, and V. Elagin, "Towards practical applications in modeling blockchain system," *Future Internet*, vol. 13, p. 125, 05 2021.
[22] T. Meng, Y. Zhao, K. Wolter, and C.-Z. Xu, "On consortium blockchain consistency: A queueing network model approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 6, pp. 1369–1382, 2021.
[23] O. Wu, S. Li, L. Liu, H. Zhang, X. Zhou, and Q. Lu, "Performance modeling of hyperledger fabric 2.0," in *The International Conference on Evaluation and Assessment in Software Engineering 2022*, ser. EASE 2022. New York, NY, USA: Association for Computing Machinery, 2023, p. 357–365. [Online]. Available: https://doi.org/10.1145/3530019.3531341
[24] P. Yuan, K. Zheng, X. Xiong, K. Zhang, and L. Lei, "Performance modeling and analysis of a hyperledger-based system using gspn," *Computer Communications*, vol. 153, 2020.
[25] X. Xu, G. Sun, L. Luo, H. Cao, H. Yu, and A. V. Vasilakos, "Latency performance modeling and analysis for hyperledger fabric blockchain network," *Information Processing & Management*, vol. 58, no. 1, p. 102436, 2021.
[26] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, p. 382–401, jul 1982. [Online]. Available: https://doi.org/10.1145/357172.357176
[27] L. Lamport, "The part-time parliament," *ACM Trans. Comput. Syst.*, vol. 16, no. 2, p. 133–169, may 1998. [Online]. Available: https://doi.org/10.1145/279227.279229
[28] S. Kasahara and J. Kawahara, "Effect of bitcoin fee on transaction-confirmation process," *Journal of Industrial & Management Optimization*, vol. 13, pp. 1–22, 01 2017.
[29] F. Ma, Q. Li, Y. Liu, and Y. Chang, "Stochastic performance modeling for practical byzantine fault tolerance consensus in blockchain," *CoRR*, vol. abs/2107.00183, 2021. [Online]. Available: https://arxiv.org/abs/2107.00183
[30] F. Ma and R.-N. Fan, "Queuing theory of improved practical byzantine fault tolerant consensus," *Mathematics*, vol. 10, p. 182, 01 2022.
[31] Y.-X. Chang, Q.-L. Li, Q. Wang, and X.-S. Song, "Dynamic practical byzantine fault tolerance and its blockchain system: A large-scale markov modeling," *arXiv preprint arXiv:2210.14003*, 2022.
[32] X. Ling, Y. Le, J. Wang, Z. Ding, and X. Gao, "Practical modeling and analysis of blockchain radio access network," *IEEE Transactions on Communications*, vol. 69, no. 2, pp. 1021–1037, 2020.
[33] S. Geissler, T. Prantl, S. Lange, F. Wamser, and T. Hossfeld, "Discrete-time analysis of the blockchain distributed ledger technology," in *2019 31st International Teletraffic Congress (ITC 31)*, 2019, pp. 130–137.
[34] F. J. Beutler, "The single server queue (j. w. cohen)," *SIAM Rev.*, vol. 25, no. 4, p. 388–412, oct 1983. [Online]. Available: https://doi.org/10.1137/1025140
[35] D. Cason, E. Fynn, N. Milosevic, Z. Milosevic, E. Buchman, and F. Pedone, "The design, architecture and performance of the tendermint blockchain network," in *2021 40th International Symposium on Reliable Distributed Systems (SRDS)*, 2021, pp. 23–33.

Bo Li received his B.S. degree from the University of Science and Technology of China in 2018. He is currently a PhD student at the Institute of Computing Technology, Chinese Academy of Sciences since 2018. His field of research includes data pricing and blockchain technologies.

Tiantian Duan received her B.S. degree from Dalian University of Technology in 2017. She is currently a PhD student at the Institute of Computing Technology, Chinese Academy of Sciences since 2017. Her field of research focuses on blockchain technologies.



Qingling Zhao received his Ph.D. degree from the Institute of Computing Technology, the Chinese Academy of Sciences, Beijing, China, in 2005. From May 2005 to August 2009, he worked as a postdoctoral researcher at the Chinese University of Hong Kong and the Hong Kong University of Science and Technology. Since September 2009, he has been with the School of Computer Science and Engineering at Macau University of Science and Technology and now he is a professor. He serves as an associate editor of IEEE Transactions on Mobile Computing and IET Communications. His research interests include blockchain and decentralization computing, machine learning and its applications, Internet of Things, wireless communications and networking, cloud/fog computing, software-defined wireless networking.

Zhaoxiong Song Zhaoxiong Song received his B.S. degree from Beijing Institute of Technology in 2017 and M.S. degree from Carnegie Mellon University in 2019. He is currently an R&D engineer at Institute of Computing Technology, Chinese Academy of Sciences. His research interests include blockchain technology and distributed system.

Yi Guo received his B.S. degree from Shandong University in 2019. Currently he is a PhD student at the Institute of Computing Technology, Chinese Academy of Sciences. His research interests include cross-chain technologies and consensus algorithms.

Hanwen Zhang received her B.S. degree from the University of Electronic Science and Technology of China in 2003 and PhD degree from the Institute of Computing Technology, Chinese Academy of Sciences in 2009. She is currently an associate professor at ICT. Her research interests include computer networks and blockchain technologies.

Zhongcheng Li received his B.S. degree in computer science from Peking University in 1983, and M.S. and Ph.D. degrees from Institute of Computing Technology, Chinese Academy of Sciences in 1986 and 1991, respectively. From 1996 to 1997, he was a visiting professor at University of California at Berkeley. He is currently a professor of Institute of Computing Technology, Chinese Academy of Sciences. His research interests include the computer networks and blockchain technology.

Yi Sun received the PhD degree in computer science from the Institute of Computing Technology (ICT), Chinese Academy of Sciences in 2007. He is currently a professor at the ICT since 2015. His research interests cover blockchain, network resource management, network architecture, and video streaming. He has already published more than 60 academic papers, and received the Outstanding Young Scientist Award of the Chinese Academy of Sciences in 2014.

Performance Modeling of Blockchains of BFT-type Consensus

Bo Li *†, Chenhao Jiang *†, Hanwen Zhang*†, Zhongcheng Li*†, Yi Sun *†
*Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China.
†University of Chinese Academy of Sciences, Beijing, China
{libo18z, jiangchenhao21s, hwzhang,zcli, sunyi}@ict.ac.cn

***Abstract*—As the requirements of different application scenarios vary, numerous blockchains have been developed. Among them, chains using Byzantine Fault Tolerance consensus (BFT chains for short) occupy a dominant position. Performance modeling is the most common method for guiding chain design and configuration. However, many lack specific models due to the large number of BFT chains. Providing a general framework for them is promising and should demonstrate two important features that BFT chains may differ in: different consensus stages and different packaging rules. This paper is the first to propose a general framework applicable to various BFT chains. We use the bulk-service model in queuing theory to reflect the above two features of various BFT chains and provide calculation methods for them. We give an expression for the confirmation delay of transactions (TXs for short) in the chain and verify it through experiments on various BFT chains. We also use the model for a brief analysis to assist designers in configuring and designing BFT chains.**

***Index Terms*—BFT-type consensus, Blockchain management, Delay, Performance modeling, General framework**

I. Introduction

A. Background

With the continuous development of blockchains in various application fields, more than 50 different consensuses have been adopted to fit different scenarios. For example, Bitcoin adopts Proof-of-Work (PoW) [1], Ethereum adopts Proof-of-Stake (PoS) [2], and Cosmos [3] and Hyperledger Fabric [4] use Practical Byzantine Fault Tolerance (PBFT) [5]. Polkadot [6], a multi-chain ecosystem, adopts BABE+GRANDPA consensus.

Among them, chains using Byzantine Fault Tolerance (BFT) type consensus occupy a very mainstream position. BFT-type consensus is a class of protocols that allow nodes to reach a consensus on a block when malicious (byzantine) nodes exist. Compared to other protocols, chains tend to adopt BFT-type consensus for two reasons: 1) Many BFT-type protocols are combined with other mechanisms to meet different requirements, e.g., PBFT, Tendermint, and BABE+GRANDPA incorporate stake assets. Additionally, there are protocols such as [7, 8] that use trusted hardware and threshold signatures to improve security, and [9] that uses the verifiable random function to improve fairness, etc. 2) BFT-type consensus is often deterministic. Each block that passes through the consensus is determined and will not be changed. Compared to PoS and PoW chains [1, 2] where blocks may be invalid due to forks, BFT chains are more suitable for more scenarios.

B. Motivation

Since performance is the major constraint for chains, modeling is a common means of evaluating performance and guiding management [10–15]. However, due to the large number of BFT chains, many do not have specific models. Additionally, existing models may be inaccurate when applied to other chains. For example, a Poisson process is used to model the consensus in [13], which reduces accuracy. Models in [10, 12] only apply to some versions of Fabric. In [15], a specialized model is introduced for a dynamic PBFT (DPBFT) chain.

Modeling BFT chains respectively can be very complicated. Therefore, providing a general performance framework for them seems promising. Such a framework can support the analysis of chains that lack specific models and help designers manage chains from a performance perspective. A general framework should demonstrate versatility in the following two features where BFT chains generally differ: 1) BFT-type consensus has varying numbers of stages. Agreement in BFT-type consensus is often achieved by a node leading other nodes to vote on blocks, including the leader's election and multiple voting stages for the nodes. The number of stages varies in chains as shown in Table I. 2) BFT chains have different packaging rules to meet different requirements. Generally speaking, the rules can be divided into empty-block-allowed, full-block-only, and other rules. The detailed description of projects is shown in Table I and illustrated in Section III.

C. Contribution

This paper first proposes a general performance framework for BFT chains. We use bulk-service queuing theory to accurately model the various stages of consensus by phase-type distributions and depict the different packaging rules. We also provide some examples of suitable chains. We express the most critical indicator in blockchain performance: the confirmation delay of TXs. We verify the model's accuracy and guide the configuration management through experiments.

The remainder of this paper is organized as follows. Section II discusses related work. Section III outlines BFT chains and introduces the features. Section IV presents and calculates the bulk-service model. Section V verifies the models and

TABLE I: Features of BFT chains and existing models

Features		Projects	Models
Number of stages	2	Zyzzyva [16], Polkadot [6], Hyperchain [17]	\
	3	Fabric [4], Cosmos [3], Algorand [9], DPBFT [15]	[12], [13], [15]
	4	Zyzzyva [16]	\
	5	RBFT [18]	\
Packaging rules	Empty blocks allowed	Cosmos [3], Polkadot [6], Hyperchain [17]	\
	Full blocks only	Fabric [4], Algorand [9], DPBFT[15],	[12], [15]
	Others	Cosmos [3], Zyzzyva [16]	[13]

simply analyzes the chain's management. Finally, Section VI summarizes the paper.

II. RELATED WORK

This section has two parts: the relevant knowledge of BFT chains and the established performance models for them.

A. BFT chains

BFT chains show much difference in consensus stages and packaging rules. PBFT is the most classic BFT-type consensus, being widely used in chains. Cosmos [3] is a relay chain system using PBFT. In [3], the leader is selected through a simple algorithm. Subsequently, different packaging rules can be chosen, and blocks are sent to other nodes for three-stage voting. Fabric [4] v0.6 also uses PBFT in its ordering phase. The leader fills the block and sends it to other nodes for PBFT. Except for PBFT, many other BFT-type protocols are used. [18] divides consensus nodes into groups, and voting only occurs within each group. The entire consensus integrates PBFT and Raft, containing five stages. [16] dynamically analyzes the voting results. Depending on whether byzantine nodes exist, the consensus ends after two stages or four stages. [6] is a multi-chain solution, including two stages called BABE and GRANDPA. [15, 19] focus on ensuring the joining and exiting of nodes and rolling back orphan blocks. Only full blocks are allowed, and the consensus has three stages. In [17], there are two stages, and empty blocks are allowed.

B. Existing Models of BFT chains

Compared to the numerous modeling papers on Bitcoin and Ethereum, there are relatively fewer studies on BFT chains. A model for DPBFT is established in [15]. The model accurately depicts the feature that DPBFT adopts the full-block-only rule. In addition, the model simplifies the duration of the three-stage consensus into a Poisson process, greatly simplifying the modeling and reducing accuracy. The authors mention that introducing phase-type distribution to increase accuracy is their future work, adopted in our paper. Similarly, models and analyses exist on the relationship between the number of nodes and the consensus efficiency in [13]. This paper also uses a Poisson process to simplify the consensus and claims phase-type distribution may improve accuracy. In addition, in [13], the leader will package each TX as a block, which is different from common chains. In [10, 12], Fabric is modeled. Unlike other chains, the PBFT is just a sub-phase instead of the entire workflow in Fabric. In [12], blocks are filled before being sent out and forced to be packaged when timed out.

III. BFT CHAINS' WORKFLOW

In this section, we introduce the general workflow of BFT chains and then summarize their features. This paper focuses on mainstream BFT chains, which usually meet the following prerequisites:

1) The BFT chains aren't pipelined. Only when the last round consensus finishes, a new round may begin[1].
2) The consensus is the main part of the chain's workflow, rather than being a sub-phase [4][2].

As shown in Fig. 1, in BFT chains, users randomly initiate TXs and send them to the blockchain nodes, and the nodes store them in the pool. At the end of each round consensus, a new leader node will be elected [3]. The leader packages TXs from the pool into a block and then broadcasts it among the consensus nodes. The block will undergo complex communication among the consensus nodes while voting. When the final stage of voting is passed, the block is confirmed.

In various BFT chains, the number of consensus stages and packaging rules may vary. Here we explain these two features. The detailed illustration of BFT chains is in Table I.

Feature 1: The various time-consuming stages.

A round of BFT-type consensus is often multi-stage, and each stage contains complex communication. The number of stages varies depending on the type of consensus. Here we mainly consider time-consuming stages. Taking PBFT in Cosmos for instance, the leader is selected through a local $O(1)$ complexity algorithm at the end of each round. Subsequently, the leader packages a block, and nodes vote on the block in three stages with a complexity of $O(n^2)$, as shown in Fig. 2. Byzantine nodes may have malicious

[1][8, 15] are examples of pipelined BFT chains

[2]Otherwise, the delay calculated by this paper can only be used as a reference for the consensus-stage instead of overall performance.

[3]In some schemes, the leader election may also be at the start of a new round. This doesn't affect our analysis.

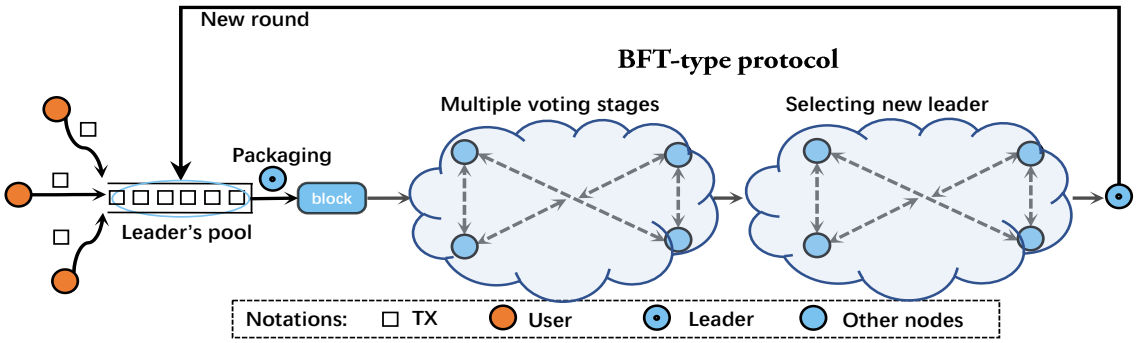


Fig. 1: A general workflow of the BFT chains.

behaviors or crashes, which interrupt the voting (as the cross in Fig. 2). Since the election of the leader is only $O(1)$, the consensus delay experienced by a block in Cosmos is mainly determined by the subsequent three stages.

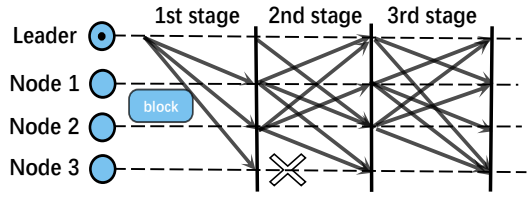


Fig. 2: Three-stage voting in PBFT of Cosmos.

In the consensus of [18], a leader for the next round needs to be elected through a two-stage Raft. In the next round, the leader packages the block and initiates a three-round BFT to vote on it. At this time, the consensus duration is determined by a total of five stages. In contrast, in Noxbft of [17], there are only the prepare stage and vote stage, i.e., two stages.

Feature 2: The different packaging rules.

Due to its functional design, BFT chains may have different packaging rules. The rules here mainly refer to how the leader packages TXs from the pool to a block.

1) Empty-block-allowed: BFT chains may use blocks as heartbeat detection. For instance, in the empty-block mode of [3][4], when the leader packages a block, even if there are temporarily no TXs in the pool, an empty block will be packaged and sent to the nodes for consensus to detect whether nodes are online.
2) Full-block-only: In [4, 9, 15], if TXs in the pool cannot fill a block, the leader needs to wait until it's full [5].
3) Other rules: In [3] , it can be set that the leader can send it out when there is at least one TX in a block. [16] does not mention a fixed strategy for packaging TXs.

[4]In Cosmos, several modes are selective.

[5]To solve possible crash of the leader, and the congestion caused by insufficient TXs for a long time, some remedial mechanisms can be selected in actual use. E.g., when a timer expires, a block may be forced to generate.

IV. Modeling

In this section, we build a general framework for BFT chains, considering that different stages and different packaginging rules may be adopted. With the help of a bulk-service model in the queuing theory, we derive the expression of the TX's average confirmation delay.

In section A, we model the BFT chains's workflow. Section B introduces how the model reflects the above two features. Section C supplements a significant variable, and section D gives the mean queue length in the pool. Section E gives the expression of the TX confirmation delay. Table II summarizes notations used throughout this paper.

TABLE II: Summary of notations

a	The lower bound of TXs a block needs to have.
b	The block size in number of TXs.
λ	The arrival rate of TXs
r	The number of time-consuming stages the consensus has.
$\mu_1, \mu_2, ..., \mu_r$	The mean process rate of each stage.
ρ	The load intensity of the chain.
$T_1, T_2, ..., T_r$	The duration of each stage.
K_n	Number of TXs arrive during the n-th round consensus.
Z_{n+1}	Number of TXs left in the pool when the n-th round ends.
R_n	Number of TXs in the pool when the n-th round begins.
ϵ	The probability that the chain is idle.
p_j^+	The probability that $Z_{n+1} = j$ when $n \to \infty$
$L_{q,0}$	Average number of TXs in the pool and the chain is idle.
$L_{q,1}$	Average number of TXs in the pool and the chain is busy..
L_q	Average number of TXs in the pool.
f_T, F_T, G_T	PDF, CPF and GF of the variable, T.
W_s	The total delay time of a TX in the chain.

A. The bulk-service model

This section builds a generalized $M/E_r^{a,b}/1$ model to reflect the BFT chains' workflow.

Firstly, we make two assumptions:

1) When the leader selects TXs from the pool for packaging, it follows the FCFS (FIFO) strategy.
2) The timer which forces leader to generate blocks when expiring is not selected, and other remedial measures are not implemented. The model will not reflect them.

Next, we introduce how the model $M/E_r^{a,b}/1$ characterizes the arrival and processing of TXs on BFT chains.

TXs' arrival: TXs are sent to the leader individually and randomly by users, waiting for leader's packaging in the pool. We assume the arrival is a Poisson process with rate λ (TXs/s).

TXs' processing: We regard the process of the packaged block being performed r-stage BFT consensus among the nodes as the service process in queuing theory. The service time is the total duration of these r stages, with the total process rate μ (rounds/s). Bulk service means that this is bulk processing for TXs. We assume that the block size is b (TXs), i.e. each block can contain up to b TXs at most. a (TXs) is the lower bound of each block.

The load intensity, ρ, has the expression:

$$\rho = \frac{\lambda}{b\mu}$$

On if $\rho \leq 1$, the steady state exists, and the subsequent calculation remains available.

B. Modeling of features

Modeling of feature 1: various time-consuming stages. In our model, We use a phase-type distribution to reflect this feature. Let us assume the duration of each stage in an r-stage BFT-type protocol is a negative distribution. The total duration is an r-stage phase-type distribution. This assumption is proposed in [13, 15], which is considered as the most accurate model to capture the duration of BFT-type protocols.

Let us assume the process rates of each stage are $\mu_1, \mu_2, ..., \mu_r$, which can be set according to chains' white papers [3, 6], measurement results, [20], and some papers which concentrate on modeling each stage [12, 13]. The total process rate is:

$$\frac{1}{\mu} = \frac{1}{\mu_1} + \frac{1}{\mu_2} ... + \frac{1}{\mu_r}$$

Let us assume random variables $T_1, T_2, ..., T_r$ denote the duration of each stage, and T denotes the total duration:

$$T = T_1 + T_2 + ... + T_r$$

Let us donate $f_T(t)$ as the probability density function (PDF) of T, and $F_T(t)$ as the cumulative distribution function (CDF). They are as follows:

$$f_T(t) = 1 - \sum_{i=1}^{r} e^{-t\mu_i} \cdot \sigma_i \mu_i \quad F_T(t) = 1 - \sum_{i=1}^{r} e^{-t\mu_i} \cdot \sigma_i$$
$$\sigma_i = \frac{1}{\Pi_{j=1, j\neq i}^{r}(1 - \mu_i/\mu_j)} \quad i = 1, 2, ..., r$$

The mean and variance of T are:

$$E(T) = \sum_{i=1}^{r} E(T_i) = \sum_{i=1}^{r} \frac{1}{\mu_i} \quad Var(T) = \sum_{i=1}^{r} \frac{1}{\mu_i^2}$$

Modeling of feature 2: different packaging rules. In this model, we introduce the thresholds a, b to reflect the packaging rules of different BFT chains. First, we introduce the physical meaning of the thresholds a, b, and then introduce how different packaging rules correspond to them.

When the leader packages a block, if the number of TXs in the pool is less than a, the leader will wait for TXs to arrive until a is exceeded, packaging all a TXs into a block. When the number of TXs exceeds a but less than b, the leader will immediately package all TXs in the pool into a block. When the number of TXs exceeds b, the leader will only package b TXs into a block instantly.

Now we can characterize different packaging rules:

- Empty-block-allowed: We set $a = 0$. When the leader needs to package TXs, even if there are no TXs in the pool, it still satisfies $0 \geq a$ and can be packaged into a block. Therefore, an empty block is generated.
- Full-block-only: We set $a = b$. When the number of TXs in the pool reaches a, it also reaches the block size b. The block will be packaged only if it is full.
- Other rules: In a mode of [3], a block can generate when at least one TX exists in a block. In this case, $a = 1$. If the designer requires a block that can be packaged at least $0.25b$ TXs exist, we set $a = 0.25b$.

Here are some examples of the cases we mentioned above, as shown in Table. III. In some cases, a or r varies in a set, meaning all these values are allowed.

TABLE III: Examples.

Projects	[16]	[4]	[17]	[3]	[6]	[9]	[18]
a	$[0, b]$	b	0	$\{0, 1\}$	0	b	0
r	$\{2, 4\}$	3	2	3	2	3	2

Now, we move to solve the model to get the delay.

C. The queue length at a post-service epoch

Let us assume that Z_{n+1} denotes the number of TXs in the pool at the moment the n-th round ends, R_n denotes the number of TXs in the pool when the n-th block is packaged, and K_n denotes the number of TX arriving at the pool during the n-th consensus. We have:

$$Z_{n+1} = R_n + K_n \tag{1}$$

As shown in Fig. 3, at the end of the n-th consensus, the number of TXs in the pool is the sum of TXs when n-th round starts and TXs receive during the consensus.

Then according to (1), we have:

$$Pr(Z_{n+1} = i) = \sum_{j=0}^{i} Pr(R_n = j) \cdot Pr(K_n = i - j)$$

where, $Pr(R_n = 0) = \sum_{j=0}^{b} Pr(Z_n = j)$.

If the n-th round of consensus clears the pool, fewer than b TXs remain in the system when the $n-1$-th round ends. For $i = 1, 2, 3...$, we have:

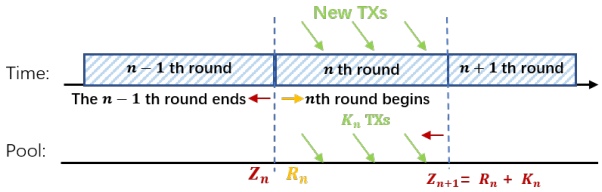


Fig. 3: The queue length at a post-service epoch.

$$Pr(R_n = i) = Pr(Z_n = i + b) \tag{2}$$

and:

$$Pr(R_0 = 0) = 1, Pr(R_0 = i) = 0$$

Their generating functions (GF) are $G_{Z_{n+1}}(z), G_{R_n}(z), G_{K_n}(z)$, and (1) can be transformed into:

$$G_{Z_{n+1}}(z) = G_{R_n}(z) \cdot G_{K_n}(z) \tag{3}$$

According to (2) and (3), we obtain:

$$G_Z(z) = \lim_{n\to\infty} G_{Z_{n+1}}(z) = \lim_{n\to\infty} \frac{\sum_{j=0}^{b-1} p_j^+ (z^b - z^i)}{\frac{z^b}{G_{K_n}(z)} - 1} \tag{4}$$

where $\lim_{n\to\infty} Pr(Z_n = j) = p_j^+$.
According to [21], $K_n(z)$ is:

$$K_n(z) = \Psi(\lambda - \lambda z) \quad \Psi(\alpha) = \int_0^\infty e^{-\alpha t} dF_T(t)$$

Denoting $\lim_{n\to\infty} E(Z_{n+1}) = L^+$,we have the mean of Z_{n+1}:

$$L^+ = G'_Z(1) = \sum_{i,j=1,i\neq j}^{r} \frac{\sigma_i}{\mu_j} - \sum_{i=b}^{b+r} \frac{1}{1-\beta_i} \tag{5}$$

$\beta_b, ..., \beta_{b+r}$ are solutions of $z^b - \Psi(\lambda - \lambda z)$ outside the unit circle. The detailed illustration of (5) is in Appendix A.

D. The mean queue length at a random instant

Based on $L+$, we can derive the mean queue length of TXs in the pool at any instant. We denote it as L_q:

$$L_q = L_{q,0} + L_{q,1}$$

$L_{q,0}$, $L_{q,1}$ are the average number of TXs in the pool, and the chain is idle or busy. Their expressions are closely related to a:

$$L_{q,0} = \sum_{i=0}^{a-1} i \cdot p_{i,0}$$

$p_{i,0}, i = 0, 1, ..., a-1$, are the probabilities that the chain is idle and i TXs are waiting. According to [21], we have:

$$p_{i,0} = \frac{\sum_{j=0}^{i} p_j^+}{\lambda\beta + \sum_{j=0}^{b-1}(a-j)p_j^+}$$

$p_j^+, j = 1, 2, ...b$ are coefficient of $G_Z(z)$ (4).
According to [21], $L_{q,1}$ has the expression based on (5):

$$L_{q,1} = (1 - \sum_{j=0}^{a-1} p_{j,0}) \cdot (L^+ + \frac{\lambda}{2\beta}(Var(T) - E(T)^2))$$

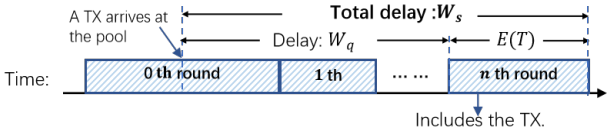


Fig. 4: Relationship between W_s and W_q.

E. TX's confirmation delay

We can now obtain the theoretical average confirmation delay of TXs in BFT chains, W_s. We define it as the duration from when the TX enters the pool of the chain leader until the block is confirmed (reaching consensus).

To denote W_s, We need the help of T_v, which is the total duration of voting stages among all the consensus stages:

$$E(T_v) = E(T_1) + E(T_2) + ...$$

Now we can derive the confirmation delay of TXs:

$$W_s = W_q + E(T_v)$$

As depicted in Fig. 4, W_q is the delay from the moment the TX enters the TX pool, to the moment it will be packaged immediately. W_q can be obtained according to Little's law:

$$W_q = L_q/\lambda$$

V. Model Verification and Analysis

In this section, we verify through experiments to show the model's accuracy on different chains with different parameters. Based on the result, we conduct an analysis to guide the management and configuration.

We set up two chains and simulated PBFT and Noxbft, respectively, to simulate Cosmos ($r = 3, a = 1$) and Hyperchain ($r = 2, a = 0$). We deployed eleven nodes on our server with Intel(R) Xeon(R) Gold 6230 CPU @ 2.10GHz. The process rate of the chain in the public network is simulated by referring to [20]. We continuously sent TXs to the chain for about 2000s and measured the average delay of TXs in the chain. In each section, we show our model's accuracy and analyze the result when the size of blocks changes and the arrival rates of TXs vary for each chain. Moreover, we also verify and analyze the packaging rules' influence.

A. Simulation of Cosmos

1) The block size varies.:

The horizontal axis represents the block size b. The delay from the Model and Experiment curves refers to the left coordinate axis, and curve ρ is the system load intensity, referring to the right coordinate axis. When it exceeds 1, the system loses its steady state.

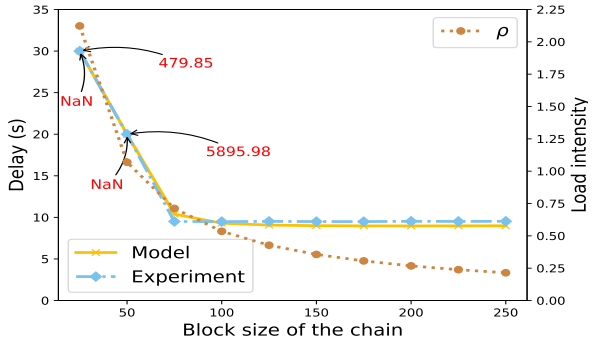


Fig. 5: $\lambda = 10$, b in $(0, 250]$.

Verification: Fig. 5 shows that the model results are very close to the experimental results. When the block size is small, the chain's processing ability is poor, and $\rho > 1$, the system's steady state does not exist. Theoretically, as the chain runs, the delay of TXs will tend to infinity. The theoretical delay doesn't exist (marked as NaN). The experiment results show that when $b = 25, 50$, the delay reached 480s and more than 5000s, respectively. When $b = 75$, the block size only increased by 50 and 25 TXs, but the delay decreased several orders of magnitude. If we continue to run the chain, the delay will tend to infinity, as mentioned above. Designers should obviously avoid the loss of steady states.

Analysis: Before the block size reaches 75, the delay reduced rapidly. But the delay will not change significantly when the block size continues to increase. In these settings, the delay of TXs is no longer sensitive to the block size when $b \geq 75$. For the chain designer and manager, enlarging the block size will not decrease the delay.

2) The arrival rate varies.:

The horizontal axis of Fig. 6 shows the TX arrival rate, and the vertical axis remains unchanged.

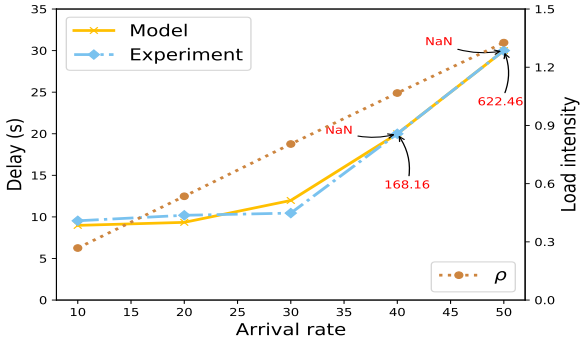


Fig. 6: $b = 200$, λ in $(0, 50]$.

Verification: Fig. 6 shows that the model results are close to the experiment measurements. As the TX arrival rate increases, the system load gradually increases. When $\rho > 1$, the system's steady state disappears, and the model returns NaN. The experimental measurement shows when $\lambda = 40, 50$, the delay reached 168.16S and 622.46s, respectively. Compared with $\lambda = 30$, the arrival rate only increased slightly, but the confirmation delay increased by 16.09 and 59.57 times.

Analysis: Furthermore, as the arrival rate increases, the TX delay initially decreases, and Blocks can be filled faster. But when the rate rises continuously, the system is instantly overloaded. This phenomenon shows designers that the current chain is very sensitive to the arrival rate. They should manage to improve the robustness of the system.

3) The packaging rule varies.:

The horizontal axis of Fig. 7 shows the lower bound of packaging (a), and the meaning of the vertical axis remains unchanged.

Verification: Fig. 7 shows the model accurately predicts the experimental results under different packaging rules.

Analysis: Experimental results show that although packaging rules do not affect the system load, the mean delay of TXs is affected and rising. When the leader wants to package TXs to generate blocks, it needs to wait for TXs exceeding the threshold a. As a increases, each block will obviously be filled more fully, and the waiting time will also become longer. When $a = b$, the delay theoretical increases 65% compared with $a = 0$. For chain designers, it is meaningful to limit a or introduce a mechanism for forced block generation upon timeout to generate blocks more quickly and reduce delay.

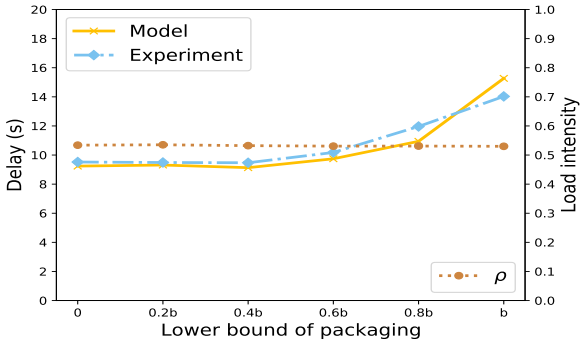


Fig. 7: $\lambda = 20, b = 100$, a in $[0, 100]$.

B. Simulation of Hyperchain

The meaning of the axes in this section is the same as the simulation of Cosmos.

1) The block size varies.:

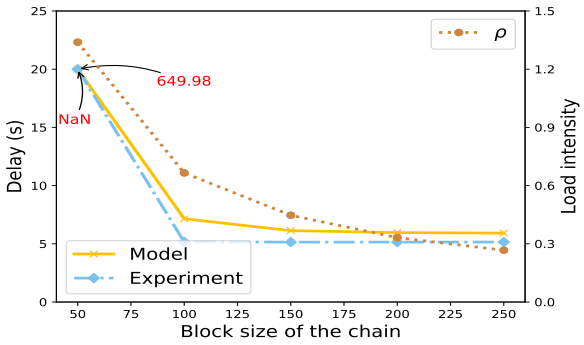


Fig. 8: $\lambda = 20$, b in $(0, 250]$.

Verification: Fig. 8 shows that the model calculation results fit the experimental results well. When the block size is small, $\rho > 1$, the system's steady state does not exist. As the chain runs, the delay of TXs will tend to infinity. The experiment results showed that when $b = 50$, the delay is 649.98s. When the block size increased to 100, the delay decreased several orders of magnitude to about 5.1s. The delay did not change significantly when the block size continued to increase.

Analysis: When configuring a blockchain, the loss of steady states should obviously be avoided first. Designers should make sure the blocks are large enough. However, after that, the delay of TXs is no longer sensitive to the block size after the size is enough. For the chain designer, enlarging the block size will no longer decrease the delay obviously.

2) The arrival rate varies.:

Verification: Fig. 9 shows that the model and experiment results are closed, especially when the load intensity is low. However, when the system load increased, the error also increased. This may be due to the increased randomness of the experiment when the system load is high. As the TX arrival rate increases, the system load gradually increases until $\rho > 1$, and the system's steady state disappears. The experimental measurement showed when $\lambda = 45, 50$, the delay reached 81.44s and 255.13s, respectively.

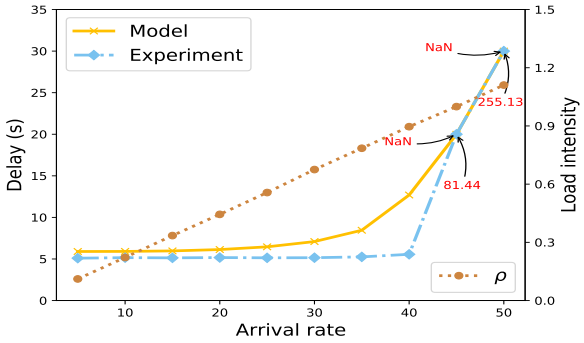


Fig. 9: $b = 150$, λ in $(0, 50]$.

Analysis: As the arrival rate increases, the confirmation delay initially decreases slightly. But when the rate rises continuously, the system is instantly overloaded. This phenomenon brings insights to managers that the chain is very sensitive to the arrival rate currently. They should manage to improve the robustness of the system.

3) The packaging rule varies.:

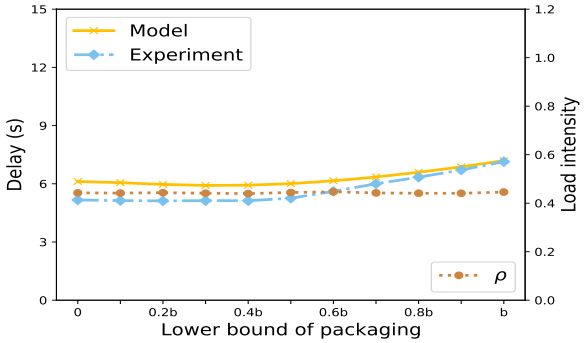


Fig. 10: $\lambda = 20, b = 150$, a in $[0, 150]$.

Verification: In Fig. 10, the model relatively accurately predicts the experimental results when adopting different packaging rules. The system load is low, so the confirmation delay is also at a low level, about 6.1s.

Analysis: Although packaging rules do not affect the system load, the mean delay of TXs is affected and rising. When the leader wants to package TXs to generate blocks, it must wait for TXs exceeding the threshold a. As a increases, each block will obviously be filled more fully, and the waiting time will also become longer. However, in terms of management, this waiting time is always low and does not fluctuate much, so there may be no need to deploy forced-block-generating mechanisms.

VI. Conclusion and Future Work

This paper proposes a general performance analysis model for BFT chains. We accurately model two distinctive features of BFT chains: different numbers of consensus stages and different packaging rules. Finally, we experimentally verify the accuracy of the model and provide a brief analysis based on it. Based on this model, designers can better select and configure BFT chains. In the future, we will supplement other analysis frameworks for BFT chains not covered in this paper. For instance, the pipelined BFT chains which can't apply to the model in this paper.

Acknowledgment

This work was supported by the National Key Research and Development Program of China (2021YFB2701103); the National Natural Science Foundation of China (61972382, U22B2032); ICT-SSC Blockchain Joint Lab; Hainan Zhongke Computing Blockchain Innovation Academy.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Business Review*, p. 21260, 2008.
[2] "Welcome to ethereum," https://ethereum.org, 2023.
[3] "Cosmos," https://cosmos.network/, 2023.
[4] "Hyperledger foundation," https://www.hyperledger.org/, 2023.
[5] M. Castro, B. Liskov *et al.*, "Practical byzantine fault tolerance," in *OsDI*, vol. 99, no. 1999, 1999, pp. 173–186.
[6] "Polkadot," https://polkadot.network/, 2023.
[7] G. S. Veronese, M. Correia, A. N. Bessani, L. C. Lung, and P. Verissimo, "Efficient byzantine fault-tolerance," *IEEE Transactions on Computers*, vol. 62, no. 1, pp. 16–30, 2013.
[8] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "Hotstuff: Bft consensus with linearity and responsiveness," in *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, ser. PODC '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 347–356.
[9] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*, ser. SOSP '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 51–68.
[10] T. Meng, Y. Zhao, K. Wolter, and C.-Z. Xu, "On consortium blockchain consistency: A queueing network model approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 6, pp. 1369–1382, 2021.
[11] A. Vladyko, A. Spirkina, and V. Elagin, "Towards practical applications in modeling blockchain system," *Future Internet*, vol. 13, p. 125, 05 2021.
[12] X. Xu, G. Sun, L. Luo, H. Cao, H. Yu, and A. V. Vasilakos, "Latency performance modeling and analysis for hyperledger fabric blockchain network," *Information Processing & Management*, vol. 58, no. 1, p. 102436, 2021.
[13] F. Ma, Q. Li, Y. Liu, and Y. Chang, "Stochastic performance modeling for practical byzantine fault tolerance consensus in blockchain," *CoRR*, vol. abs/2107.00183, 2021.
[14] O. Wu, S. Li, L. Liu, H. Zhang, X. Zhou, and Q. Lu, "Performance modeling of hyperledger fabric 2.0," in *The International Conference on Evaluation and Assessment in Software Engineering 2022*. New York, NY, USA: Association for Computing Machinery, 2023, p. 357–365.
[15] Y.-X. Chang, Q.-L. Li, Q. Wang, and X.-S. Song, "Dynamic practical byzantine fault tolerance and its blockchain system: A large-scale markov modeling," *arXiv preprint arXiv:2210.14003*, 2022.
[16] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong, "Zyzzyva: Speculative byzantine fault tolerance," *ACM Trans. Comput. Syst.*, vol. 27, no. 4, jan 2010.
[17] "Hyperchain," https://hyperchain.readthedocs.io, 2023.
[18] G. Huang, "Raft cluster practical byzantine fault tolerance algorithm combined with bls signature," *Computer Science and Application*, vol. 12, pp. 1728–1736, 01 2022.
[19] X. Hao, L. Yu, L. Zhiqiang, L. Zhen, and G. Dawu, "Dynamic practical byzantine fault tolerance," 05 2018, pp. 1–8.
[20] D. Cason, E. Fynn, N. Milosevic, Z. Milosevic, E. Buchman, and F. Pedone, "The design, architecture and performance of the tendermint blockchain network," in *2021 40th International Symposium on Reliable Distributed Systems (SRDS)*, 2021.

[21] F. J. Beutler, "A first course in bulk queues (m. l. chaudhry and j. g. c. templeton)," *SIAM Review*, vol. 27, no. 2, pp. 263–264, 1985. [Online]. Available: https://doi.org/10.1137/1027073

APPENDIX A

We focus on $\Psi(\alpha)$:

$$\Psi(\alpha) = \int_0^\infty e^{-\alpha t} \mathrm{d}F_T(t) = \sum_{i=1}^{r} \frac{\beta_i^{r-1}}{\Pi_{j=1,j\neq i}^{r}(\beta_i - \beta_j)} \frac{1}{\beta_3 \alpha + 1}$$

Then we have,

$$\Psi(\lambda - \lambda z) = \sum_{i=1}^{r} \frac{\sigma_i}{\beta_i(\lambda - \lambda z) + 1} = \frac{\psi_1(z)}{\psi_2(z)} \quad \Psi(0) = 1 \tag{6}$$

where

$$\psi_2(z) = \Pi_{i=1}^{r}[\beta_i(\lambda - \lambda z) + 1]$$

$$\psi_1(z) = \sum_{i=1}^{r} \sigma_i \cdot \{\Pi_{j=1,j\neq i}^{r}[\beta_j(\lambda - \lambda z) + 1]\}$$

Analyzing the equation, $\Psi(\lambda - \lambda z) - z^b = 0$, $\mu_0, \mu_1, ..., \mu_{b+r-1}$ are $b + r$ distinct roots inside and outside the unit circle. According to Rouche's Theorem, we notice:

$$|\Psi(\lambda - \lambda z)| \leq 1 = |z^b|.$$

$$z^b = z^b - \Psi(\lambda - \lambda z) + \Psi(\lambda - \lambda z).$$

Similar to z^b, $z^b - \Psi(\lambda - \lambda z)$ also has $b - 1$ zeros inside the unit circle $(\mu_1, \mu_2, ..., \mu_{b-1})$, r zeros outside the unit circle $(\mu_b, ..., \mu_{b+r-1})$, and $\mu_0 = 1$. Then we have:

$$g_Z(z) = \sum_{i=0}^{\infty} p_i^+ z^i = \frac{\Psi(\lambda - \lambda z) \sum_{i=0}^{b-1}(z^b - z^i) p_i^+}{z^b - \Psi(\lambda - \lambda z)}$$

$$\sum_{i=0}^{b-1}(z^b - z^i) p_i^+ = (\sum_{i=0}^{b-1} p_i^+) \Pi_{i=0}^{b-1}(z - \mu_i)$$

$\frac{z^b}{\Psi(\lambda - \lambda z)} - 1$ has $b + r$ above zeros. We get the followings:

$$\frac{z^b}{\Psi(\lambda - \lambda z)} - 1 = \frac{z^b \psi_2(z) - \psi_1(z)}{\psi_1(z)}$$

$$g_Z(z) = \frac{\psi_1(z)(\sum_{i=0}^{b-1} p_i^+) \Pi_{i=0}^{b-1}(z - \mu_i)}{z^b \psi_2(z) - \psi_1(z)} = \frac{\psi_1(z)(\sum_{i=0}^{b-1} p_i^+)}{C \Pi_{i=b}^{b+r-1}(z - \mu_i)}$$

For $g_Z(1) = 1$ and $\psi_1(1) = \sum_{i=1}^{r} \sigma_i = 1$ we have:

$$g_Z(z) = \frac{\psi_1(z) \Pi_{i=b}^{b+r-1}(1 - \mu_i)}{\Pi_{i=b}^{b+r-1}(z - \mu_i)}$$

Then we have the result of (5)

$$L^+ = g_Z'(1)$$

$$g_Z'(1) = \frac{(\psi_1'(1)) \cdot \Pi_{i=b}^{b+r-1}(1 - \mu_i) - \psi_1(1) \cdot [\Pi_{i=b}^{b+r-1}(z - \mu_i)]'|_{z=1}}{\Pi_{i=b}^{b+r-1}(1 - \mu_i)}$$

$$L^+ = g_Z'(1) = \sum_{i,j=1,i\neq j}^{r} \sigma_i \beta_j - \sum_{i=b}^{b+r-1} \frac{1}{1 - \mu_i}$$

2023 IEEE International Performance, Computing, and Communications Conference (IPCCC) | 979-8-3503-0293-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/IPCCC59175.2023.10253823

Performance Modeling of Blockchains with Fixed Block Intervals

Bo Li *†, Hanwen Zhang*†, Chenhao Jiang *†, Zhongcheng Li*†, Yi Sun *†
*Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China.
†University of Chinese Academy of Sciences, Beijing, China
{libo18z, hwzhang, jiangchenhao21s, zcli, sunyi}@ict.ac.cn

***Abstract*—With the emergence of various application scenarios, various chains have been developed to meet their requirements. Among them, chains with fixed block intervals (fixed chains for short) occupy an increasingly significant position. Performance has always been a key bottleneck of blockchains and modeling for them is the most common method for performance analysis. But til now, few models for fixed chains exist and they are not precise and applicable enough. This paper proposes a model for fixed chains via the bulk-service queuing theory, which can reflect the real scenario more precisely and apply to the high load. We consider the continuous time and the transaction (TX for short) pool with limited capacity and reflect the chains' features of fixed intervals and empty blocks to improve accuracy and applicability. We give an expression for three significant measurements: the average confirmation delay of TXs, the blockchain throughput, and the TX rejection rate. We use Ethereum to validate our model. And moreover, we use the model for analysis to assist designers in operating chains.**

***Index Terms*—Blockchain, Delay, Throughput, Performance analysis, Fixed interval, Proof-of-Authority**

I. Introduction

A. Background

In the past decade, blockchain technology has developed rapidly, resulting in the emergence of numerous well-known chains. In these traditional projects, the frequency of block generation is often uncertain. For example, in Bitcoin [1], which uses Proof-of-Work (PoW) consensus, miners compete to generate blocks through hash calculations with random seeds. They are expected to generate blocks every 10 minutes. However, blocks with intervals of several hours usually happen [2, 3], and sometimes the interval may even be negative! Whenever such situations happen, many transactions (TX) are blocked, and the market experiences great panic. In Cosmos [4], and Bitxhub [5], a leader generates one block per round, and nodes use byzantine fault tolerance (BFT) consensus to vote on the block before moving on to generate the next block. In a public network, the interval between blocks is mainly determined by the delay of voting. However, due to the existence of byzantine nodes [6], this delay is often not stable. According to [7], 10% voting delays in Cosmos are 134% longer than the other 90%.

However, the uncertainty of this frequency makes it difficult to predict when a block will be generated. The performance of the chain is not stable and does not meet the requirements of many practical scenarios. For instance, in blockchains of supply chains [8, 9], enterprises hope that the chain can update information timely and regularly. Therefore, chains with fixed block generation intervals have gradually emerged (we call them fixed chains for short), e.g., Ethereum Proof-of-Authority (PoA) [10], XuperChain[11], Polkadot [12], and FISCO BCOS [13][1]. In these chains, the generation of blocks is no longer uncertain but follows a fixed interval. Compared with traditional methods such as PoW, the time for generating blocks in fixed chains can be determined and the performance is stable. In addition, by generating blocks at fixed times, the online status of each node can be continuously monitored through heartbeat detection, which makes fixed chains more widely used.

B. Motivation

Performance has always been a key bottleneck limiting the application of blockchains in various fields. Modeling blockchains is an important method for analyzing their performance and thus enabling their reasonable configuration. So far, various models have emerged to analyze different types of chains. For example, [14–20] are models for Bitcoin and [21–26] are oriented towards the performance of chains with various BFT-type consensuses.

However, most of these models focus on traditional, non-fixed interval chains. They are designed for simplicity, unlike the more intricate demands of fixed chains, and thus fail to encompass the following two critical features: 1) These models often use stochastic processes such as the Poisson process to characterize the mining and voting within chains. Only after the Poisson process ends can the next block be generated. This block interval is uncertain and the Poisson process is simpler than what fixed chains need. 2) Because fixed chains may use a fixed interval mechanism for heartbeat detection, empty blocks are allowed. When a block needs to be generated at a specified time, even if the TX pool is empty, an empty block will still be generated and sent to each node for consensus. This is rarely allowed in other chains, and the corresponding models rarely capture this feature. [2]

[27] is the only model available for fixed chains, but to simplify the scenario, it is not accurate enough to depict the real scenario, and its usability is also limited. 1) [27] assumes that the time while working is discrete and the size of the

[1]The Two-Phase PBFT mode in [13]

[2]For fixed chains, various block generation rules may be selective, but generating empty blocks when necessary is a basic and commonly used rule.

TX pool is infinite, which is obviously different from the real world where time is continuous and the pool must have a capacity limit, resulting in an inaccurate result. 2) [27] only applies to scenarios where the chain load is low. Furthermore, the computational complexity of [27] is usually very high[3], which limits the use of the model.

Therefore, modeling and analyzing fixed chains is of great significance, but the existing model is not accurate and applicable enough, motivating us to carry out this paper.

C. Contribution

This paper proposes an accurate and applicable performance analysis model for fixed chains, provides expressions for three main performance indicators, and analyzes the blockchain configuration. Specifically, the contributions of this paper can be divided into the following two parts:

1) This paper proposes a performance analysis model for fixed chains, taking into account the two features of fixed intervals and generating empty blocks. Moreover, we model and analyze continuous-time and finite-capacity TX pools that conform to real-world scenarios, even if the load is high, providing theoretical expressions for the average values of three indicators: TX confirmation delay, blockchain throughput, and TX pool rejection rate.
2) This paper validates the model using a fixed chain, Ethereum PoA, in a laboratory environment. Furthermore, this paper analyzes how the configuration of the chain affects its performance, to guide designers in designing and configuring chains.

The remainder of this paper is organized as follows. Section 2 discusses related work. Section 3 outlines fixed chains and introduces the features. Section 4 models fixed chains and presents three measurements. Section 5 verifies the model and analyzes the chain configuration. Finally, Section 6 summarizes the paper.

II. Related work

This section summarizes the current work on blockchain performance models. Bitcoin [1] and chains with BFT-type consensus (e.g., Fabric [28]) are famous chains on which most performance modeling work cast effort. [27] is the only model which applies to chains with fixed intervals.

Mining is the most important feature of Bitcoin. [14] and [15] use bulk-service queues to model the mining process and analyze the effect of TX fees on their delays. In [17], the generation of blocks is divided into two parts: mining and broadcasting, each with a Poisson process to describe. In [29], the mathematical innovation in the modeling of chains using queuing theory is emphasized, a new calculation method for the TX confirmation delay based on [17]. In [16], $M_{NET+BC}/Pa/v$ and $M_{NET+BC}/Pa/1$ models are established to analyze Bitcoin's workflow in a real network environment. However, in [16], numerous parameters need to be derived from measurements. A model to fix this problem appears in [18]. In [20], they analyze the TX fees and security of Bitcoin and Ethereum.

Fabric [28] is the most famous BFT-type consensus chain, containing three different stages. [24, 25, 30, 31] focus on modeling the different stages of various versions of Fabric and $M/H_2/1$, $M/M/1$, and $M/E_r/1$ models are adopted. The models that concentrate on other BFT chains also exist. [26] goes into details of PBFT consensus, modeling the relationship between the duration of Practical BFT (PBFT) and the number of nodes with a QBD process. [21] proposes a new blockchain that combines DPoS and PBFT together, and then analyzes it based on [26]. In [22, 23, 32], a complex PBFT blockchain is modeled with dynamic nodes and rolling-back protocol. [19] is a general performance model, which emphasizes the broadcasting of blocks on wireless networks.

As far as we know, [27] is currently the only model applicable to chains with fixed intervals. [27] is a discrete-time model that assumes time is a series of discrete points, and TXs can only arrive at the time point according to a certain distribution. This assumption makes the model less accurate. In the real world, TXs can arrive at any time. In addition, [27] assumes that the TX pool of the chain is infinite, but in actual use, a TX pool must have a limited capacity size (Ethereum is 5120 TXs by default). The pool will reject new TXs when it is full. The infinite pool not only makes the model unable to reflect the real scenario but also limits the application. On the one hand, the model cannot compute the rejection rate. On the other hand, this model can only be used when the load is low. When the load is high, the queue in the pool extends indefinitely, and no theoretical delay for TXs exists (it is infinite in fact). Besides, the method used to compute queue length in [27] is very complicated.

In conclusion, as the chains with fixed intervals become more and more important, no models are precise and applicable enough to help designers analyze the performance. Therefore, we are motivated to model them in this paper.

III. Fixed chains' workflow

In this section, to establish models for fixed chains, we introduce their basic workflow and summarize their features.

Generally speaking, a block in a blockchain system goes through two processes: being generated by a block-generating node and being broadcast to other nodes for confirmation (reaching consensus). Depending on the generation of new blocks and the confirmation of previous blocks are serial or pipelined, fixed chains can mainly be divided into two types, serial type, and pipelined type.

A. Serial type

In serial chains, the confirmation of previous blocks and the generation of new blocks are always serial. The next block can only be generated after the previous block is confirmed. The interval between blocks is fixed based on the design of consensus protocols, e.g., Ethereum [10] and XuperChain [11] with PoA are their representatives. PoA consensus is the most

[3]The algorithm includes a loop that accumulates four layers of nested calculations, each layer needs to calculate infinite times.

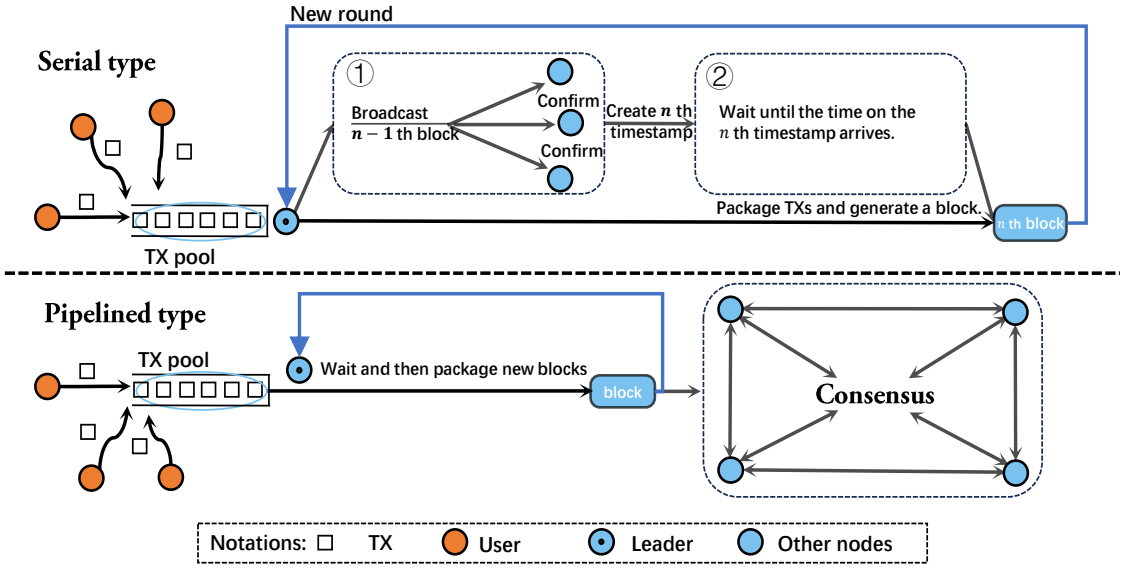


Fig. 1: Workflow of two types fixed chains.

well-known protocol that keeps the consensus duration fixed, so we also call them PoA-type.

Generally, users send TXs to the nodes of the blockchain at random and irregular instants, and the TXs are stored in the TX pool. In PoA consensus, there is a leader responsible for generating blocks. At the beginning of round n, the leader first enters stage ①, broadcasting the block generated in the previous round to all other nodes, as the $n-1$-th block shown in Fig. 1. After nodes confirm that the block comes from the leader, the block is confirmed and the consensus reaches. *After that, the leader*[4] *calculates the n-th timestamp based on the timestamp of $n-1$-th block and the preset block interval.* Next, the leader enters stage ②, waiting for the time on the n-th block's timestamp arrives. While waiting, the leader packages TXs into the n-th block and fills the n-th timestamp into it. When n-th block's timestamp arrives, the block is generated. Even if it is not full or even an empty block, it will no longer change. Then it will be sent to other nodes in round $n+1$-th.

Therefore, PoA consensus uses the mechanism of pre-calculating timestamps and waiting to guarantee fixed intervals. The interval is determined by a preset value.

It is worth noting that serial type also includes some other chains. For example, in some chains using crash fault tolerance (CFT) / BFT-type consensus, after generating a block in round n, the leader sends it to nodes for multi-stage voting to reach consensus. While voting, the leader can select TXs for $n+1$-th block. After voting is completed, $n+1$-th block will be officially generated. However, due to reasons such as not having to consider byzantine nodes in these scenarios, voting may have a very constant duration and can be roughly considered as having a fixed interval between blocks. According to [7], when ignoring crash faults and byzantine faults, 96% rounds of PBFT have delays between 2.3s and 2.7s. When ignoring byzantine faults only, delays of 62% rounds are around 2.65s. For such chains, our model can also be roughly applicable.

[4]One of the nodes which confirmed the $n-1$-th block is usually switched as the new leader for generating the n-th block. But this is not the point we care about in this paper and we simplify the description.

B. Pipelined type

In addition to chains represented by PoA, there is another type of fixed chain that keeps generating blocks in the pipeline. At this time, it is no longer the case that after nodes confirm the previous block, the leader generates the next block. Instead, the leader continuously generates blocks with fixed intervals, regardless previous blocks are confirmed or not, e.g., [12, 13].

As shown in Fig. 1, TXs are randomly sent to the pool of blockchain nodes, and the leader node is responsible for generating blocks. After the block from the previous round is sent to other nodes for consensus, the leader does not wait for consensus to end but waits until the preset interval expires to generate the next block. Similarly, empty blocks will be generated when there are no TXs. At this time, depending on the specific design of the blockchain and the efficiency of consensus, it is not certain whether the consensus of the previous block has ended.

Overall, fixed chains exhibit the following features:

1) The blocks generated by the leader have a fixed time interval, which is often determined by the setting.
2) When generating blocks, even if the pool is empty at that time, the leader is able to generate an empty block. Nodes also need to confirm the empty block as usual.

In addition, it is worth noting that in both types of fixed chains, the leader generates the block at the last moment of each generation interval. This means that regardless of serial type or pipelined type, TXs received during this generation interval can be continuously included in the current block.

IV. Modeling

In this section, we build a model for fixed chains based on bulk-service queuing theory and derive three key measurements. In section 4.1, we describe the bulk-service queue we build for the fixed chains. Sections 4.2 and 4.3 introduce two significant variables for our modeling. And in section 4.4, we give out the final expressions of performance measurements we care about, the TX confirmation delay, throughput, and probability that the pool rejects TXs.

Table I summarises the notations we use in this paper.

TABLE I: Summary of notations

b	Block size in number of TXs.
λ	The arrival rate of TXs.
β	The interval between blocks.
s	Size of TX pool in number of TXs.
Z_{n+1}	Number of TXs in the pool when the n-th block will generate.
d_i	The probability that i TXs arrive during an interval.
X	Number of TXs at a random instant.
k_i	The probability that i TXs arrive after last block generates.
L	Average number of TXs in the pool.
W	Average delay of TXs before their block generates.
ϵ	The rate that a TX is rejected.
TPS	Average throughput in number of TXs.
W_{delay}	The confirmation delay of a TX in the chain.

A. The bulk-service model

In this section, we build a bulk-service model for fixed chains according to [33]. In queuing theory, we denote it as an $M/D^{[0,b]}/1$ model. The majority of modeling is the same for both types of fixed chains. The only difference is shown in section 4.4 while presenting measurements.

First, in order to abstract fixed chains into a mathematical model, we make two assumptions:

1) When the leader selects TXs from the TX pool for packaging, the First Come First Serve strategy (FCFS, also known as FIFO) is adopted.
2) The block size will not change automatically when the chain operates, [5], and each TX has the same size.

TXs' arrival:

TXs are sent by users randomly and individually. Their arrivals at the TX pool can be denoted as a stochastic process, a Poisson process, with the arrival rate λ (TXs/s).

Service process:

TXs are processed from the pool in blocks (called bulk service in queuing theory). The interval of blocks is the service process, the time of which is fixed and denoted as β. To be specific, for serial chains, the service process is broadcasting the last block and then waiting for generating the current one; for pipelined chains, the service process is the interval the leader waits before generating a block.

Service rule:

The leader's packaging is a bulk-service process for TXs. TXs in the pool are packaged according to FCFS. In each block, TXs no more than the block size, b, can be packaged. When there are less than b TXs existing in the pool, all TXs are packaged. Especially, when there are 0 TXs, an empty block is generated, without waiting for TXs. Moreover, the block under service is accessible, which means new TXs can be packaged into the block immediately when the current block is not full.

Limited TX pool capacity:

We consider the TX pool with a limited capacity, s. When the TX pool is full, it will reject TXs.

B. Queue length before service finished

Firstly, we introduce an important random variable, queue length before the service ends. We denote it as $Z_{n+1} = i, i = 0, 1, ..., s$, meaning that i TXs are waiting in the pool before the nth round service ends (i.e. before the n-th block is generated). Moreover, we have the probabilities $Z_{n+1}(i), i = 0, 1, ..., s$:

$$Z_{n+1}(i) = Pr(Z_{n+1} = i), i = 0, 1, ..., s$$

Let us assume that $d_i, i = 0, 1, 2, ...$ denote the probabilities that i TXs arrive during a round of service, even though these TXs may be rejected by the pool when the pool is full. For the reason that TXs arrive at the pool as a Poisson process, we have the expression:

$$d_i = \frac{(\lambda\beta)^i}{i!} e^{-\lambda\beta}$$

To be specific, we denote $d_i = 0$ for $i < 0$.

Now we are able to derive the relationship between Z_{n+1} and Z_n via d_i.

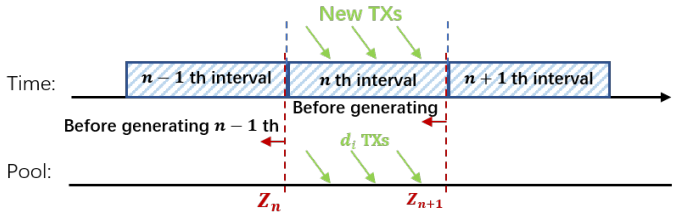


Fig. 2: The queue length before a service finishes.

For $i = 0, 1, 2, ..., s - b$:

$$Z_{n+1}(i) = \sum_{j=0}^{b} d_i \cdot Z_n(j) + \sum_{j=b+1}^{b+i} d_{i-(j-b)} \cdot Z_n(j) \quad (1)$$

When no more than $s - b$ TXs are waiting in the pool at the moment n-th service ends, there are two situations: 1) if $Z_n \leq b$, all Z_n TXs can be packaged in the $n-1$-th block.

[5]In some latest versions of Ethereum, the block size may vary according to the previous load intensity automatically.

All i TXs arrive during the n-th interval. To be specific, when $i = 0$ and $j = 0$, the empty blocks is considered. 2) if $Z_n > b$, b TXs can be packaged in the $n-1$-th block, and Z_n is at most $b+i$. There are $i-(j-b)$ TXs arriving during the n-th interval.

For $i = s-b+1, ..., s-1$:

$$Z_{n+1}(i) = \sum_{j=0}^{b} d_i \cdot Z_n(j) + \sum_{j=b+1}^{s} d_{i-(j-b)} \cdot Z_n(j) \quad (2)$$

This equation is similar to the previous one, for the reason that $Z_n(j)$ is at most s, we replace it as s.

For $i = s$:

$$Z_{n+1}(s) = \sum_{j=0}^{b}(\sum_{l=s}^{\infty} d_l) \cdot Z_n(j) + \sum_{j=b+1}^{s}(\sum_{l=s+b-j}^{\infty} d_l) \cdot Z_n(j) \quad (3)$$

1) If $Z_n \leq b$, all Z_n TXs are included in the $n-1$-th block. At least s TXs arrive during the n-th interval. 2) If $Z_n > b$, b TXs can be packaged in the $n-1$-th block. Z_n is at most s, and at least $s+b-j$ TXs arrive during the n-th interval.

Subsequently We have:

$$Z_{n+1}e = 1, \quad Z_{n+1} = Z_n Q$$

Where e is a column vector of ones with proper dimension, Q is the following $(1+s) \times (1+s)$ metric from the previous equations.

$$\begin{array}{c} \\ 0 \\ 1 \\ \cdots \\ b \\ b+1 \\ b+2 \\ \cdots \\ s \end{array} \begin{array}{c} \begin{array}{ccccccccc} 0 & 1 & 2 & \cdots & b & \cdots & s-b & \cdots & s \end{array} \\ \begin{pmatrix} d_0 & d_1 & d_2 & \cdots & d_b & \cdots & d_{s-b} & \cdots & \sum_{l=s}^{\infty} d_l \\ d_0 & d_1 & d_2 & \cdots & d_b & \cdots & d_{s-b} & \cdots & \sum_{l=s}^{\infty} d_l \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ d_0 & d_1 & d_2 & \cdots & d_b & \cdots & d_{s-b} & \cdots & \sum_{l=s}^{\infty} d_l \\ 0 & d_0 & d_1 & \cdots & d_{b-1} & \cdots & d_{s-b} & \cdots & \sum_{l=s-1}^{\infty} d_l \\ 0 & 0 & d_0 & \cdots & d_{b-2} & \cdots & d_{s-b-2} & \cdots & \sum_{l=s-2}^{\infty} d_l \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 0 & \cdots & d_0 & \cdots & \sum_{l=b}^{\infty} d_l \end{pmatrix} \end{array}$$

$$Z_1 = (d_0, d_1, ..., d_{s-1}, 1 - \sum_{i=0}^{s-1} d_i)$$

According to section 7 in [33], the limit, $Z = \lim_{n\to\infty} Z_n$, exits, which means the following steady-state equation can be solved:

$$Ze = 1, \quad Z = ZQ$$

C. Queue length at a random instant

With the help of Z, we can derive the number of TXs in the pool at a random instant at the steady state. Let us use a random variable, X to denote it. And the probabilities are:

$$X(i) = Pr(X = i), i = 0, 1, ..., s$$

Let us assume that $k_i, i = 0, 1, 2, ...$ denote the probabilities that i TXs arrive from the moment the last block will be packaged until this instant, even though these TXs may be rejected by the pool when the pool is full. We have the expression:

$$k_i = \frac{1}{\beta}\int_0^{\beta} \frac{(\lambda t)^i}{i!} e^{-\lambda t} \mathrm{d}t$$

For $i = 0, 1, 2, ..., s-b$, similar to (1), we have:

$$X(i) = \sum_{j=0}^{b} k_i \cdot Z(j) + \sum_{j=b+1}^{b+i} k_{i-(j-b)} \cdot Z(j)$$

For $i = s-b+1, ..., s-1$, similar to (2), we have:

$$X(i) = \sum_{j=0}^{b} k_i \cdot Z(j) + \sum_{j=b+1}^{s} k_{i-(j-b)} \cdot Z(j)$$

For $i = s$, similar to (3), we have:

$$X(s) = \sum_{j=0}^{b}(\sum_{l=s}^{\infty} k_l) \cdot Z(j) + \sum_{j=b+1}^{s}(\sum_{l=s+b-j}^{\infty} k_l) \cdot Z(j)$$

Therefore, we are able to get the mean of TXs waiting in the pool at a random instant, denoted as L:

$$L = E(X) = \sum_{i=0}^{s} i \cdot X(i)$$

D. Measurements

1) The rejection rate:
The rejection rate refers to the rate that a newly-arriving TX finds the TX pool full and it is rejected to queue in the pool. Therefore, we denote it as ϵ:

$$\epsilon = X(s)$$

2) The throughput:
The throughput is the number of TXs processed by the chain in a unit of time, denoted as TPS. Apparently, those TXs rejected by the pool cannot be processed by the chain.

$$TPS = \lambda \cdot (1 - \epsilon)$$

However, for some pipelined chains where the generated blocks should queue for consensus, the throughput in this section is more like the speed leader packages TXs into blocks.

3) The confirmation delay of a TX:
The confirmation delay of a TX is denoted as W_{delay}, from the instant the TX enters the pool (if it is not rejected), until the block containing it is confirmed by other nodes. For serial and pipelined chains, the expressions of W_{delay} have a little difference. To express W_{delay}, we need to denote another delay firstly, W, which is the delay from the TX entering the pool, to the moment the block containing it will generate immediately. W has the same expression in both kinds of chains according to Little's law:

$$W = \frac{L}{\lambda(1-\epsilon)}$$

For serial chains, W_{delay} has a delay of broadcasting more than W, denoting it as μ_1:

$$W_{delay} = W + \mu_1$$

For pipelined chains, W_{delay} has a delay of a round of consensus more than W, denoting it as μ_2. μ_2 can be found in white papers or estimated by the consensus rate.

$$W_{delay} = W + \mu_2$$

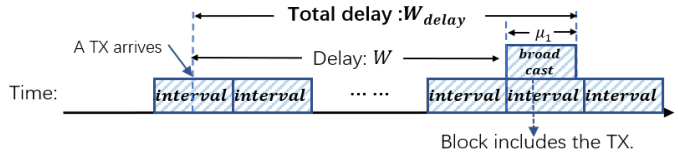


Fig. 3: The delay of serial chains.

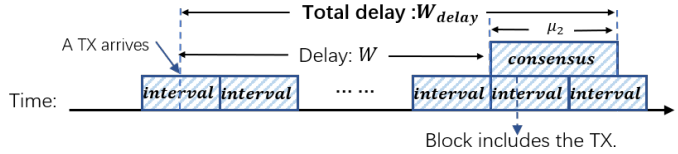


Fig. 4: The delay of pipelined chains.

V. Model verification and analysis

This section shows that the model's accuracy is better than [27] in different parameters through experiments. Moreover, based on the model's results, we have conducted a brief analysis to guide the configuration of the chain.

Because most serial and pipelined chains' models are the same, we do not verify them respectively. An isolated Ethereum blockchain with PoA, which is the most well-known fixed chain, is deployed in our laboratory to verify our model with four nodes and one user. In each section, only one parameter varies and others are stable. Each time, we continuously run the chain for at least 3600s to observe the measurements when they are steady. The broadcasting delay, μ_1, is measured as 1.3s on average. Section A shows the results of the model when the block size changes, section B shows that the model has high accuracy when the interval varies, and section C has different arrival rates of TXs. In section D, the capacity of the TX pool varies.

At first, we introduce ρ to represent the load intensity of the chain, which is quite fundamental in queuing theory. $\rho > 1$ means the chain is overloaded and it cannot process such many TXs. When $\rho < 1$, the load is not saturated and the chain can process all arrived TXs. $\rho = 1$ means the system is in a critical state. ρ has the expression:

$$\rho = \frac{\lambda \cdot \beta}{b}$$

A. The block size varies.

In Fig. 5, the block size varies from 50 TXs to 500 TXs. Our model's TX confirmation delay, rejection rate, and throughput fit well with Ethereum's experimental results, proving the model's accuracy. The discrete model in [27] only computes the confirmation delay without consideration of throughput and rejection rate. When the block size is small and the system load is high, the discrete model cannot solve and returns *INF* only, meaning it thinks the theoretical delay is infinite. When the block size is large, the results from the discrete model are a little smaller than the experimental and our results. The reason may be we have to cut down the four-layer infinite computations in [27] into finite computations.

Furthermore, we analyze the model results to guide the configuration. As the block size increases, the TX confirmation delay gradually decreases and then no longer changes. The rejection rate decreases linearly to 0 and then no longer changes. The throughput increases to its maximum value and then keeps stable. Critical changes in these three indicators occur when the block size is 250 TXs, i.e., $\rho = 1$, and the system is in a critical state. At this time, continuing to increase the block size will not result in better performance., but reducing the block size will cause the performance to decline. Therefore, we recommend operators set the block size to ensure ρ is nearly 1.

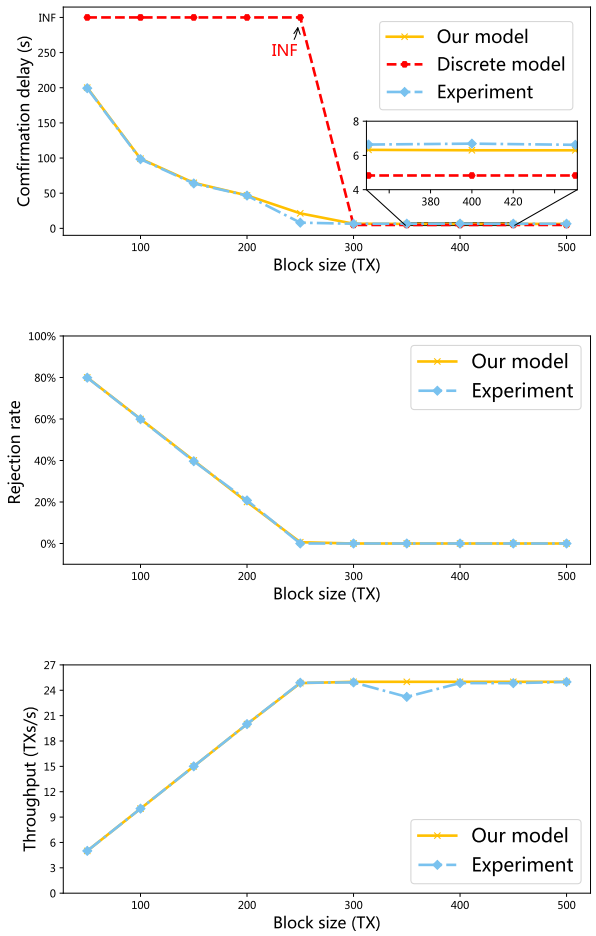


Fig. 5: $\lambda = 25, \beta = 10, s = 1000, b \in [50, 500]$.

B. The interval varies.

In Fig. 6, intervals of blocks are set to change from 5s to 60s. Although the experiment brings uncertainty, the model can still accurately fit overall. The discrete model returns INF immediately.

As the interval increases, the TX confirmation delay, rejection rate, and generation throughput initially do not change. When the interval grows to more than 10s, the delay increases rapidly. The pool rejects many TXs, and the rejection rate grows gradually slower. After the throughput decreases, it gradually becomes smoother. The critical point is $\beta = 10s$, $\rho = 1$. Therefore, setting β to make $\rho = 1$ is best.

C. The arrival rate varies.

In Fig. 7, we simulate the user randomly sending TXs to Ethereum at a rate from 10 TXs/s to 100 TXs/s. The model accurately predicts the measured results of the experiment. Delay from the discrete model in [27] is a little smaller than the experiment and our model. When the arrival rate increases, it cannot work.

We can see that as λ increases, the delay first rises slowly, then rises rapidly, and finally returns to a slow-rising period. This brings insight to designers or operators that to have better performance, the system delay should be in the slow-rising period when the current TX arrival rate is reached. The delay returns to a slow rise later because the arrival rate is very high at this time, and the pool is often full. As blocks are generated and leave the pool, the vacancies in that block are quickly filled by new TXs. The higher the arrival rate, the faster

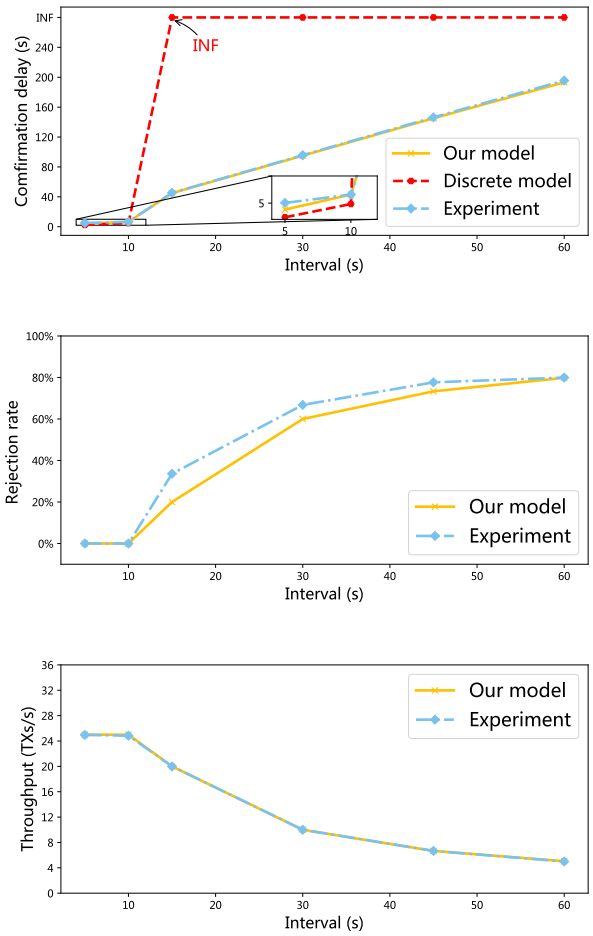


Fig. 6: $\lambda = 25, b = 300, s = 1000, \beta \in [5, 60]$.

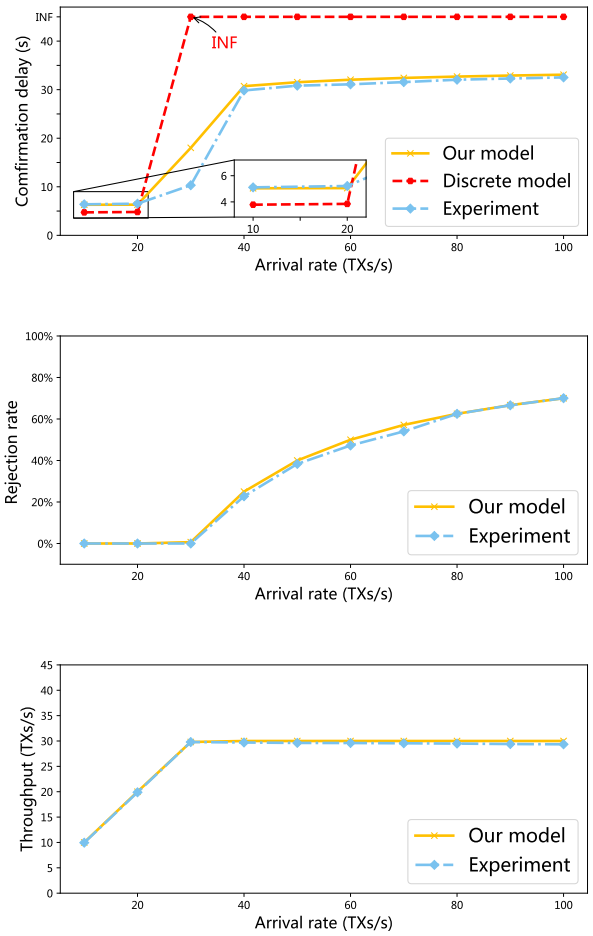


Fig. 7: $b = 300, \beta = 10, s = 1000, \lambda \in [10, 100]$.

the pool is filled. In addition, the rejection rate also rises slowly, and the throughput no longer changes after rising.

D. The pool capacity varies.

The discrete model [27] has no consideration of the pool capacity, so we do not show its results in Fig. 8. Instead, we show two sets of parameters. One has a block size of 300 TXs, and a pool capacity of 300 to 2000 TXs, and the other one has a block size of 200 TXs, and a pool capacity of 200 to 2000 TXs. Their ρ is 0.833 and 1.25 respectively. The model and experimental results fit well.

When $\rho < 1$, the three indicators tend to stabilize. The chain can process the TXs, and not too many TXs are queuing in the pool. Almost no TXs will be rejected. This brings insight to the operator, i.e., when $\rho < 1$, the size of the pool will hardly affect the performance. However, when $\rho > 1$, as the pool capacity gets larger, the confirmation delay increases. The rejection rate maintains around 20%. The design of pool capacity will affect the delay but not the other two indicators. However, the rejection rates for ρ_1, ρ_2 are a little larger initially. When the pool is too small, it is easy to fill, and a few more TXs may be rejected.

VI. Conclusion and Future Work

This paper models blockchains with fixed block intervals and provides expressions for the average confirmation delay of TXs, blockchain throughput, and rejection rate. Subsequently, this paper validates the model via Ethereum PoA and analyzes the configuration to help designers choose the configuration reasonably.

In future work, we will focus on more urgent and specific blockchain performance issues. For example, in order to break through the performance bottleneck brought by a single chain, there are now solutions such as sharding and cross-chain interoperability. Studying performance improvement in such solutions is also a research point worth paying attention to.

Acknowledgment

This work was supported by the National Key Research and Development Program of China(2022YFB2702902); the National Natural Science Foundation of China (U22B2032, 61972382); ICT-SSC Blockchain Joint Lab; Hainan Zhongke Computing Blockchain Innovation Academy.

References

[1] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” *Decentralized Business Review*, p. 21260, 2008.
[2] “Bitcoin questions,” https://bitcoin.stackexchange.com/questions/77783, 2023.
[3] “Bitinfocharts,” https://bitinfocharts.com/comparison/bitcoin-confirmationtime.html, 2023.
[4] “Cosmos,” https://cosmos.network/, 2023.
[5] “Bitxhub,” https://bitxhub.cn/, 2023.
[6] M. Castro, B. Liskov *et al.*, “Practical byzantine fault tolerance,” in *OsDI*, vol. 99, no. 1999, 1999, pp. 173–186.
[7] D. Cason, E. Fynn, N. Milosevic, Z. Milosevic, E. Buchman, and F. Pedone, “The design, architecture and performance of

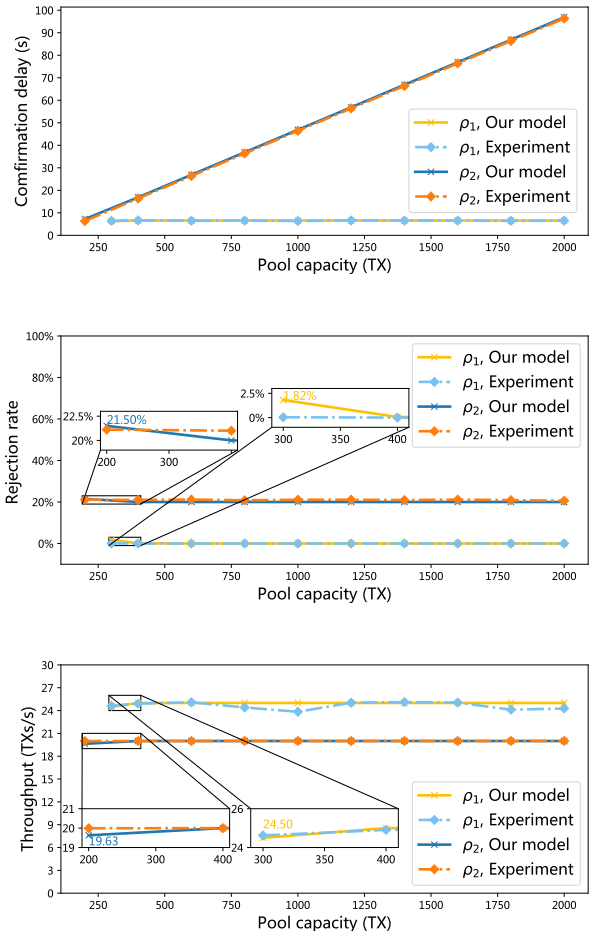


Fig. 8: $\rho_1 = 0.833 : \lambda = 25, b = 300, \beta = 10,\ s \in [300, 2000]$. $\rho_2 = 1.25 : \lambda = 25, b = 200, \beta = 10,\ s \in [200, 2000]$.

the tendermint blockchain network," in *2021 40th International Symposium on Reliable Distributed Systems (SRDS)*, 2021, pp. 23–33.

[8] N. N. Ahamed, P. Karthikeyan, S. Anandaraj, and R. Vignesh, "Sea food supply chain management using blockchain," in *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2020, pp. 473–476.

[9] R. Wang, "Application of blockchain technology in supply chain finance in beibu gulf region," in *2021 International Wireless Communications and Mobile Computing (IWCMC)*, 2021, pp. 1860–1864.

[10] "Welcome to ethereum," https://ethereum.org, 2023.

[11] "Xuperchain," https://xuper.baidu.com/, 2023.

[12] "Polkadot," https://polkadot.network/, 2023.

[13] "Fisco bcos," https://fisco-bcos-doc.readthedocs.io, 2023.

[14] S. Kasahara and J. Kawahara, "Effect of bitcoin fee on transaction-confirmation process," *Journal of Industrial & Management Optimization*, vol. 13, pp. 1–22, 01 2017.

[15] Y. Kawase and S. Kasahara, "Transaction-confirmation time for bitcoin: A queueing analytical approach to blockchain mechanism," in *Queueing Theory and Network Applications*, W. Yue, Q.-L. Li, S. Jin, and Z. Ma, Eds. Cham: Springer International Publishing, 2017, pp. 75–88.

[16] A. Vladyko, A. Spirkina, and V. Elagin, "Towards practical applications in modeling blockchain system," *Future Internet*, vol. 13, p. 125, 05 2021.

[17] Q.-L. Li, J.-Y. Ma, and Y.-X. Chang, "Blockchain queue theory," in *Computational Data and Social Networks*, X. Chen, A. Sen, W. W. Li, and M. T. Thai, Eds. Cham: Springer International Publishing, 2018, pp. 25–40.

[18] J. Mišić, V. B. Mišić, X. Chang, S. G. Motlagh, and M. Z. Ali, "Modeling of bitcoin's blockchain delivery network," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 3, pp. 1368–1381, 2020.

[19] X. Ling, Y. Le, J. Wang, Z. Ding, and X. Gao, "Practical modeling and analysis of blockchain radio access network," *IEEE Transactions on Communications*, vol. 69, no. 2, pp. 1021–1037, 2020.

[20] J. He, G. Zhang, J. Zhang, and R. Q. Zhang, "An economic model of blockchain: The interplay between transaction fees and security," *Big Data & Innovative Financial Technologies Research Paper Series*, 2020.

[21] F. Ma and R.-N. Fan, "Queuing theory of improved practical byzantine fault tolerant consensus," *Mathematics*, vol. 10, p. 182, 01 2022.

[22] C. Yanxia, L. Quanlin, W. Qing, and S. Xingshuo, "Dynamic practical byzantine fault tolerance and its blockchain system: A large-scale markov modeling," *arXiv preprint arXiv:2210.14003*, 2022.

[23] Y.-X. Chang, Q.-L. Li, Q. Wang, and X.-S. Song, "Dynamic practical byzantine fault tolerance and its blockchain system: A large-scale markov modeling," *arXiv preprint arXiv:2210.14003*, 2022.

[24] X. Xu, G. Sun, L. Luo, H. Cao, H. Yu, and A. V. Vasilakos, "Latency performance modeling and analysis for hyperledger fabric blockchain network," *Information Processing & Management*, vol. 58, no. 1, p. 102436, 2021.

[25] T. Meng, Y. Zhao, K. Wolter, and C.-Z. Xu, "On consortium blockchain consistency: A queueing network model approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 6, pp. 1369–1382, 2021.

[26] F. Ma, Q. Li, Y. Liu, and Y. Chang, "Stochastic performance modeling for practical byzantine fault tolerance consensus in blockchain," *CoRR*, vol. abs/2107.00183, 2021. [Online]. Available: https://arxiv.org/abs/2107.00183

[27] S. Geissler, T. Prantl, S. Lange, F. Wamser, and T. Hossfeld, "Discrete-time analysis of the blockchain distributed ledger technology," in *2019 31st International Teletraffic Congress (ITC 31)*, 2019, pp. 130–137.

[28] "Hyperledger foundation," https://www.hyperledger.org/, 2023.

[29] Q. L. Li, J. Y. Ma, Y. X. Chang, F. Q. Ma, and H. B. Yu, "Markov processes in blockchain systems," *Computational Social Networks*, vol. 6, no. 1, pp. –, 2019.

[30] O. Wu, S. Li, L. Liu, H. Zhang, X. Zhou, and Q. Lu, "Performance modeling of hyperledger fabric 2.0," in *The International Conference on Evaluation and Assessment in Software Engineering 2022*, ser. EASE 2022. New York, NY, USA: Association for Computing Machinery, 2023, p. 357–365.

[31] X. Xu, G. Sun, L. Luo, H. Cao, H. Yu, and A. V. Vasilakos, "Latency performance modeling and analysis for hyperledger fabric blockchain network," *Information Processing & Management*, vol. 58, no. 1, p. 102436, 2021.

[32] X. Hao, L. Yu, L. Zhiqiang, L. Zhen, and G. Dawu, "Dynamic practical byzantine fault tolerance," 05 2018, pp. 1–8.

[33] H. Gold and B. Frötschl, "Performance analysis of a batch service system," in *The Fundamental Role of Teletraffic in the Evolution of Telecommunications Networks*, ser. Teletraffic Science and Engineering, J. LABETOULLE and J. W. ROBERTS, Eds. Elsevier, 1994, vol. 1, pp. 155–168.

Connection Science

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/ccos20

A pricing model for subscriptions in data transactions

Bo Li, Minrui Wu, Zhongcheng Li & Yi Sun

To cite this article: Bo Li, Minrui Wu, Zhongcheng Li & Yi Sun (2022) A pricing model for subscriptions in data transactions, Connection Science, 34:1, 529-550, DOI: 10.1080/09540091.2021.2024146

To link to this article: https://doi.org/10.1080/09540091.2021.2024146

Published online: 20 Jan 2022.

CONNECTION SCIENCE
2022, VOL. 34, NO. 1, 529–550
https://doi.org/10.1080/09540091.2021.2024146

A pricing model for subscriptions in data transactions

Bo Li[a,b], Minrui Wu[c], Zhongcheng Li[a,b] and Yi Sun[a,b,d]

[a]Blockchain Lab, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, People's Republic of China; [b]School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing, People's Republic of China; [c]School of Computer Science and Technology, Huazhong University of Science and Technology, Hubei, People's Republic of China; [d]Shandong Key Laboratory of Blockchain Finance, Shandong, People's Republic of China

ABSTRACT

With the increasing demands for data, the subscription scheme came into being in the face of pricing for an extensive and unfixed number of data items. However, in the existing subscription scheme, a diversity of customers in the real market may lead to the lack of stability, which means risking the failure of pricing. Additionally, the study involves arbitrage-free, an essential economics concept, which is not reasonable on data items. To address these problems, this paper provides insights for designing an improved subscription scheme that includes two components: the calculation and the specific validity. On the one hand, the calculation improves the existing scheme by building a new structure that combines different customers' behaviours instead of the separated calculation in the existing scheme, and can steadily set prices for subscriptions to maximise the sellers' profit even in a real market. On the other hand, the specific validity shows the improvement towards arbitrage-free by taking the characteristics of data subscriptions into account. In other words, the specific validity endows the scheme with more rationality.

ARTICLE HISTORY
Received 31 May 2021
Accepted 24 November 2021

KEYWORDS
Data; pricing model; subscription; arbitrage-free

1. Introduction

As data are becoming more widely used, the demand for data transactions is also increasing, and therefore, so is the demand for pricing models of data transactions. In recent years, many large platforms which provide data transaction service are becoming indispensable, such as FACTUAL (2021), crunchbase (2021), Hu et al. (2021), Fernandez et al. (2020), SDTE from Dai et al. (2020) and GBDEX (2021). Different demands towards data transactions emerge in these platforms, resulting in the need of pricing models. For example, individuals may plan to buy a definite and limited number of data items each time, thus, pricing for every single item precisely is necessary. According to the precise data pricing schemes, e.g. query-based models and game theory methods, after waiting for an inefficient and unintuitive calculation, individuals can obtain the precise price. Meanwhile, large organisations, such as firms and research institutes, may have continuous and extensive demands over a

CONTACT Yi Sun sunyi@ict.ac.cn Blockchain Lab, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, People's Republic of China

continuing period, e.g. a potential demand of 200–300 items each year. In order to choose a data supplier for cooperation, they may mainly focus on the total expenditure rather than the cost of any specific item, not willing to wait for the unintuitive and inefficient pricing time and time again. Therefore, an intuitive and efficient pricing strategy for the continuous and enormous demand is in need, which is commonly called the subscription model.

Generally speaking, the subscription scheme is the strategy which intuitively instructs customers how much they should pay when choosing a number of items. Specifically, the subscription scheme has three main features. Firstly, a subscription model usually sets the price of certain data items in total in advance and ignores which items are purchased by the customers. Thus, the subscription model is efficient and intuitive, which is completely different from precise pricing schemes. Secondly, to set the price, the subscription model is used to take the behaviours of customers and sellers into account. When a pricing model attempts to ignore the difference among items, whether customers and sellers can reach agreement according to the subscription scheme and determine the sellers' profit. Thirdly, the subscription scheme is usually multi-stepped and customers can choose the step they need and then pay for it. Each step contains a price and an upper bound of items customers can purchase. Thirdly, the subscription scheme is usually multi-stepped and customers can choose the step they need and then pay for it. Each step contains a price and an upper bound of items customers can purchase.

The features above have brought some new but essential challenges which have not been all solved at once. Firstly, a well-designed scheme should find a way to set the multi-stepped model efficiently and intuitively. It shall be able to tell customers the price to pay in advance. Secondly, while the scheme ignores what items customers actually choose, it is therefore necessary to take the behaviours of customers and sellers into pricing. How do you ensure the agreement maximises the sellers' profit? Thirdly, the pricing scheme should work steadily in a real market, despite the various kinds of customers. Finally, the subscription model should meet arbitrage-free to be reasonable, which is an essential but basic economic requirement for pricing. When considering the characteristics of data transactions, what kind of arbitrage-free should the data subscription schemes meet and how to meet it remain ongoing challenges.

To overcome the challenges in subscription schemes, this paper firstly analyses the existing subscription scheme. The existing mainstream subscription scheme, proposed by Kushal et al. (2011), takes the behaviours of customers and sellers to build a model to overcome the first two challenges. However, there are two remaining challenges. Since its separated calculation on each type of customer may not cover the maximum profit, it may not work steadily in many real conditions. The detailed analysis is in Section 4. Besides, its requirement for a subscription model in data transactions towards arbitrage-free is not reasonable enough. While pricing for data items, the features of data products should be considered.

Facing the two remaining challenges, this paper improves the existing scheme to form a new strategy including two components: the calculation and the specific validity. By combining the behaviours of each type of customer in advance, the new calculation fixes the shortages of the separated calculation. After that, the process shows the reason why it works steadily. Considering the irregularity of steps in data subscriptions, this paper claims a new standard towards arbitrage-free, demonstrating that the standard is reasonable, and then presents an operation to reach it.

In summary, this paper has three main contributions as follows.

(1) The problems in existing subscription models are analysed, as well as the way of improvement.
(2) The combined calculation is put forward to build a new subscription model that can target different kinds of customers. In this way, the model works steadily
(3) This paper proves why the common requirements in arbitrage-free do not fit data items. A new standard is proposed towards arbitrage-free in data subscriptions and then the way to reach the standard is also presented.

2. Related work

Though Muschalle et al. (2012) and Balazinska et al. (2011) previously study the requirements of data pricing, however, to the best of our knowledge, there are not any mainstream pricing schemes. Different schemes have different focuses, and the advantages and disadvantages vary among them. These schemes can be roughly divided into two parts: the traditional methods and the modern methods.

Some large platforms, e.g. GBDEX (2021), still use traditional methods, such as auction and bargain, to reach an agreement. Though some research, e.g. Zhao et al. (2020), attempted to enhance the efficiency of data auctions, the traditional methods are still relatively inefficient.

The modern pricing schemes can be divided into two types according to the purpose, ones aiming at pricing for every item precisely and other ones aiming at pricing for items in total.

The schemes aiming at pricing data items one by one contain two main algorithms, the query-based model (Tang et al., 2013, 2015; Zhang et al., 2020) and the methods based on game theories (Bataineh et al., 2021; Liu et al., 2019; Niyato et al., 2016; Xu et al., 2020). Koutris et al. (2012) firstly illustrate the requirements for the query-based model. After that, some data markets, e.g. (crunchbase, 2021), find a model to offer the final price after a query. At first, the data provider defines some views with price in its local data warehouse. After that, every query creates an undefined new view, which should be divided into a combination of defined views, and then the price of the new view is generated according to the combination. Tang et al. (2015) evaluate the scheme and conclude that it can even precisely offer a price for a single tuple. Considering that dividing the new view into the combination of defined views is an NP-hard combination problem, Tang et al. (2013) find a way, named Minicon, to accelerate the computing process to keep the price returned in time. Moreover, Zhang et al. (2020) enhance Minicon to fit their market of IoT data. Query-based models work well in providing prices for each query precisely, at the same time however, there are also many disadvantages. For instance, these models seriously limit the scale in data trading. Though Tang et al. (2013) and Zhang et al. (2020) enhance them, a lot of time is still needed to calculate the price. Nowadays, many purchasers' demands towards data transactions are continuous and extensive, who are not willing to wait for pricing hundreds of times. Additionally, the defined view may not cover all the views, in other words, the new view cannot be divided into a combination of the defined views. Customers possibly have to wait for the remedies.

Niyato et al. (2016) offer a scheme based on a game theory, the stackleberg model. This scheme divides participants into three groups, namely, the data exchanges, data vendors, and data purchasers. When the data exchange gets data from the vendors, machine learning is used to score the quality of every dataset. Then, with the help of the stackleberg model, the exchange determines a final price for the dataset. This scheme aims to maximise the difference between the cost it pays to vendors and the revenue it can get from customers. In other words, the main difference from other schemes is that this scheme aims at maximising the interests of the exchanges instead of the sellers'. Besides, the stackleberg model is used by Xu et al. (2020) to build a blockchain-based data platform of car sharing and by Liu et al. (2019) to set a two-step data pricing scheme. However, these schemes are confronted with a serious challenge that before these exchanges gain dominance, why vendors agree to supply data if they cannot get the largest profit? How can the purchasers and the vendors trust in the third party instead of trading with each other directly? Furthermore, these models can neither tell the price in total for many data products towards the continuous and extensive demand. At last, these models take customers' willingness into consideration, but still ignore the variety of customers.

Considering that many purchasers have a constant demand for data, the price of every single item is no longer the most necessary issue. As a consequence, many data markets attempt to use subscription schemes that focus on pricing for data items in total. These schemes can replace the precise strategies as well as being a supplement to them. In a subscription model, purchasers can choose a step and acquire as many items as they wish, but no more than the upper bound. Instead of scientifically based however, the origin subscription model is usually experience-based. Then Kushal et al. (2011) gives a multi-stepped model to maximise the sellers' profit and a operation make the model valid and Chen et al. (2019) give an example of providing customers with subscription services. But unfortunately, when facing the real complex market with various customers, Kushal et al. (2011) do not work steadily. Furthermore, the standard and operations towards arbitrage-free are not reasonable enough in data transactions, which also limits the actual utilisation in the real market.

Owing to the problems different pricing models have, determining a well-designed pricing scheme facing the extensive and continuous demand in the real market is challenging. A subscription model is presented in the paper, which works steadily to maximise the sellers' profit. Moreover, it also enables the pricing models to fit arbitrage-free better by considering the characteristics of data transactions.

3. Basic frame work

This section introduces the common basis of subscription schemes in data pricing from Kushal et al. (2011). Since either of them is changed, this section is simple to understand. Three basic concepts are presented in data subscriptions, namely, the basic assumptions, roles that the participates play and the requirements for it.

3.1. Assumptions

Here are the assumptions of the pricing model in subscription schemes.

(1) The pricing model only depends on the number of items customers select. In other words, the pricing model is purchased-amount invariant.
(2) The basic price of each item is determined independently before modeling.
(3) There is cost associated with the production of the dataset only when it is published on the data market. There is no more cost for the seller every time it is sold.
(4) Each kind of CWTP function is the same in each kind of customer. In a sense, this can be justified by averaging the kind of customers.

3.2. Roles in subscription models

In common subscription models, the roles of participants can be roughly divided into three categories: sellers, customers and data markets.

(1) Sellers:
The sellers allow the customers to check the data items and choose what they want. In any event, sellers concentrate on getting the profit as large as possible.
(2) Customers:
Customers are those who are interested in buying data. Generally, they only agree to have a deal when the price the seller asks for is lower than their subjective threshold. In a real market, the willingness of customers may vary.
(3) Data markets:
A data market brings sellers and customers together. For the reason that the commission of the data market is a constant number, it does not affect the pricing scheme. The sellers set the price independently.

3.3. Requirements

To maximise the sellers' profit, there are some requirements a well-designed pricing scheme should meet. The pricing model, $P(n)$, which is an increasing function of variant n, is the amount of items customers determine to buy.

(1) Arbitrage-free: A pricing model should be concerned with arbitrage-free, an important concept in economics described in Section 6.
(2) Diminishing Returns: The price should vary sub-linearly according to the purchase amount, which means $P(n)/n$ is a non-increasing function.
(3) Consumer Buying Power: The pricing model should capture and closely follow the buying power of the customers. i.e. it should be able to represent customers' maximum willingness to pay.

4. The calculation part of subscription schemes

This part introduces the existing subscription scheme and our improvements towards it. Firstly, Section 4.1 shows the basic modelling framework on which both Kushal et al. (2011) and this paper are based. Then, the second subsection illustrates, though Kushal et al. (2011) claim that its calculation can work in the complex market, the reason it is still unstable in

many common conditions. Therefore, the subscription pricing schemes when facing different types of customers remain a problem. After that, the improvement of the subscription scheme to solve this problem is shown. At last, a contrast between the existing scheme and the proposed model is carried out.

4.1. Basic modeling

Now that the roles are divided into three categories, two functions are established, aiming at characterising the sellers and customers' behaviours.

4.1.1. Cumulative willingness to pay

Cumulative willingness to pay (CWTP(n)) is defined as the average maximum willingness of customers to pay for a total of n items, which has the following properties.

(1) CWTP(n) ≥ 0;
(2) CWTP is continuous, smooth and non-decreasing on n.
(3) CWTP is a concave function of n. i.e. $(CWTP)^{''}(n) \leq 0$
(4) The domain of CWTP(n) must be continuous.

In the real market, there are usually different kinds of customers. According to Choudhary (2010), all the customers for information goods can be divided into two types, type c, and type d. Consequently we have two CWTP functions, CWTP_c and CWTP_d to characterise customers' behaviours. When type c customers want to buy one more item, the added price they are willing to afford is fixed. More specifically, CWTP_c is linear. In contrast, type d customers have a declining willingness with a sub-linear CWTP_d. Let $(1-k)$ be proportion of type d and type c has k. e.g. CWTP_c and CWTP_d generally have the form as follows:

- $\text{CWTP}_c = wn, w > 0$
- $\text{CWTP}_d = a\sqrt{n}, a > 0$

When customers are divided into two types, the functions seem clear. In Section 4.2, it is illustrated why setting the subscription scheme facing two CWTP functions remains unsteady in the existing work.

4.1.2. Pricing model

The pricing model gives the price of the items sellers ask to pay. $P(n)$ means the purchaser needs to pay $P(n)$ for n selected data items, which shall meet the following conditions.

(1) $P(n) \geq 0$;
(2) $P(n)$ is non-decreasing;
(3) The domain of $P(n)$ must be continuous.

A subscription scheme is absolutely a pricing model. A subscription scheme usually contains multiple levels with the form $(P_1, T_1), (P_2, T_2), \ldots, (P_m, T_m)$ where m is the total number of steps. Besides, $\{P_j\}$, $j = 1, \ldots, m$, as well as $\{T_j\}$, $j = 1, \ldots, m$, is an increasing sequence. This structure means that customers can buy up to T_j items by paying P_j. For instance, a

customer who chooses the step (20.00, 100) can buy up to 100 data items with the cost of 20.00.

4.2. The calculation part in existing subscription scheme

Let us recall the calculation part from Kushal et al. (2011). The shortage of the calculation is analysed when customers type c and d both exist. Though has claimed to work, a steady subscription scheme in the real market is still a problem.

4.2.1. A recall on the existing subscription scheme

This section recalls the way the existing subscription models set the pricing scheme. Considering the two kinds of customers in the market, this part analyses the relations between the pricing model and the customers' willingness. Next, the scheme calculates the profit from each type of customer separately and sums them up together. And the price which should be set on each step by maximising the total profit is also explored.

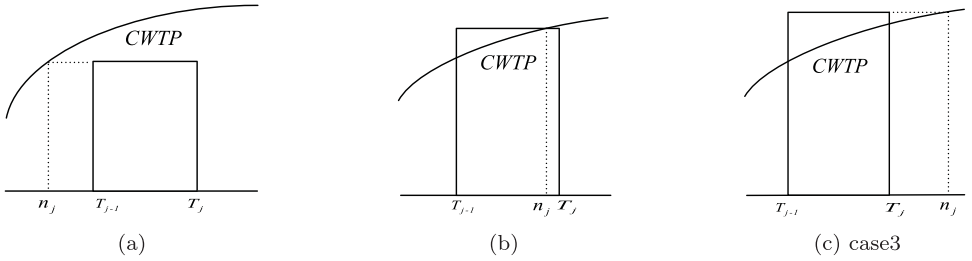
Analysis: The figures above present the relations between customers' behaviours and sellers. According to the definition of CWTP above, and considering every fixed n, customers are willing to pay for n items only if $\mathrm{CWTP}(n) \geq P(n)$. The point of intersection is defined as n_j, $\mathrm{CWTP}(n_j) = P_j$. In each step, based on the relations among n_j, T_{j-1}, T_j, three different cases may appear in Figure 1. Case1 means $cwtp(n)$ is above the pricing model while $cwtp(n)$ is under the pricing model in case3. When a smart seller sets the subscription model, however, only case2 which means $cwtp(n)$ crosses the pricing model can happen. Kushal et al. (2011) show a detailed proof.

In case2 (i.e. $T_{j-1} < n_j < T_j$), customers decide to buy n items from level j when $n_j \leq n < T_j$ and T_i from level i when $T_{j-1} \leq n < n_j$, given by $\max i \leq j | (P_i \leq \mathrm{CWTP}(T_i))$. π_j, the profit from level j can be calculated by using the formula as follows:

$$\pi_j = \int_{T_{j-1}}^{n_j} P_i \, \mathrm{d}n + \int_{nj}^{Tj} P_j \, \mathrm{d}n. \tag{1}$$

In conclusion, the profit in jth is as follows when $T_{j-1} < n_j < t_j$:

$$\pi_s = \sum_{j=1}^{n} \pi_j = \sum_{j=1}^{n} \left(\int_{T_j-1}^{n_j} P_i \, \mathrm{d}n + \int_{nj}^{Tj} P_j \, \mathrm{d}n \right). \tag{2}$$



(a) (b) (c) case3

Figure 1. The figures show three cases between pricing models and CWTP on each step. (a) case1. (b) case2. (c) case3.

To calculate the profit from various kinds of customers, the scheme computes the profit from each type separately and then sums them up.

The profit from type c only: The profit from type c only has the following form. The critical point is given by $P_j = wn_j$, and P_j's are constrained by $T_{j-1} \leq P_j/w \leq T_j$.

$$\pi_j = \int_{T_{j-1}}^{P_j/w} kP_{j-1}\,\mathrm{d}n + \int_{P_i/w}^{T_j} kP_j\,\mathrm{d}n \tag{3}$$

$$\pi_s = \sum_{j=1}^{m} \pi_j = \frac{k}{w}\left(\sum_{j=1}^{m-1} P_j P_{j+1} - \sum_{j=1}^{m} P_j^2 + P_m T_m w\right) \tag{4}$$

The seller can maximise his profit by searching for the local extremum point. After solving m equals from $\partial\pi_s/\partial P_j = 0$, the price should be set on the jth interval is

$$P_j^* = \frac{wj \cdot T_m}{m+1}. \tag{5}$$

The profit from type d only: In the same way, the profit from customers type d only is as follows. This scheme can help us find the critical point n_j, while P_j's are constrained by $T_{j-1} \leq P_j^2/a^2 = n_j \leq T_j$.

$$\pi_j = \int_{T_{j-1}}^{P_j^2/w^2} (1-k)P_{j-1}\,\mathrm{d}n + \int_{P_j^2/w^2}^{T_j} (1-k)P_j\,\mathrm{d}n \tag{6}$$

$$\pi_s = \sum_{j=1}^{m} \pi_j = \frac{1-k}{a^2}\left[\sum_{j=1}^{m-1} P_{j+1}^2 P_j - \sum_{j=1}^{m} P_j^3 + P_m T_m a^2\right] \tag{7}$$

Let $\partial\pi_s/\partial P_j = 0$, the solutions are very complicated. For example, let $m = 2$. The prices on the intervals are,

$$P_1^* = \sqrt{\frac{T_m}{9-2\sqrt{3}}}a, P_2^* = \sqrt{\frac{3T_m}{9-2\sqrt{3}}}a. \tag{8}$$

The profit from all customers in total: After calculating the profit from type c and d separately, the scheme finally focuses on the real market which contains different kinds of customers, and directly sums up the profit from type c in Equation (4), and type d in Equation (7), put together as π_s,

$$\begin{aligned}\pi_s = \sum_{j=1}^{m} \pi_j = &\frac{k}{w}\left(\sum_{j=1}^{m-1} P_j P_{j+1} - \sum_{j=1}^{m} P_j^2 + P_m T_m w\right) \\ &+ \frac{1-k}{a^2}\left[\sum_{j=1}^{m-1} P_{j+1}^2 P_j - \sum_{j=1}^{m} P_j^3 + P_m T_m a^2\right].\end{aligned} \tag{9}$$

After that, to find the extremum point on each step to set the price, m equations ($\partial\pi_s/\partial P_j = 0$) need to be solved. However, this scheme neither shows the exact results in this complex condition nor the constraints of P_j's domain. In fact, this scheme may not work steadily.

4.2.2. The problems of the existing scheme in real markets

Here we analyse the problems in the existing scheme. When we solve equations ($\partial \pi_s / \partial P_j = 0$), usually, we cannot obtain the results directly. The analysis is also helpful for us to find a way to improve the subscription to work steadily for both kinds of customers. Here are three definitions and one proposition for the following analysis. Here are three definitions and one proposition for the following analysis.

Definition 1: *Solvable*: Regardless of the constraints on the domains of the variables, when the equations to find extremum points have solutions, we call this scheme solvable. Otherwise, it is unsolvable.

Definition 2: *Feasible*: When the scheme to find extremum points is solvable, and the solutions obey the constraints on the domains, the scheme is feasible. Otherwise, when the solutions are beyond the domains, it is infeasible.

Definition 3: *Replaceable*: According to the extreme value theorem, when the scheme is infeasible or unsolvable, it may be meaningful to take the maximum point on the boundary instead of the extremum point. The scheme is replaceable when meaningful and unreplaceable when meaningless.

Claim 1: *A subscription scheme is able to set an optimised price on each step to maximise the sellers' total profit only if it is feasible or replaceable. Such a scheme is called workable.*

Proof: (*sufficiency*) The proof of sufficiency is apparent. The target of the scheme is maximising the sellers' profit on each step. When the equations are feasible or replaceable, solutions can always be found to maximise the profit, which is workable. ■

Proof: (*necessity*) The maximum profit can be found from $\partial \pi_s / \partial P_j = 0$ or on the bound of the domain. When the scheme is infeasible, the extremum points do not exist. At this time, if the scheme is unreplaceable, picking up the optimised P_j^* on the boundary is meaningless. As a result, it is not possible to maximise the profit on the step, which in other words, the scheme is not workable. ■

Proposition 1: *A subscription scheme is workable only if it is feasible or replaceable.*

According to the definitions and proposition 1, the following paragraphs present why the scheme by Kushal et al. (2011) is not workable steadily. As a result, it cannot be used to optimise the subscription model. Table 1 shows when m is equal to 3,4,5,6,7, respectively, it is neither solvable nor feasible.

Unsolvable: First and foremost, this scheme may be unsolvable with some common m's. For instance, when m is equal to some common values, e.g. $m = 3, 4, 5$, it returns no solutions for any step at all. At this moment, this scheme cannot be used to set the price.

Table 1. Whether the scheme from Kushal et al. (2011) is solvable or feasible with different m's.

The number of steps: m	2	3	4	5	6	7
(solvable, feasible)	(Y,Y)	(N, N)	(N, N)	(N, N)	(N, N)	(N, N)

(Y, Y) means the scheme is both solvable and feasible while (N, N) means it is neither solvable nor feasible.

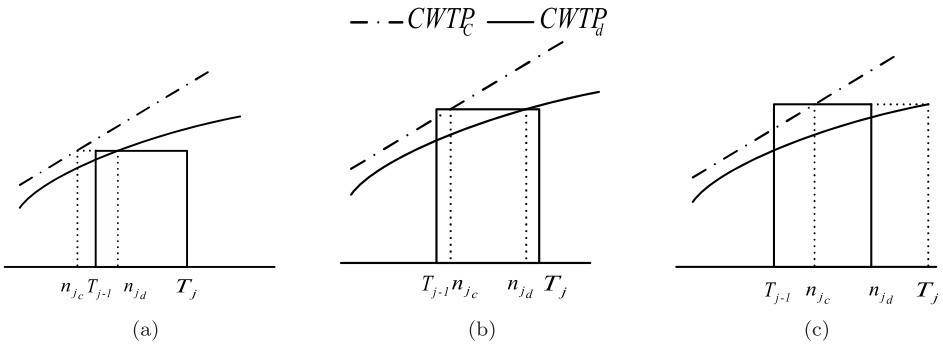


Figure 2. The figures show the complex situations which the existing scheme does not cover. (a) $CWTP_c$ falls in case1 when $CWTP_d$ falls in case2. (b) Both CWTPs fall in case2. (c) $CWTP_c$ falls in case2 when $CWTP_d$ falls in case3.

Here is the analysis of the reason. The integration of this framework is based on the intelligent sellers pursuing a pricing model in case2 of Figure 1. But in this complex condition, when both CWTP curves exist, not both curves satisfy in case2. For example, one CWTP in case1 and the other one in case2, together with one CWTP in case3 and the other one in case2, may lead to a larger profit. Specifically, Figure 2 shows when P_j declines from (b) to (a), the profit of $CWTP_c$ goes down while the profit of $CWTP_d$ gets larger. It is difficult to tell whether the profit in total gets larger or not. In the same way, whether the profit in (b) is larger than (c) is unknown. But the integration above only includes the condition that both two types are in case2. In a word, the calculation sums up the two separated profit integration may not cover the extremum point in many conditions, which may result in either an infeasible or unsolvable scheme.

Infeasible: Though the solutions may exist, when focusing on the expansion of each P_j^*, there is no guarantee that P_j^* lies in its domain, which is from T_{j-1} to T_j. In other words, the scheme may be infeasible.

Unreplaceable: As we all know, a function must have a maximum point on a domain of finite close interval. The constraint of P_j is a finite close interval. When the extremum point does not exist, the maximum point always exists on the boundary of the domain. In this case, when the scheme is unsolvable or infeasible, is it possible to pick up the solutions on the boundary instead?

The answer is no. If the scheme sometimes is not feasible or solvable according to the exact equations, picking up the solutions on the boundary is meaningful. But if the analytical solutions do not exist at all with some m's, the scheme cannot work steadily. Do you want to choose a subscription model which cannot work when you want to set a three or five-step model? Therefore, picking up the solutions on the boundary is meaningless in this work, which is unreplaceable.

In conclusion, facing the complex situation where different kinds of customers exist, the existing scheme may be unsolvable, infeasible, or unreplaceable. In other words, according to proposition 1, the scheme is not workable in many conditions. Although the existing scheme is established, a steady subscription model towards complex customers in the market is still a problem remains unresolved.

4.3. The new calculation part

Considering the disadvantages of the existing scheme, finding a new way to set up the subscription model in the real market catches our eyes. As common sense, the data pricing scheme should be workable no matter how many steps sellers choose. The way of calculating the profit separately in advance and then summing up together cannot work steadily. Naturally, combining customers' behaviours in advance and maximising the profit later is focused on in this paper.

This part is presented with an analysis of the combined calculation and an introduction to the new scheme. Besides, it is proved in the paper that the proposed model, the new subscription scheme, is workable in the real market by theoretical analysis.

4.3.1. Analysis and the properties

The main reason why the existing scheme cannot work steadily is that calculating the profit from each type of customer separately may not cover the maximum points. Therefore, attention shall be paid to combining the customers' behaviours in advance as a mixed CWTP, $\mathrm{CWTP}_{\mathrm{mix}}$ and optimising the pricing model.

This part analyses the way to combine the behaviours of type c and type d customers together in advance. Besides, their properties are presented.

Analysis: According that the degree of type d customers is generally around 0.5 instead of the fixed 0.5. Let us talk about a more general situation, where $\mathrm{CWTP}_c = wn$ and $\mathrm{CWTP}_d = an^x, 0 < x < 1$. Considering every fixed n, k customers are willing to pay up to $\mathrm{CWTP}_c(n)$ for n items while $(1-k)$ fraction of customers are able to afford no more than $\mathrm{CWTP}_d(n)$. Meanwhile, $\mathrm{CWTP}_{\mathrm{real}}$, which combines two types of customers, actually has the following form at n,

$$\mathrm{CWTP}_{\mathrm{real}}(n) = E(k\mathrm{CWTP}_c(n) + (1-k)\mathrm{CWTP}_d(n)) = kwn + (1-k)an^x \tag{10}$$

The single curve, $\mathrm{CWTP}_{\mathrm{real}}$, should fall in case2 in each step. On the jth step, the critical point n_j is the solution to the following equation,

$$P_j = kwn_j + (1-k)an_j^x \tag{11}$$

Using this $\mathrm{CWTP}_{\mathrm{real}}(n)$ to calculate the profit $\pi_s = \sum_{j=1}^{m} \pi_j$ is the best and easiest way to represent both kinds of customers. In this way however, the m equations, $\partial\pi_s/\partial P_j = 0$, may be unable to solve. As we all know, the value of n_j is necessary while integrating. For example, when $x = \frac{1}{2}$, n_j can be easily solved from a quadratic equation with one variable, though π_s may be very complicated. But furthermore, on the one hand, if x varies, there is not an analytical solution that fits all the x's. On the other hand, it is very tough to find n_j due to every x. In particular, when $x \leq \frac{1}{5}$, equivalent equations with an exponent of more than five even do not have an analytical solution itself, which was once a classic mathematical problem in history proved by Abel.

Properties: Being aware that $\mathrm{CWTP}_{\mathrm{real}}(n) = kwn + (1-k)an^x$ clearly does not have analytical solutions, finding a replacement to $\mathrm{CWTP}_{\mathrm{real}}(n)$ is in need. The new CWTP function, $\mathrm{CWTP}_{\mathrm{mix}}(n)$ should have the following properties,

(1) It should mainly meet the basic requirements for a normal CWTP function, e.g. non-decreasing, non-negative, continuous, and concave.

(2) It can represent the behaviours of both kinds of customers with a single expression, which means it can replace CWTP_c and CWTP_d in general. Meanwhile, the loss between the new $\mathrm{CWTP}_{\mathrm{mix}}(n)$ and $k\mathrm{CWTP}_c(n) + (1-k)\mathrm{CWTP}_d(n)$ is small enough to be ignored.
(3) According to this $\mathrm{CWTP}_{\mathrm{mix}}$, no matter what the parameters are, the critical points, n_j, always exist with an analytical form.
(4) The subscription scheme must be workable steadily, which means it should be either feasible (and solvable) or replaceable.

4.3.2. The calculation part in the new subscription scheme

This part introduces a new model combining the customers' behaviours in advance and then maximises the total profit. In the beginning, the way to set the model to meet the requirements above is presented, especially the parameters. After that, it is proved that the proposed scheme is always solvable on each step compared to the existing scheme. Then, considering the condition that some P_j^*s might lay beyond the interval, an operation to guarantee that the scheme is feasible is presented. In this way, the sellers could always get an advised model after calculation. As a result, the scheme is workable.

The way to build the new model: Here, the way to build the new model is introduced. Considering the common ranges of x and k from Choudhary (2010), and after a research on different kinds of expressions, $bn^{k+(1-k)x}$, $b = \sqrt{k/x}w$ and so on, $\mathrm{CWTP}_{\mathrm{mix}} = bn^{(x+1)/2}, b = 2\sqrt{k(1-k)aw}$ is chosen, which has the best performance and can apparently meet the requirement (1). Moreover, the following part presents the reasons why $\mathrm{CWTP}_{\mathrm{mix}} = bn^{(x+1)/2}, b = 2\sqrt{k(1-k)aw}$ meets the requirements (2), (3) and (4).

Firstly, we show why we choose $b = 2\sqrt{k(1-k)aw}$ in $\mathrm{CWTP}_{\mathrm{mix}}$ to represent both kinds of customers' behaviours. Cautiously, we use *dif* to represent the difference between $\mathrm{CWTP}_{\mathrm{mix}}$ and $k\mathrm{CWTP}_c + (1-k)\mathrm{CWTP}_d$, and more specifically, *dif* is defined as the integration of $L1$ loss,

$$dif = \int \left| kwn + (1-k)a \cdot n^x - bn^{(x+1)/2} \right| \mathrm{d}n, 0 < x < 1. \tag{12}$$

To try our best to simulate CWTP_c and CWTP_d, b is set at the minimum point,

$$b = \mathrm{argmin}_b \mathrm{dif} = \mathrm{argmin}_b \int \left| kwn + (1-k)a \cdot n^x - bn^{(x+1)/2} \right| \mathrm{d}n. \tag{13}$$

After substitution and calculation, the minimum point of dif is obtained at $b = 2\sqrt{k(1-k)aw}$. As a result, $\mathrm{CWTP}_{\mathrm{mix}}$ is calculated in the following formula,

$$\mathrm{CWTP}_{\mathrm{mix}} = 2\sqrt{k(1-k)awn}^{(x+1)/2}. \tag{14}$$

Figure 3 shows that L1 loss between $\mathrm{CWTP}_{\mathrm{mix}}$ and the target function, $\mathrm{CWTP}_{\mathrm{real}}$, on some common x's around 0.5 and a fixed $w = 2.5$. MAE is small enough when n is not too large. As far as we know, in most data markets, each data item itself contains a large number of data records or even datasets. Therefore, the number of transactions is strictly limited. Besides, the MAE of the common substitution function, $b * n^{k+(1-k)x} (b = \sqrt{k/x}w)$, which is labelled as the comparison objects is also presented in the figure. Just as we prove, various CWTP_c and CWTP_d give various b which enables the $\mathrm{CWTP}_{\mathrm{mix}}$ to describe the behaviours of CWTP_c and CWTP_d in total. Furthermore, it is proved to be solvable and feasible strictly.

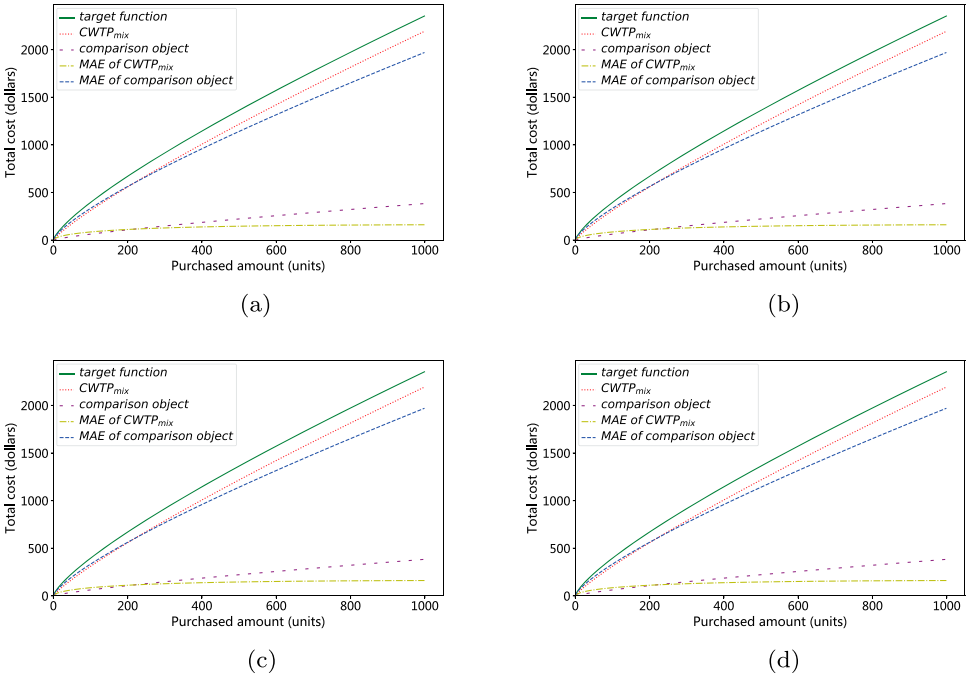


Figure 3. The figures show the performance of CWTP_{mix}. (a) $x = 0.3, k = 0.7, a = 257.3$. (b) $x = 0.5, k = 0.5, a = 68.5$. (c) $x = 0.7, k = 0.3, a = 18.2$. (d) $x = 0.6, k = 0.4, a = 35.3$.

Why the new scheme is solvable: Now let us come to why CWTP_{mix} makes Equations (18) solvable on each step. The operation on how to keep the scheme feasible is in the next paragraph.

When CWTP_{mix} is used to represent both kinds of customers, the intelligent sellers, analysed in Kushal et al. (2011), should guarantee that the scheme falls into case2 and crosses every step. Accordingly, we have the following integrations (with no limitations on the domain of P_j now) on the jth step,

$$\pi_j = \int_{T_j}^{n_j} P_{j-1}\,\mathrm{d}n + \int_{n_j}^{T_j} P_j\,\mathrm{d}n. \tag{15}$$

And owing to $bn_j^x = P_j$, each n_j is calculated as follows:

$$n_j = \left(\frac{P_j^2}{4k\,(1-k)\,aw}\right)^{1/(x+1)}, \quad j = 1, 2, \ldots, m. \tag{16}$$

After summing all the π_j' up, the total profit π_s is obtained as follows,

$$\pi_s = \sum_{j=1}^{m} \pi_j = \sum_{j=1}^{m-1} P_j n_{j+1} - \sum_{j=1}^{m} P_j n_j + P_m T_m. \tag{17}$$

To maximise the total profit, the following m equations are solved to find the price that should be set at each step,

$$\frac{\partial \pi_s}{\partial P_j} = 0, \quad j = 1, 2, \ldots, m. \tag{18}$$

P_m^* and the recurrence relations among P_j^*s are obtained,

$$P_m^* = \frac{(4k(1-k)aw)^{1/2} T_m^{(x+1)/2}}{\left(\frac{x+3}{x+1} - \frac{2}{x+1}\alpha_{m-1}\right)^{(x+1)/2}}, \tag{19a}$$

$$P_j^* = \alpha_j P_{j+1}^*, \quad j = 1, 2, \ldots, m-1. \tag{19b}$$

Apparently, P_j has a recursive relationship depending on α_j. Among α_j's, there are also recursive relations. It is not presented the final expression of P_j^*, because calculating each P_j^* and α_j step by step according to the equations is the way to get the extremum points.

$$\begin{aligned} \alpha_1 &= \left(\frac{x+3}{x+1}\right)^{-(x+1)/2}, \quad j = 1, 2, \ldots, m-2, \\ \alpha_{j+1} &= f(\alpha_j) = \left(\frac{x+3}{x+1} - \frac{2}{x+1}\alpha_j\right)^{-(x+1)/2}. \end{aligned} \tag{20}$$

Then each P_j^* due to P_m^* and α_j can be obtained from Equations (19b) and 20. In other words, the equations of extremum point, Equation (18), have a result on every step. At this time, the scheme is solvable. The operation on how to keep the scheme feasible is in the next paragraph.

The operation to keep the scheme feasible: Now we focus on how to guarantee the results above, P_j^*, to become solutions in its domain. As we can see, the expression of P_m^* only has relationship with T_m and P_j^*s are all up to P_m^*. As a result, T_j's except T_m do not affect the extremum points. In this way, we can move T_j to ensure that every T_j is between n_{j-1} and n_j. According to the equations, such operation not only keeps the profit sellers have but also ensures each P_j^* lays between T_{j-1} and T_j.

Specifically, the moving-boundary operation is as follows. This operation guarantees that the structure falls into case2 on every step. Moreover, the operation is an $O(n)$ algorithm which uses T_j^* to replace T_j for $j = 1, 2, \ldots, m-1$,

Operation 1:

$$\forall j = 1, 2, \ldots, m-1, T_j^* = \begin{cases} T_j, & \text{if } n_j < T_j < n_{j+1} \\ n_j, & \text{otherwise} \end{cases} \tag{21}$$

As a result, the CWTP_{mix} falls into case2 in each step. After the above operations, the constraints of P_j^* remain on P_m^* only,

$$P_m^* \le b T_m^{(x+1)/2} \tag{22}$$

We have $x < 1$ according to CWTP_d which is sub-linear. Thus $\alpha_1 < 1$. After that, every $\alpha_j < 1$. Thus, the denominator of $P_m^* > 1$. Therefore, Equation (22) is satisfied. Now the restraint

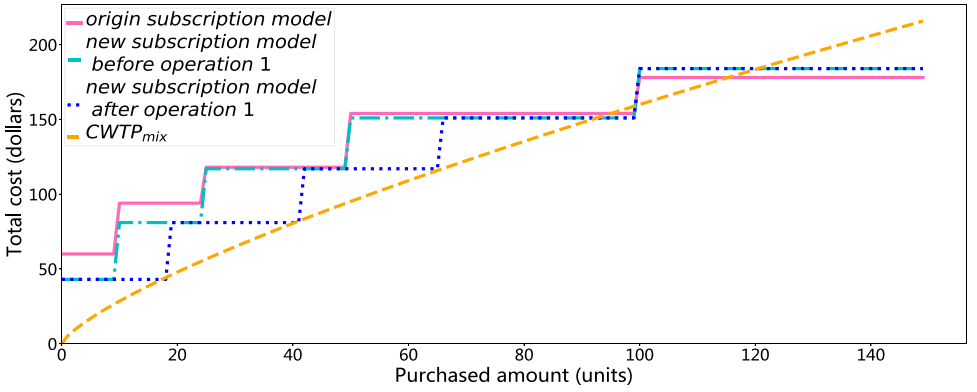


Figure 4. The figure shows the origin model and the new model before and after the attached operation.

is satisfied, so the scheme is feasible. P_j^* for all the j's are stagnation points. Furthermore, according to the expansions of Equation (18), P_j^* is an extremum point, and specifically, it is the local maximum point. In a word, now the P_j^*s are solutions to Equation (18). Precisely, P_j^*s are the prices sellers should set in each step after optimisation in our scheme (Figure 4).

In this way, the optimisation in this work is solvable and feasible. Therefore, it is always feasible to give a way to maximise the profit step by step. The optimised subscription scheme in this work is as follows:

$$(P_1^*, T_1^*), (P_2^*, T_2^*) \ldots (P_m^*, T_m). \tag{23}$$

4.4. A comparison between the existing scheme and the new subscription scheme

It is proved in the paper that no matter how many steps sellers choose and what the parameters are, the scheme can always make sellers get the largest profit after moving T_j. Here, a comparison between the scheme in this paper and the existing scheme is as follows.

Table 2. The comparison between the existing scheme and the new subscription scheme.

T_j	P_j	T_j^* from Kushal et al. (2011)	P_j^* from Kushal et al. (2011)	T_j^* from this work	P_j^* from this work
seller a					
200	100.00	200	154.32	247	172.98
1000	300.00	1000	275.91	1000	326.58
seller b					
50	57.12	null	null	93	41.67
100	62.88	null	null	218	78.69
500	120.00	null	null	500	113.36
seller c					
10	60	null	null	18	43.10
25	94.80	null	null	41	81.37
50	118.80	null	null	61	117.23
100	154.80	null	null	100	151.51
150	178.80	null	null	150	184.64

Let us take the same sellers, a, b and c from Kushal et al. (2011) for instance in Table 2 and there are 2,3 and 5 steps in it. The first two columns are the origin transaction limits (i.e. T_j) and prices on each step (i.e. P_j), then the third and fourth columns are (T_j^*, P_j^*) after the calculation from Kushal et al. (2011) and the fifth and sixth columns are from the optimisation in this work.

As shown in the table, the calculation from the existing scheme usually returns no result, which is labelled as 'null', e.g. m equals 3,5. This is because the separated calculation in the existing scheme sometimes cannot cover the largest profit. Therefore, $\partial\pi_s/\partial P_j = 0$ returns nothing. We cannot help the sellers enlarge the profit. After the combined calculation in this work, whenever m equals to 2,3 or 5, the sellers always get T_j^*'s to enlarge the expected profit. These three cases are just examples, the comparison will be more obvious if more tests are carried out.

4.5. Summary

To summarise, the paper shows that the separated calculation in the existing subscription scheme does not work steadily in complex markets. After that, CWTP_{mix} is presented to combine customers' behaviours in advance with an attached operation, so that the new model can face different customers. The new model is always solvable and feasible, it is therefore workable. At last, the comparison shows that the scheme in this paper works much more steadily.

5. The specific validity part

This part introduces the work towards arbitrage-free. This part can either be carried out after the calculation part or proceed independently. Firstly, there is an introduction of arbitrage-free, which is an important requirement in economics. After that, we illustrate the traditional requirements towards data subscriptions of arbitrage-free is not reasonable. As a result, we present a more reasonable requirement of arbitrage-free towards data items, named specific validity, together with an operation to reach it. At last, we show why the operation towards specific validity is more reasonable in the same condition.

5.1. The concept of arbitrage-free

Arbitrage-free is an essential concept in economics, which means a pricing model should guarantee that there is no arbitrage in a risk-free market. In other words, when the market is risk-free, the price of the union should be equal to the prices of the individual units, i.e. $P(n_1 + n_2) = P(n_1) + P(n_2)$. Otherwise, if $P(n_1 + n_2) > P(n_1) + P(n_2)$, customers buy n_1 and n_2 separately and sell them as $n_1 + n_2$ to get arbitrage. So does $P(n_1 + n_2) < P(n_1) + P(n_2)$. When arbitrage exists, the pricing scheme may not work reasonably from the view of economics. In a word, arbitrage-free is a necessary requirement to meet though it may lessen the profit from the calculation part,

However, considering that the market in real life is not risk-free, the arbitrage-free requirement is usually weaker, i.e. $P(n_1 + n_2) \leq P(n_1) + P(n_2)$. It is also the requirement existing subscription schemes meet, and there is a more formal expression. If $O(t)$ is the least price to get t transactions and $I(t)$ is the price by the pricing mode, e.g. $I(t) = P_{j+1}$

if $T_j < t < T_{j+1}$. When $O(t) = I(t)$ happens at all the t's, it is called validity. The existing expression of arbitrage-free in form of proposition is shown as follows,

Proposition 2:

$$P(n_1 + n_2) \leq P(n_1) + P(n_2), \forall n_1, n_2 \Leftrightarrow O(t) = I(t), \forall t$$

Proposition 3: *A multi-step pricing function* (P, T) *is valid iff* $P_{k+1} \leq \min_{1 \leq j \leq k} P_j + I(T_k + \varepsilon - T_j), \forall k = (0, 2, \ldots, m-1), \varepsilon \to 0$.

5.2. Analysis

Nevertheless, considering the characteristics of data subscriptions, such a requirement is still too much. In a data subscription model, the length of step may be much more uneven than traditional products. As a result, the existing requirement for data subscriptions is not reasonable enough. Therefore, a more detailed example is put forward as follows, a two-step pricing model with constant T.

Example 1:

$$P((0, T]) = P_1, T > 0$$
$$P((T, aT]) = P_2, a > 1$$

Accordingly we have,

$$O(T + \varepsilon) = \min(P_1 + O(\varepsilon), P_2) = \min(2P_1, P_2) \leq 2P_1, \varepsilon \to 0.$$

Due to the conditions of arbitrary free, P_2 is strongly limited,

$$P_2 = I(T + \varepsilon) = O(T + \varepsilon) \leq 2P_1.$$

As a result, P_2 on the second interval $(T, aT]$ should be less than $2P_1$.

At this time, however, parameter a is arbitrary only if $a \geq 1$. As a result, the price from the model on $(T, aT]$ must be less than $2P_1$, regardless of the exact value of a among $2, 3, \ldots, \infty$. The price on an interval thoroughly has no relationship with its upper bound! As common sense, the pricing model on a step should be determined by the bounds and length of interval together. For some common products, the length of steps may be uniform which means parameter a is around 2. But in data markets, the steps are uneven, e.g. the subscription examples from Kushal et al. (2011) showed that a can be 5 or even larger. That may lead to a big difference.

5.3. Specific validity, a new standard to meet arbitrage-free in data items

This part introduces the improvement of a subscription model towards arbitrage-free, which fits the data markets better.

To take the uneven steps of data items into account, we take some relaxations on validity requirements, named specific validity. There is no need that the price $I(t)$ from the pricing model be equal to the least price $O(t)$ all the way.

Definition 4: *Specific validity*: When a subscription model satisfies $I(t) = O(t), \forall t, T_{j+1} \geq t > (T_j + T_{j+1})/2$, $j = 1, 2, \ldots, m-1$, it is called specific validity.

A claim is presented as follows, along with the proof to connect the specific validity with the multi-stepped model.

Claim 2: *A multi-step pricing function* (P, T) *is specific valid iff* $P_{k+1} \leq \min_{1 \leq j \leq k} P_j + I((T_k + T_{k+1})/2 + \varepsilon - T_j)$, $\forall k = (0, 2, \ldots, m-1), \varepsilon \to 0$.

Proof: *(sufficiency)* The proof of sufficiency is obvious. When $T_{j+1} \geq t > \frac{1}{2}(T_j + T_{j+1})$, $I(t) = O(t)$ means $P_{k+1} = O(t)$. According to the definition, $O(t) \leq \min_{1 \leq j \leq k} P_j + I(\frac{1}{2}(T_k + T_{k+1}) + \varepsilon - T_j)$. So far, the sufficiency is proved. ■

Proof: *(necessity)* Here we use the mathematical introduction to finish the proof.

basis: When $k = 0$, the necessity is satisfied.

introduction hypothesis: Assuming that the necessary works are carried out on a fixed k, now we pay attention to $k + 1$.

introduction step: The definitions of $I(t)$ and $O(t)$ mean $O(t) \leq I(t)$ for all the t's, so does it on $T_{j+1} \geq t > \frac{1}{2}(T_j + T_{j+1})$. If $I(t) \leq O(t)$ is happened simultaneously, then $I(t) = O(t)$. Considering about $k + 1$, $P_{k+1} = I(t)$. Now it can be concluded that,

$$I(t) \leq \min_{1 \leq j \leq k} P_j + I\left(\tfrac{1}{2}(T_k + T_{k+1}) + \varepsilon - T_j\right)$$

$O(t)$ must be a combination of the smaller steps, which is expressed as,

$$O(t) = \min_{1 \leq j \leq k} P_j + O\left(\tfrac{1}{2}(T_k + T_{k+1}) + \varepsilon - T_j\right)$$

As the assumptions of the mathematical introduction tell us, $O(t) = I(t)$ works on the first k steps. Consequently,

$$O\left(\tfrac{1}{2}(T_k + T_{k+1}) + \varepsilon - T_j\right) = I\left(\tfrac{1}{2}(T_k + T_{k+1}) + \varepsilon - T_j\right)$$

$$O(t) = \min_{1 \leq j \leq k} P_j + I(\frac{1}{2}(T_k + T_{k+1}) + \varepsilon - T_j)$$

As a result, $I(t) \leq O(t)$ can be satisfied. Step by step, now let us come to $I(t) = O(t)$ for $k + 1$. According to mathematical introduction, the equation holds for every $T_{j+1} \geq t \geq \frac{1}{2}(T_j + T_{j+1})$. ■

The proof of sufficiency and necessity is completed, therefore, this claim becomes a proposition, which claims the requirements a model should meet for specific validity.

Proposition 4: *A multi-step pricing function* (P, T) *is specific valid iff* $P_{k+1} \leq \min_{1 \leq j \leq k} P_j + I((T_k + T_{k+1})/2 + \varepsilon - T_j)$, $\forall k = (0, 2, \ldots, m-1), \varepsilon \to 0$.

5.4. Operation

After discussing the requirements for arbitrage-free in the pricing scheme, an operation is needed to improve the pricing scheme to meet specific validity. An original subscription model without the calculation in Section 4 can also carry out the operation here to meet arbitrage-free. To satisfy Proposition 4, it is necessary to transfer (P, T) into (P'', T) but not too far from (P, T). This operation 2 is an $O(n^2)$ algorithm. The dual operation which transfers (P, T) to (P, T'') will not be repeated here.

Operation 2:

$$\min \sum_j \left| P_j'' - P_j \right| \cdot (T_j - T_{j-1}) \; st. P_k'' \leq P_j'' + I\left(\frac{T_j + T_{j+1}}{2} + \varepsilon - T_j \right),$$

$$\forall k = 1, 2, \ldots m-1, 1 \leq j \leq k-1, \varepsilon \to 0, P_1'' \leq P_2'' \leq \ldots P_m''$$

5.5. Comparison

Here is a comparison on a pricing model between validity and specific validity. Let us look at the same example now. To meet the specific validity, we have the following equations due to the proposition.

$$P_2 = P\left(\frac{T + aT}{2} + \varepsilon \right) = I\left(\frac{T + aT}{2} + \varepsilon \right) = O\left(\frac{T + aT}{2} + \varepsilon \right)$$

$$O(\frac{T + aT}{2} + \varepsilon) = \min(\lceil \frac{a+1}{2} + \varepsilon \rceil P_1, P_2)$$

$$P_2 \leq \left\lceil \frac{a+1}{2} + \varepsilon \right\rceil P_1 \tag{24}$$

Equation (24) can precisely describe the lengths and the distributions of the steps. When a gets larger, P_2 increases linearly. For example, if $a = 4, 6, 10$, then $P((T, aT]) \leq 3T, 4T, 6T$ instead of a fixed $2T$. The scheme which meets specific validity performs much more reasonably than validity.

Table 3 shows the comparison between validity and specific validity. The first two rows are the original scheme (P, T) from seller a,b,c when the third and fourth row show that P_j' meets validity and P_j'' meets specific validity. When the step is uneven, e.g. seller a, the P_j'' performs more reasonable while P_j' is too tight. When the steps are even, e.g. seller b,c, neither validity nor specific makes a big difference to the original scheme.

Table 3. The comparison between the scheme meet validity and specific validity.

T_j	seller a 100	600	seller b 25	100	500	seller c 10	25	50	100	150
P_j	150.00	700.00	37.00	57.18	120.00	30.00	44.50	68.75	100.80	130.05
P_j' meets validity	150.00	300.00	37.00	57.18	114.36	30.00	44.50	68.75	100.80	130.05
P_j'' meets specific validity	150.00	600.00	37.00	57.18	120.00	30.00	44.50	68.75	100.80	130.05

In a word, considering the uneven steps in data subscriptions, and aiming at meeting arbitrage-free, the scheme which meets specific validity performs much more reasonably than validity. Specific validity can help sellers set the model better from the view of economics.

6. Conclusions and future work

After introducing the current work on data pricing, this paper points out the problems of existing schemes, which mainly focus on a single record or dataset, and pricing on some items together is either an important supplement or replacement to them. Based on the existing subscription model that connects customers' buying power with the sellers' pricing model, this paper puts forward a way to improve it to face both kinds of customers. This paper also rigorously proves that the new scheme works steadily in a complex market, which can always optimise the model to enlarge the sellers' profit. Furthermore, the paper proves that the normal requirement of arbitrage-free is too strict for data products such as datasets and data records. In data markets, data pricing models have their characteristics. As a result, the paper proposes a specific validity and an $O(n^2)$ program to reach it.

However, the work in this paper has some shortages. For instance, though we propose a better calculation part, there is still difference between $\mathrm{CWTP}_{\mathrm{mix}}$ and $\mathrm{CWTP}_{\mathrm{real}}$. How to set a better $\mathrm{CWTP}_{\mathrm{mix}}$? Moreover, although it is considered to be worthy, specific validity lessens the profit sellers can get from the calculation part. How to balance the arbitrage-free and the expected profit more reasonably? The Game theory maybe useful here. At last, the paper sets the subscription model independently from precise models. In actual use, they are usually closely related.

In the future, we may concentrate on the shortages above, especially the relations between subscription models and precise models. The balance between two kinds of pricing schemes is necessary to cosmically commercial in data exchange. Besides, as Wu et al. (2021), Wang et al. (2019), Chen et al. (2020) and Sreeja (2019) mention, the quality of data items has an important impact in actual use. It is our concern to combine data quality with subscription pricing schemes. Furthermore, Zhang et al. (2021) and Liang et al. (2018) illustrate the importance of the security of the data exchange system, and then Hu et al. (2021), Dai et al. (2020), Yu et al. (2020) and Murugajothi and Rajakumari (2020) focus on building a secure exchange system. But such system designs are lack of consideration on pricing methods. As mentioned in related work, pricing schemes, the ways of data exchanging and the system design are closely related to each other. Integrating the pricing schemes into the exchange system design deserves more attention in the future.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was supported by the National Key Research and Development Program of China (Grant Number 2019YFB1404903), National Natural Science Foundation of China (Grant Numbers 61772502, 61972382), Natural Science Foundation of Inner Mongolia (Grant Number 2020MS06017), Xiong'An Independently Controllable Blockchain Infrastructure Project (2020).

References

Balazinska, M., Howe, B., & Suciu, D. (2011). Data markets in the cloud: An opportunity for the database community. *Proceedings of the VLDB Endowment*, *4*(12), 1482–1485. https://doi.org/10.14778/3402755.3402801

Bataineh, A. S., Bentahar, J., Mizouni, R., Wahab, O. A., Rjoub, G., & Barachi, M. E. (2021). Cloud computing as a platform for monetizing data services: A two-sided game business model. *Transactions on Network and Service Management*. https://doi.org/10.1109%2Ftnsm.2021.3128160

Chen, L., Shang, S., Zheng, K., & Kalnis, P. (2019, April). *Cluster-based subscription matching for geo-textual data streams*. 2019 IEEE 35th international conference on data engineering (ICDE). IEEE. https://doi.org/10.1109/2Ficde.2019.00084

Chen, Y. H., Chang, C. C., & Hsu, C. Y. (2020, April). Content-based image retrieval using block truncation coding based on edge quantization. *Connection Science*, *32*(4), 431–448. https://doi.org/10.1080/2F09540091.2020.1753174

Choudhary, V. (2010). Use of pricing schemes for differentiating information goods. *Information Systems Research*, *21*(1), 78–92. https://doi.org/10.1287/isre.1080.0203

crunchbase (2021). *Infochimps*. https://www.crunchbase.com/organization/infochimps

Dai, W., Dai, C., Choo, K. K. R., Cui, C., Zou, D., & Jin, H. (2020). SDTE: A secure blockchain-based data trading ecosystem. *IEEE Transactions on Information Forensics and Security*, *15*, 725–737. https://doi.org/10.1109/2Ftifs.2019.2928256

FACTUAL (2021). *FACTUAL + FOURSQUARE. Factual + foursquare*. https://www.factual.com/

Fernandez, R. C., Subramaniam, P., & Franklin, M. J. (2020, August). Data market platforms. *Proceedings of the VLDB Endowment*, *13*(12), 1933–1947. https://doi.org/10.14778/2F3407790.3407800

GBDEX (2021). *Global big data exchange*. http://www.gbdex.com/website/view/dealRule.jsp

Hu, D., Li, Y., Pan, L., Li, M., & Zheng, S. (2021, July). A blockchain-based trading platform for big data. *Computer Networks, 191,* 107994. https://doi.org/10.1016/j.comnet.2021.107994

Koutris, P., Upadhyaya, P., Balazinska, M., Howe, B., & Suciu, D. (2012, August). QueryMarket demonstration: Pricing for online data markets. *Proceedings of the VLDB Endowment*, *5*(12), 1962–1965. https://doi.org/10.14778/2367502.2367548

Kushal, A., Moorthy, S., & Kumar, V. (2011). *Pricing for data markets*. Technical report.

Liang, F., Yu, W., An, D., Yang, Q., Fu, X., & Zhao, W. (2018). A survey on big data market: Pricing, trading and protection. *IEEE Access*, *6*, 15132–15154. https://doi.org/10.1109/2Faccess.2018.2806881

Liu, K., Qiu, X., Chen, W., Chen, X., & Zheng, Z. (2019, December). Optimal pricing mechanism for data market in blockchain-enhanced internet of things. *IEEE Internet of Things Journal*, *6*(6), 9748–9761. https://doi.org/10.1109/2Fjiot.2019.2931370

Murugajothi, T., & Rajakumari, R. (2020, October). Blockchain based data trading ecosystem. *Materials Today: Proceedings*. https://doi.org/10.1016/2Fj.matpr.2020.09.626

Muschalle, A., Stahl, F., Löser, A., & Vossen, G. (2012). Pricing approaches for data markets. In *Birte*.

Niyato, D., Alsheikh, M. A., Wang, P., Kim, D. I., & Han, Z. (2016). *Market model and optimal pricing scheme of big data and Internet of Things (IoT)*. 2016 IEEE international conference on communications (ICC) (pp. 1–6). IEEE. https://doi.org/10.1109%2Ficc.2016.7510922

Sreeja, N. K. (2019). A weighted pattern matching approach for classification of imbalanced data with a fireworks-based algorithm for feature selection. *Connection Science*, *31*(2), 143–168. https://doi.org/10.1080/09540091.2018.1512558

Tang, R., Wu, H., Bao, Z., Bressan, S., & Valduriez, P. (2013). The price is right. In *Lecture Notes in Computer Science* (pp. 380–394). Berlin: Springer. https://doi.org/10.1007/978-3-642-40173-2_31

Tang, R., Wu, H., He, X., & Bressan, S. (2015). *Valuating queries for data trading in modern cities*. IEEE international conference on data mining workshop (ICDMW). IEEE. https://doi.org/10.1109/icdmw.2015.11

Wang, S., Minku, L. L., Chawla, N., & Yao, X. (2019). Learning from data streams and class imbalance. *Connection Science*, *31*(2), 103–104. https://doi.org/10.1080/09540091.2019.1572975

Wu, S., Liu, Y., Zou, Z., & Weng, T. H. (2021). S_I_LSTM: stock price prediction based on multiple data sources and sentiment analysis. *Connection Science*, 1–19. https://doi.org/10.1080/09540091.2021.1940101

Xu, C., Zhu, K., Yi, C., & Wang, R. (2020, December). Data pricing for blockchain-based car sharing: A stackelberg game approach. In *GLOBECOM 2020–2020 IEEE global communications conference*. IEEE. https://doi.org/10.1109/2Fglobecom42002.2020.9322221

Yu, L., Duan, Y., & Li, K. C. (2020, November). A real-world service mashup platform based on data integration, information synthesis, and knowledge fusion. *Connection Science*, *33*(3), 463–481. https://doi.org/10.1080/2F09540091.2020.1841110

Zhang, S., Yao, T., V. K. A. Sandor, Weng, T. H., Liang, W., & Su, J. (2021). A novel blockchain-based privacy-preserving framework for online social networks. *Connection Science*, *33*(3), 555–575. https://doi.org/10.1080/09540091.2020.1854181

Zhang, Y., Xiong, Z., Niyato, D., Wang, P., & Han, Z. (2020, January). Information trading in internet of things for smart cities: A market-oriented analysis. *IEEE Network*, *34*(1), 122–129. https://doi.org/10.1109/2Fmnet.001.1900064

Zhao, Y., Xu, K., Yan, F., Zhang, Y., Fu, Y., & Wang, H. (2020). *Auction-based high timeliness data pricing under mobile and wireless networks*. ICC 2020 – 2020 IEEE international conference on communications (ICC) (pp. 1–6). IEEE. https://doi.org/10.1109%2Ficc40277.2020.9149197

Orbit: A Dynamic Account Allocation Mechanism In Sharding Blockchain System

***Abstract*—The account allocation mechanism is a crucial component affecting the performance of sharding blockchain systems. A well-designed account allocation mechanism must reduce the number of cross-shard transactions while balancing the workload across shards. State-of-the-art mechanisms, which are semi-static, typically adjust account partitions based on historical transactions at regular intervals. However, in real-world applications, unpredictable new scenarios in historical transactions or sudden workload changes can impact shard performance. These existing mechanisms can neither foresee such scenarios to avoid cross-shard transactions nor dynamically adjust account partitions to improve workload issues, leading to suboptimal performance until the next re-allocation. To this end, we propose Orbit, a dynamic account allocation mechanism based on the pending transactions in the pool. Orbit can promptly detect new situations and changes in pending transactions and provide updated allocation strategies. Moreover, through its off-chain scheduling mechanism, Orbit can deploy these strategies before transaction packaging to enhance shard performance. Experimental results show that compared to the state-of-the-art allocation mechanisms, Orbit improves throughput by 2.02 times, reduces cross-shard transactions to 11.9%, and achieves a more balanced shard workload. Additionally, Orbit excels in various other aspects, including latency, transaction queue size, and bandwidth overhead, outperforming the state-of-the-art mechanisms.**

***Index Terms*—Blockchain, Sharding, Dynamic Allocation**

I. Introduction

Sharding [1] is a promising solution for enhancing blockchain performance. The basic idea is to divide all accounts into multiple groups (called shards) to enable parallel processing of transactions (TX for short). Each shard maintains ledger data for a portion of accounts and focuses on processing TXs related to these accounts. TXs confined to one shard are called intra-shard TXs (intra-TX for short), while those involving accounts across multiple shards are called cross-shard TXs (cross-TX for short). Compared to intra-TXs, cross-TXs typically incur the overhead several times larger, as they require collaboration among multiple shards [2]–[4].

To ensure the system high performance, a well-designed accounts allocation in sharding should consider the following two aspects. 1) Reducing cross-TXs. A good method should guarantee the accounts involved in frequent TXs will be allocated in the same shard. As we know, the less cross-TXs, the higher the performance. 2) Balancing workload among shards. When the whole blockchain has to process a large amount of TXs, if the workload is imbalance, some hot shards may be overloaded, and some may be idle. The processing ability of the overall system is not fully utilized.

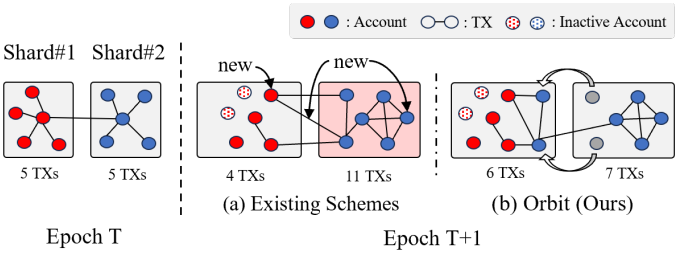


Fig. 1. Existing schemes suffer the high cross-TX portion and imbalance workload.

However, in traditional static allocation mechanisms where accounts are randomly allocated across shards once forever, over 90% TXs are cross-TXs [5]. This high rate of cross-TXs significantly increases the workload. Moreover, the workload of shards are imbalance according to the random allocation.

Therefore, to improve the system's performance from the previous two aspects, many semi-static studies come out. They are called semi-static mechanisms because they may stop operating the shards and re-allocate the accounts based on historical data after an epoch (e.g., one day). To reduce the ratio of cross-TXs, existing research [6], [7] have proposed various semi-static account allocation methods. They transform historical TXs into graph models and regularly apply graph partitioning algorithms [8] or community detection methods [9] to cluster accounts that frequently interact into the same shard, thereby preventing the occurrence of cross-TXs. To balance workload, studies [10], [11] take the workload intensity into account when allocating accounts.

Motivation: However, most of these mechanisms behave charming in tests, and they do not function as well as expected in actual use. The main reason is their features of being semi-static and relying on history data. When relying on history data, they are not well prepared for new situations and some sudden changes in actual use. Moreover, when these stuff happen, because of being semi-static, they cannot re-allocate accounts in time, only waiting for the next epoch.

In terms of reducing cross-TXs, many new users and TXs between new pairs of accounts appear in an epoch, and the cross-TXs are not reduced as expected. Our analysis on Ethereum blocks[1] reveals that only 30.7% of cross-TXs are related to historical TXs, while 44.3% introduce new accounts and 25% result from TXs between new pair of existing accounts. When relying on history data, the new users and TXs between new pairs are lack of data and cannot be properly

[1]We use Ethereum blocks from 12,000,000 to 12,120,000 to assess current solutions. The first 100,000 blocks are used as historical data, and the next 20,000 evaluate the allocation Mechanism's effectiveness.

allocated, as in Fig.1. Moreover, as a semi-static system, though some data generates as shards operate, accounts cannot be re-allocated immediately. The cross-TXs remain until next epoch and lowers the performance. The test results for a ten-shard system show that the actual cross-shard transaction ratio (28.3% to 33.4%) under a semi-static mechanism is significantly higher than the historically predicted results (18.5% to 23.9%), leading to a decrease in the system's performance.

In terms of workload balancing, some sudden changes may happen, which are hard to predict, and the system remain imbalance. While using historical data to balance workload, this mechanism accounts for long-term statistics and ignores short-term fluctuations in each epoch, let alone ensure real-time balance. In Fig. 1, Shards 1 and 2 may have the same workload in epoch T, but suddenly there is a noticeable surge of TXs in shard 2 at the start of epoch T+1. It's hard to predict based on history data. And as a semi-static strategy, the system cannot re-allocate accounts instantly, but only suffering the overload.

Therefore, a pending-TX-dependent and dynamic account allocation mechanism is actually in need, which can analyze TXs the chains just received and allocate related accounts instantly. On one hand, when the chain directly analyzes pending TXs, it is sensitive to new situations and sudden changes, and far from the restrictions of using historical data. On the other hand, as a dynamic mechanism, when new situations and sudden changes happen, instead of waiting for a stop to re-allocate, it schedules accounts immediately in time and synchronously with shard's operating.

Challenge: However, compared to the semi-static and history-based mechanisms, designing a dynamic and pending-TX-dependent mechanism is more challenging. It has a high requirement on the efficiency. The pending TXs have to wait for scheduling accounts, and then they can be processed. Moreover, as a dynamic mechanism, this will happen frequently. Naturally, improving the efficiency of a dynamic account allocation mechanism can be divided into two aspects:

How to efficiently schedule accounts from one shard to another dynamically? In semi-static strategies, account re-allocation often involves consensus from all shards, a time-consuming process will take several minutes [7], [12]. For a dynamic scheduling mechanism, it is inadequate. Designing a effective method is necessary. Furthermore, in a dynamic strategy, account scheduling is synchronous with chains' operating, so the atomicity and safety of operations should also be guaranteed.

How to set an optimal allocation strategy to reduce the operation of scheduling accounts? In semi-static systems, when allocating accounts, shards do not operate, and they do not care about which are involved. However, in a dynamic strategy, the more accounts are involved, the less effective the system is. Moreover, multiple objectives also need to be considered, e.g., the number of cross-TXs and the throughput, which makes finding an optimal strategy more challenging.

Contribution: To address the challenges above and ensure the sharding system high performance, we propose Orbit, a dynamic account allocation mechanism, which relies on the pending TXs. As shown in Fig.1, to reduce cross-TXs, it can instantly transform cross-TXs into intra-TXs by scheduling accounts involved into the same shard before packaging. To balance workload, Orbit can allocate TXs to idle shards when some shards suddenly become overloaded. We overcome the above challenges as follows,

1) We propose an off-chain dynamic account scheduling mechanism. Based on Byzantine Fault Tolerance (BFT) consensus rules, we design an off-chain account permission. Shards can schedule accounts from other shards through off-chain permission transfers, thereby avoiding the on-chain consensus process and enhancing scheduling efficiency. Besides, to ensure the atomicity and security of the scheduling process, we develop corresponding rules for permission locking, unlocking, and timeouts to prevent system failures.
2) We propose an optimization algorithm for allocation strategies. This algorithm calculates an optimal allocation strategy for transactions and accounts based on the current pending TXs and the latest partitioning of accounts. It aims to balance multiple objectives to maximize the overall system performance, including minimizing the number of cross-TXs, balancing the workload across shards, and reducing the cost of account scheduling.
3) We conduct a TX-driven simulation using a real-world Ethereum dataset. The simulation results show that compared to state-of-the-art baselines, Orbit improves throughput by 2.02 times, reduces cross-shard transactions from 32.8% to 11.9%, and achieves a more balanced shard workload. Additionally, Orbit also excels in various other aspects, including latency, transaction queue size, and bandwidth overhead, outperforming the state-of-the-art mechanisms.

The rest of this paper is organized as follows. Section II reviews the state-of-the-art research. Section III describes the design of the scheduling mechanism. Section IV details the optimization of the scheduling strategy. Section V presents the evaluation results of Orbit. Finally, Section VI concludes the paper.

II. BACKGROUND AND RELATED WORKS

In this section, we introduce some background knowledge and related work on account allocation mechanism.

A. Cross-Shard Transaction

In existing sharding solutions, cross-TXs are inevitable. Therefore, how to handle these TXs is a critical issue that sharding systems must address. To this end, current studies have designed various complex cross-shard protocols. For example, in systems based on Unspent TX Outputs (UTXO), RapidChain [4] introduced a transfer mechanism that consolidates all involved UTXOs into the same shard for TX execution. In account/balance-based systems, Monoxide [2]

proposed a relay scheme, splitting cross-TXs into multiple intra-shard sub-TXs that are sequentially committed in the respective shards. OmniLedger [3] introduced a client-driven two-phase commit protocol to ensure the atomicity of cross-TXs. ChainSpace [13] presented a cross-TX processing scheme supporting smart contracts. Pyramid [14], through a hierarchical architecture, designated a specific shard to handle cross-TXs, thereby reducing their overhead. Nevertheless, the high complexity of cross-TXs means that their workload is often several times that of intra-TXs.

B. Account Allocation

Therefore, reducing the number of cross-TXs has become a primary direction for improving the performance of sharding systems. In the systems above, accounts are allocated to shards based on hash values, such as randomly assigning accounts to one of 2^k shards according to the first k bits of the account address. This static random mechanism overlooks the history of TXs between accounts, resulting in a ratio of cross-TXs exceeding 90%.

To reduce cross-TXs, many studies have focused on optimizing account allocation mechanisms by analyzing historical TX data. The main idea is to analyze TX relationships in the previous epoch and place frequently interacting accounts in the same shard in the next epoch to minimize cross-TXs, which is a semi-static mechanism. For example, BrokerChain [7] proposed an account allocation mechanism based on Metis [15], modeling historical TXs as a graph with accounts as nodes and TXs as edges, and using the Metis algorithm to partition this graph to minimize inter-shard TXs. Optchain [16] utilizes the PageRank algorithm to allocate TXs to the shard with the highest association degree, thereby avoiding cross-TXs as much as possible. However, while these algorithms reduce the number of cross-TXs, they often result in very large shards, leading to an imbalance in workload distribution. To address this, Transformers [10] proposed a constrained label propagation algorithm to reduce cross-TXs while minimizing shard workload differences. TxAllo [11] introduced G-Allo and A-Allo community detection methods to balance the load among shards and enhance overall system performance.

Thus, existing work primarily utilizes historical TXs to predict future TX trends and periodically reallocate accounts to reduce cross-TXs and balance shard workloads. However, as discussed in Section I, real-world applications often encounter unpredictable new scenarios in historical transactions or sudden changes in workload. For these unforeseen circumstances, existing allocation mechanisms cannot accurately predict or respond swiftly, resulting in performance degradation until the next reallocation. Consequently, the performance of these systems in real-world applications often falls short of expectations, which is the problem this paper addresses.

III. Mechanism Design of Orbit

In this section, we present the design of the dynamic account allocation mechanism in Orbit, detailing how accounts can be safely and efficiently scheduling from one shard to another.

A. Overview of Orbit

In Orbit, nodes are divided into two roles: allocators and consensus nodes. Allocators are responsible for collecting all TXs sent by users and distributing these TXs to shards. Consensus nodes focus on maintaining shard ledger data and running shard consensus protocols (e.g., PBFT [17]) to process received TXs.

Similar to existing sharding solutions [3], [4], [7], Orbit runs in fixed periods called epochs. At the beginning of each epoch, randomness is generated through a publicly verifiable, bias-resistant, and unpredictable method to assign roles and shard locations to each node randomly [14]. To prevent Sybil attacks, each node must register its public key on an identity blockchain by solving a PoW puzzle before joining the sharding network [3], [7]. Therefore, each node's public key identity, role, and shard location are public during the epoch and can be verified through the identity chain and randomness.

Orbit adopts an account/balance model to represent ledger data, where each user is associated with a pair of an account and a balance. Like Ethereum [18] and Brokerchain [7], each shard in Orbit uses a state tree structure [19] to maintain the state of all accounts within the shard, and the root hash of the state tree, *StateRoot*, represents the current world state of the shard. However, unlike existing solutions where the set of accounts maintained by a shard is static within each epoch, it is dynamically changing in Orbit, as shards can schedule accounts from other shards

The main workflow of Orbit is illustrated in Fig.2, and these stages are briefly introduced as follows:

1) **Transaction Allocation Phase.** The allocator collects all pending TXs and the latest account partitioning. Before the shard consensus, it will formulate an optimal strategy for the allocation of both accounts and TXs. This strategy is subsequently distributed to each shard to reduce the number of cross-TXs and achieve a balanced workload across the shards while processing these pending TXs. The specific allocation strategy will be detailed in Section IV.
2) **Account Scheduling Phase.** Upon receiving allocated TXs, a shard first checks whether all accounts involved in the TXs are within the shard. If an account is missing, the shard attempts to schedule this missing account from another shard to its own based on the given allocation strategy. Once the scheduling is successful, the TX is converted into a less costly intra-TX. Otherwise, the shard continues to process this cross-TX following the relay mechanism in Monoxide [2].
3) **Shard Consensus Phase.** Finally, the shard packages and executes all allocated TXs, updates the account states, and reaches consensus on the new block.

Next, we describe the critical operations and related data structures in Orbit: the account scheduling, the deadlock resolution.

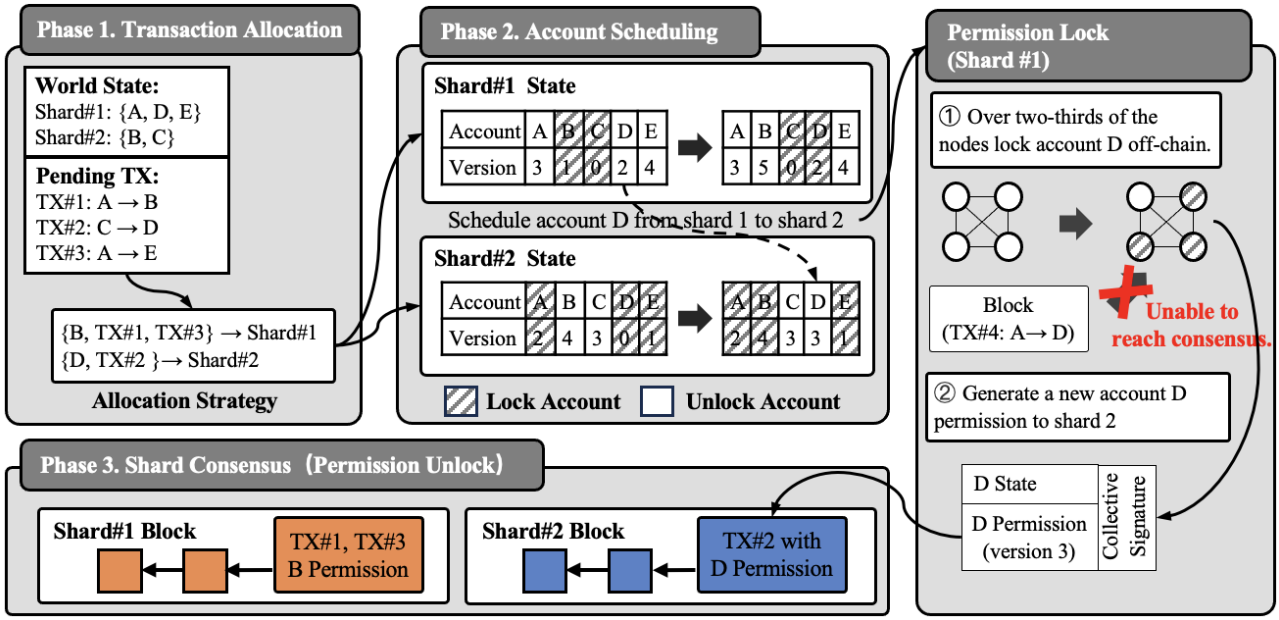


Fig. 2. The workflow of Orbit.

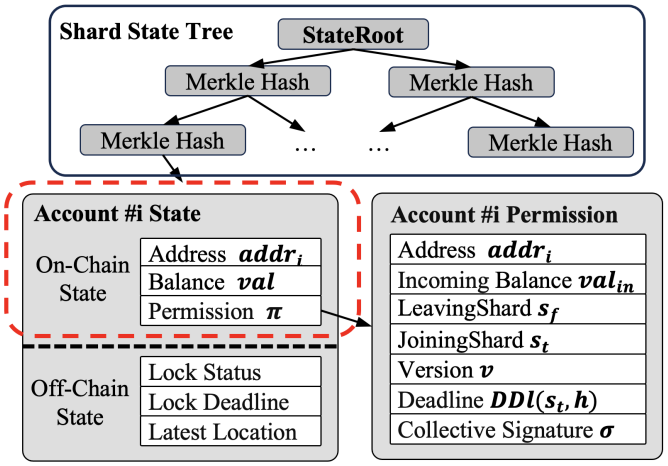


Fig. 3. Data structure of the account state and the account permission.

B. The Account Scheduling

In Orbit, the efficiency of account scheduling directly affects whether the allocation strategy can be deployed before transactions are packaged. However, existing scheduling methods [7], [20], which rely on shard consensus for security, do not meet the efficiency demands of dynamic account allocation.

To address this challenge, we design an off-chain permission that allows for off-chain account scheduling through locking and unlocking permissions in corresponding shards, bypassing on-chain consensus and improving efficiency. To achieve this, we design a new data structure for the on-chain and off-chain account states. Additionally, we establish rules for locking and unlocking account permissions to ensure the security of the scheduling process.

Account State Design. In Orbit, account state is divided into two parts: on-chain and off-chain, as shown in Fig.3.

The on-chain state consists of the basic data of an account, including its address $addr$, balance val, and permission certificate π. The permission certificate π serves as valid proof for an account's migration from one shard to another, containing several key elements: the account address $addr$, the balance val_{in} at the time of migration, the leaving shard s_t, the joining shard s_t, the version number v, the deadline for the certificate's validity ddl, and the collective signature of the leaving shard σ_*. These on-chain states are stored in the state trees of each shard and can only be updated through shard consensus, preventing any independent modifications by a single node.

The off-chain state of an account mainly refers to its scheduling status during operation, including the current lock status of the account, the validity period of the lock message, and the account's latest location within the system. The lock status is a critical factor in determining whether a node can update an account's on-chain state. When an account is locked by a node, it implies that the account might have been scheduled to another shard. Therefore, the node will reject blocks involving updates to that account until it is unlocked. Unlike the on-chain state, the off-chain state is stored locally on the node, allowing the node to modify it freely.

In Orbit, as an account may move between multiple shards, the state data of the same account can exist in several shards simultaneously, with each shard having a different version of the state. To ensure that only one shard's account state is valid in the system, we introduce an incrementing version number. Each time an account is scheduled, its version number also increases accordingly. Only the account state in the shard with the highest version number is considered valid. In other words, that account's *LockStatus* is only unlocked in the nodes of this valid shard, while it remains locked in the nodes of all other shards.

Next, we introduce how nodes in orbit implement account scheduling by locking and unlocking accounts. For clarity, we use the example of Shard 2 scheduling account D from Shard 1 to illustrate the specific process, as shown in Fig.2.

Permission Locking. The leader of Shard 2 sends a lock

request to Shard 1 in the format $\langle lock, addr_D, ddl(s_2, h)\rangle_{\sigma_2}$, where $addr_D$ is the address of account D, σ_2 is the leader's signature, and $ddl(s_2, h)$ indicates that the request must be completed before the block at height h in Shard 2; otherwise, it expires.

Upon receiving the lock request, the leader of Shard 1 checks the *LockStatus* of account D. If account D is already *locked*, the node rejects this locking operation. Otherwise, the node sets its *LockStatus* to *locked* and updates the *LockDeadline* to $ddl(s_2, h)$. Once account A is locked, the leader generates a new account permission certificate $\pi' = \langle addr_D, val, s_1 \to s_2, v_{s_1} + 1, ddl(s_2, h)\rangle_{\sigma_1}$, where val and v_{s_1} are the latest balance and version number of account D, respectively. Then, the leader broadcasts this lock request within Shard 1 to lock other nodes. If other nodes successfully lock account D with the same balance, they sign the certificate π' and broadcast it within the shard. To reduce communication overhead, we use collective signatures [3], [21], which can generate a multi-signature among a group of nodes. Therefore, as long as the collective signature σ in π' includes more than two-thirds of the majority of nodes, the permission lock is successful. Finally, Shard 1 returns the newly generated permission π' to Shard 2.

Once an account is successfully locked, any block attempting to update or read the balance of account D will fail to reach consensus within Shard 1. This situation arises because the locked nodes will reject any blocks involving account D. Since over two-thirds of the nodes are locked, even if one-third of these locked nodes are malicious, the block will not be able to obtain approval from more than two-thirds of the nodes, thus failing to achieve consensus within the shard. Additionally, this process does not affect other accounts that are not locked.

Permission Unlocking. When Shard 2 receives the permission certificate π', it packages π' along with the TX (Tx#2:C→D) into a block for consensus. The nodes in Shard 2 verify the permission π' according to the following rules:

- Check that the joining shard is s_2 to ensure the account is scheduled to the correct shard.
- Verify that the new version $v_{s_1} + 1$ is greater than the local version v_{s_2} to ensure up-to-date account data.
- Verify if the block height meets $ddl(s_2, h)$ requirement.
- Verify that the signature σ is correct and contains sufficient nodes in shard 1.

If the new permission passes all verifications, Shard 2 updates the on-chain balance and permission of account D to reflect its new state $\{addr_A, val, \pi'\}$, and changes the off-chain *LockStatus* of account D from *locked* to *unlocked*. Subsequently, Tx#2:C→D is treated as an intra-TX and executed, updating the balances of accounts C and D. After the consensus is completed, Shard 2 broadcasts a notification to other nodes to update the latest location of account D to s2.

Additionally, nodes stop waiting if the lock request is rejected or the account is not successfully locked within the specified deadline $ddl(s_2, h)$. In such cases, all TXs involving accounts that fail to be scheduled are processed following the rules for cross-TXs.

C. Deadlock Resolution

However, since the locking process occurs off-chain without shard consensus, deadlock situations may arise, especially in the event of malicious attacks. For instance, when two shards attempt to lock the same account simultaneously, each shard's lock request only receives support from half of the nodes. This situation leads to a deadlock for the account: the old permission is locked and invalid, but neither of the two new permissions can take effect due to insufficient signatures. In such a scenario, no shard can update that account.

To resolve potential deadlock issues, we set a deadline ddl for each lock request. Suppose Shard 2 does not receive new permissions within the given deadline $ddl(s_2, h)$. In that case, Shard 2 can broadcast a withdrawal message $\langle addr_D, s_1 \to s_2, \pi_{s_2}, \omega\rangle$ in Shard 1, where π_{s_2} is the permission state of account D in the block at height h of Shard 2, and ω is a Merkle state tree path to prove the validity of π_{s_2}. Nodes in Shard 1 can determine whether the lock request of Shard 2 is successful by comparing version numbers in the current on-chain state π_{s_1} and the provided π_{s_2}. If the version in π_{s_2} is lower than in π_{s_1}, it indicates a lock failure, meaning Shard 2 did not successfully obtain account D's permission before $ddl(s_2, h)$. Therefore, Shard 1 can unlock the account and clear the *LockDeadline*, thereby resolving the deadlock.

Furthermore, in our scheduling strategy's design(section IV), we limit an account to moving only once per scheduling cycle to prevent deadlock from multiple shards trying to schedule the same account simultaneously. Additionally, we set the lock deadline to 1-2 blocks in the future, ensuring lock success while minimizing the duration of potential deadlocks. This way, even if a lock fails, deadlocks can be quickly resolved, preventing accounts from being locked for extended periods.

D. Design Refinements

Parallel Optimization In the above design, a shard locking an account while handling scheduling requests from other shards can hinder its internal consensus. However, if the account to be scheduled is not associated with the block currently under consensus, it can be processed in parallel.

Therefore, we make a parallel improvement to Orbit. During the consensus process, nodes can set the *LockStatus* of accounts involved in the consensus block to an *unlockable* status and restore it after the consensus commitment. Any scheduling request attempting to lock an *unlockable* account is suspended until the status is restored, thereby avoiding conflicts between the scheduling request and the consensus block. To further reduce the potential for conflict, Orbit first locks the leader node before locking other nodes to ensure the block does not contain TXs conflicting with the scheduling request. Therefore, Orbit can parallelly complete the block consensus and account scheduling.

Pipeline Optimization. Since initiating scheduling requests does not conflict with shard consensus, Orbit can simultaneously schedule accounts for the next block according to the

scheduling strategy while the previous block is undergoing consensus, which can reduce the TX processing delay.

Lockstatus synchronization. Since the locking of accounts occurs off-chain without consensus, there are minor discrepancies in the off-chain account state sets among nodes. For example, an account that has already been transferred may still appear as unlocked in the databases of a few nodes. To avoid significant data discrepancies between nodes, Orbit periodically reaches a consensus on the newly locked accounts to help nodes synchronize with the latest lock status.

E. Security analysis

Atomicity. As long as the number of malicious nodes in a shard is less than 1/3, account scheduling operations in Orbit are atomic. Consider the scenario where shard 2 schedules account D from shard 1. If shard 2 successfully obtains the permission of account D within a specified deadline, shard 1 can no longer update the states of account D. Since more than two-thirds of the nodes in shard 1 have already locked account D, and the number of malicious nodes is less than one-third, any operation on account D by shard 1 cannot gain the support of more than two-thirds of the nodes, whether for consensus or for generating new permissions directed to other shards. Suppose shard 2 fails to lock within the given deadline. In that case, all nodes that have already locked can unlock using a publicly verifiable withdrawal message, which anyone can generate based on the public block information of shard 2. Additionally, the presence of version numbers prevents the reuse of expired permission credentials, thus avoiding replay attacks. Consequently, as long as the shard is safe, the account scheduling operations in Orbit are atomic.

Long-term locking problem. The long-term locking issue is when an account is locked for an extended period, rendering it unusable. We implement several measures in Orbit to mitigate the issue of long-term account locking by malicious nodes. First, we require that each scheduling request includes the signature of the allocator. This signature validates that the request is part of the intended scheduling strategy, not a product of malicious intent, effectively guarding against DDoS attacks. Additionally, we limit the deadline for account locks to only 1-2 future blocks, significantly reducing the window for potential malicious locking. Furthermore, we plan to introduce economic incentives as a deterrent to increase the cost of malicious behavior, such as charging fees for instances where an account is locked for an extended period without being used.

IV. ALLOCATION STRATEGY OPTIMIZATION

In Orbit, the allocator dynamically adjust the account partitions based on pending TX data and allocates these TXs to the corresponding shards. Since TXs can essentially be allocated to any shard, there are countless potential allocation strategies. A allocation strategy's effectiveness directly impacts the sharding system's performance. A poor allocation strategy could lead to an increase in cross-TXs, an imbalance in workload distribution among shards, or excessive account scheduling overhead, thereby harming the overall performance of the sharding system.

In this section, we introduce the optimization algorithm of the allocation strategy in Orbit, specifically addressing which accounts and TXs should be allocated to which shards to optimize the overall performance of the Orbit.

A. Problem Statement

To formalize the allocation problem, we model the pending TX data as a directed graph $G(V, E)$, where $V = \{1, 2, ..., N\}$ denotes the set of nodes in the graph, representing the accounts to be updated, and $E = \{e_{i,j}\}$ represents the set of directed edges, indicating the TX relationships between accounts. The edge weight $e_{i,j}$ corresponds to the number of TXs from account i to j. However, an edge can only connect two nodes, but a TX may involve multiple inputs and outputs. We address this issue by decomposing the TX into multiple one-to-one edges with the same weight. Suppose a TX has $|V_{in}|$ inputs and $|V_{out}|$ outputs, the TX is split into $|E_{tx}| = |V_{in}| \cdot |V_{out}|$ edges and each with a weight of $1/|E_{tx}|$, to ensure the total weight of the TX remains 1.

Assuming there are K shards in the system, we use a two-dimensional array $X = \{x_{i,k} | i \in V, k \in \{1, 2, ..., K\}\}$to represent the distribution of account permissions, where $x_{i,k} = 1$ if account i belongs to shard k, and $x_{i,k} = 0$ otherwise. Additionally, $x_{i,k}$ must satisfy $\sum_{k=1}^{K} x_{i,k} = 1$, as each account can be updated by only one shard at any specific moment. Since account allocation in Orbit is dynamic, we use $X^t = \{x_{i,k}^t\}$ to represent the account allocation result in Orbit during the t-th round of consensus. Similarly, $E^t = \{e_{i,j}^t\}$ represents the pending TXs in the t-th round of consensus

Next, we model the workload of each shard. In Orbit, the workload of shard k, denoted as W_k, includes intra-TXs, cross-TXs, and account scheduling workload. Assuming the workload of a single intra-TX is 1, the workload of intra-TXs is the number of TXs occurring between accounts within the same shard, which can be expressed as:

$$W_{k,intra} = \sum_{i=1}^{N} \sum_{j=1, j \neq i}^{N} e_{i,j}^t \cdot x_{i,k}^t \cdot x_{j,k}^t, \tag{1}$$

where N is the number of accounts.

Similarly, the workload for cross-TXs is represented by the TX weights between accounts in different shards, which is expressed as

$$W_{k,cross} = \sum_{l=1, l \neq k}^{K} \sum_{i=1}^{N} \sum_{j=1, j \neq i}^{N} e_{i,j}^t \cdot x_{i,k}^t \cdot x_{j,l}^t + \sum_{i=1}^{N} e_{i,i}^t \cdot x_{i,k}. \tag{2}$$

Since Orbit employs the relay mechanism to handle cross-TXs, the cross-TX (i→j) is first executed in the sender shard and then a relay-TX (j→j) is sent to the receiver shard. Therefore, in equation (2), the first term calculates the workload of cross-TXs initiated by shard k, and the second term calculates the workload of relay TXs received by shard k.

Additionally, the workload for account scheduling is related to the number of accounts being scheduled. We set the workload of scheduling one account as λ, where λ is related to the size and complexity of the account data. Therefore, the workload of account scheduling can be expressed as:

$$W_{k,shedul} = \lambda \sum_{i=1}^{N} |x_{i,k}^{t} - x_{i,k}^{t-1}|. \tag{3}$$

Therefore, the total workload of shard k is

$$W_k = W_{k,intra} + W_{k,cross} + W_{k,shedul} \tag{4}$$

And, the variance of the shard workload is

$$Var_w = \sum_{i=1}^{K} (W_k - \overline{W})^2 / K, \tag{5}$$

where $\overline{W} = \sum_{i=1}^{K} W_k / K$.

Finally, we model the throughput of each shard. Based on [11], we assume that the maximum workload capacity of each shard per consensus round is γ. If a shard's workload is within this capacity, i.e., $W_k < \gamma$, it can process all allocated TXs. Otherwise, only a portion of the TXs is successful. Unlike existing semi-static allocation schemes, the throughput of shards in Orbit is influenced by the TX execution order, as accounts are dynamically allocation. A cross-TX may become an intra-TX in the subsequent allocation round. Therefore, Orbit prioritizes packing intra-TXs to achieve optimal throughput. The throughput of shard k can be defined as:

$$T_k = \begin{cases} W_{k,intra} + \frac{1}{2} \cdot W_{k,cross}, & if \quad W_k < \gamma \\ \gamma - W_{k,shedul} + T_{k,cross}, & if \quad W_{k,intra} < \gamma \\ \gamma - W_{k,shedul}, & if \quad W_{k,intra} \geq \gamma \end{cases}. \tag{6}$$

where $T_{k,cross} = \frac{1}{2} min(W_{k,cross}, \gamma - W_{k,shedul} - W_{k,intra})$.

Based on this model, we can express the account allocation issue as an following optimization problem. The goal is to find a allocation strategy X^t that maximizes the throughput of shards given the pending TX graph G^t.

$$\begin{aligned} \arg\max_{X^t} \quad & T(X^t) = \sum_{i}^{K} T_k(X^t) \\ s.t. \quad & \textstyle\sum_i^K x_{i,k}^t = 1, \quad \forall i \in \{1, 2, ..., N\} \\ & x_{i,k}^t \in \{0, 1\} \quad \forall i \in \{1, 2, ..., N\}, k \in \{1, 2, ..., K\} \end{aligned} \tag{7}$$

B. Strategy Resolution

To solve this problem, we propose an optimization algorithm based on TxAllo [11] to calculate the optimal account allocation strategy. Our algorithm takes as input the pending TX data, the current distribution of account permissions, shard processing capabilities, and a convergence threshold. It outputs a account and TX allocation results, which are then distributed to the sharding system for TX processing. The pseudocode for strategy resolution in Orbit is shown in Algorithm 1.

Similar to TxAllo, the strategy resolution in Orbit is divided into initialization and optimization phases. In the initialization phase, as shown in lines 1-5 of the Algorithm 1, we randomly allocate new accounts to various shards to form the initial values for the algorithm based on the current distribution of accounts. In the optimization phase, we iteratively calculate the best shard for each account. For example, for account i, we calculate the throughput gain ΔT by moving it from its current shard p to other shards. Then, we assign it to the shard where the gain ΔT is maximized, thus achieving the highest performance.

Algorithm 1: The allocation strategy in Orbit

Input : Pending transactions $G^t = (V, E^t)$; Current account allocation X^{t-1}; Shard processing capacity γ; Convergence threshold ε.

Output: The result of account and transaction allocation, X^t and $\{E_1^t, E_2^t, ..., E_k^t\}$.

1 $X^t \leftarrow X^{t-1}$;
2 **for** $i \in V$ **do**
3 **if** $\sum_{l=1}^{K} x_{i,l}^t == 0$ **then**
4 $k \leftarrow random(0, K)$;
5 $x_{i,k} \leftarrow 1$;
6 **while** $\Delta T > \varepsilon$ **do**
7 **for** $i \in V$ **do**
8 $p \leftarrow$ the current shard of account i;
9 **for** $q \in \{1, ..., p-1, p+1, ..., K\}$ **do**
10 *Calculate TPS gain $\Delta T_{i,p,q}$ if account i is moved from shard p to shard q*;
11 $q \leftarrow \arg\max_q \Delta T_{i,p,q}$;
12 **if** $\Delta T_{i,p,q} > 0$ **then**
13 $x_{i,p}^t, x_{i,q}^t \leftarrow 0, 1$;
14 *Update the TPS and workload of shard p and q*;
15 $\Delta T += \Delta T_{i,p,q}$
16 $\{E_1^t, E_2^t, ..., E_k^t\} \leftarrow$ Allocate the pending transactions E^t to various shards based on X^t;
17 **return** X^t *and* $\{E_1^t, E_2^t, ..., E_k^t\}$;

However, the introduction of scheduling overheads in Orbit make this calculation more complex. When account i moves from shard p to shard q, the TXs between account i and accounts in shard q are transformed into intra-TXs. The increased intra-TX load can be represented as follows:

$$W'_{q,intra} = W_{q,intra} + \sum_{j=1, j \neq i}^{N} (e_{i,j}^t + e_{j,i}^t) \cdot x_{j,q}^t. \tag{8}$$

Additionally, other TXs initiated by account x and the relay TXs it receives become part of the cross-TX load for shard q, represented as:

$$W'_{q,cross} = W_{q,cross} + e_{i,i}^t + \sum_{l=1, l \neq q}^{K} \sum_{j=1, j \neq i}^{N} e_{i,j}^t \cdot x_{j,l}^t. \tag{9}$$

In addition, the scheduling of the account i introduces new scheduling overheads for shard q, represented as:

$$W'_{q,shedul} = W_{q,shedul} + \lambda \cdot (1 - x_{i,q}^{t-1}) - \lambda \cdot x_{i,q}^{t-1}. \quad (10)$$

Where $x_{i,q}^{t-1} = 0$ indicates that account i has migrated to a new shard, which adds a scheduling overhead of λ to shard q. Conversely, $x_{i,q}^{t-1} = 1$ means that account i has returned to its initial shard q , and we need to deduct the cost λ previously incurred when the account left shard q.

Similarly, we can also calculate the workload change caused by the account $i$leaving shard p:

$$W'_{p,intra} = W_{p,intra} - \sum_{j=1,j\neq i}^{N} (e_{i,j}^{t} + e_{j,i}^{t}) \cdot x_{j,p}^{t}. \quad (11)$$

$$W'_{p,cross} = W_{p,cross} - e_{i,i}^{t} - \sum_{l=1,l\neq p}^{K} \sum_{j=1,j\neq i}^{N} e_{i,j}^{t} \cdot x_{j,l}^{t}. \quad (12)$$

$$W'_{p,shedul} = W_{p,shedul} + \lambda \cdot x_{i,p}^{t-1} - \lambda \cdot (1 - x_{i,p}^{t-1}). \quad (13)$$

Subsequently, we can use Equation (6) to evaluate the performance gain obtained by moving account i from shard p to shard q, defined as $\Delta T_{i,p,q} = T'_{i,p} - T_{i,p} + T'_{i,q} - T_{i,q}$, where $T'_{i,p}$ and $T'_{i,q}$ are the throughput performances of shards p and q after scheduling, respectively. Therefore, we can maximize the shard throughput performance by assigning account i to the shard with the highest gain $\Delta T_{i,p,q}$, as shown in line 10 of Algorithm 1.

To accelerate convergence, we introduce an update threshold ε. When the gain obtained from strategy optimization falls below this threshold, the system will stop and return the optimal allocation result. Since Orbit is based on analyzing pending TXs, its convergence speed is only related to the number of TXs in the TX pool and does not extend with the increase in the number of blocks. In stable system conditions, the convergence speed of the algorithm in each round can be considered consistent.

C. Security analysis

In Orbit, the allocation strategy serves as a recommendation for block generation rather than a critical security component. The effectiveness of this strategy impacts the system's performance but not its security. Evaluating the performance of a allocation strategy is relatively easy; it involves analyzing aspects like cross-TXs, load balancing, and the presence of any conflicting schedules. Therefore, Orbit employs an optimistic approach, allowing shards to check the efficiency of the received strategy initially. A more thorough assessment of the system's overall performance is conducted at the end of each epoch, determining whether the allocator receives rewards or penalties.

V. Performance Evaluation

A. Settings

Simulation implementation: To evaluate the proposed Orbit mechanism, we use Python to develop a TX-driven shard simulator, which replays historical TXs to measure the performance of the scheduling mechanism in detail. In the simulation experiment, each block contains a maximum of 1,000 TXs, and consensus latency is set to an average of 5 seconds. The parameters λ, γ, and ε in the scheduling strategy are set to 0.1, 1000, and 1.0, respectively.

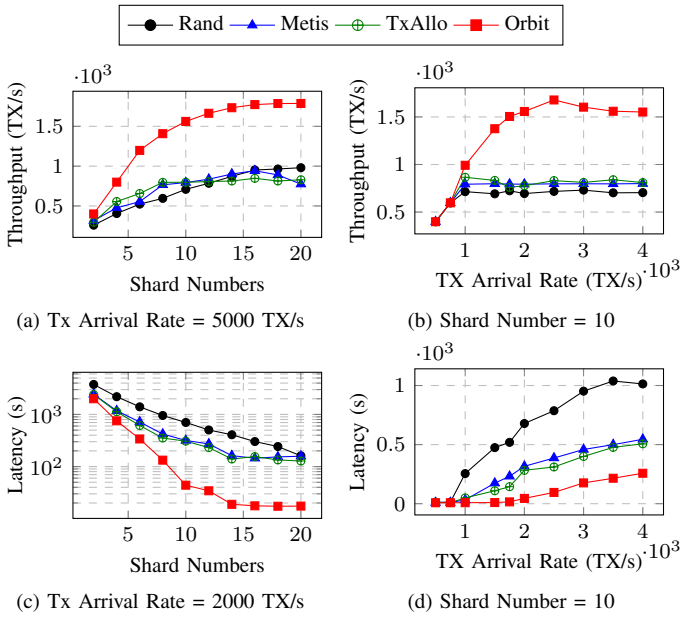
Baseline: We compare the Orbit mechanism with three following schemes: Rand [2], which allocates accounts randomly based on the first few bits of the address in existing sharding systems; Metis [7], a well-known graph-based partitioning method that optimizes account placement in sharding systems; and TxAllo [11], a state-of-the-art account partitioning scheme emphasizing load balance through community detection.

Dataset and Usage: We use blocks from Ethereum [18], ranging from height 12,000,000 to 12,200,000, as our dataset, containing approximately 39.98 million TXs. The test period is set to 20 epochs, with an average of 10,000 blocks processed in each epoch. The first epoch serves to initialize the baseline systems. Afterward, accounts are reallocated at the beginning of each epoch based on previous historical TXs, and TXs are replayed at a specific rate to measure system performance.

Metrics: We study the impact of critical parameters such as the number of shards, TX arrival rate, and account scheduling overhead on system performance, including throughput, TX latency, TX pool queue size, and shard workload variance. Simultaneously, we also focus on additional overhead metrics introduced by Orbit, such as lock latency, strategy computation time, and bandwidth costs.

B. Throughput and TX Latency

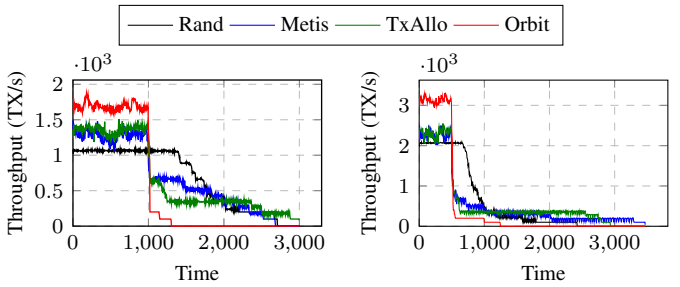
We first evaluate the system average throughput and TX latency under various shard numbers and TX arrival rates. Simulation results are presented in Fig. 4. Fig. 4a indicates that, with a fixed TX arrival rate of 5000 TX/s, the average throughput of the sharding system increases with the number of shards. However, as the shard number exceeds 10, the throughput growth rate significantly slows down and eventually converges. Orbit outperforms other solutions, exhibiting the highest throughput performance, ranging from 1.82 to 2.15 times that of other approaches. In Fig.4b, we fix the shard number at ten and increase the TX arrival rate from 500 TX/s to 4000 TX/s. The average throughput of Rand, Metis, and TxAllo reaches the maximum when the TX arrival rate reaches 1000 TX/s and remains constant thereafter. In contrast, Orbit can keep TPS growing until the TX arrival rate reaches 2500 TX/s. At this time, Orbit's average throughput reaches the maximum of 1678 TPS, surpassing other schemes by 2.02 to 2.34 times. Figs. 4c and 4d compare the TX latency of Orbit with existing solutions. In Fig. 4c, when the shard number is below 14, Orbit's TX latency decreases as the number of shards increases. This is because the TX arrival



(a) Tx Arrival Rate = 5000 TX/s (b) Shard Number = 10

(c) Tx Arrival Rate = 2000 TX/s (d) Shard Number = 10

Fig. 4. The system average throughput and TX latency under various shard numbers and TX arrival rates

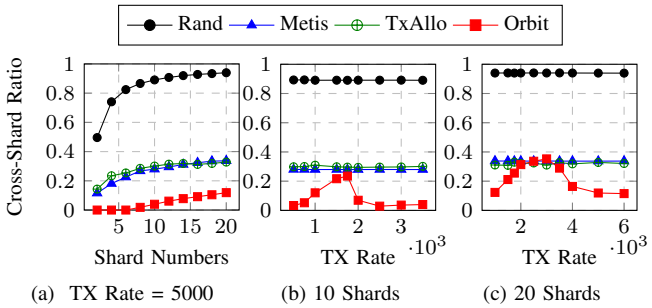


(a) 10 Shards, 2000 TX/s Arrival Rate (b) 20 Shards, 4000 TX/s Arrival Rate

Fig. 5. The real-time throughput of the system during a single epoch while processing 2 million transactions.

rate of 2000 TX/s exceeds the system processing capacity, and congestion occurs. When the shard number surpasses 14, congestion alleviates, and Orbit's TX latency converges, reaching a minimum of 17.44 seconds with 20 shards, reducing latency by 86.3%-89.1% compared to other approaches. Fig. 4d demonstrates that in a 10-shard system, Orbit can maintain TX latency below 15.25 seconds when the TX arrival rate is below 1750 TX/s, reducing latency by 89%-97% compared to other solutions. However, as the TX arrival rate exceeds 1500, Orbit's TX latency rapidly increases due to system congestion. Nevertheless, Orbit's TX latency remains significantly lower than that of other approaches.

To further demonstrate the throughput of the system, we measure the real-time changes in the system's throughput during a single epoch while processing 2 million TXs, as shown in Fig. 5. In Fig. 5a, we observe significant fluctuations in the real-time performance of the 10-shard system under the Rand, Metis, and TxAllo mechanisms. During the first 1000 seconds, the real-time throughput of all four schemes remains relatively stable, with Orbit achieving the highest at 1873 TPS, while Rand, Metis, and TxAllo reach their peak at 1089, 1492, and 1508 TPS, respectively. However, after



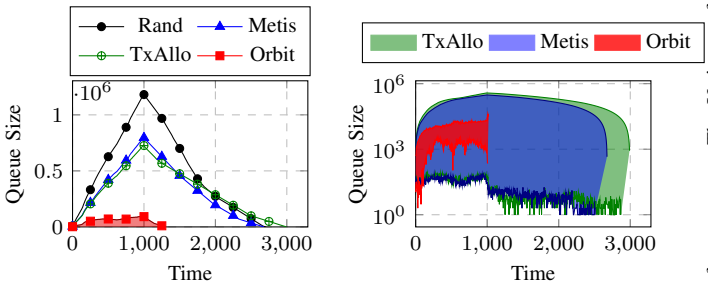
(a) TX Rate = 5000 (b) 10 Shards (c) 20 Shards

Fig. 6. The ratio of cross-shard TX under various shard numbers and TX arrival rate

1000 seconds, the real-time throughput of Metis and TxAllo gradually decreases to 400-600 TPS and further drops to less than 200 TPS at 2400 seconds, eventually completing all TXs in nearly 3000 seconds. This happens because at 1000 seconds, all test TXs are sent, and most shards pause due to a lack of TXs. However, due to the imbalance of shard workloads, a few shards still have a large backlog of unprocessed TXs, causing the system to continue operating at a very low performance for another 2000 seconds to complete the TX processing. In contrast, Orbit, through its dynamic scheduling mechanism based on pending TXs, can transfer backlog TXs to idle shards for processing, thus balancing the workload of the shards. Therefore, it achieves higher real-time performance in the first 1000 seconds and completes all TX processing within 300 seconds after the TXs stop. Fig. 5b shows that even with 20 shards and a TX arrival rate of 4000 TX/s, Orbit still maintains a higher real-time TPS and faster completion time than existing solutions. Notably, in both Figs. 5a and 5b, Orbit ceases operation around 1250-1300 seconds. This is because 10% of the test TXs are related to the same account, meaning that no matter how the TXs are scheduled, the shard containing that account has a minimum completion time of at least 1000 seconds, and Orbit comes very close to this time.

C. Cross-Shard Ratio

We evaluate the ratio of cross-TXs under various scheduling mechanisms with different shard numbers and TX arrival rates, as Fig.6 shows. In Fig. 6a, fixing the TX arrival rate at 5000 and increasing shard numbers from 2 to 20, Orbit consistently has the lowest cross-TX ratio among the four schemes. For 2 to 6 shards, Orbit's cross-TX ratio is 0%, and it increases with the number of shards, reaching a peak of 11.9% at 20 shards, compared to 94%, 33.7%, and 32.8% for Rand, Metis, and TxAllo, respectively. Thus, Orbit's dynamic account scheduling effectively minimizes cross-TXs. In Fig.6b and Fig.6c, with fixed shard numbers, we increase the TX arrival rate from 500 to 6000 TX/s. Rand, Metis, and TxAllo maintain constant cross-TX ratios due to their static account partitioning within an epoch. In contrast, Orbit's cross-TX ratio initially increases, then decreases, and eventually stabilizes. This change occurs as Orbit prioritizes shard workload balancing over reducing cross-TXs when TX arrival rates are low but some shards are overload. For instance, in Fig.6b, as the TX arrival rate increases from 500 to 1500 TX/s, Orbit schedules backlog TXs



(a) Queue size of the system TX pool. (b) Queue size of each shard TX pool.

Fig. 7. Queue size of the TX pool in a 10-shard system with a TX arrival rate of 2000 TX/s. In 7b, each region's upper and lower boundaries correspond to the maximum and minimum queue sizes of the shards, respectively.

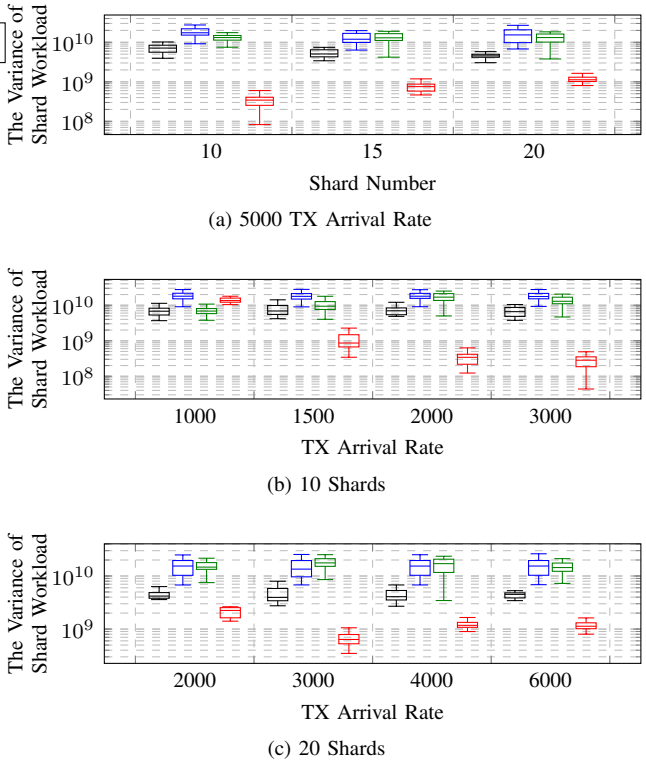
to idle shards rather than the most relevant ones, increasing cross-TXs but also enhancing system throughput, as shown in Fig. 4b. When the rate rises from 1500 to 3000, and all shards become overloaded, Orbit prioritizes reducing cross-TXs to lower overall load, directing accounts and TXs to the most relevant shards. A similar trend is observed in 20-shard systems, as shown in Fig.6c.

D. TxPool Size

We also assess TX pool changes by monitoring queue sizes in a 10-shard system, as shown in Fig.7. We sent 2 million TXs at a rate of 2000 TX/s. Fig. Fig.7a shows changes in the overall system TX pool: queue size grows initially due to continuous TX injection and decreases after TXs stop being sent. Orbit maintains a more stable and shorter queue than Rand, Metis, and TxAllo. Fig.7b further shows TX distribution across shards. Compared to Metis and TxAllo, Orbit shows smaller variations in shard queue sizes, consistently maintaining the minimum queue size above 1000 (the size of a block). When a shard becomes idle (< 1000), Orbit quickly redistributes more TXs to idle shards to maintain efficiency, whereas Metis and TxAllo frequently encounter scenarios with idle shards.

E. The Variance of Shard Workload

To evaluate Orbit's workload balancing, we measure shard workload variance under different scheduling mechanisms across various shard numbers and TX arrival rates. We record the variance over 19 epochs in each test set, with the results presented in Fig.8. In Fig.8a, at consistent TX arrival rates, Orbit's shard workload variance significantly surpasses the baseline mechanism, gradually increasing as the number of shards rises. Fig.8b and Fig.8c illustrate that Orbit's shard workload variance is comparable to the baseline mechanism at a low TX arrival rate. However, it progressively decreases and eventually stabilizes as the TX arrival rate climbs. This occurs because, at lower rates, all shards are idle, prompting Orbit to focus on scheduling accounts to reduce cross-TXs, as shown in Fig.6a. As the rate escalates and some shards begin to overload, Orbit redirects TXs toward idle shards, balancing the load across shards.



(a) 5000 TX Arrival Rate

(b) 10 Shards

(c) 20 Shards

Fig. 8. The workload variance under various shard numbers and TX arrival rates. In the above boxplot, the black, blue, green, and red boxes respectively correspond to Rand, Metis, TxAllo, and Orbit.

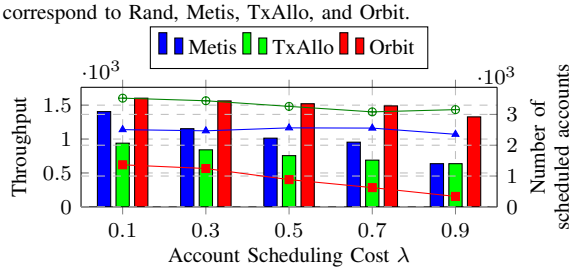


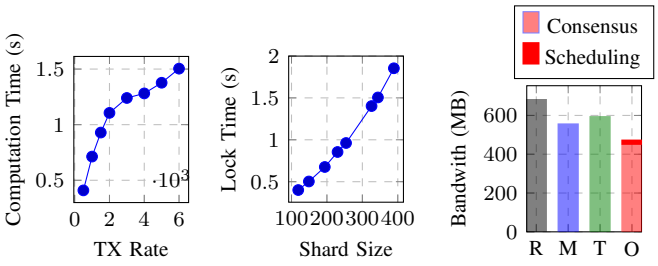
Fig. 9. The system's average throughput and the number of scheduled accounts per round under different account scheduling costs. The bar graph represents throughput, and the line graph indicates the number of scheduled accounts.

F. Account Scheduling Cost

To compare the efficiency of scheduling strategies, we integrate Metis and TxAllo into our dynamic scheduling mechanism. We measure the system's average throughput and the average number of accounts scheduled per consensus round under various account scheduling costs, as shown in Fig.9. In these three schemes, Orbit consistently shows the highest average throughput and schedules fewer accounts than Metis and TxAllo, with the number decreasing as the cost increases.

G. Time cost and bandwidth overhead.

In Fig.10, we assess the average solution time for different scheduling strategies at various TX arrival rates and the average time required for account locking under different shard sizes. In Fig.10a and 10b,the scheduling process completes within 3.5 seconds, even in the worst-case scenarios. This duration is generally shorter than the consensus interval in



(a) Strategy Solution (b) Lock Time (c) Bandwidth Cost

Fig. 10. Time cost and bandwidth overhead of the scheduling mechanism. In 10c, the labels R, M, T,and O respectively represent Rand, Metis, TxAllo, and Orbit.

existing shard systems [2]–[4], enabling Orbit to complete the account schedule before the next block begins.

Additionally, we evaluate the average bandwidth consumption of nodes processing 2 million TXs under different scheduling mechanisms, as depicted in Fig. 10c. With the dynamic scheduling feature, Orbit's bandwidth expenditure comprises consensus and scheduling costs. These scheduling costs encompass account off-chain locking and unlocking, off-chain state synchronization, and updates on shard positions. In Fig. 10c, The bandwidth overhead in Orbit is 475.5MB, which is 30.6%, 14.9%, and 20.3% lower than Rand, Metis, and TxAllo, respectively. Scheduling expenses account for 5.7% of Orbit's total costs, indicating that the additional bandwidth costs introduced by Orbit's dynamic scheduling are substantially offset by the savings from reducing cross-TXs and balancing workloads.

VI. Conclusions

In this paper, we propose Orbit, a dynamic account scheduling mechanism based on pending TXs. Orbit achieves efficient and dynamic movement of accounts across different shards through an off-chain account scheduling mechanism. This mechanism, coupled with a strategic optimization approach, effectively reduces cross-shard TXs and achieves load balancing. Simulation experiments demonstrate that Orbit excels in various aspects, including throughput, latency, cross-shard TX ratio, load balancing, and bandwidth overhead, outperforming state-of-the-art sharding methods.

References

[1] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: ACM, 2016, pp. 17–30. [Online]. Available: http://doi.acm.org/10.1145/2976749.2978389

[2] J. Wang and H. Wang, "Monoxide: Scale out blockchains with asynchronous consensus zones," in *Proceedings of the 16th {USENIX} Symposium on Networked Systems Design and Implementation*, ser. NSDI '19, 2019, pp. 95–112.

[3] E. K. Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "Omniledger: A secure, scale-out, decentralized ledger via sharding," in *Security and Privacy (SP), 2018 IEEE Symposium on*. Ieee, 2018.

[4] M. Zamani, M. Movahedi, and M. Raykova, "Rapidchain: Scaling blockchain via full sharding," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: ACM, 2018, pp. 931–948. [Online]. Available: http://doi.acm.org/10.1145/3243734.3243853

[5] G. Wang, Z. J. Shi, M. Nixon, and S. Han, "Sok: Sharding on blockchain," in *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, ser. AFT '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 41–61. [Online]. Available: https://doi.org/10.1145/3318041.3355457

[6] E. Fynn and F. Pedone, "Challenges and pitfalls of partitioning blockchains," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, 2018, pp. 128–133.

[7] 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 INFOCOM 2022 - IEEE Conference on Computer Communications*, 2022, pp. 1968–1977.

[8] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical review E*, vol. 76, no. 3, p. 036106, 2007.

[9] S. E. Garza and S. E. Schaeffer, "Community detection with the label propagation algorithm: A survey," *Physica A: Statistical Mechanics and its Applications*, vol. 534, p. 122058, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378437119312026

[10] 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)*, 2022, pp. 284–294.

[11] Y. Zhang, S. Pan, and J. Yu, "Txallo: Dynamic transaction allocation in sharded blockchain systems," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, 2023, pp. 721–733.

[12] L. Jia, Y. Liu, K. Wang, and Y. Sun, "Estuary: A low cross-shard blockchain sharding protocol based on state splitting," *IEEE Transactions on Parallel and Distributed Systems*, pp. 1–16, 2024.

[13] A. Sonnino, "Chainspace: A sharded smart contract platform," in *Network and Distributed System Security Symposium 2018 (NDSS 2018)*, 2018.

[14] Z. Hong, S. Guo, P. Li, and W. Chen, "Pyramid: A layered sharding blockchain system," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, 2021, pp. 1–10.

[15] G. Karypis and V. Kumar, "Metis: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices," 1997.

[16] L. N. Nguyen, T. D. T. Nguyen, T. N. Dinh, and M. T. Thai, "Optchain: Optimal transactions placement for scalable blockchain sharding," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, 2019, pp. 525–535.

[17] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *3rd Symposium on Operating Systems Design and Implementation (OSDI 99)*. New Orleans, LA: USENIX Association, Feb. 1999. [Online]. Available: https://www.usenix.org/conference/osdi-99/practical-byzantine-fault-tolerance

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

[19] H. S. de Ocáriz Borde, "An overview of trees in blockchain technology: Merkle trees and merkle patricia tries," 2022.

[20] M. Król, O. Ascigil, S. Rene, A. Sonnino, M. Al-Bassam, and E. Rivière, "Shard scheduler: Object placement and migration in sharded account-based blockchains," in *Proceedings of the 3rd ACM Conference on Advances in Financial Technologies*, ser. AFT '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 43–56. [Online]. Available: https://doi.org/10.1145/3479722.3480989

[21] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Advances in Cryptology—EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003 Proceedings 22*. Springer, 2003, pp. 416–432.

DOI:10.7544/issn1000-1239.202330284
卷(期):起止页，年

PieBridge：一种按需可扩展的跨链架构

段田田[1,2] 郭仪[1,2] 李博[1,2] 张瀚文[1,2] 宋兆雄[1] 李忠诚[1,2] 张珺[3] 孙毅[1,2]

[1]（中国科学院计算技术研究所 北京 100190）

[2]（中国科学院大学 北京 101408）

[3]（内蒙古大学 内蒙古 呼和浩特 010021）

（duantiantian@ict.ac.cn）

PieBridge：An On-demand Scalable Cross Chain Architecture

Duan Tiantian[1,2], Guo Yi[1,2], Li Bo[1,2], Zhang Hanwen[1,2], Song Zhaoxiong[1], Li Zhongcheng[1,2], Zhang Jun[3] and Sun Yi[1,2]

[1]（*Institute of Computing Technology, Chinese Academy of Sciences*, *Beijing* 100190）

[2]（*University of Chinese Academy of Sciences, Beijing* 101408）

[3]（*Inner Mongolia University, Hohhot 010021, China*）

Abstract Due to its decentralized and traceable nature, blockchain has been widely used in various fields such as digital currency, supply chain finance, and smart healthcare. As the demand for applications continues to expand, the need for independent blockchains to collaborate to build a broader value internet is increasing, making it urgent to study cross-chain technology. However, as blockchain develops towards the application chain model, the number of blockchains has greatly increased, and the scale of cross-chain interaction has also increased. Existing cross-chain research cannot meet the challenges posed by this trend in terms of architectural scalability and cross-chain requirements diversity. Therefore, featured by the idea of "on-demand domain", this paper proposes an on-demand and scalable cross-chain architecture called PieBridge and raises a four-layer cross-chain interaction protocol stack for the first time. By decoupling cross-chain transmission, verification, transactions and applications, our protocol stack meets the diverse requirements of cross-chain applications in terms of privacy, security, and performance. This paper implements the PieBridge prototype system and further verifies PieBridge's scalability and flexible support for diverse cross-chain interactions from both theoretical and experimental perspectives.

Key words blockchain; cross-chain; atomicity; cross-chain transfer; queuing theory

摘要 区块链由于其去中心、可溯源等特性，已被广泛应用于数字货币、供应链金融、智慧医疗等不同领域．随着应用需求的不断拓宽，各独立区块链协作以构建更广泛价值互联网的需求日益增强，因而迫切需要研究跨链技术．然而当前区块链生态规模不断扩大、丰富，异构/同构区块链间的互联互通需求也随之快速增长．现有跨链研究无法应对这种趋势对跨链研究在架构可扩展性与跨链需求多样性方面提出的挑战．针对上述问题，基于“按需建域”的理念，提出一种按需可扩展的跨链架构 PieBridge，并首次提出了一套四层跨链交互协议栈，解耦跨链传输、验证、事务与应用，满足不同跨链应用在隐私、安全、性能等方面的差异化需求．同时实现了 PieBridge 原型系统，并通过建模分析与实验验证证明了 PieBridge 的可扩展性，以及其对差异化跨链交互需求的灵活支持．

关键词 区块链；跨链；原子性；跨链传输；排队论

中图法分类号 TP391

自比特币提出以来，区块链得到了学术界、工业界的极大关注，由于其去中心、可溯源等特性，被广泛应用于数字货币、供应链金融、智慧医疗等不同领域，形成了多链、异构的区块链生态系统[1].

尽管区块链实现了区块链网络范围内的价值流通，但是各独立区块链之间呈现明显的孤岛效应．随

收稿日期：修回日期：

基金项目：国家重点研发计划(2022YFB2702902)；国家自然科学基金项目(61972382)；内蒙古自然科学基金项目(2020MS06017). This work was supported by the National Key Research and Development Program of China (2022YFB2702902); the National Natural Science Foundation of China (61972382), and the Natural Science Foundation of Inner Mongolia (2020MS06017).

通信作者：张瀚文（hwzhang@ict.ac.cn）

着应用需求的不断拓宽，各独立区块链相互联通、协作以构建更广泛价值互联网的需求日益增强. 如何融合同构、异构区块链，突破去中心化应用边界，构建更加开放、易于协作、多方共赢的区块链生态，成为当今的迫切需求. 因此，区块链的互操作性，即跨链研究，自 2016 年起得到了越来越多的关注[2].

跨链技术是跨越单一区块链的数据可信边界(共识机制作用范围)，实现独立区块链间信息、价值流通的技术.

早期跨链研究通过在任意两条有跨链需求的区块链间建立连接实现直接的跨链交互[3-5]. 然而，区块链连接建立过程繁琐，需要完成区块链的适配，包括数据结构匹配、区块链验证规则加载以及必要数据同步[1]等工作，当彼此存在跨链交互需求的区块链较多时，两两连接的方式导致每条区块链的适配复杂度高(O(n))，跨链方案可扩展性较差.

因而，当前的跨链方案大多引入中继链连接所有交互区块链[6-8]. 存在跨链交互需求的区块链仅需与中继链进行适配、建立连接后，即可在中继链的桥接下，实现与其它区块链的跨链交互，每条区块链的适配复杂度降至常数级（O(1)），但中继链的适配复杂度仍然维持在O(n).

随着区块链生态应用类型不断丰富，跨链互通的规模不断扩大，现有跨链方案面临着挑战，主要体现在以下两个方面：

1）跨链架构的可扩展性.

无论是公链技术体系还是联盟链技术体系下，异构区块链数量都在快速增长，异构/同构区块链间的互联互通需求也随之快速增长. 在公链技术体系下，由于性能、可定制性、价值捕捉等原因，开发者越来越多的转向构建独立的应用链而不是在公共区块链平台上部署智能合约[9]，Cosmos[6]、Polkadot[7]、zkSync[10]等多种平台先后出现，用于帮助开发人员快速构建应用链，仅 Cosmos 主网（2021 年启动）就有 49 条活跃应用链，测试网有 247 条应用链[11]. 在联盟链技术体系下，更是涌现出了涌现出 Fabric、FiscoBicos、长安链、趣链等大量形态各异的区块链技术平台，基于这些异构区块链技术平台，面向金融、政务服务等不同领域的行业应用，构建了大量区块链服务平台[12]. 各区块链在既有用户和价值积累的基础上，产生了与其它区块链交互的外延需求，因此区块链间互联互通的需求也快速增长，给跨链解决方案的可扩展性带来新的挑战.

然而，现有跨链方案中，中继链承载着所有跨链消息的路由、转发和验证等工作，随着接入区块链数量增多、跨链交易增多，中继链将会出现拥塞并成为跨链系统的性能瓶颈. 尽管 BitXHub[8]等方案将中继链扩展成中继链网络以支持更多区块链的接入，但这使得大量跨链交易需要经过多跳中继链处理，导致跨链交互时延大幅增加.

2）跨链需求的多样性.

一方面，不是所有区块链间都存在相同强度的跨链需求，例如：医院区块链、康复养老区块链之间存在频繁的跨链交互，医院区块链却与供应链金融的区块链之间几乎不存在跨链交互. 构建于区块链之上的应用，往往要求有序、可靠、可信的数据流通，而现有的“一通百通”的中继链跨链方案[2]难以满足不同区块链间的按需连通.

另一方面，现有跨链研究大多针对资产转移、代币互换等数字货币应用，然而区块链在各领域的广泛应用带来了更多类型的跨链应用，同时这些跨链应用对跨链交互在性能、安全、隐私等方面提出了差异化的要求，例如，跨链代币互换为了避免双花问题更注重安全性；跨链医疗数据共享为了保护患者隐私更注重隐私性；跨链城市数据共享为了保证数据的实时性更注重性能. 现有跨链架构与单一、僵化的跨链交互机制难以满足不同区块链以及不同跨链应用在隐私、安全、性能等方面的差异化跨链需求.

针对上述挑战，本文提出了一种按需可扩展的跨链架构——PieBridge，并在此基础上提出了具有独立事务层的四层跨链交互协议栈，保证跨链架构可扩展的同时支持差异化的跨链交互.

具体而言，本文引入中继域，基于“按需建域”的理念，将有交互需求的区块链划分在一个中继域内，各中继域基于域内中继链实现域内区块链的跨链互联. 一方面通过域的划分，实现跨链系统负载的合理切分，保证跨链架构的可扩展性；另一方面满足域内区块链间的按需连通，以及与它域区块链间的安全隔离. 在按需建域的基础上，针对单中继域内跨链交易数量增加造成的中继链拥塞问题，本文设计了中继域按需扩容机制，通过域内中继链复制与多中继负载均衡机制，将跨链交易约束在单一中继链中进行处理，在实现中继域通量提升的同时，避免跨链交易时延的增加. 进一步，本文针对跨链交互需要解决的跨链信息传输、跨链信任传递与跨链事务处理三个基本问题，提出了具有独立事务层的四层跨链交互协议栈. 在跨链传输层、验证层、事务层实现多种基础协议，支持应用对各层协议的灵活选择，满足不同跨链应用在性能、安全、隐私等方面的差异化需求.该协议栈首次

[1] 例如，SPV 验证需要区块头链.

[2] “一通百通”是指区块链接入跨链系统后即可与该系统内所有区块链进行跨链交互.

解耦了跨链事务与应用，抽象出独立的跨链事务层，一方面简化跨链应用的设计与开发，支持跨链应用的灵活、快速构建；另一方面为跨链事务提供系统级保障，包括跨链事务的原子性与事务间的隔离性，避免跨链应用在事务保障方面的设计、实现缺陷为所在跨链系统引入安全漏洞.

本文的主要贡献包括以下 3 个方面：

1）提出了一种按需可扩展的跨链架构——PieBridge. 基于“按需建域”的理念，将需要跨链交互的区块链按需组建成中继域，通过域的管理与性能优化，保证跨链架构的可扩展性.

2）提出了一种具有独立事务层的四层跨链交互协议栈. 在对跨链交互的功能分层的基础上，在各层实现多种基础协议，支持应用在交互过程中对各层协议的灵活选择，满足不同跨链应用在隐私、安全、性能等方面的差异化需求. 同时首次解耦跨链事务与应用，支持跨链应用简易、灵活、快速构建的同时，实现系统级跨链应用事务性保障.

3）基于所提跨链架构实现了 PieBridge 原型系统，并通过理论分析与实验验证，证明了 PieBridge 的可扩展性，以及其对差异化跨链交互需求的灵活支持.

1 相关工作

跨链研究包括跨链架构与跨链交互机制两个层面的工作，其中跨链交互机制是在跨链架构的基础上，实现跨链交互的具体方法.

1.1 跨链架构

现有跨链研究根据是否需要借助中继链可以分为直连跨链架构与中继跨链架构.

直连跨链解决方案的基本思想是有跨链需求的区块链直接实现区块链互操作.

BTCRelay[3]被认为是最早的跨链方案，其在以太坊上部署可以接收并处理比特币区块头与交易的智能合约，并借助 Relayer 在以太坊上实现基于最长链规则的比特币轻客户端，从而实现了比特币区块链与以太坊区块链的单向跨链操作.

RSK[4]是锚定比特币区块链的一个开源智能合约平台，其目标是将智能合约以可操作的形式带入比特币，实现即时支付以及高扩展性，从而为比特币生态系统增加价值和功能. 与 BTC Relay 实现的单向跨链操作不同的是，RSK 实现了与以太坊的双向跨链操作，支持在 RSK 上根据比特币资产锁定交易生成资产以及在比特币上根据 RSK 资产锁定交易解锁资产.

上述两种方案是针对特定两种区块链跨链的方案，针对性强，可扩展性差，难以直接移植用于其它区块链跨链交互.

WeCross[13]面向区块链互联互通问题，旨在构建一套未来区块链互联基础设施，虽然其支持多链跨链，但是由于跨链交互不需要借助其他区块链，本质上依旧是基于直连跨链架构的方案. WeCross 的核心组件是跨链路由，每条参与跨链交互的区块链都有一个由部署该区块链的机构搭建的跨链路由，区块链强信任其跨链路由. WeCross 的交互区块链通过跨链路由直接建立连接实现跨链交互. 与前两种方案相比，WeCross 面向更通用的区块链跨链交互，并将适配复杂度由链上卸载至跨链路由，但是跨链路由间依旧需要进行(O(n))次适配，适配复杂度高.

基于中继链的跨链方案，其基本思想是有跨链需求的区块链借助其他区块链作为中继，通过两跳跨链操作实现区块链互操作. 单个中继链的处理能力有限，当存在大规模跨链需求时，中继链可以进一步扩展成中继链网络，有跨链需求的区块链借助中继链网络，通过多跳跨链操作实现区块链互操作.

Cosmos[6]针对区块链存在的扩展性、可用性以及独立性问题，提出了构建区块链互联网的设想，为此设计了一种区块链网络架构，并提出了该架构下的跨链交互方案. Cosmos 的区块链网络架构分为两部分，独立区块链被称作分区（Zone），连接分区的特殊分区被称作枢纽（Hub），分区借助枢纽实现跨链交互，在大量分区存在交互需求时，这种架构可以减少区块链之间的适配复杂度.然而由于 Cosmos Hub 性能是有限的，其可连接 Zone 的数量也是有限的，无法满足大规模跨链交互场景下跨链架构可扩展性的要求.

Polkadot[7]针对区块链在可扩展性、可伸缩性、安全性等方面普遍存在的问题. Polkadot 由一条/多条中继链（Relay Chain）以及多条平行链(Parachain)构成. 其中，平行链负责具体业务的执行;中继链负责与其直接连接的所有平行链的最终性共识. 为了在保证安全性的前提下实现中继链上复杂的操作，Polkadot 设计了四种角色共同维护网络:收集者、渔夫、提名者以及验证者. 与 Cosmos 类似，由于 Polkadot 需要在中继链上实现所有平行链的全局共识，这使得 Polkadot 的可扩展性极大的被中继链性能所约束，目前理论上仅 100 条平行链的接入.

1.2 跨链交互机制

跨链交互机制是在跨链架构的基础上实现交互功能的具体方法. 跨链交互机制研究需要解决跨链信息传输、跨链信任传递与跨链事务处理三个层面的基本问题：

1）跨链信息传输是指将一个区块链中的数据传

送到另一个区块链．与传统网络信息传输不同的是，区块链是一个封闭的去中心化系统，无法向其它外部用户或者区块链发送信息．

2）跨链信任传递是指目的区块链接收跨链信息后确认该信息来自于源区块链并已通过源区块链共识．由于不同区块链采用不同的共识算法，其信任传递（即跨链信息验证）方式相应的也有所不同．

3）跨链事务是指为实现某个跨链交互逻辑而在两个（或多个）交互区块链内分别执行的一组链内交易（在不同链执行），跨链事务处理需要保证事务的原子性，即组成跨链事务的链内交易对所有链上状态的改变是原子的，要么所有涉及到的合约变量状态全部改变成功，要么全部不改变．

现有跨链研究面向跨链应用，提供一套完整的、但各层关键技术紧耦合的跨链交互机制．

哈希时间锁协议（Hashed Timelock contract，HTLC）是典型的针对跨链资产交换应用的跨链交互机制[14-18]．具体地，HTLC 基于相同的哈希锁保证各个组成跨链事务的链内交易可以一致执行,基于引入差异化的时间锁,保证跨链操作最终一致回滚的同时,保证参与者有足够的时间提交交易.在此过程中，HTLC 并没有在区块链间传输跨链信息，而是通过用户在不同区块链上释放哈希锁密钥代替特定的跨链信息．同时 HTLC 也没有直接实现跨链信任传递，而是根据预先设置的哈希锁验证哈希锁密钥的正确性，从而间接验证特定跨链信息有效. HTLC 无法抵御审查攻击[19-21]、洪泛攻击[22]等攻击，且仅适用于跨链资产交互应用，无法拓展至其它应用场景．

为了提供更通用的跨链交互功能，简化跨链应用的设计与实现，Cosmos、Polkadot、BitXHub 等研究将跨链传输验证与应用进行了解耦．

Cosmos 提出了区块链间通信协议（the interblockchain communication protocol，IBC 协议）[23]．IBC 协议是一种端到端、面向连接、有状态的协议，实现了独立分布式账本上模块之间的可靠、有序和认证通信．IBC 协议具体由客户端、连接、通道与中继器组成．其中，客户端负责验证跨链交易；连接与通道维护了跨链信息传输的状态，实现对对端链与应用的认证以及传输有序性的支持；中继器实现跨链信息在连接的转发．

Polkadot 设计了平行链间跨链信息传输协议（the cross-consensus message passing，XCMP）[24]，XCMP 通过允许双向通信的跨链消息传递通道在平行链间传递数据．由于 XCMP 仍在设计实现阶段，目前平行链通信使用资源开销更大的、借助中继链通过多跳传输实现的水平中继路由信息传输（horizontal relay-routed message passing，HRMP）．Polkadot 通过中继链上的全局共识实现跨链信息验证．

BitXHub[8] 提出的链间消息传输协议（inter-blockchain transfer protocol，IBTP 协议）是一种类似 TCP/IP 的链间传输协议，其通过引入转发节点（Pier）实现了记录了跨链信息的 IBTP 数据包的传输，包括直接交互模式与中继链转发模式. BitXHub 通过链上 SPV 实现对跨链信息的验证．

尽管上述方案实现了最基本的传输验证与应用分离的两层结构，但是传输与验证紧耦合，事务与应用紧耦合．在传输与验证方面，上述方案无法满足不同区块链差异化跨链数据验证的要求．在事务与应用方面，各跨链应用的设计、实现需要保证事务的基础特性，设计、开发难度大且可能因为跨链应用在事务性基础特性保障方面的设计、实现缺陷为所在跨链系统引入安全漏洞．

2 按需可扩展跨链架构

本文提出了一种按需可扩展的跨链架构——PieBridge，并在此基础上提出了具有独立事务层的四层跨链交互协议栈，保证跨链架构的可扩展性的同时支持差异化的跨链交互，本节将介绍按需可扩展跨链架构．

本节首先介绍跨链架构的整体设计，以便了解 PieBridge 的整体设计思路，区分其与现有跨链架构；然后从中继域全生命周期管理与跨链交互的基本流程两个方面介绍了 PieBridge 的详细设计．

2.1 整体设计

本文引入中继域，并基于“按需建域”的跨链架构设计理念，设计了如图 1 所示的按需可扩展跨链架构 PieBridge．与现有“一通百通”的基于中继链的跨链架构不同，“按需建域”考虑到区块链之间差异化的跨链需求强度，仅将有共同跨链交互需求的区块链连接在同一条中继链上，形成一个个独立的中继域．各独立中继域在组建过程中可以根据跨链交互需求配置不同的规模、性能、安全等级的中继链．通过“按需建域”，一方面将跨链交易隔离在中继域内，可以实现跨链系统负载的合理切分，保证跨链架构的可扩展性；另一方面，可以在中继域内按需配置中继链，从架构层面支持差异化的跨链交互需求．

在“按需建域”的基础上，本文针对单中继域内平行链数量增加、跨链交易比例升高等导致中继链拥塞问题，设计了中继域动态扩容机制．具体地，在中继域内复制一条新的中继链，并将跨链交易均衡分流

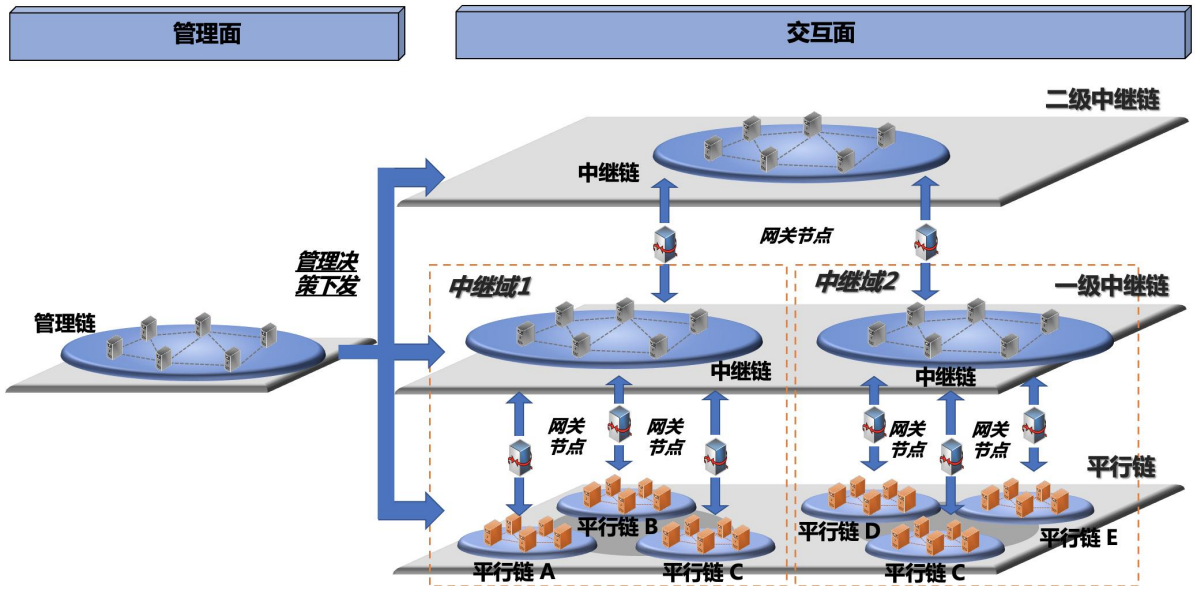


Fig. 1 The overview architecture of PieBridge

图 1 PieBridge 架构图

至不同中继链进行处理. 通过中继域动态扩容，保证实现中继域处理能力提升的同时，避免多跳中继链处理带来的跨链交互时延增加问题.

进一步地，考虑到中继域是随跨链交互需求动态生成、更新的，本文秉持去中心化的理念，引入了一条专门区块链对中继域进行去中心化的管理.

本文设计的按需可扩展的跨链架构 PieBridge 如图 1 所示，包括管理面与交互面.

PieBridge 管理面由管理链组成，实现对交互面去中心、去信任、公开透明的管理. 管理链是为 PieBridge 提供去中心管理服务的区块链，其以智能合约的形式部署了具体的管理规则，通过管理决策下发机制可以将管理决策发送给交互面执行，具体实现方法在 2.2 中继域全生命周期管理中详细介绍.此外，管理链还管理了一组用于构建中继链与网关节点的节点.

PieBridge 交互面实现具体的跨链应用，由平行链、中继链、网关节点与按需组建的一系列中继域组成. 这些组成元素的具体描述如下：

1）中继域. 中继域是 PieBridge 中进行跨链交互的基本单元，区块链需要加入中继域才能进行跨链交互. 中继域由有共同跨链交互需求的平行链、连接这些平行链的中继链以及在区块链之间转发跨链信息的网关节点组成. 域内平行链大多为相同行业领域的区块链，而由于部分平行链可以应用于不同行业、与不同行业平行链存在跨链交互需求，例如供应链、保险等行业的跨链应用均需要引入银行区块链进行结算，因此平行链可以根据应用需求加入多个中继域.

2）平行链. 平行链是存在跨链需求的独立区块链，其通过组建或者加入中继域实现跨链交互.

3）中继链. 中继链即按需组建的、在域内连接指定平行链的区块链，其通过与域内平行链适配、建立连接，为它们提供去中心的中继服务[3].

4）网关节点. 网关节点一般指域内网关节点，即在域内平行链与中继链之间转发跨链数据，支撑跨链信息传输的节点. 此外，网关节点还包括域间网关节点，即在中继链之间转发跨域数据的节点. 后文网关节点均指域内网关节点. 网关节点既可以由平行链提供，也可以由跨链系统提供，甚至可以通过设置激励机制引入，本文不讨论这部分内容.

2.2 中继域全生命周期管理

为了实现对中继域按需生成、按需更新的支持，本文设计、实现了对中继域的全生命周期管理，具体包括中继域的组建、中继域新增/删除平行链、多粒度的跨链资源访问控制与中继域动态扩容.

域的组建可以分为三个阶段，首先是请求发起阶段，各平行链代表向管理链提交建域请求，声明要组建一个域，并提出期望的共识算法、出块频率等中继链选型信息;随后进入决策阶段，管理链根据收到的建域请求生成中继链创世区块，实现了对中继链的按需配置与域内网关节点的初始化授权，并选择一组节点作为中继链初始共识节点；最后进入决策执行阶段，管理链基于网络将决策信息分发给中继链初始共识节点，这些节点验证该决策已在管理链中达成一致后，

[3]为了应对域间不频繁、非时间敏感的跨链交互需求，可以引入连接中继链的二级中继链，以实现域间交互.

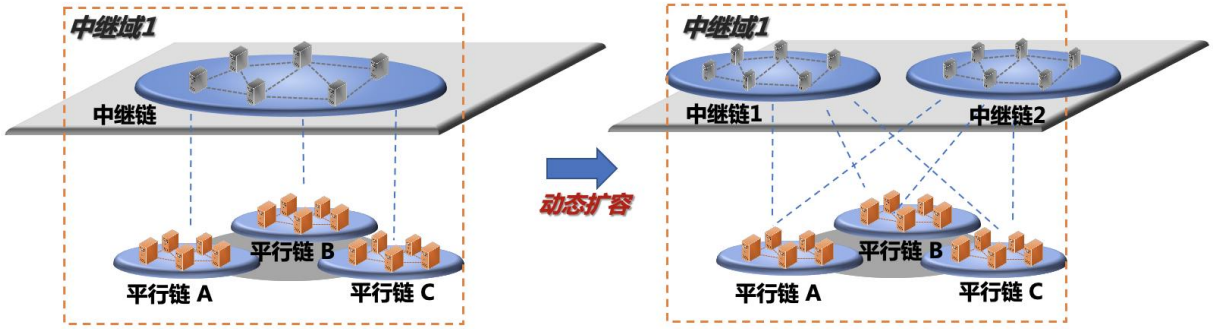


Fig. 2 The dynamic expansion of the relay domain

图 2 中继域动态扩容示意图

依据该决策启动中继链，基于已授权的网关节点完成平行链与中继链的连接，实现域的构建.

域新增、删除平行链与域组建类似，但是在决策执行阶段管理链需要将决策信息分发给运行中的中继链，无法直接基于网络传输实现. 为此，PieBridge 引入了中继链治理者，由治理者以中继链交易的形式实现决策分发. 而为了避免引入信任与单点故障问题，PieBridge 进一步设计了基于分布式私钥[25]的治理决策分发机制，由管理链共同担任中继链治理者，保证决策分发拥有与管理链相同的安全等级与去中心化程度. 具体地，管理链基于分布式私钥技术生成中继链治理节点公钥地址以及由管理链共识节点共同持有的分布式私钥，当管理链处理平行链组加域、退域请求时，管理链共识节点根据管理链决策分别使用其持有的分布式私钥签署中继链交易，当分布式私钥签名达到一定数量时该交易生效.

在多粒度的跨链资源访问控制方面，PieBridge 通过中继链对平行链网关节点发交易的权限的控制，保证只有授权网关节点所在平行链的跨链信息能被转发至中继链，从而实现区块链级别的资源访问控制；通过在平行链上部署基于区块链标识、跨链应用标识与用户全局标识的资源访问控制列表，基于跨链请求与资源访问控制列表的匹配，实现应用与用户级别的资源访问控制.

由于中继链需要处理域内所有跨链交易、是中继域的性能瓶颈，因此随着接入平行链增多、跨链交易增多，中继链将率先出现拥塞，并导致域内跨链交互高时延问题.为此，PieBridge 提出了如图 2 所示的动态扩容：管理链在域内复制一条与域内平行链连接的中继链，将域内原本由单一中继链处理的跨链交易分流至两条中继链上.该方案通过适度增加平行链上的存储、计算开销，可以有效缓解单中继链拥塞问题，并保证域内跨链交易无需多跳中继链处理，从而降低跨链系统平均跨链交易时延.

域的扩容同样可以分为三个阶段，首先是请求发起阶段，平行链代表向管理链提交扩容请求，声明中继域达到性能瓶颈；随后进入决策阶段，管理链依据该域中继链生成新中继链创世区块、选择一组节点作为新中继链初始共识节点并生成跨链数据分流策略；最后进入决策执行阶段，管理链基于网络将创世区块分发给新中继链初始共识节点，由这些节点启动新中继链，基于网络将分流策略分发给已授权的网关节点，网关节点加载分流策略，同时平行链适配新增中继链，实现域的扩容.

2.3 跨链交互基本流程

PieBridge 中继域内，平行链借助中继链实现跨链交互，因此平行链与中继链均需要部署跨链交互功能. 跨链应用的具体功能需要基于跨链交互实现，例如，在资产类应用场景下，跨链转账应用需要资产区块链 A 对资产区块链 B 上用户资产状态的跨链更新实现；在医疗保险应用场景下，跨链核保应用需要保险区块链对医院区块链患者医疗数据的跨链读取实现. 而由于跨链交互无法协调交互区块链完成不存在的应用功能，因此交互区块链均需要部署相应的应用，在上述跨链转账应用中，资产区块链 B 需要本身支持对用户资产状态的管理.

本文抽象了 PieBridge 中继域中跨链交互的流程，具体如图 3 所示，包括跨链请求发起阶段、跨链请求中继阶段、跨链请求处理阶段以及跨链回执反馈阶段. 为了便于描述，本文将发起跨链交互的平行链称为源平行链，接收并处理跨链请求的平行链称为目的平行链，在上述例子跨链核保应用中，保险区块链是源平行链，医院区块链是目的平行链.

1）跨链请求发起阶段. 用户在源平行链上以交易的形式调用跨链应用合约并触发跨链请求. 源平行链跨链交互合约将该跨链请求打包成标准的跨链数据包.

2）跨链请求中继阶段. 源网关节点通过监听源平行链获取跨链数据包，整理出证明数据包在源平行

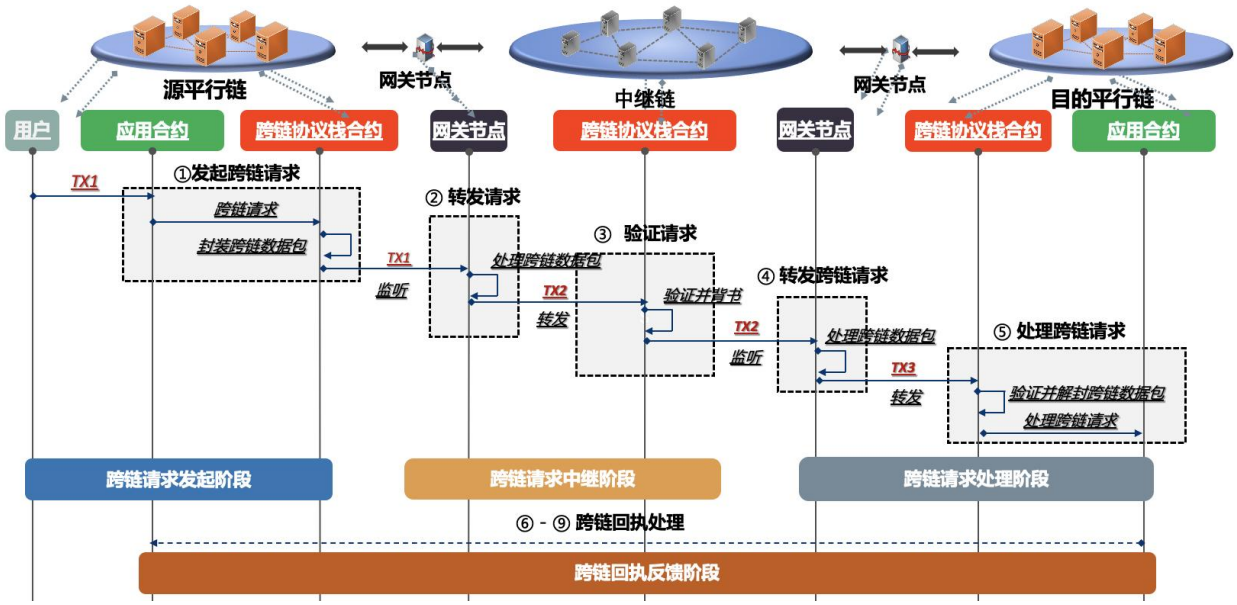


Fig. 3 The workflow of cross-chain interoperation

图 3 跨链交互基本流程图

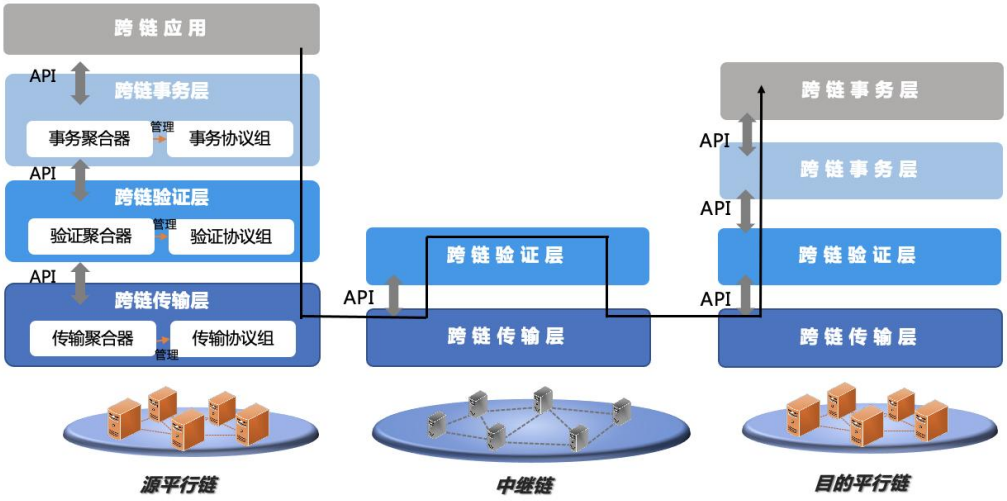


Fig. 4 The four-layer cross-chain protocol stack

图 4 四层跨链交互协议栈

链中达成一致的证据，随后将该跨链数据包与证据以交易的形式发送到中继链中. 中继链跨链交互合约基于证据验证跨链数据包在源平行链达成一致，并将验证结果写入链中实现背书.

3）跨链请求处理阶段. 目的网关节点通过监听中继链获取跨链数据包与中继链背书的证据，并将该数据包以交易的形发送到目的区块链中. 目的平行链跨链交互合约解析跨链数据包，并通过验证中继链对跨链请求的背书间接验证该跨链请求在源区块链上达成一致，随后调用对应的应用合约，完成跨链请求响应.

4）跨链回执反馈阶段. 跨链请求处理回执以同样逐跳传输、验证的方式传回源平行链.

复杂的跨链应用需要多次跨链交互实现，PieBridge 跨链交互协议的具体设计与实现将在第 3 节四层跨链交互协议栈中详细展开.

3 四层跨链交互协议栈

本节介绍具有独立事务层的四层跨链交互协议栈，首先概述了分层跨链交互协议栈的整体设计思路，随后详细介绍协议栈各层的内容.

3.1 协议栈整体设计

针对跨链交互需要解决的跨链信息传输、跨链信任传递与跨链事务处理三个层面的基本问题，本文通过抽象跨链元操作与跨链事务，首次将跨链事务与跨链应用解耦，提出了具有独立事务层的四层跨链交互协议栈，如图 4 所示. 独立事务层的抽象，一方面简

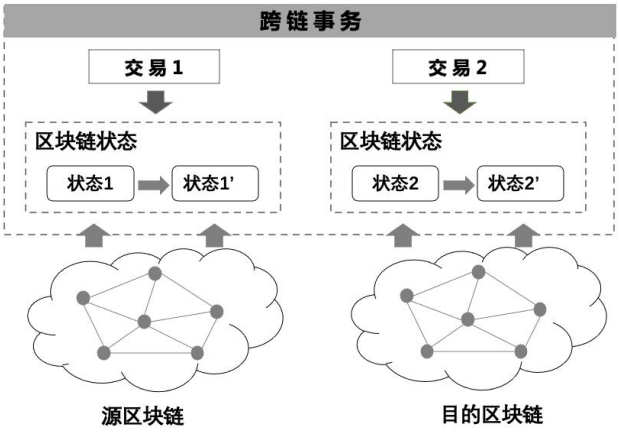


Fig. 5 The illustration of the cross-chain transaction

图 5 跨链事务示意图

化了跨链应用的设计与开发，支持跨链应用的灵活、快速构建；另一方面为跨链应用的事务性安全提供了系统级保障，避免跨链应用在事务处理方面的设计、实现缺陷为所在跨链系统引入安全漏洞，有利于跨链应用生态系统的发展．在协议分层解耦的基础上，本文在协议栈各层均实现了多种基础协议，并采用适配器设计模式[26]，为协议栈各层引入聚合器，设计通用层间接口，将层间多协议的复杂交互转换为聚合器间的简单交互，降低了多种协议的管理复杂度，保证了层内协议的灵活性，满足不同跨链应用在性能、安全、隐私等方面的差异化需求．

四层跨链交互协议栈具体包括跨链应用层、跨链事务层、跨链验证层与跨链传输层．

1）跨链应用层．跨链应用层部署在平行链上，包括多种跨链应用，各跨链应用设计、实现需要跨链交互的应用功能．跨链应用需求将会被封装成跨链事务，交由跨链事务层处理．

2）跨链事务层．跨链事务如图 5 所示，由在两个（或多个）交互区块链内分别执行的一组链内交易（在不同链执行）组成．跨链事务层部署在平行链，提供了独立的事务协议，在保证跨链事务原子性的同时，保证跨链事务与跨链事务间、跨链事务与链内交易间的隔离性，即组成跨链事务的交易与组成其它跨链事务的交易及链内交易互不影响．

3）跨链验证层．跨链验证层解决跨链信任传递的问题，与跨链传输层一起支撑跨链事务处理．其部署在平行链、中继链以及网关节点上，提供了多种区块链验证方法与对应的证据生成机制，实现了平行链与中继链对基于跨链传输层所获取的跨链数据的可信性验证．

4）跨链传输层．跨链传输层解决跨链信息传输的问题，与跨链验证层一起支撑跨链事务处理．其部署在平行链、中继链以及网关节点上，提供了多种跨链传输协议，实现了跨链数据在交互区块链间的传输．其中，由于区块链无法主动向外部发送信息[18]，各跨链传输协议均通过网关节点监听平行链与中继链获取跨链信息、转发跨链信息实现．

本文采用适配器设计模式，如图 4 所示，为协议栈各层引入聚合器，一方面实现层内协议管理，另一方面实现层间交互.在层内协议管理方面，本文通过协议注册与协议地址映射表实现新协议的加载与协议的调用.在层间交互方面，本文设计了如表 1 所示的各层控制信息．在跨链请求发送阶段，上层聚合器将处理好的跨链数据以及控制信息发给下层聚合器，下层聚合器根据控制信息进行协议调度，数据经过层内协议处理过后，聚合器再将本层数据包打包，连同控制信息一并发给下层聚合器;在跨链请求处理阶段，数据处理与发送向相反．各层的数据包拆封与解封方式与传统网络协议栈类似，每层的数据都是由本层的数据包头以及来自上层的数据包体组成，发送时层层封装，接收时层层解封．

3.2 分层协议栈详细设计

（1）跨链事务层

跨链事务层解耦应用与事务，为跨链事务提供多种事务协议保障其原子性与隔离性.

本文针对事务协议的共性需求，抽象出资源锁服务、回滚日志服务、事务时钟服务三种基础服务.

资源锁服务是保证跨链事务隔离性的重要模块.跨链事务隔离性，本质上需要解决并行的多个跨链事务对同一公共资源的竞争问题.资源锁服务通过在链上维护资源锁映射表，记录当前世界状态下指定对象的操作权限.跨链事务在访问区块链资源前需要成功申请资源锁，在跨链事务完成后需要释放锁.资源锁具体包括读锁、共享写锁和独占写锁.

Table 1 Control information of the protocol stack's layers

表 1 跨链交互协议栈控制信息

协议层	名称	类型	说明
传输层	xid	String	事务 ID
	sourceApp	String	源应用
	destApp	String	目的应用
	tsctType	String	事务协议类型
	tsctstatus	Int	事务状态
	tsctSpec	Bytes	事务协议数据单元
验证层	verifyType	String	验证类型
传输层	tsptType	String	传输类型
	tsptStatus	String	传输状态

回滚日志服务是保证跨链事务原子性的重要模块.回滚日志服务通过预写日志和回滚日志的方法实现事务处理失败时所涉及区块链状态的回滚.具体而言，跨链事务在执行前需要将回滚操作加载至回滚日志服务中，若事务处理失败，则回滚日志服务会按照事务预先注册的回滚策略进行事务回滚.

事务时钟服务主要解决区块链缺少统一时钟带来的事务协议可终止性问题.事务时钟服务通过链上记录事务有效时间与链下计时器监听、触发事务超时事件完成对跨链事务超时情况的监测.

基于上述基础服务，本文设计了三种基本的事务协议，包括基础事务协议（BTP）、自动事务协议(ATP)、多链事务协议(MTP).BTP 协议仅适用于有补偿操作的跨链应用，且需应用显示的给出事务处理失败时的补偿操作，其基于补偿保证原子性，基于资源锁服务保证隔离性.ATP 协议基于回滚日志服务保证原子性，基于资源锁服务保证隔离性，适用范围更广.MTP 协议适用于多链事务，其原子性和隔离性保障方式和 BTP 相同.

（2）跨链验证层

跨链验证层提供多种跨链验证方法保证跨链交互过程中跨链信息的可信性，即跨链信息是经过源链共识的、在源链主链中的数据.

本文实现了基于 SPV 的跨链数据验证方法与基于公证人的跨链数据验证方法.基于 SPV 的验证方法通过区块头链的同步保证区块头的可信性；通过状态树验证，保证指定状态数据在有效区块头中.基于公证人的验证方法通过可信公证人背书实现跨链数据的验证.

跨链验证层部署在平行链、中继链以及网关节点上，其中平行链与中继链实现具体的验证方法；网关节点根据不同验证协议生成不同的证据.

（3）跨链传输层

跨链传输层实现链到链的信息传输.由于区块链需要借助链下网关节点监听转发实现跨链信息传输，在此过程中需要解决如何在源链上发现并获取待转发的跨链信息这一基本问题.

为此，本文设计了跨链事件抛出与跨链数据队列存储两种基础服务.其中，跨链事件抛出服务是指利用大多区块链自有的事件机制，将封装好的跨链数据包以事件形式抛出，以便网关节点监听. 跨链数据队列存储服务是指利用智能合约维护一个存储跨链数据的队列，新生产的跨链信息将存储在队列中等待传输，链下网关需要不断查询这个队列来判断是否有新的数据包到来.

在此基础上，本文进一步设计了基础跨链传输协议与可靠跨链传输协议.其中，基础跨链传输协议仅实现跨链信息监听、转发.而可靠传输协议为跨链数据包引入唯一的序号，实现重复跨链信息过滤；基于源、目的平行链上维护的下一笔跨链数据包发送序号、跨链数据包接受序号，实现跨链信息有序交付以及数据包丢失重传.

4 验证与分析

本节从理论分析与实验分析两个角度对 Pie Bridge 的可扩展性与差异化跨链交互进行了验证. 首先，通过理论分析，对比中继链网络方案与 PieBridge 在扩展性方面的优劣. 接着，通过两组实验，对比了单中继链方案与 PieBridge 在可扩展性方面的优劣，并验证了 PieBridge 对差异化跨链交互的支持.

4.1 理论分析

本文选取跨链交易的时延作为指标，由于所有跨链交易都需要经过源区块链、目的区块链处理，因此此处时延仅指跨链交易在中继链中的时延. 若跨链系统在时延可接受、波动较小的情况下，可以接入更多区块链，则表明该跨链系统扩展性良好；反之，若跨链系统为接入更多区块链，造成了严重的阻塞，甚至是无限制排队，导致时延很高，则跨链系统可扩展性不佳.

4.1.1 建模

首先，本文对于一条中继链上跨链交易的到达、处理作出一些基本假设. 假设交易的到达、处理是服从参数为λ，μ的泊松过程，这种假设将每条中继链的交易处理过程简化为一个$M/M/1$ 模型[27]. 此处的λ，μ的单位并非区块链中常考量的 tps，而是以中继链的区块为单位计数的频率，例如若中继链从一条区块链收到跨链交易的频率是 5tps，中继链处理交易的速度是 100tps，且区块能打包 500 笔交易，那么在本模型中对于中继链而言，此时每秒钟会从一条平行链收到$\lambda=\frac{5}{500}$个中继链区块，处理$\mu=\frac{100}{500}$个中继链区块，即:

$$\lambda=\frac{TPS_1\cdot\gamma}{\mathrm{B}},\ \mu=\frac{TPS_2}{\mathrm{B}}, \tag{1}$$

TPS_1是一条源平行链的交易处理速率，γ是其中的跨链交易比例，B 是一个中继链区块所能容纳的交易数目；TPS_2是中继链处理交易速率.

接下来，讨论每种方案的理论时延.

（1）中继链网络

记W_{relay}为中继链网络下，一笔跨链交易从被发

送至中继链到上链处理完毕的时延，有如下表示：

$$W_{relay} = p_1 \cdot W + p_2 \cdot 2W.$$

因为中继链随机连接一组平行链，对于一笔跨链交易，其源平行链与目的平行链连接同一条中继链，也可能连接不同的中继链，两种情况对应的概率分别记做p_1，p_2. W为经历一次中继链的时延，则当源链、目的链在同一中继链群组时，跨链交易会经历一次的中继链时延，源链、目的链分处于不同中继链群组时，跨链交易经历两次中继链时延. p_1，p_2有：

$$p_1 = \frac{1}{k},\ p_2 = \frac{k-1}{k}.$$

k为中继链网络下，中继链群组的数目，他是下面方程的解.

$$k - 1 + \frac{n}{k} = \theta \cdot m_{max}.$$

其中$k-1$是指当前中继链需要与所有其他中继链互联，n为整个网络中需要提供服务的平行链数量，则$\frac{n}{k}$为每个群组中随机分配到的平行链数量，这二者之和不得超过中继链的极限接入能力，m_{max}. 根据我们前文抽象的排队论$M/M/1$模型，每条中继链的负载强度ρ不得大于 1，否则中继链会发生阻塞，跨链交易无限制排队下去，由此可得m_{max}，

$$\rho = \frac{m_{max}\lambda}{\mu'} \le 1,\ m_{max} = \frac{\mu'}{\lambda}.$$

需要注意的是，(1)中的μ是中继链处理交易的能力，但是在中继链网络下，除跨链交易外，每个源平行链区块头也需要被中继链同步，这会占据中继链的处理能力，我们假设平均q笔跨链交易，会有一个区块头需要同步，因此

$$\mu' = \frac{q}{q+1}\mu.$$

θ是本文引入的参数，代表中继链接入链数目占其极限数目的百分比，$\theta < 1$，记$m = \theta \cdot m_{max}$.

由$M/M/1$模型可知，单一中继链的时延是，

$$W = \frac{1}{\mu' - m\lambda}.$$

因此有

$$W_{relay} = \frac{2k-1}{k(\mu' - m\lambda)}. \quad (2)$$

（2）PieBridge

本文分别为 PieBridge 不进行中继域扩容的基础方案与进行中继域扩容的优化方案建模.

1）基础方案

在“按需建域”的 PieBridge 中，一笔跨链交易的源平行链、目的平行链总是存在于同一中继域，因此有，

$$W_{PieBidge\ basic} = W = \frac{1}{\mu' - m\lambda}. \quad (3)$$

m的表达式同上文，不过需要注意的是，PieBridge 中继域根据跨链需求生成，各中继域内平行链数量不一，本文建模对此进行了简化. 同时，由于各中继域独立，中继链不需要互联，因此每条中继链所连接的所有区块链都是需要服务的平行链，这将使得在全网区块链数目n一定的情况下，本方案需要更少的中继链群组、建立更少的中继链，可以达到更低的时延.

2）优化方案

在支持中继域扩容的 PieBridge 中，网关节点根据跨链交易分流策略，实现中继链之间的负载均衡. 假设每个中继域有s条中继链，此时整个中继链的模型不再是单链模型$M/M/1$的组合，而是相当于一个多服务台的$M/M/s$系统[27].

根据图 6，中继链时延，$W_{PieBridge\ advanced}$可以表示为，

$$W_{PierBidge\ advanced} = \frac{p_0\rho^s\rho_s + \rho s!(1-\rho_s)^2}{s!(1-\rho_s)^2 m\lambda}. \quad (4)$$

其中p_0为中继链空载概率，

$$p_0 = \left(\sum_{n=0}^{s-1} \frac{\rho^n}{n!} + \frac{\rho^s}{s!\,(1-\rho_s)} \right)^{-1}.$$

其中ρ代表整体中继链负载强度，

$$\rho = m\lambda/\mu'',$$

ρ_s代表此时每条中继链负载强度，

$$\rho_s = \frac{m\lambda}{s\mu''}.$$

此处μ''为该方案下每条中继链对跨链交易的实际处理能力，需要注意的是，此时即使一笔跨链交易没有经由某条中继链处理，但是其区块头仍然需要s

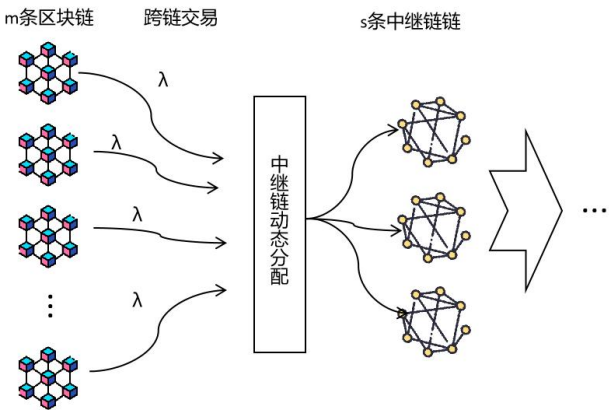


Fig. 6 The multi-server queuing model

图 6 多服务台排队模型

条中继链全部同步，即：

$$\mu'' = \frac{q}{q+s}\mu.$$

m_{max}由$\rho_s = 1$取得，m仍然满足$m = \theta \cdot m_{max}$.值得注意的是，此时m_{max}扩展为了$\frac{s(q+1)}{q+s}$倍.尽管支持中继域扩容的 PieBridge 方案中，由于域内中继链均需要适配所有平行链，每条中继链上的平行链数据同步开销将随着平行链接入而不断增加，因此m_{max}无法扩展为s倍.但是，仍然可以极大的提升了跨链架构的可扩展性，且提升效果在跨链交易密集的情况下更好.

4.1.2 分析

本文根据对 Cosmos 网络中跨链交易的比例的测量以及现有相关测量工作[27]，进行了如表 2 所示的配置，并讨论了表 2 配置下 PieBridge 与 Cosmos 的端到端跨链时延随接入平行链数量变化时的模拟实验结果.

Table 2 Parameters when number of chains varies

表 2 接入平行链数量变化时的参数

n	TPS_1	γ	TPS_2	θ	q	s	B
变量	438	2%	438	95%	5	2	1000

对比如图 7 所示的三种方案在不同接入区块链数目下的延时变化情况：

1）中继链网络的时延始终高于 PieBridge 的两种方法，这是因为中继链网络下，跨链交易有很大的概率需要跨越中继链群组，随着区块链数目越多、中继链群组数目越多，跨链交易需要涉及其他群组的概率也就越大，因此其时延不断增长.进一步的，因为中继链既需要与一定数目接入区块链相连，还需要负责连接数目越来越多的其他中继链，当接入区块链数目超过某一临界值后，所有中继链彻底拥塞，跨链交易开始无限制排队，此时如图中红线所示，理论时延不是红线对应的数值 120s，而是随着无限排队趋于无穷，因此中继链网络的可扩展性瓶颈可见一斑.

2）PieBridge 的两种方法因为不涉及到跨链交易跨中继链群组传播，因此时延显著低于中继链网络方案，且因为中继链不需要连接其他中继链，系统不会因为中继链群组、接入区块链数目的增多而发生拥塞，展现了良好的可扩展性.

3）PieBridge 的优化方案展现出了极佳的性能，在每个中继链群组只额外引入一条区块链的情况下，系统的时延由 55s 大幅度下降到了 4.21s，该时延已经接近中继链本身的共识时延，基本可用达到“跨链交易产生，即被中继链处理”的效果.此外，如模型中讨论，PieBridge 优化方案的区块链可接入数量得到

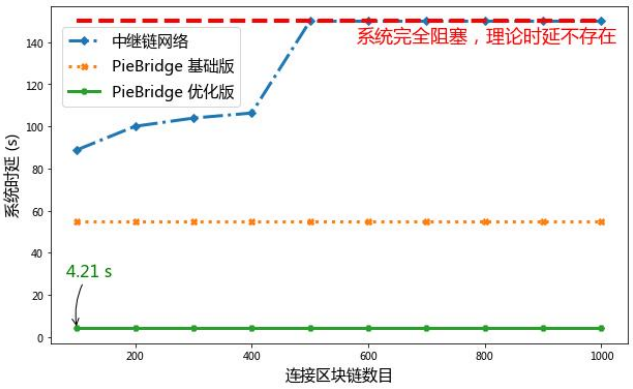


Fig. 7 Delay of transactions in the relay chain system when the number of chains varies

图 7 随连接区块链数据变化的中继链时延

极大的提升，系统的可扩展性优异.

4.2 实验分析

本文选取中继链吞吐量（中继链单位时间内处理交易的数目）作为指标，进行了性能实验.通过增加接入中继链的平行链数量，比较中继链吞吐量，衡量跨链系统性能.若中继链吞吐量趋于平稳，则表明中继链已到达性能瓶颈，无法接入更多的平行链.

然后本文验证了 PieBridge 对差异化跨链交互的支持.根据供应链数据存证场景与跨城市公共服务场景对跨链交互在可扩展性与时效性上的不同需求，针对性地构建了不同类型的中继域，并通过实验比较两个中继域在性能方面的差异性，验证了 PieBridge 对差异化跨链交互的支持.

考虑到实际跨链应用中的网络问题，本文选择部署在不同地区的云服务器进行实验.服务器使用了 2 个 CPU,每个 CPU 配置为 2.10GHz、20 核，服务器内存为 256G.

4.2.1 性能试验

本文将 Cosmos 跨链系统与按需建域的 PieBridge 进行比较，通过比较相同实验环境下 Cosmos 与 PieBridge 吞吐量随接入平行链数量的变化，验证 PieBridge 在性能方面的可扩展性.

在该实验中，本文设置 Cosmos 跨链系统和 PieBridge 跨链系统的中继链配置相同，均采用 4 个共识节点，分别部署在 4 台服务器上，中继链每个区块可容纳最大交易数量为 5000；同时使用了进程模拟平行链出块过程，假设平行链出块间隔时间为 6s，每个区块中的跨链交易比例为 10%，每当平行链有新的区块产生，区块内的跨链交易都会被逐个发送到中

继链.

实验结果如图 8 所示. Cosmos 吞吐量首先随接入平行链数量的增多持续增大，并在接入 *500* 条平行链后趋于稳定，这意味着跨链系统到达性能瓶颈，无

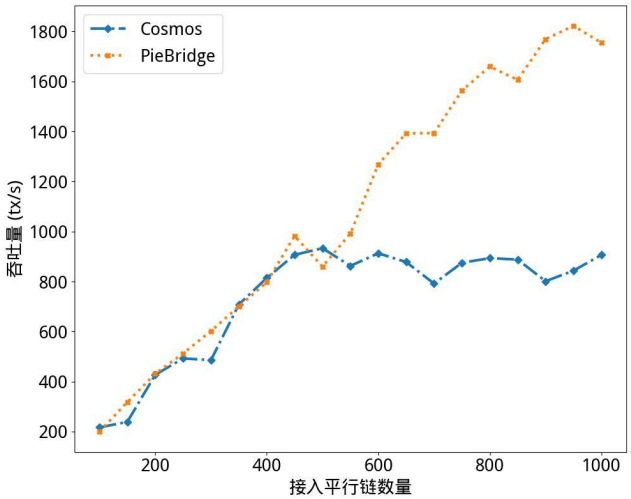


Fig. 8 Throughput of the system when the number of chains varies

图 8 随连接区块链数量变化的跨链系统吞吐量

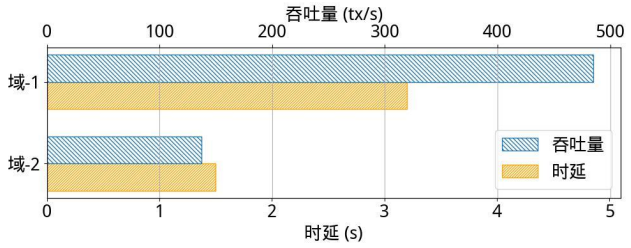


Fig. 9 Diversity of cross-chain interactions

图 9 差异性跨链交互

法接入更多的平行链、处理更多的跨链交易；而 PieBridge 由于将平行链接入了不同的中继域，并且当中继域到达性能瓶颈时会动态扩容，其吞吐量随接入平行链数量的增多持续增大. 该结果表明，PieBridge 可以通过域的扩容，按需提升跨链系统的吞吐量，保证跨链性能不随跨链交互的增加而降低.

4.2.2 差异化交互实验

本文依据供应链数据存证场景与跨城市公共服务场景对跨链交互在可扩展性与时效性上的不同需求，构建两个中继域，并通过比较两个中继域的平均吞吐量与平均跨链交互时延，验证 PieBridge 对差异化跨链交互的支持.

在该实验中，供应链数据存证场景更加注重中继域（Zone-1）的吞吐量，且需要保证存证数据完整可靠，中继链配置了较大的区块大小与 PoA 共识机制，跨链交互协议栈在验证层配置了基于 SPV 的跨链数据验证方法，传输层配置了可靠传输协议；面向跨城市公共服务场景的中继域（Zone-2）注重跨链交互的时效性，中继链配置了较小的出块间隔与 tendermint 共识机制，跨链交互协议栈在验证层配置了基于公证人的跨链数据验证方法，传输层配置了基础跨链传输协议.

实验结果如图 9 所示，Zone-1 拥有更高的平均吞吐量，Zone-2 拥有更低的跨链交互时延. 这是因为 Zone-1 配置了较大的区块大小，单区块可包含的交易数量增多；Zone-2 配置了较小的出块间隔，且采用基于公证人的跨链数据验证方法，缩短了交易等待时间与跨链数据验证时间. 该结果符合不同应用场景对跨链交互提出的需求，这意味着 PieBridge 可以通过中继链与跨链交互协议的配置实现差异化的跨链交互.

5 总结

本文基于“按需建域”的理念提出了一种按需可扩展的跨链架构 PieBridge，并在此基础上提出了具有独立事务层的四层跨链交互协议栈，保证跨链架构可扩展的同时支持差异化的跨链交互. 本文实现了 PieBridge 原型系统，并通过建模分析与实验验证证明了 PieBridge 的可扩展性，以及其对差异化跨链交互需求的灵活支持.

未来，我们将在本文工作的基础上，针对当前跨链交互性能低的问题，从以下两个层面开展工作：

1）架构层面，针对中继链上交易类型单一、交易分区明显的特点，研究中继链并行化方法，通过提升单中继链性能，降低跨链交互时延；

2）交互协议层面，针对跨链事务处理机制在多主体、长流程的场景下低效率的问题，研究单事务并行执行技术与多事务并发执行技术，通过提升跨链事务处理效率，提升跨链交互性能.

作者贡献声明：作者一提出系统、实现系统并撰写论文，作者二实现系统、进行实验，作者三对系统进行理论分析，作者四、六、七、八负责方案设计指导与论文修改，作者五负责系统实现指导与论文修改.

参 考 文 献

[1] Tiancheng Xie, Jiaheng Zhang, Zerui Cheng, et al. ZkBridge: Trustless Cross-chain Bridges Made Practical. In Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22). Association for Computing Machinery, New York, NY, USA, 2022, 3003–3017

[2] Rafael B, André V, Sérgio G, et al. A Survey on Blockchain Interoperability: Past, Present, and Future Trends. ACM Comput. Surv. 54, 8, Article 168 (November 2022), 41 pages

[3] BTC-relay: Ethereum contract for Bitcoin SPV [OL]. [2022-12-20] https://github.com/ethereum/btcrelay

[4] Lerner SD. RSK white paper overview, [OL]. [2022-12-05] https://bravenewcoin.com/assets/Whitepapers/RootstockWhitePaperv9-Overview.pdf

[5] Frauenthaler P, Sigwart M, Spanring C, et al. ETH relay: Acost-efficient relay for ethereum-based blockchains. the IEEE InternationalConference on Blockchain (Blockchain). IEEE, 2020: 204-213

[6] Kwon J, Buchman E. Cosmos: A network of distributed ledgers, [O-L]. [2022-11-18] https://github.com/cosmos/cosmos/blob/master/WHITE-PAPER.md

[7] Wood G. Polkadot: Vision for a heterogeneous multi-chain framework, [OL]. [2022-11-18] https://github.com/polkadot-io/polkadotpaper/-raw/master/PolkaDotPaper.pdf

[8] BitXHub Whitepaper, [OL]. [2022-11-18] https://upload.hyperchain.cn/BitXHub%20Whitepaper.pdf

[9] Application-Specific Blockchains: The Past, Present, and Future, [OL]. [2023-03-03] https://medium.com/1kxnetwork/application-specific-blockchains-9a36511c832

[10] Zero-Knowledge Blockchain Scalability,[OL]. [2023-01-08] https://ethworks.io/assets/download/zero-knowledge-blockchain-scaling-ethworks.pdf

[11] Map of zones - Cosmos network explorer, [OL]. [2022-08-06] https://mapofzones.com/home?columnKey=ibcVolume&period=24h

[12] China Academy of Information and Communications Technology. Blockchain Whitepapee (2022). 2022. https://mp.weixin.qq.com/s/WvVni45u-knqfTBWwoRe_Q (in Chinese)
(中国信息通信研究院. 区块链白皮书 (2022))

[13] WeCross, [OL].[2022-11-27] https://wecross.readthedocs.io/zh_CN/latest/

[14] Nolan T. Alt chains and atomic transfers, [OL]. [2022-11-24] https://bitcointalk.org/index.php?topic=193281.0

[15] HTLC implementation in the wallet, [OL]. [2023-01-14] https://github.com/bitcoin/bips/blob/master/bip-0199.mediawiki

[16] Decred-compatible cross-chain atomic swapping, [OL]. [2023-01-28] https://github.com/decred/atomicswap

[17] The Komodo Organization. BarterDEX: Atomic Swap DecentralizedExchange of Native Coins, [OL]. [2023-02-05] https://github.com/SuperNETorg/komodo/wiki/barterDEX-Whitepaper-v2

[18] Zyskind, Guy et al. “Enigma Catalyst: A machine-based investing pl-atform and infrastructure for crypto-assets.”, [OL]. [2023-03-20] https://www.enigma.co/enigma_catalyst.pdf

[19] WinzerF, Herd B, Faust S. Temporary censorship attacks in the presence of rationalminers. 2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW). IEEE, 2019: 357-366

[20] TsabaryI, Yechieli M, Manuskin A, et al. MAD-HTLC: because HTLC is crazy-cheap toattack. 2021 IEEE Symposium on Security and Privacy (SP). IEEE, 2021: 1230-1248

[21] NadahalliT, Khabbazian M, Wattenhofer R. Timelocked bribing. International Conference onFinancial Cryptography and Data Security. Springer, Berlin, Heidelberg, 2021:53-72

[22] HarrisJ, Zohar A. Flood & loot: A systemic attack on the lightning network. Proceedingsof the 2nd ACM Conference on Advances in Financial Technologies. 2020: 202-213

[23] Goes C. The interblockchain communication protocol: An overview, [OL]. [2023-03-20] https://arxiv.org/pdf/2006.15918.pdf

[24] Web3 Foundation Research.Cross-chain MessagePassing, [OL]. [2023-02-05] https://research.web3.foundation/en/latest/polkadot/XCMP/index.html

[25] Boneh, D., Drijvers, M., Neven, G. (2018). Compact Multi-signatures for Smaller Blockchains. In: Peyrin, T., Galbraith, S. (eds) Advances in Cryptology – ASIACRYPT 2018. Lecture Notes in Computer Science(), vol 11273. Springer, Cham

[26] Design Principles and Patterns, [OL]. [2023-03-15] http://staff.cs.utu.fi/staff/jouni.smed/doos_06/material/DesignPrinciplesAndPatterns.pdf

[27] introduction-to-prob-models-11th-edition, [OL]. [2023-02-08] http://mitran-lab.amath.unc.edu/courses/MATH768/biblio/introduction-to-prob-models-11th-edition.PDF

[28] Daniel C; Enrique F; Nenad M, et al. "The design, architecture and performance of the Tendermint Blockchain Network," 2021 40th International Symposium on Reliable Distributed Systems (SRDS), Chicago, IL, USA, 2021, pp. 23-33

Duan Tiantian, born in 1996. PhD candidate. Student member of CCF. Her main research interests include blockchain technologies and distributed systems.

段田田，1996 年生，博士研究生，CCF 学生会员.主要研究方向为区块链和分布式系统.

Guo Yi, born in 1997. PhD candidate. Student member of CCF. His main research interests include cross-chain technologies and consensus algorithms.

郭仪，1997 年生，博士研究生，CCF 学生会员.主要研究方向为区块链互操作与区块链共识.

Li Bo, born in 1996. PhD candidate. Student member of CCF. His main research interests include blockchain, data pricing and performance analysis.

李博，1996 年生，博士研究生，CCF 学生会员.主要研究方向为区块链、数据定价与性能分析.

Zhang Hanwen, born in 1981. PhD. associate professor. Senior member of CCF. Her research interests include computer networks and blockchain technologies.

张瀚文，1981 年生，博士，副研究员，CCF 高级会员.主要研究方向为计算机网络和区块链.

Song Zhaoxiong, born in 1993. Master, Engineer. Member of CCF. His main research interests include blockchain technology and distributed system.

宋兆雄，1993 年生，硕士，工程师，CCF 会员.主要研究方向为区块链和分布式系统.

Li Zhongcheng, born in 1962. PhD. professor, PhD supervisor. Senior member of CCF. His research interests include the computer networks and blockchain technology.

李忠诚，1962 年生，博士，研究员，博士生导师，CCF 高级会员.主要研究方向为计算机网络和区块链.

Zhang Jun, born in 1975. PhD, associate professor. Senior member of CCF. Her main research interests include blockchain, future Internet and network security.

张珺，1975 年生，博士，副教授，CCF 高级会员.主要研究方向为区块链、下一代互联网和信息安全.

Sun Yi, born in 1979. PhD, professor, PhD supervisor. Distinguished Member of CCF. His main research interests include blockchain, distributed system and network architecture.

孙毅，1979 年生，博士，研究员，博士生导师，CCF 杰出会员.主要研究方向为区块链、分布式系统和网络体系结构.

软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software [doi: 10.13328/j.cnki.jos.006950]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

区块链互操作技术综述*

段田田[1,2], 张瀚文[1,2], 李 博[1,2], 宋兆雄[1], 李忠诚[1,2], 张 珺[4,5], 孙 毅[1,2,3]

[1](中国科学院 计算技术研究所, 北京 100190)

[2](中国科学院大学, 北京 100049)

[3](山东省区块链金融重点实验室, 山东 济南 250014)

[4](内蒙古大学, 内蒙古 呼和浩特 010021)

[5](雄安新区智能城市创新联合会 区块链实验室, 河北 保定 071700)

通信作者: 张瀚文, E-mail: hwzhang@ict.ac.cn

摘 要: 区块链技术被认为是构建价值互联网的基石, 然而彼此独立的区块链系统形成了数据、价值孤岛. 区块链互操作 (也被称为跨链操作) 是打破链间壁垒、构建区块链网络的关键技术. 在区分狭义与广义区块链互操作的基础上, 重新定义狭义区块链互操作, 并抽象出跨链读与跨链写两类基本操作; 分析总结实现狭义区块链互操作需要解决的 3 个关键技术问题: 跨链信息传输、跨链信任传递、跨链操作原子性保障; 系统梳理这 3 个问题的研究现状, 并分别从多角度进行比较; 在此基础上, 从关键技术问题的角度分析具有代表性的整体解决方案; 最后指出几个值得进一步探索的研究方向.

关键词: 区块链; 区块链互操作; 跨链; 原子性

中图法分类号: TP393

中文引用格式: 段田田, 张瀚文, 李博, 宋兆雄, 李忠诚, 张珺, 孙毅. 区块链互操作技术综述. 软件学报. http://www.jos.org.cn/1000-9825/6950.htm

英文引用格式: Duan TT, Zhang HW, Li B, Song ZX, Li ZC, Zhang J, Sun Y. Survey on Blockchain Interoperability. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/6950.htm

Survey on Blockchain Interoperability

DUAN Tian-Tian[1,2], ZHANG Han-Wen[1,2], LI Bo[1,2], SONG Zhao-Xiong[1], LI Zhong-Cheng[1,2], ZHANG Jun[4,5], SUN Yi[1,2,3]

[1](Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China)

[2](University of Chinese Academy of Sciences, Beijing 100049, China)

[3](Shandong Key Laboratory of Blockchain Finance, Jinan 250014, China)

[4](Inner Mongolia University, Hohhot 010021, China)

[5](Blockchain Lab, Xiong'an Intelligent City Innovation Federation, Baoding 071700, China)

Abstract: Blockchain is the basis of the Internet of value. However, data and value silos arise from independent blockchain systems. Blockchain interoperability (also known as cross-chain operability) is essential for breaking inter-chain barriers and building a blockchain network. After differentiating between the blockchain interoperability in the narrow sense and that in the broad sense, this study redefines the former concept and abstracts out two primary operations: cross-chain reading and cross-chain writing. Subsequently, it summarizes three key technical problems that need to be resolved for achieving the blockchain interoperability in the narrow sense: cross-chain information transmission, cross-chain trust transfer, and cross-chain operation atomicity guarantee. Then, the study reviews the current research status of the three problems systematically and makes comparisons from multiple perspectives. Furthermore, it analyzes some representative holistic solutions from the perspective of the key technical problems. Finally, several research directions deserving of further

* 基金项目: 国家重点研发计划 (2021YFB2700404); 国家自然科学基金 (U22B2032, 61972382); 内蒙古自然科学基金 (2020MS06017); 河北省重点研发计划 (20310105D); 未来区块链与隐私计算高精尖创新中心项目 (GJJ-22-025)
收稿时间: 2022-10-17; 修改时间: 2023-01-13; 采用时间: 2023-04-06; jos 在线出版时间: 2023-09-06

exploration are also presented.

Key words: blockchain; blockchain interoperation; crosschain; atomicity

区块链是一种以安全、可验证、透明的方式在对等网络上存储交易的分布式账本[1], 自 2008 年被提出以来, 经过两个阶段的发展, 从概念走向应用, 逐步与实体经济融合, 开始助力经济社会发展, 被看作是构建未来价值互联网的重要支撑技术. 依据区块链技术的应用范围可以将其大致划分为两个阶段.

(1) 区块链 1.0 阶段, 以分布式账本为标志性技术. 区块链技术起源于比特币[2], 首次从技术上突破了中心化信任模式的桎梏, 解决了在一组互不信任的多方之间实现一致的状态转移问题, 从而在不可信多方间实现了可信的信息/价值流通的应用范式. 本阶段, 区块链技术对业务逻辑的实现能力有限, 因此其应用主要局限在数字货币领域.

(2) 区块链 2.0 阶段, 以支持图灵完备的智能合约为标志性技术. 图灵完备的智能合约技术使得区块链具备了实现上层业务逻辑及承载部分垂直行业应用的能力, 通过与多种信息化技术的集成重构, 在金融、民生、政务等不同行业催生了新的商务及管理模式, 优化了现有生产关系, 出现了用于实际生产环境的应用方案, 开始从不同角度助力经济社会的发展.

经过前两阶段的发展, 基于异构平台实现的面向不同领域的上层应用, 在既有用户和价值积累的基础上, 产生了与其他区块链及区块链应用交互的外延需求. 如何在不破坏区块链去中心、去信任特性的前提下融合异构的底层技术平台, 打破上层应用边界, 成为当今区块链技术发展的迫切需求, 因此, 跨链技术应运而生.

跨链技术是指跨越单一区块链系统的数据可信边界 (共识机制作用范围), 实现区块链互操作, 即互不影响的两个或多个区块链间的有效协作, 进而实现可信的信息/价值跨链流通的技术.

区块链本质上是基于点对点分布式账本技术实现的多副本状态机, 区块链系统的链内操作实际上是对区块链状态的操作. 具体而言, 区块链共识节点基于共识算法对交易序列达成一致, 并生成不可篡改的区块链账本; 区块链全节点 (状态机) 各自在本地维护着区块链状态, 其基于一致的初始状态 (创世区块), 通过按序执行由区块链账本所记录的一致的交易, 实现一致的状态更新, 而全局一致的状态更新则承载了上层应用逻辑的实现. 区块链系统的链内操作可以抽象为不更改区块链状态的读操作和更改区块链状态的写操作两类基本操作.

与链内操作一样, 跨链操作同样可以抽象为跨链读操作与跨链写操作两类基本操作, 但与链内操作不同的是, 跨链操作的对象在另一个区块链系统内, 跨链操作会被拆分为多个区块链链内操作. 为了实现跨链读、写操作, 需要解决几个关键的技术问题.

(1) 跨链信息传输, 即将一个区块链系统的状态数据或者账本数据传送到另一个区块链系统, 其中, 前者被称作源区块链, 后者被称作目的区块链.

(2) 跨链信任传递, 即目的区块链确认所接收的跨链数据已在源区块链中达成一致.

(3) 跨链操作原子性保障, 即保证跨链操作在各区块链上的链内操作同时执行或者同时不执行.

早期跨链研究基于具体的应用需求, 如跨链验证、跨链资产原子交换、跨链资产转移等, 设计解决方案. 根据应用需求的不同, 研究重心也有差异: 跨链验证研究主要解决跨链信息传输与信任传递问题[3−6]; 跨链代币兑换、跨链代币原子交换研究主要解决跨链操作原子性保障问题[7−12]. 随着对跨链需求的深入认识, 跨链研究转向系统级的解决方案, 在设计跨链架构的基础上提出兼顾上述 3 种关键技术问题的整体解决方案[13−16].

目前已有多项综述工作针对跨链研究进行了梳理, 但是其聚焦于跨链整体解决方案, 缺乏技术体系层面系统的分析与梳理. Buterin 对跨链方案进行了分类与回顾, 将跨链方案分为公证人、侧链/中继与哈希时间锁 3 类[17]; Belchior 等人在定义区块链互操作框架与标准的基础上, 对跨链方案进行了更全面的分类与回顾, 将区块链互操作分为公共连接器、区块链的区块链和混合连接器 3 类, 其中公共连接器包含 Buterin 整理的 3 类方案[18]; Singh 等人[19]与 Johnson 等人[20]的工作聚焦于侧链技术, 对相关研究与项目进行了回顾与分析; Zamyatin 等人[21]与 Robinson[22]从跨链通信的角度分析和综述了已有方案; Schulte 等人则是从跨链资产转移与跨链合约调研两类跨链应用的角度分析和综述了已有方案[23]. 为了帮助后续研究者更深入地理解跨链, 从而更快地加入这一领域的研究, 本文首次系统性地分析了跨链需要解决的关键技术问题, 并综述总结了各个关键技术问题的研究现状, 为读者

分析可行的跨链方案打下基础; 同时从关键技术问题的角度分析典型的跨链整体解决方案, 以便读者更好地理解并评价现有研究工作.

本文第 1 节在定义区块链互操作的基础上, 系统性地分析区块链互操作需解决的跨链信息传输、跨链信任传递、跨链操作原子性保障 3 种关键技术问题. 第 2–4 节针对各关键技术问题, 分别综述分析现有研究工作. 第 5 节对代表性的跨链整体解决方案从关键技术问题的角度进行分析与讨论. 最后, 在第 6 节指出跨链研究领域尚待解决的问题与挑战.

1 跨链问题分析

1.1 区块链互操作定义

跨链技术发展过程中, 不同研究团队对区块链互操作有不同诠释, 具体可以分为广义和狭义两类. 广义的区块链互操作包括区块链与区块链的互操作 (也被称为跨链操作) 以及区块链与其他信息系统的互操作, 中国信息与通信技术研究院区块链研究团队认为区块链互操作是指区块链系统实例与其他区块链系统实例及所有区块链外部系统实例交换信息, 并对所交换信息加以使用[24]. 狭义的区块链互操作仅包括不同区块链系统间的互操作, Jin 等人认为跨链互操作定义是在保留区块链自身不可逆转、可追溯等特性的同时, 实现不同区块链系统间有效的通信与直接的信息交换[25], Lafourcade 等人认为跨链互操作是两个区块链系统协同工作[26].

本文仅针对狭义区块链互操作相关研究进行综述, 将区块链互操作定义为跨越区块链数据可信边界 (共识机制作用范围), 在独立区块链系统间实现的、与区块链链内操作效果一致的可信信息获取与协同状态更新.

由于区块链本质上是基于点对点分布式账本技术实现的多副本状态机, 区块链的链内操作实际上是对区块链状态的操作, 因此链内可以抽象为不更改区块链状态的读操作和更改区块链状态的写操作两类基本操作.

(1) 读操作

读操作是指读取区块链数据, 包括读取区块链状态数据 (如读取账户余额) 和读取区块链账本数据 (如读取某条交易信息). 由于所有区块链全节点均在本地维护了区块链状态, 而读操作不更改区块链状态, 因此区块链的读操作可以仅在一个区块链全节点上执行, 而不需要所有区块链全节点均执行. 当任意账户存在读需求时, 一般通过向一个或多个可信的区块链全节点发起请求, 通过获取这些区块链全节点的本地执行结果获取所需数据.

(2) 写操作

写操作是指更改区块链状态数据 (如更新账户余额). 由于写操作需要更改区块链状态, 而区块链的状态由系统内所有全节点在各自本地独立维护, 因此, 当任意账户存在写需求时, 一般以交易的形式将该写操作打包至区块链账本. 所有全节点通过执行打包进账本中的交易, 在本地执行该写操作, 从而一致地完成对区块链状态的指定更改.

与链内操作一样, 跨链操作同样可以抽象为不更改区块链状态的跨链读操作和更改区块链状态的跨链写操作两类基本操作, 而与链内操作不同的是, 跨链操作的对象在另一个区块链系统内, 一个跨链操作会被拆分为多个交互区块链链内的子操作.

(1) 跨链读操作

跨链读操作是指一个区块链系统 $Chain_1$ 读取另一个区块链系统 $Chain_2$ 的数据, 包括读取区块链系统 $Chain_2$ 状态数据和账本数据. 例如, Alice 在司法区块链 $Chain_1$ 上读取存证区块链 $Chain_2$ 上指定的存证信息 x 用于案件判决, 该跨链读操作被拆分为:

- ACTION $Chain_1$: Alice 发起对存证区块链 $Chain_2$ 上存证信息 x 的跨链读请求并获取存证信息 x.
- ACTION $Chain_2$: 读取存证信息 x 并将该信息返回给区块链 $Chain_1$.

(2) 跨链写操作

跨链写操作是指一个区块链系统 $Chain_1$ 更改另一个区块链系统 $Chain_2$ 的状态数据, 包括跨链原子写操作与跨链非原子写操作.

跨链原子写操作包括跨链研究中常见的资产原子交换、跨链资产转移等, 例如: Alice 与 Bob 在区块链

$Chain_1$ 与区块链 $Chain_2$ 上均拥有账户, Alice 在区块链 $Chain_1$ 上有 10 个代币, Bob 在区块链 $Chain_2$ 上有 5 个代币, Alice 通过跨链写操作使用其在区块链 $Chain_1$ 上的 10 个代币交换 Bob 在区块链 $Chain_2$ 上的 5 个代币, 该跨链原子写操作被拆分为:

- ACTION $Chain_1$: Alice 发起与 Bob 进行代币交换的请求, 并将 10 个代币转移给 Bob.
- ACTION $Chain_2$: Bob 将 5 个代币转移给 Alice.

跨链原子写操作的正确执行需要保证子操作的原子性, 即需要保证 ACTION $Chain_1$ 与 ACTION $Chain_2$ 同时执行或者同时不执行.

跨链非原子写操作包括跨链信息同步等, 例如: 区块链 $Chain_1$ 记录了信息 x, 区块链 $Chain_2$ 记录了信息 x 的备份, 区块链 $Chain_1$ 中信息更新后, 跨链更新区块链 $Chain_2$ 中的备份信息, 该跨链写操作被拆分为:

- ACTION $Chain_1$: 区块链 $Chain_1$ 发起对区块链 $Chain_2$ 上信息 x 的跨链更新请求.
- ACTION $Chain_2$: 更新信息 x.

跨链非原子写操作不需要保证子操作的原子性, 若 ACTION $Chain_2$ 因为网络拥塞等原因没有执行, ACTION $Chain_1$ 不需要撤销更新操作.

现有跨链研究根据跨链操作的实现方式可以分为链上触发模式和链下触发模式两种, 链上触发模式在交互区块链上发起跨链请求、解析跨链请求并根据解析结果触发对应的子操作[13−16,27]. 链下触发模式在链下解析跨链需求、拆分跨链操作, 并在此基础上通过按序触发不同区块链的子操作达到预期跨链效果[13,17,28]. 链下触发模式的核心逻辑均在链下, 链上子操作彼此独立, 区块链仅作为资源使用.

1.2 跨链关键问题

跨链读、写操作的实现需要解决 3 个层面的关键问题: (1) 解决区块链系统间信息传输的问题, 实现链间信息互通. (2) 解决区块链系统间信任传递的问题, 保证链间信息可信. (3) 解决跨链操作原子性保障的问题, 保证相互依赖的跨链子操作一致执行或者一致不执行.

(1) 跨链信息传输

跨链读、写操作可以拆分为各交互区块链内部的子操作, 由彼此独立的各交互区块链协同工作实现, 协同工作的基础是必要的信息互通. 但是区块链获取数据需要区块链系统内的所有节点都获取一致的数据, 而即使在外部数据源与区块链网络连通的情况下, 由于无法保证区块链系统中的每一个节点对该外部数据源具有相同的访问权限和视图, 因此无法仅依靠网络传输实现区块链对外部数据的获取.

因此, 为了实现跨链读、写操作, 首先需要解决跨链信息传输问题, 保证区块链系统可以获取其他区块链系统的数据. 例如在上述跨链读操作的例子中, 司法区块链 $Chain_1$ 需要能够获取存证区块链 $Chain_2$ 内部的存证信息 x.

跨链信息传输是指将一个区块链系统的状态数据或者账本数据传送到另一区块链系统中. 为了更好地描述跨链信息传输, 本文将一次跨链传输中发起跨链信息传输的区块链系统称为源区块链, 将接收跨链信息的区块系统链称为目的区块链.

(2) 跨链信任传递

区块链的数据是指区块链系统内所有节点达成一致的数据, 由于区块链共识机制只能保证共识范围内节点对区块链账本数据与状态数据达成一致, 而协同完成跨链操作的区块链彼此独立, 区块链无法直接确认其获得的对端区块链账本数据与状态数据的可信性. 在上述跨链读操作中, 司法区块链 $Chain_1$ 获取存证信息 x 后并不能确定这是在存证区块链系统 $Chain_2$ 中达成了一致的数据.

因此为了实现跨链读、写操作, 在跨链信息传输的基础上, 还需要打破区块链可信性的边界, 实现区块链之间的信任传递问题, 让司法区块链 $Chain_1$ 的节点可以确认通过跨链传输获取的区块链 $Chain_2$ 状态数据或者账本数据的可信性.

跨链信任传递是指在跨链传输的基础上, 实现目的区块链对跨链信息的可信性确认, 即确认所获取的跨链信息已在源区块链中达成一致.

(3) 跨链操作原子性保障

读操作与写操作作为区块链系统的元操作, 其执行只有成功与失败两种状态, 若操作执行失败, 则区块链系统需要回滚至操作执行前的初始状态. 由于读操作在执行过程中并不会更新区块链状态, 因此只有写操作在执行失败时会进行区块链状态的回滚.

跨链读、写操作的执行也只有成功与失败两种状态, 跨链写操作执行失败时, 跨链系统需要进行状态的回滚. 跨链系统的状态是交互区块链状态的并集, 一次跨链写操作可能导致目的区块链或者目的区块链与源区块链的状态更新. 若跨链写操作同时涉及目的区块链与源区块链的状态更新, 则在跨链写操作执行失败时, 需要同时回滚目的区块链与源区块链上已有的状态更新. 在上述资产交换例子中, 若 ACTION $Chain_1$ 正确执行, 将 Alice 的 10 个代币转移给了 Bob, 而 ACTION $Chain_2$ 由于网络拥塞、攻击等原因执行失败, 则 Bob 需要在区块链 $Chain_1$ 上将已收到的 10 个代币退还给 Alice, 否则将会导致 Alice 损失 10 个代币, 不符合资产交换成功或者失败的两种预期结果.

因此为了实现跨链写操作, 面向跨链转账、跨链资产交换等同时涉及源区块链与目的区块链状态更新的场景, 还需要保证跨链操作的原子性, 这类写操作就是跨链原子写操作.

跨链操作的原子性是指拆分在多个区块链系统内执行的子操作要么全部执行成功, 要么均不执行.

2 跨链信息传输关键技术

跨链信息传输是指将一个区块链系统的状态数据或者账本数据传送到另一区块链系统中. 由于区块链系统是一个封闭的系统, 无法主动向外部系统发送数据, 只能记录向外部系统发送特定数据的意图[29], 因此跨链信息传输方案一般通过引入链下转发实体实现区块链系统间的数据转发.

跨链信息传输的基本流程可以总结抽象为图 1, 主要包括获取跨链信息与转发跨链信息两个阶段.

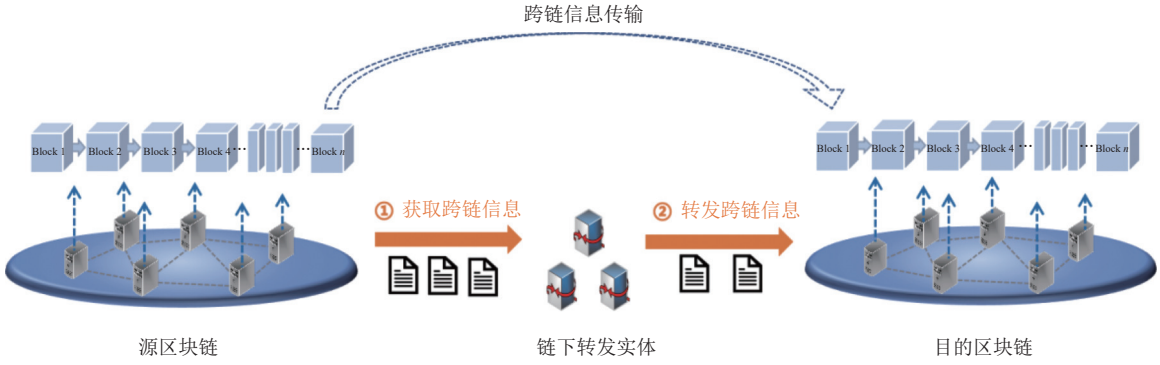


图 1 跨链信息传输示意图

(1) 获取跨链信息

链下转发实体通过监听源区块链获取待传输的跨链信息. 具体地, 链下转发实体监听源区块链记录跨链信息的对象, 包括交易、收据以及状态. 通过交易记录跨链信息是指用户在源区块链发起交易时将跨链信息直接存储在交易的某些字段中[6,30−32]; 通过收据记录跨链信息是指源区块链在处理用户跨链请求时, 以事件的形式将用户的跨链意图存储在区块链的收据日志中[14,15,33,34]; 通过状态记录跨链信息是指源区块链基于智能合约将跨链信息存储在区块链状态中[15].

(2) 转发跨链信息

链下转发实体将待传输的跨链信息转发至目的区块链. 链下转发实体一般通过在目的区块链上发起包含跨链信息的交易 (后文称为转发交易) 实现消息转发, 这是因为区块链作为多节点组成的系统, 需要保证系统中的所有节点均获取相同的信息, 而基于共识机制可以对要执行的交易 (即待处理的跨链操作) 达成一致.

2.1 跨链信息传输方案

BTC Relay[3]在以太坊上构建比特币轻客户端, 使得以太坊具备验证比特币交易的能力, 其在传输层面实现了两类信息从比特币到以太坊的单向跨链传输: 一类是比特币交易信息, 通过用户监听、转发特定交易实现; 一类是比特币区块头, 通过基于激励机制引入的链下转发节点 Relayer 监听、转发比特币区块头实现, 为以太坊上比特币交易的验证提供基础. 在 BTC Relay 中, 用户在以太坊上验证比特币交易需要向提交交易所在区块头的 Relayer 支付手续费, 从而激励 Relayer 参与跨链传输与 BTC Relay 的构建. BTC Relay 支持多 Relayer, 若多个 Relayer 向以太坊转发同一高度的比特币区块头, 仅第 1 笔被以太坊打包的转发交易生效, 以保证跨链信息仅被处理一次, 即保证跨链传输的幂等性. Peace Relay[4]、ETH Relay[5]仿照 BTC Relay 设计、实现了类以太坊区块链之间的跨链交互. RSK[6]、Loom[35]等侧链研究同样基于用户实现跨链信息在主链、侧链之间的传输.

Cosmos、BitXHub、Polkadot 等方案针对跨链信息传输问题提出了专门的传输协议.

Cosmos 提出的区块链间通信协议 (IBC 协议) 是一种端到端、面向连接、有状态的协议, 实现了独立分布式账本上模块之间的可靠、有序和认证通信[29]. IBC 协议抽象了轻客户端、连接 (connection)、通道 (channel) 以及转发节点 (Relayer). 轻客户端抽象封装验证区块链数据的属性及方法, 为跨链信任传递提供支撑. 连接维护区块链间的关联状态, 与轻客户端一起实现跨链信任传递. 通道维护不同区块链的应用模块间信息传输的状态, 提供有序传输与无序传输两种模式, 其中有序模式基于 IBC 数据包序列号与跟踪传输状态实现, 并基于 IBC 数据包的超时时间提供丢包检测功能. 转发节点完成区块链间的信息传输, 与 BTC Relay 类似, Cosmos 通过激励机制引入转发节点, 但其核心协议中不包括对应激励机制, 由应用提供具体激励机制. 同样地, 当多个转发节点转发同一跨链信息时, 为了避免目的区块链重复处理, 仅最先提交的生效. Cosmos 的跨链数据包括两类: 一类是记录了应用跨链信息的 IBC 数据包, 一类是用于跨链信息验证的区块头. 特别地, 在获取跨链数据阶段, Cosmos 支持监听事件与通道状态两种方法; 在转发跨链数据阶段, 转发节点可以将多个 IBC 数据包打包在一笔目的链交易中.

BitXHub 提出的链间消息传输协议 (IBTP 协议) 是一种类似 TCP/IP 的链间传输协议, 其同样通过引入转发节点 (跨链网关, Pier) 实现了记录了跨链信息的 IBTP 数据包的传输[14,36]. IBTP 在不同的区块链交互模式, 跨链传输的实现方式不同.

● 区块链直接进行交互时, 源区块链跨链网关通过 P2P 网络将 IBTP 数据包传输给目的区块链跨链网关, 再由目的区块链跨链网关将 IBTP 数据包以交易的形式提交给目的区块链.

● 区块链借助中继链进行交互时, 跨链传输需要通过多跳传输实现: 源区块链跨链网关将 IBTP 数据包以交易的形式提交给中继链, 再由目的区块链跨链网关将 IBTP 数据包以交易的形式提交给目的区块链. 在大规模跨链场景下, 还需要实现 IBTP 数据包在中继链间的传输.

与 Cosmos 在区块链上实现跨链传输的有序性不同的是, BitXHub 在跨链网关上借助缓冲池对 IBTP 数据包的排序实现跨链传输的有序性. 同时, BitXHub 考虑到单跨链网关可能产生恶意行为, 采用跨链网关集群的方式增强跨链网关可信度, 并设计了主备节点模式保证交互区块链间只有一个活跃的主节点处理跨链请求, 从而避免跨链信息的重复传输. 备用节点在主节点宕机时主动升级为主节点处理跨链请求.

Polkadot 针对区块链在可扩展性、可伸缩性、安全性等方面普遍存在的问题, 分离共识架构中的一致性和有效性, 由各平行链并行出块, 中继链统一共识[16]. 为此, Polkadot 设计了收集者角色收集平行链交易, 打包平行链区块. Polkadot 设计的平行链间跨链信息传输协议 XCMP 也需要借助各平行链收集者实现连接同一中继链的平行链间的信息传输[37]. 具体地, 源区块链收集者将跨链信息、目的链与时间戳一起放入源区块链的出队列, 目的区块链收集者通过定期向源区块链收集者请求跨链数据获取目的链与其匹配的跨链信息, 并将其放入目的区块链入队列, 收集者出块时将出、入队列信息打包在内, 从而实现跨链信息传输. XCMP 的目标是高效、有序、可验证、无遗漏的传输, 但是由于目前仍在设计阶段, 具体的实现方式还不明确. 由于 XCMP 仍在设计实现阶段, 目前平行链通信使用资源开销更大的水平中继路由信息传输 (HRMP) 借助中继链通过多跳传输实现.

WeCross[13]、HTLC[8]等方案尽管实现了跨链交互, 但是并没有直接实现区块链间的信息传输, 而是通过链下触发的方式, 由目的区块链的特殊用户/转发节点接收跨链信息并触发目的区块链上对应的操作.

WeCross 借助由部署区块链的机构为区块链搭建的、可信的跨链路由 (对应转发节点) 实现链下触发. 如图 2 所示, 源区块链跨链路由通过监听源区块链输出队列获取记录了目标区块链资源路径、目标接口、调用参数等信息的跨链操作请求, 并通过网络将跨链操作转发给目的区块链的跨链路由, 实现跨链信息从源区块链到目的区块链跨链路由的传输. 目的区块链跨链路由根据获取的跨链操作请求在目的区块链上触发对应操作.

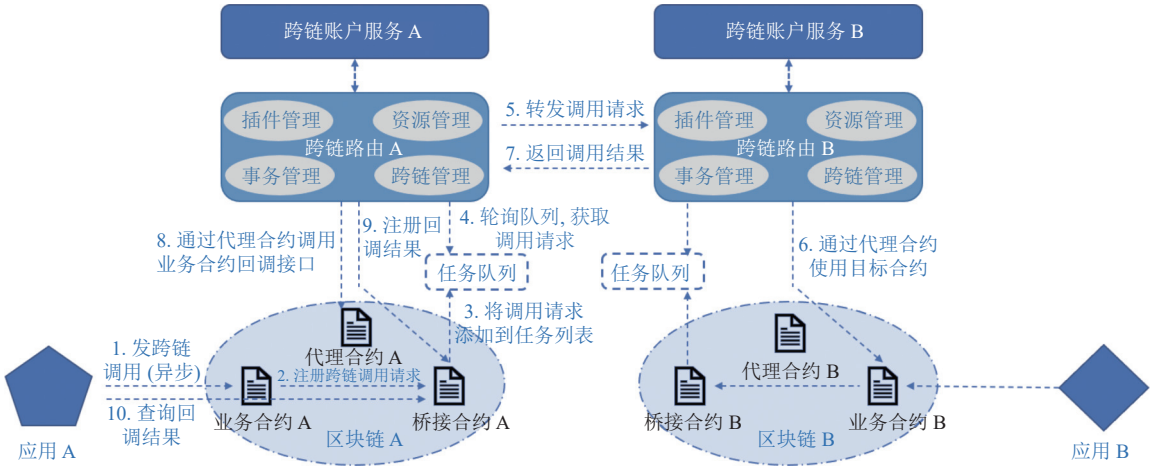


图 2 WeCross 交互示意图[13]

HTLC 借助参与跨链交互的用户实现链下触发, 特别地由于用户同时在源区块链上与目的区块链上, 因此没有引入额外的角色进行跨链信息传输, 而是由用户自行监听源区块链获取跨链信息并触发目的区块链对应操作.

Xiao 等人[38]、Luo 等人[39]、Belchior 等人[40]提出的方案同样引入转发节点实现链下触发. 其中, Luo 等人[39]提出的方案还引入了路由区块链, 用于维护区块链的路由节点信息 (对应转发节点), 从而支撑源、目的区块链路由节点间的通信. Belchior 等人进一步提出了容忍崩溃的网关节点 Hermes[41], 支持网关节点自恢复和主备两种崩溃恢复方法.

2.2 跨链信息传输方案评价

(1) 评价指标

根据本节汇总的跨链信息传输方法, 我们整理得到表 1, 从转发节点安全假设、幂等性、传输开销、可扩展性、有序性以及可靠性几个方面进行分析与评价.

表 1 跨链信息传输方案评价

文献	转发节点安全假设	幂等性	传输开销	可扩展性	有序性	可靠性			
						转发节点数量	转发节点奖励机制	转发节点恢复机制	跨链传输丢包处理
[3−5]	拜占庭	可保证	高	低	不支持	多个	用户向转发节点支付服务费	无	不支持
[14]	拜占庭	可保证	低	低	支持	多个 (主备)	无	无	不支持
[15]	拜占庭	可保证	高	高	支持	多个	传输协议不包含激励机制, 应用模块提供	无	仅支持丢包检测
[16]	拜占庭	可保证	低	低	支持	多个	传输协议不包含激励机制, 平行链提供	无	状态根保证
[13]	诚实	可保证	低	低	不支持	单个	无需激励	无	不支持
[38]	诚实	可保证	低	低	不支持	单个	无	无	不支持
[39]	诚实	可保证	低	低	不支持	单个	无	无	不支持
[41]	诚实	可保证	低	低	不支持	多个 (主备)	无	基于日志的主备节点恢复	不支持

1) 转发节点安全假设: 安全假设包括诚实节点假设以及拜占庭节点假设. 诚实节点假设下, 节点只存在崩溃问题; 拜占庭节点假设下, 节点还存在伪造跨链信息、丢弃跨链信息等问题.

2) 幂等性: 区块链系统不会重复处理重复提交的跨链信息.

3) 传输开销: 一次跨链信息传输在目的区块链上的计算、存储开销, 若交互区块链间存在多个转发节点且均执行转发操作, 则传输开销包含所有转发节点在目的区块链上的计算、存储开销.

4) 可扩展性: 即跨链信息传输方案扩展至其他、多元应用的能力.

5) 有序性: 目的区块链系统按照源区块链系统发送跨链信息的顺序进行处理.

6) 可靠性: 确保跨链信息能够被转发到目的区块链上, 若跨链信息无法被转发到目的区块链, 源区块链能够及时得知. 我们面向转发节点可靠性与传输协议可靠性两个角度, 分别从转发节点数量、转发节点奖励机制、转发节点恢复机制与跨链传输丢包处理几个方面进行评估.

a) 转发节点数量: 交互区块链间用于转发跨链信息的节点数量 (包括主备节点), 用于评估跨链传输方案对转发节点崩溃问题的容忍程度. 跨链传输最基本的需求是交互区块链间至少有一个诚实的转发节点, 转发节点越多, 对转发节点崩溃问题的容忍程度越高.

b) 转发节点奖励机制: 转发节点提供跨链传输服务的动力. 跨链传输服务需要大量成本, 转发节点缺乏提供跨链传输服务的动力, 因此需要额外引入奖励机制, 保证交互区块链间有足够的转发节点提供跨链传输服务.

c) 转发节点恢复机制: 转发节点崩溃宕机后的恢复机制, 用于评估跨链传输方案对转发节点崩溃问题的容忍程度.

d) 跨链传输丢包处理: 包括丢包检测方法与对丢包的恢复方法, 用于评估跨链传输方案对丢包问题的.

(2) 方案对比

● 幂等性方面: 幂等性是跨链交互对跨链传输的基本要求, 若跨链信息被重复转发至目的区块链并被重复处理, 将影响跨链交互的正确性. BTC Relay[3]、Peace Relay[4]、ETH Relay[5]、Cosmos[15]几种方案通过目的区块链识别并忽略重复的跨链信息保证跨链传输的幂等性, 其他方案通过可靠的单一转发节点或者主备节点保证跨链信息仅被传输一次. 现有方案均能保证幂等性.

● 转发节点安全性假设方面: WeCross [13]等基于代理传输的方案均采用诚实节点假设, 而 BTC Relay 等基于链上传输的方案均采用拜占庭节点假设, 与前者相比在实现可靠传输时需要考虑更多的节点可靠性问题.

● 传输开销方面: BTC Relay、ETH Relay、Peace Relay、Cosmos 几种方案交互区块链间有多个转发节点处理跨链信息, 目的区块链会接收多笔该跨链信息的转发交易, 因此传输开销高. 其他方案目的区块链只会接收到一笔该跨链信息的转发交易, 因此传输开销低.

● 可扩展性方面: 仅 Cosmos 的 IBC 协议采用松耦合的设计模式, 区分链间连接与应用间通道, 支持多个通道对连接的复用, 使得应用层面的扩展无需关注链间连接, 可扩展性较高.

● 有序性方面: Cosmos、BitXHub 与 Polkadot[16]支持有序传输, 其中 Cosmos 基于通道内传输的 IBC 数据包序列号以及消息确认机制, 保证跨链信息传输的有序性; BitXHub 在跨链网关上借助缓冲池对 IBTP 数据包的排序实现跨链传输的有序性, Polkadot 则是通过输入输出队列保证跨链信息传输的有序性.

● 可靠性方面: 1) 采用诚实节点假设的方案. 仅 Hermes[41]在交互区块链间引入了多个转发节点, 且针对转发节点可能存在的崩溃宕机问题设计了基于日志的恢复机制, 可以容忍转发节点的崩溃, 其他方案在转发节点崩溃或者被攻击时将无法提供跨链传输服务. 除 WeCross 转发节点由部署区块链的机构搭建, 无需考虑动力问题外, 其他方案均不提供转发节点奖励机制, 在实际应用中可能存在可用性问题. 上述方案均未设计丢包检测机制与丢包恢复机制, 无法处理因目的区块链拥塞等原因导致丢包问题. 2) 采用拜占庭节点假设的方案. 上述方案均在交互区块链间引入了多个转发节点, 但是均未设计转发节点崩溃恢复机制, 对崩溃问题的容忍程度与网关节点数量相关. 除 BitXHub 外, 其他方案均有对转发节点奖励机制的讨论, BTC Relay、ETH Relay、Peace Relay 针对转发节点设计了跨链服务奖励, 但是缺乏对激励机制有效范围与可行性的进一步讨论与分析, 在实际应用中存在可用性问题: BTC Relay 中 Relayer (对应转发节点) 的跨链服务奖励远小于其成本, 项目上线一年后所有 Relayer 都撤出平台; Cosmos 与 Polkadot 的核心机制并不包括对转发节点的激励, 而是将这部分工作留给了应用与平行链. 仅

Cosmos 与 Polkadot 提供了丢包处理方法, 其中 Cosmos 仅在有序模式下支持丢包检测, 其基于 IBC 数据包超时时间实现, 若出现丢包则关闭通道; Polkadot 基于中继链上存储的状态根, 验证跨链数据的完整性, 从而避免丢包; 其他方案即使转发节点向目的区块链发送了跨链信息, 也可能因为目的区块链拥塞等原因而“丢包”.

3 跨链信任传递关键技术

跨链信任传递是指在跨链传输基础上, 实现目的区块链对跨链数据可信性的确认, 保证跨链数据已在源区块链中达成共识并写入主链. 其中, 跨链数据包括源区块链账本数据及状态数据: 账本数据一般指交易, 验证源区块链账本数据的可信性本质上是验证源区块链交易的存在性, 即一笔交易是否被写入源区块链主链; 状态数据是各区块链节点在本地维护的全局一致的区块链状态.

为了确保目的区块链的所有节点对跨链数据达成一致, 最需要关注的两个问题.

(1) 由谁来执行跨链数据验证?

(2) 如何实现跨链数据验证?

下文将围绕上述问题, 从跨链数据验证主体以及跨链数据验证方法两个角度分别综述当前研究, 并进行总结评价. 为实现对跨链研究的思路整理及扩展, 所综述的方案不局限于各跨链方案, 还包括其他区块链研究中可用于跨链信任传递的相关技术.

3.1 跨链数据验证主体

目的区块链对跨链数据的验证, 既可以通过目的区块链自行验证跨链数据实现, 也可以通过目的区块链确认跨链数据的验证结果实现. 前者在目的区块链链上进行跨链数据验证, 后者在链下进行跨链数据验证, 根据验证发生的位置可以将二者分别称为链上验证模式与链下验证模式.

(1) 链上验证模式

链上验证模式中, 目的区块链将自行验证跨链数据可信性, 即目的链区块链节点各自完成跨链数据验证并对验证结果进行共识最终达成一致. 链上验证模式的跨链信任流可以抽象为如图 3, 信任从源区块链直接传递至目的区块链.

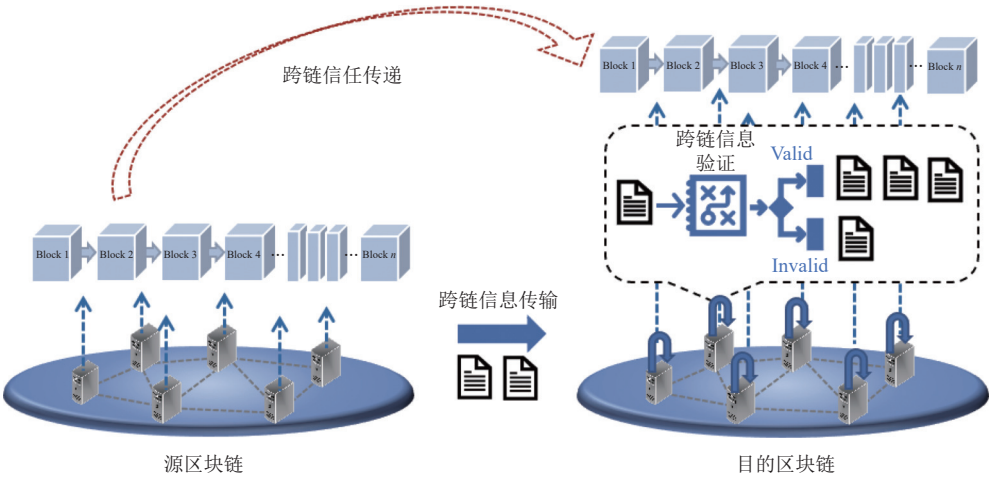


图 3 跨链信任传递链上验证模式示意图

链上验证模式一般以在目的区块链上部署源区块链数据验证智能合约的形式实现. 例如: BTC Relay 在以太坊上部署智能合约实现对比特币区块头与交易的验证[3]; Cosmos 的 IBC 协议抽象出客户端模块实现对对端区块链的验证[29].

链上验证模式下验证算法公开透明, 安全程度高. 然而, 该方法只适用于支持智能合约的区块链, 整体性能与开销取决于源块链数据验证算法, 若源区块链验证算法复杂, 则整体性能较低且需要大量的链上计算资源与存储

资源: ETH Relay 指出在类以太坊区块链上实现以太坊轻客户端的开销极大, 即使是优化后的验证方案, 仅以太坊共识主体的验证就需要大约 300 万 gas (一笔普通转账交易大约需要 21 000 gas)[5].

(2) 链下验证模式

链下验证模式中, 目的区块链将跨链数据可信性验证委托给一个或者一组链下代理, 目的区块链节点仅确认链下代理的验证结果. 链下验证模式的信任流可以抽象为图 4, 源区块链与目的区块链间的信任传递由源区块链与链下代理间的信任传递以及链下代理与目的区块链间的信任传递组合实现.

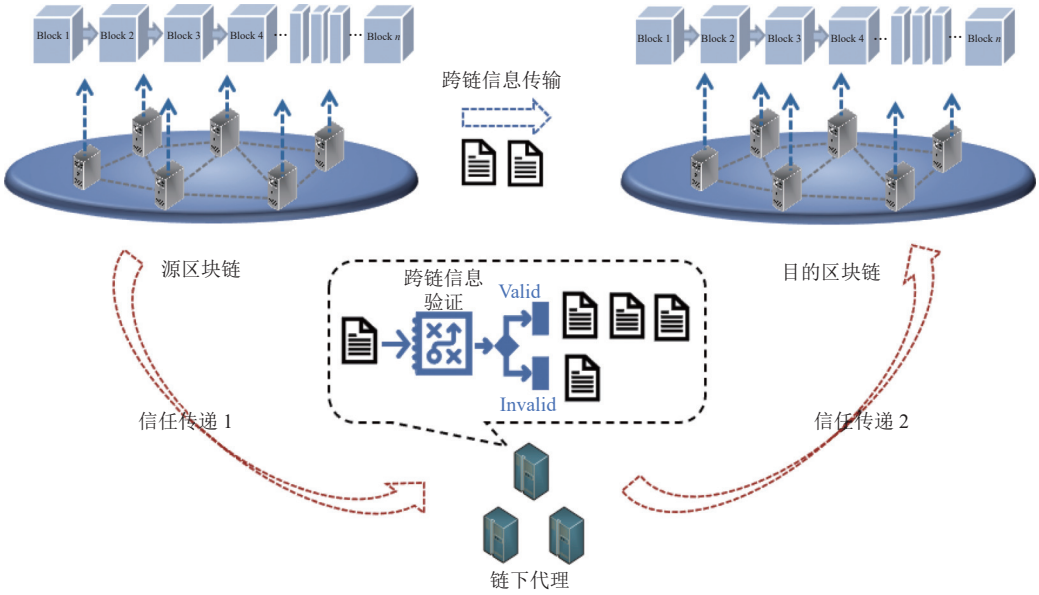


图 4 跨链信任传递链下验证模式示意图

链下代理的选择将影响链下代理与目的区块链间信任传递的方式, 根据目的区块链与链下代理信任传递构建的原理, 可以将链下验证模式分为基于身份的链下验证模式以及基于行为验证的链下验证模式.

1) 基于身份的链下验证模式

在基于身份的链下验证模式中, 目的区块链信任链下代理, 仅需验证收到的跨链信息是否有正确的链下代理背书. 其中, 链下代理可以是单点也可以是多点.

WeCross 的链下代理是区块链部署机构所搭建的跨链路由, 区块链信任其对应跨链路由[13,42]. 跨链路由代替目的区块链完成跨链数据验证后触发目的区块链子操作, 目的区块链仅需检测跨链子操作由跨链路由发起. 该方案链上操作简单, 整体性能较高, 但是仅面向联盟链场景, 需要信任单一的跨链路由, 存在单点故障风险.

Loom 的链下代理是网关 Oracle, 其一般运行在参与区块链 DPoS 共识的委托人节点上[35]. 当代币从侧链撤回至主链时, 网关 Oracle 基于用户提交的跨链交易与相关证明进行验证, 并为验证通过的交易签名, 主链仅需检测跨链交易具有网关 Oracle 签名. 与 WeCross 类似, 该方案引入了中心化信任的网关 Oracle, 同时网关 Oracle 的处理将会影响跨链交互的性能.

Drivechain[43]的链下代理是参与联合挖矿的矿工节点, 同时为源区块链与目的区块链出块的矿工组在目的区块链上对跨链数据的可信性进行投票, 实现目的区块链对跨链数据的确认. 该方案在不支持智能合约的区块链上也适用, 同时通过多矿工投票避免了单点故障. 但是 Drivechain 的安全性仅由联合挖矿矿工保证, 只适用于区块链强相关的场景 (主链与侧链), 无法满足独立区块链跨链交互的需求, 同时若联合挖矿算力不够高, 攻击者可以轻易地实现对 Drivechain 的攻击.

RSK[44]在不同阶段使用不同的链下代理: 初期使用可信联邦, 即多家社会中有较高声誉的组织、公司作为链下代理, 通过多签投票向目的区块链提供跨链数据背书; 随着参与联合挖矿算力的不断提高, 加入联合挖矿的矿工

节点作为链下代理, 并逐渐增大矿工投票比重直至最终只有矿工投票, 在此过程中去中心化程度不断提升.

基于身份的链下验证模式中, 目的区块链信任链下代理, 链上仅需验证链下代理的签名, 与链上模式相比方案易于实现、性能显著提升且链上开销低. 但是该模式引入了对链下代理的信任, 不适用于不存在可信链下代理的跨链场景.

2) 基于行为验证的链下验证模式

在基于行为验证的链下验证模式中, 目的区块链不信任代替它进行数据验证的链下代理, 而是信任链下代理的验证行为本身. 通过技术手段, 目的区块链确认链下代理的验证行为后, 信任可以传递到目的链. 当前主要基于可信执行环境与零知识证明技术保证链下代理行为可验证.

可信执行环境 (trusted execution enviroment, TEE) 是通过硬件隔离手段对运算和操作进行保护的技术, 在不破解硬件的前提下可以保证执行不可篡改. 可信执行环境中的代码是公开、可审计的, 通过提前审计以及可信执行环境签名可以保证输出是特定程序的执行结果[45].

Robinson 等人提出的跨链方案基于 TEE 实现链下代理[46]. 链下代理在 TEE 中部署源区块链数据验证方法, 并生成存储在 TEE 中的公私钥对. 目的区块链审计 TEE 中的程序, 并存储 TEE 公钥. 链下代理在 TEE 中完成跨链数据验证后, 使用 TEE 私钥对跨链数据验证结果进行签名背书. 目的区块链根据交易的 TEE 签名可以确认链下代理在 TEE 中执行了已审计的程序, 完成了对源区块链数据的验证.

此类方案保留链下验证高性能、低开销优势的同时通过代码审计以及硬件保证削减了对链下代理的信任, 但是由于可信执行环境内的认证密钥以及硬件隔离均由制造商提供, 转而需要信任制造商不会留有后门, 而由于可信执行环境存储空间有限, 链下代理的功能在一定程度上会被限制.

零知识证明技术是一种证明者在不向验证者提供任何有用信息情况下, 通过生成一个数学证明, 使得验证者相信对于某个待验证的问题, 其持有对应答案的技术[47].

Zkrelay 基于零知识证明实现链下验证[48], 其链下代理对源区块链跨链信息进行验证, 并在成功时生成证明其链下行为的零知识证明文件, 同时将该证明文件转发至目的区块链. 目的区块链基于提前存储的验证密钥以及证明文件可以确认跨链信息确实通过了链下代理的验证.

基于零知识证明技术构建链下代理无需引入任意程度的信任, 但是链下代理生成零知识证明文件的过程非常复杂, 而链上对于零知识证明文件的验证也比许多跨链数据验证的复杂度更高, 整体性能低.

3.2 跨链数据验证主体评价

(1) 评价指标

根据本节综述的各方案, 我们整理出表 2 对相关研究工作进行总结与对比, 其中的维度包括去中心化程度、信任锚点、开销、性能以及各自独特的优劣.

表 2 跨链数据验证主体评价

文献	验证主体分类	去中心化程度	信任锚点	开销		性能		其他
				存储开销	计算开销	吞吐量	时延	
[3]	链上	区块链	区块链共识	高	高	中	中	—
[4]	链上	区块链	区块链共识	高	高	低	高	未实现
[15]	链上	区块链	区块链共识	高	高	中	中	—
[13,42]	链下 (身份)	单节点	可信跨链路由	链上低 链下高	链上低 链上高	高	低	—
[44]	链下 (身份)	少数节点/区块链	联邦/区块链共识	链上低 链下高	链上低 链下低	高	中	局限性强, 只适用于挖矿节点同时为两条链工作的场景
[46]	链下 (行为)	单节点/少数节点	可信执行环境	链上低 链下高	链上低 链下高	高	低	TEE可能存在后门; 存储空间有限
[48]	链下 (行为)	单节点	密码学	链上低 链下高	链上高 链下非常高	高	高	可以通过降低验证、同步源链数据的频率, 降低开销

1) 去中心化程度: 即验证主体的去中心化程度, 用于描述方案抵御验证主体单点故障与联合作恶的能力.

2) 信任锚点: 即当前方案对验证主体的信任来源, 信任锚点的选择会影响跨链信任传递的安全性.

3) 开销: 包括存储开销与计算开销, 链下验证模式将分别标注链上与链下的存储、计算开销.

4) 性能: 包括吞吐量与时延, 其中吞吐量是指单位时间内完成验证的次数, 时延是指完成一次验证的时间, 为了能够横向比较不同验证主体的性能, 假定所有方案跨链信息验证复杂度一致.

5) 其他: 各方案独特的优缺点.

(2) 方案对比

● 去中心化程度方面: 链上验证方案去中心化程度与区块链相同, 去中心化程度较高. 链下验证方案中 RSK[41] 与 Robinson 等人的方案[46]引入了多个验证节点, 但是 RSK 后期依赖联合挖矿, 理论上若所有矿工均加入联合挖矿则其去中心化程度与区块链相当. WeCross[13,42]与 Zkrelay[48]仅引入单个验证节点, 若验证节点故障、被攻击或者作恶, 则无法提供链下验证功能, 甚至提供虚假的验证结果, 影响跨链操作安全性.

● 信任锚点方面: 链上验证方案的信任来源是区块链本身. 链下验证方案中, 基于身份的链下验证方案的信任来源是验证者本身, 其中, RSK 后期的验证者是联合挖矿矿工, 矿工是区块链信任构建的基础, 若 RSK 有足够的联合挖矿矿工, 则后期其信任来源等同于区块链; 基于行为验证的链下验证方案的信任来源是验证链下行为的技术, Robinson 等人的方案的信任源自 TEE 技术, 但是由于 TEE 很有可能存在厂商预留的后门, 因此一定程度上该方案还需要信任 TEE 厂商, 而 Zkrelay 的信任源自密码学, 无需其他额外信任.

● 开销方面: 链上验证方案需要在链上存储源链数据并完成验证, 一般计算开销与存储开销较大, 这极大制约了方案的落地. 链下验证方案中, 基于身份的链下验证方案与基于 TEE 技术的链下验证方案的链下部分承担了源区块链跨链数据存储与验证功能, 链上仅需验证签名, 开销远小于链上验证方案. 然而基于零知识证明的链下验证方案链下需要存储源链数据并基于密码学生成证明文件, 计算过程非常复杂, 计算开销显著高于其他方案, 一般通过批处理降低验证频率, 保证该方案的开销可被接受.

● 性能方面: 链上验证方案在区块链上进行跨链数据验证, 具体吞吐量与时延取决于区块链本身的设计, 采用确定性共识、模组化的 Cosmos[15]性能均优于采用概率性共识的类以太坊 (Peace Relay), 但是由于链上计算需要多节点共识, 整体性能均较低. 链下验证方案除 Zkrelay 外, 其他方案性能均优于链上方案, Zkrelay 链下证明生成过程非常复杂, 性能较低, 当跨链验证较为简单时, 其性能甚至不如链上验证.

3.3 跨链数据验证方法

跨链数据验证 (可信性验证) 是指验证跨链数据在源区块链中是否达成了一致, 其中跨链数据包括源区块链账本数据及状态数据. 账本数据一般指交易, 验证源区块链账本数据的可信性本质上是验证源区块链交易的存在性, 即一笔交易是否被写入源区块链主链. 状态数据是各区块链节点在本地维护的全局一致的区块链状态 (如: 比特币区块链的 UTXO 集合[2], 以太坊区块链的状态树[49]), 区块链节点通过按序执行账本中的交易完成本地维护的区块链状态的更新. 面向账本数据验证和状态数据验证两个问题, 下面将分别综述其验证方法.

(1) 账本数据验证

账本数据验证一般需要解决如下两个问题: 1) 交易打包验证, 即验证交易 Tx 是否被打包在区块 B_n 中. 2) 区块一致性验证, 即验证区块 B_n 是否在源区块链主链上.

1) 交易打包验证

验证跨链交易 Tx 是否被打包在区块 B_n 中最直接的方法是验证者查询交易 Tx 是否在区块 B_n 包含的全部交易序列$\{Tx\}$中, 然而这种方法需要验证者存储包含交易序列$\{Tx\}$的区块 B_n, 而一个区块可能包含几百甚至几千笔交易, 存储开销相对较大, 尤其是在链上验证模式中, 所有区块链全节点都是验证者, 整体存储开销随区块链规模线性增长.

BTC Relay、Cosmos、WeCross 等大多数跨链研究均基于区块链交易树进行交易打包验证[3,13−15]. 大多数区块链在生成新区块 B_n 时, 会以该区块包含的交易序列$\{Tx\}$为叶子节点构建 Merkle Tree[50]、Merkle Patricia Tree[51] 等形式的交易树, 并将交易树树根存储在区块头 BH_n 中. 验证者基于交易树树根、跨链交易以及对应的验证路径即可验证跨链交易是否在交易树中. 这种方法仅需要验证者存储包含交易树树根的区块头 BH_n, 与存储完整区块

的方案相比开销较小, 但是无法适用于 Fabric 等没有交易树的区块链.

2) 区块一致性验证

验证者如何验证区块是否在源区块链主链上, 取决于源区块链的共识算法. 区块链的共识算法分为确定性共识以及概率性共识两大类[52], 前者只会形成一条区块链, 区块一旦形成不可更改; 后者需要在众多区块链中确定一条主链, 由于主链可能发生变更, 因此链上区块包含的交易也可能被回滚.

当源区块链的共识算法是 PBFT[53]等确定性共识算法时, 因为只有一条区块链存在, 验证者仅需验证区块是否符合源区块链出块规则. 例如, Cosmos 通过验证区块头是否拥有足够的可信验证者签名投票实现对采用 tendermint 共识的区块链的验证[29]. Cosmos 检验区块头的验证者的投票权重是否超过可信验证者集合总投票权重的 2/3, 如果通过验证则认定为有效区块. 同时, 如果该区块的验证者集合与目的区块链维护的可信验证者集合不一致, 则在认定该区块有效后, 将可信验证者集合更新为该区块的验证者集合, 从而完成可信验证者的更新.

当源区块链的共识算法是工作量证明类[2,54]、权益证明类[55−57]等概率性共识算法时, 验证者还需基于源区块链主链确定规则进一步验证区块是否在源区块链主链上. 例如, BTC Relay 实现了对比特币区块链的验证. 比特币区块链采用 PoW 共识算法, 并通过最长链规则确定主链. BTC Relay 从一个确定的区块高度开始不断同步后续区块头; 同时设置了确认期, 保证目标区块后足够多的有效区块, 从而保证所同步的比特币区块头链为最长链.

由于验证概率性共识区块链中的区块要同步该区块前的所有区块, 验证开销与区块高度呈线性正相关. 即使仅同步、验证源区块链区块头, 随着区块高度的不断增加以及跨链场景下区块链数目的增多, 验证者依旧需要消耗大量的存储、计算资源.

为了降低跨链场景下验证者的开销, ETH Relay 通过乐观接收区块头的方法实现区块一致性验证[5]. ETH Relay 中目的区块链乐观地接收转发节点在每个区块高度提供的第 1 个源区块链区块, 并对该区块进行公示. 公示期间任意链下客户端可以对该区块进行链下审核, 并在审核不通过时在目的区块链上发起挑战, 只有当链下客户端发起挑战时, 目的区块链才依据源区块链规则验证该区块是否有效. 这种优化方法在合适的奖惩机制的配合下可以很好地降低目的区块链上的计算开销, 但是无法降低目的区块链上的存储开销.

Garoffolo 等人基于 NIPoPoW 构建简洁的区块一致性证明, 实现无需信任的 PoW 侧链[32]. NIPoPoW 是一种超轻客户端, 其实现了链外节点对区块链一致性的简洁的验证[58]. PoPoW 是 NIPoPoW 的基础版本[59].

PoPoW 设计了一种基于跳表的结构体 Interlink, 添加在每个区块中. 该结构体结合每个区块在共识过程中求解 PoW 的结果, 指向前序中的一些其他区块, 形成如图 5 所示的多级索引. 验证者不需要完整的区块链, 而是仅基于所有未确认的 (如比特币中默认 6 个区块后确认) 区块以及各级索引中一定数目的区块, 即可在一定安全程度上完成区块一致性验证, 该方法可以将存储、验证开销从线性降低到对数级, 但是该方案只可用于难度不变的 PoW 区块链, 且需要验证者与全节点反复通信.

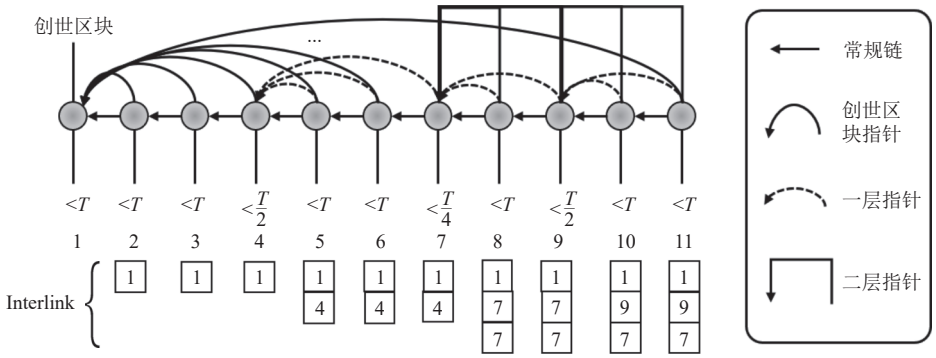


图 5 Interlink 结构示意图

NIPoPoW 是 PoPoW 的改进版, 其同样基于 Interlink 构造由各级索引中一定数目的区块组成的证明. 验证者对比多个证明者提交的证明, 找到多个证明的最低公共祖先 B, 并对公共祖先 B 之后的区块按照区块索引层级进

行打分. NIPoPoW 认为综合得分最高的证明有较高可能性来自于长度最长且真实有效的区块链. NIPoPoW 可以用于难度调整的区块链, 减少了轻节点与完整节点通信的次数, 实现了更加快速、方便的区块头有效性验证.

FlyClient[60]是另一种超轻客户端, 其为每个区块引入如图 6 所示的默克尔山脉 (Merkle mountain range, MMR) 的根, 并基于 MMR 扩展不影响前述叶子节点默克尔根的特效, 通过区块抽样将区块一致性验证的存储、验证开销从线性降低到对数级. 与 PoPoW 及 NIPoPoW 相同, 这种校验是基于一定安全程度进行的猜测, 而非确定性的校验.

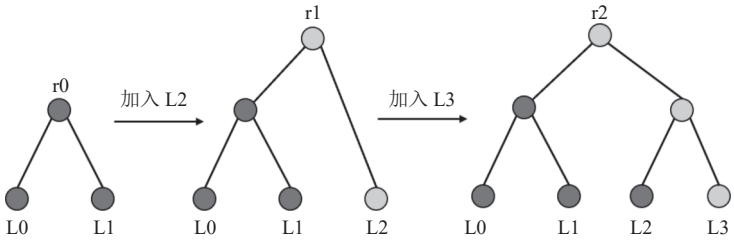


图 6 默克尔山脉示意图[60]

超轻客户端研究应用于跨链领域时, 需要源区块链已经包含有指定的数据结构, 或通过分叉的方式进行更新, 这制约了他们在跨链系统中的实施性. 但是, 随着越来越多的区块链项目对超轻客户端的支持, 如 zcash[61], 它在跨链研究中的使用前景也越发明朗.

(2) 状态数据验证

状态数据验证 (可信性验证) 是指验证者验证状态数据是否在源区块链中达成了一致. 由于区块链系统中节点就区块数据进行共识, 因此如果区块结构中包含状态数据, 验证者可以通过验证状态是否包含在源链主链中实现, 而如果区块数据中不包含状态数据, 验证者则需要通过其他方法实现.

1) 区块包含状态数据

当源区块链的区块数据包含状态数据 (如, 状态树) 时, 状态数据的验证可以拆分成:

a) 状态打包验证, 即验证状态 S 是否被打包在区块 B_n 中.

b) 区块一致性验证, 即验证区块 B_n 是否在源区块链主链上.

其中, 区块一致性验证在账本数据验证部分已有详细总结, 此处不再赘述. 下面将综述总结状态打包验证相关研究.

Westerkamp 等人提出的智能合约移植[62]是指将源区块链上的智能合约复制到目的区块链上, 智能合约移植完成后, 源区块链和目的区块链均能通过交易更新合约状态. 在移植过程中, 目的链需要验证目的区块链上重构的智能合约状态是否与源区块链智能合约状态一致. 该方案通过链下重放交易构造待移植合约的状态, 并通过直接比较源、目的区块链中该合约的存储根是否一致, 实现对智能合约状态的验证. 但是该方案仅当源区块链与目的区块链状态树构建的方式一样时才可使用.

与交易数据一样, 部分区块链状态数据也以状态树的形式存储在区块头中[49], 因此验证者同样可以基于轻客户端思想, 通过状态树树根以及状态 S 对应的验证路径实现状态打包验证.

Fynn 等人提出的智能合约转移[63]是指将源区块链上的智能合约移动到目的区块链上, 智能合约转移完成后, 源区块链只能读取而不能更新被锁定的合约状态, 目的区块链则可以通过交易更新合约状态. 在移动过程中, 目的区块链同样需要验证智能合约状态. 该方案基于轻客户端思想, 通过源区块链区块头中存储的状态树根以及合约状态所在路径实现状态打包验证.

由于状态数据是全局数据而非特定区块的数据, 数据量大且类型负责复杂, 与交易验证相比, 基于轻客户端思想进行状态验证复杂度更高、开销更大、耗时更长.

SmartSync 在智能合约移植的基础上, 通过定期更新智能合约状态, 实现目的区块链对源区块链指定智能合约状态数据的同步[64]. 为了减少状态同步、验证的开销, 该方案在同步过程中并不同步、验证合约的所有状态, 而是仅基于轻客户端思想同步并验证更新的状态, 同时通过将证明路径中的新状态替换成目的区块链存储的当前状态, 并将生成的状态根与目的区块链当前的合约状态根对比, 确保状态更新同步的完整性.

2) 区块不包含状态数据

当源区块链的区块数据不包含状态数据时, 验证状态数据性最直接的方法是验证者执行区块链主链交易, 在本地生成并维护源区块链的可信状态, 但是这种方法会有持续的计算开销与不断增长的存储开销.

部分跨链解决方案将状态数据验证转换成账本数据验证: 通过将目标状态写入区块链事件 event, 实现本地状态信息写入区块链账本, 目的区块链通过交易信息验证方法实现对交易与事件的验证, 并从事件中可以获取目标状态信息. 但是将区块链状态验证转换成交易验证的方法需要在区块链上通过额外的交易将状态信息写入账本中, 增长了跨链流程, 增大了跨链时延与开销.

此外, 无状态客户端[65]研究对此场景下的状态数据验证具有借鉴意义. 目前该方向的研究可以分为基于哈希树累加器的无状态客户端[66−68]、基于 RSA 累加器的无状态客户端[68,69]以及基于双线性映射累加器的无状态客户端[70,71]. 其基本原理是不存储区块链的完整状态, 而是基于上述累加器聚合并存储区块链状态, 在此基础上, 借助对应的状态承诺可以证明区块链状态有效性. 然而现有无状态客户端研究聚焦于降低存储开销, 其计算复杂度较高, 无法直接应用于跨链场景.

3.4 跨链数据验证方法评价

(1) 评价指标

根据本节综述的区块链账本数据跨链验证方法与区块链状态数据跨链验证方法, 我们整理出表 3、表 4, 展示验证方法的分类、方法所解决的问题, 并从开销、安全性、实现难度、适用场景、验证主体以及各自独特的优劣进行评价.

表 3 区块链账本数据跨链验证方法评价

文献	验证主体	安全性	适用区块链	实现难度	开销		其他
					存储开销	计算开销	
[3,15]	链上	哈希	支持轻客户端	中	线性	线性	—
[32,58]	链上/链下	概率	PoW共识	高	对数级	对数级	单次通信
[59]	链上/链下	概率	PoW共识 (难度固定)	高	对数级	对数级	多次通信
[60]	链上/链下	概率	概率性共识	高	对数级	对数级	单次通信
[5]	链上	哈希	支持轻客户端	中	线性	非线性	—

表 4 区块链状态数据跨链验证方法评价

文献	验证主体	安全性	适用区块链	实现难度	开销		其他
					存储开销	计算开销	
[62]	链下	哈希	区块有状态根	低	常数级	低	状态根构造相同链间
[63]	链上	哈希	区块有状态根	中	线性	较高	—
[64]	链上	哈希	区块有状态根	中	线性	中	—
[66−68]	链下	哈希	UTXO	高	对数级	较高	—
[68,69]	链下	RSA	账户/UTXO	高	常数级	高	10亿账户; 非成员验证
[70]	链下	ECC	账户	高	常数级	高	可聚合承诺

1) 验证主体: 即为前文所述的验证主体, 包括链上链下两种.

2) 安全性: 即评价该跨链数据验证方案的安全性是如何保障的, 例如有的验证方案因为其算法, 只能概率性地保证安全性; 有的方案的安全性通过密码学算法予以保障.

3) 适用区块链: 本文所综述的方案不局限于各跨链方案, 还包括了一些非跨链场景技术的借鉴, 其中很多方案对于区块链本身因为作出了较强的假设, 因此, 此处主要评价这些方案所适用的区块链场景.

4) 实现难度: 方案在跨链场景下的实现难度, 用以评价方案的可实施性.

5) 开销: 包括存储开销与计算开销. 对于解决不同问题的方案, 其开销的评价角度、比较对象不一而同, 难以跨问题评价, 我们将在后面的统一评价中进行展开.

6) 其他: 各方案独特的优缺点.

(2) 方案对比

1) 区块链账本数据验证

● 安全性方面: BTC Relay[3], ETH Relay[5], Cosmos[15]等大部分跨链方案通过维护源区块链轻客户端, 实现对源区块链账本数据的验证, 其安全性来源于哈希计算的防碰撞性; PoPoW[59]、NIPoPoW[58]、FlyClient[60]基于“抽取”的思想, 不再同步所有区块头, 而仅通过同步对数级数目的区块头确定主链, 但是其对于主链的确定是概率性的, 抽取的区块头越多, 确定的概率越大, 安全性越高.

● 适用区块链方面: BTC Relay, ETH Relay, Cosmos 等通过构建源区块链轻客户端的方案需要源区块链支持轻客户端; PoPoW、NIPoPoW、FlyClient 是针对概率性共识区块链验证方案的优化, 其中 PoPoW、NIPoPoW 方案仅适用于 PoW 共识区块链.

● 实现难度方面: BTC Relay, ETH Relay, Cosmos 等方案的具体难度取决于源区块链轻客户端构建难度; PoPoW、NIPoPoW、FlyClient 这 3 种方案均修改了区块链结构, 尽管已有一些区块链进行了升级适配[61], 但是无法直接用于大多数区块链的跨链验证, 需要源区块链分叉进行支持, 实现难度较大.

● 开销方面: BTC Relay 等大部分跨链方案通过逐个验证并存储区块头的方式实现源区块链轻客户端的构建与维护, 其存储开销与计算开销均线性增长, 具体开销与实现复杂度取决于源区块链; ETH Relay 采用乐观接收区块头的方式, 可以降低链上计算开销, 但是存储开销依然线性增长; PoPoW、NIPoPoW、FlyClient 基于“抽取”的思想, 将存储开销与计算开销降低到对数级别, 对跨链验证不频繁场景下的跨链验证优化研究具有一定参考价值.

2) 区块链状态数据验证

● 安全性方面: 基于 RSA 累加器的无状态客户端[68,69]与 Pointproofs[70]的安全性分别来源于 RSA 与 ECC 椭圆曲线; 而智能合约移植[62]、智能合约转移[63]、智能合约同步[64]以及基于哈希树累加器的无状态客户端[66−68]均通过哈希计算聚合、验证状态, 相对 RSA 与 ECC 较弱.

● 实现难度方面: 智能合约移植、智能合约转移、智能合约同步实现难度较小, 而 3 类基于累加器的无状态客户端方案因为其数学计算的复杂程度, 缺乏在链上的可实现性, 应用在跨链场景时更适用于链下代理作为验证主体的方案.

● 开销方面: 智能合约移植不进行状态打包验证, 通过比较源、目的区块链合约存储根验证状态是否一致, 开销小但是需要引入合适的奖惩机制保证方案的可用性, 且仅适用于源、目的区块链状态树设计一致的场景; 智能合约转移以及智能合约同步基于链上维护的源区块链轻客户端, 在链上根据状态树实现状态打包验证, 具体开销与复杂度取决于源区块链状态树设计与状态树大小, 但由于状态树包含源区块链全局状态数据, 因此一般开销较大, 耗时较长, 其中智能合约同步通过仅验证更新状态降低计算开销. 3 类基于累加器的无状态客户端无需存储区块链的状态, 而是通过获取的证据直接验证状态, 因此, 表格中此处的存储开销, 主要指每次通信时, 客户端需要获取的证据大小. 基于哈希树累加器的无状态客户端每次通信的证据大小, 相比所有区块数目是对数级的, 其计算开销主要来源于哈希计算, 是 3 类无状态客户端中最小的; 基于 RSA 累加器的无状态客户端可以将每次提交的证据大小优化为常数大小, 显著优于哈希累加器方案的表现, 并且拓宽了方案的适用范围, 其计算开销主要来源于 RSA 的验证计算, 计算复杂度高; Pointproofs 使用 ECC 椭圆曲线来充当双线性映射, 相比 RSA 累加器方案, 也提供了常数级的证据大小, 并且在相同证据大小时, 能够提供更高的安全性, 当然, 这也带来了更高的验证计算复杂度.

4 跨链操作原子性保障关键技术

跨链操作原子性是指跨链操作拆分成的在多个区块链上子操作执行的原子性, 即多个区块链上的子操作需要一致执行或者一致不执行.

原子性问题在数据库领域已经历了数十年的研究, 这一问题的基本解决思路是两阶段提交.

(1) 准备阶段

各节点执行本地操作后进入准备状态, 等待其他节点进入准备状态.

(2) 提交阶段

1) 提交操作: 如果所有节点都可以提交, 则所有节点提交本地操作.

2) 回退操作: 如果存在节点不能提交, 则所有节点撤销本地操作.

跨链操作原子性问题的基本解决思路同样可以归纳为两阶段提交.

(1) 准备阶段

各区块链通过执行跨链子操作证明其具备执行跨链子操作的能力, 并且将子操作相关的区块链状态锁定进入准备状态, 等待其他区块链的准备状态.

(2) 提交阶段

1) 提交操作: 如果所有跨链操作相关区块链都进入准备状态, 即相关区块链都具备执行跨链子操作的能力, 则所有相关区块链均提交子操作状态.

2) 回滚操作: 如果存在跨链操作相关区块链未进入准备状态, 即存在相关区块链不具备执行跨链子操作的能力, 则所有相关区块链均回滚子操作状态.

两阶段提交的关键在于根据准备阶段的状态协调各区块链完成一致的提交或者回滚操作. 依据协调对象的不同, 可以将跨链原子性保障关键技术分为基于用户自协调的原子性保障机制与第三方协调的原子性保障机制. 下面将分别综述各类方案的研究现状, 并提出评价指标分类评估现有方案.

4.1 基于用户自协调的原子性保障机制

基于用户自协调的原子性保障机制是指由参与跨链操作的用户作为协调者依据跨链操作相关区块链的准备状态, 对提交阶段的操作进行决策的方案.

哈希时间锁协议 (hashed timelock contract, HTLC) 是基于用户自协调保障原子性的经典方案[8], 其使用哈希锁[72]与时间锁[73]保证多个操作的原子执行, 一般用于解决跨链资产原子交换问题[74].

具体地, HTLC 需要选举出一个用户担任发起者与协调者, 并由该用户在进行跨链操作前设置哈希锁 $hash(x)$, 哈希锁原像 x 仅有该用户持有. 准备阶段, 用户使用相同的哈希锁 $hash(x)$ 与差异化的时间锁 T_1 与 T_2 锁定资产, 交易对手需要使用哈希锁原像 x 才能进行提交, 用户需要等待时间锁计时完毕才能进行回滚. 提交阶段, 若时间锁到期前协调者确认所有跨链操作相关区块链都进入准备状态, 则释放哈希锁原像 x 完成提交操作, 其他参与用户获取哈希锁原像 x, 随后完成提交操作; 若时间锁到期, 用户执行回滚操作取回锁定资产.

HTLC 基于相同的哈希锁保证各个区块链上的子操作可以一致提交, 基于差异化的时间锁, 保证跨链操作可终止, 且参与者有足够的时间提交操作. 但是 HTLC 的提交操作与回滚操作并不是完全互斥: 攻击者在执行完提交操作后, 可以通过审查攻击[75−77]、洪泛攻击[78]等方法阻止其他用户执行提交操作, 并在超时后执行回滚操作, 这种情况下 HTLC 无法保证跨链操作的原子性. 目前在实际应用中一般通过将时间锁与其差值设置得足够长减少审查攻击、洪泛攻击带来的影响. 但是 HTLC 需要参与者串行地在各个区块链上构造正确的哈希时间锁, 足够长的时间锁设定会导致跨链操作完成的最大时延过长, 而当跨链操作涉及多个参与者时, 这种现象将更突出[79].

HTLC 的实现需要参与区块链均支持脚本或者智能合约, 但是并非所有区块链都支持智能合约 (或者脚本). Zie 等人对哈希时间锁协议进行了扩展, 通过智能合约技术 (或者脚本) 与多签技术实现了支持智能合约的区块链与不支持智能合约的区块链间的原子交换[80].

该方案由在不支持智能合约 (或者脚本) 的区块链上锁定资产的用户担任发起者与协调者. 下面基于跨链原子写操作例子进行描述, 其中区块链 $Chain_1$ 不支持智能合约, Alice 为协调者.

在准备阶段, Alice 将资产锁定在多签账户中, 需要 Alice 与 Bob 的签名才能进行解锁; Bob 提前为区块链 $Chain_1$ 上的提交操作 tx_1 (将 $Chain_1$ 锁定资产转给 Bob) 与回滚操作 tx_2 (将 $Chain_1$ 锁定资产退给 Alice) 签名, 并将资产锁定在智能合约中. 在提交阶段, 若 Alice 确认 Bob 进入准备状态, 则为区块链 $Chain_1$ 上的提交操作签名, 并使用该签名触发区块链 $Chain_2$ 上的预提交操作 (将锁定资产标记为解锁), Bob 学习到该签名后使用该签名触发 $Chain_1$ 上的提交操作, 一定时间后 Alice 触发 $Chain_2$ 上的提交操作 (将 $Chain_2$ 锁定资产转给 Alice); 否则, Alice 为区块链 $Chain_1$ 上的回滚操作签名, 并使用该签名触发区块链 $Chain_2$ 上的回滚操作 (将锁定资产退给 Bob).

该方案拓展了基于用户自协调的原子性保障机制的适用范围, 但是由于协调者在提交阶段可以通过审查攻击在不同区块链上触发相反的操作, 从而破坏跨链操作的原子性, 即跨链操作的原子性依赖于同为参与者的协调者遵守协议.

基于用户自协调的方案依赖于参与者之间的同步假设, 需要各个参与者实时在线并关注区块链的更新情况, 对跨链操作参与者提出了较高的要求.

4.2 基于第三方协调的原子性保障机制

基于第三方协调的原子性保障机制是指引入外部第三方作为协调者依据跨链操作相关区块链的准备状态, 对提交阶段的操作进行决策的方案. 根据第三方的性质, 基于第三方协调的原子性保障机制又可以分为公证人作为协调者以及区块链作为协调者两类.

(1) 基于公证人协调的原子性保障机制

以跨链资产原子交换为典型的原子性问题出现之初, 主要通过公证人解决原子性问题[7,80,81].

AC3TW[80]的中心化可信公证人模式借助跨链操作参与者共同选择的公证人 Trent 保证跨链操作的原子性. 准备阶段, 参与者将资产锁定智能合约中, 该智能合约根据公证人 Trent 的指令 RD 与 RF 分别执行提交操作与回滚操作. 在任意时刻, 跨链操作参与者都可以向 Trent 发起提交请求或者回滚请求从而进入提交阶段. 提交阶段, Trent 指令一旦发布就不会再更改, 保证了提交操作与回滚操作互斥, 从而保证跨链操作的原子性.

该方案完全依赖于公证人 Trent, 一旦 Trent 不遵守协议, 则无法保证跨链操作的原子性. 然而, 由于各个区块链在组成节点、安全等级等方面存在较大差异, 并不是所有跨链操作都能找到一个可信的公证人.

Lipton 等人提出的方案[81]借助交互区块链各自选择的公证人保证跨链操作的原子性, 为了保证多公证人间无争议的协调, 其基于经典的两阶段提交协议[82,83]或者其他变体 (例如三阶段提交协议) 实现, 具体过程如图 7 所示. 交互区块链的网关节点首先建立连接并交换跨链信息, 接着源区块链网关节点 G1 向目的区块链网关节点 G2 提供源链待转移资产已锁定的相应证明, 同时 G2 返回该证据的签名收据, 随后 G1 与 G2 通过两阶段提交保证交互区块链上同时提交.

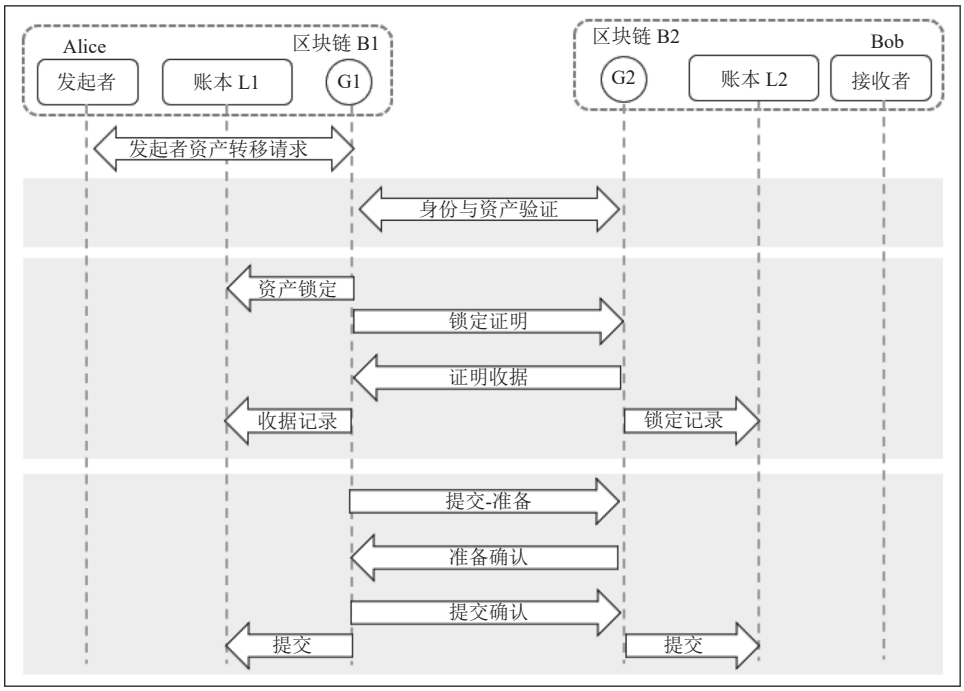


图 7 Lipton 等人方案流程图[81]

该方案拓展了基于公证人协调的原子性保障机制的适用范围, 但是这种“多公证人”的模式并没有降低用户对公证人的信赖, 反而需要更强的安全假设: 由于交互区块链预锁操作顺序执行且不提供回滚操作, 如果存在后手公证人不按照协议执行或者遭受攻击, 则跨链操作无法终止.

上述两类方案存在单点故障的隐患, 为了降低原子性保障方案对单一可信公证人的依赖, Interledger 原子模式[7]将单一公证人扩展成公证人集合, 同时通过拜占庭容错算法实现公证人集合一致的协调, 从而降低对单一公证人的信任程度.

基于公证人协调的原子性保障机制与基于用户自协调的原子性保障机制相比, 不需要用户保持同步, 降低用户复杂度的同时可以实现准备阶段与提交阶段中各用户并行操作, 在多参与者的场景中可以有效降低整体时延, 但是引入了额外的可信公证人, 增强了安全假设.

(2) 基于区块链协调的原子性保障机制

虽然 Interledger 将单公证人协调拓展成了多公证人协调, 提高了协调者的去中心化程度, 提升了原子性保障机制的容错能力, 但是公证人集合大小不能无限扩展, 去中心化程度有限. 为了进一步提升原子性保障机制的容错能力, 不少研究将公证人扩展成公证链.

AC3TW[79]的非许可公证人网络模式借助一条非许可链 (公证链) 进行协调保证跨链操作的原子性, 其中, 公证链需要部署与公证人 Trent 有相同协调功能的智能合约. 该方案原子性保障的原理及流程与前文所述的中心化可信公证人模式一致, 只是将公证人 Trent 替换成公证链, 相应的, 参与者与 Trent 间的通信也替换成了交互区块链与公正链间通信.

除了使用第三方区块链外, 参与跨链交互的区块链也可以作为协调者保障跨链操作的原子性. Zhao 等人的方案[84]基于链间传输同步假设, 借助任意一个交互区块链作为 leader, 通过分布式系统两阶段提交, 在崩溃容忍场景下保证跨链操作的原子性, 同时通过被动心跳方法检测节点故障, 从而避免阻塞.

与基于公证人协调的原子性保障机制相比, 基于区块链协调的原子性保障机制保持用户友好、支持并行等优势的同时, 进一步提升了协调者的去中心化程度, 同时这类机制基于智能合约实现协调协议, 协议内容公开透明, 支持用户进行验证与确认, 将对公证人身份的信任转换成对区块链技术的信任. 相对应的, 协调功能需要在区块链上实现部署, 复杂度提升, 而由于引入链间通信, 跨链操作时延增大.

4.3 原子性保障方案评价

(1) 评价指标

综合前文介绍的各原子性保障方案, 我们整理得到表 5, 从基础要求与扩展要求两个层面对其进行分析与评价.

表 5 原子性保障方案评价

文献	安全性	可终止性	去中心化程度	用户实时在线要求	智能合约支持要求	实现难度	多参与方原子性保障扩展	性能		
								回滚模式	执行模式	协调时间开销
[79]	满足	满足	公证人	不需要	双方	低	适用	主动	并行	低
[81]	满足	不满足	公证人	不需要	双方	低	适用	无	并行	低
[79]	满足	满足	公证区块链	不需要	双方	中	适用	主动	并行	高
[84]	满足	满足	公证区块链	不需要	双方	中	适用	无	并行	高
[8]	不满足	满足	单用户	需要	双方	低	改进后适用	时间锁	串行	低
[80]	不满足	满足	单用户	需要	单方	低	不确定	主动+时间锁	串行	低

原子性保障方案的基础要求, 包括安全性与可终止性, 一般而言, 只有当方案同时满足了安全性与可终止性, 才实现了跨链操作的原子性的保障.

1) 安全性: 即使在客户端崩溃故障、网络分区、消息延迟、参与者偏离协议的环境下, 协议执行完成后, 所有参与方的终止状态仍满足“all or nothing”的需求, 即要么所有参与方均回滚至初始状态, 要么都迁移至执行子操作后的状态.

2) 可终止性: 所有跨链操作相关区块链都会进入提交阶段而不会永久地停留在准备阶段.

原子性保障方案在实现与应用方面的扩展性要求, 具体包括如下几个方面.

1) 性能: 在实际系统中, 因为跨链操作原子性保障方案不是独立实现的, 需要综合诸多其他关键技术, 例如跨链信息传输技术, 因此, 评价原子性保障机制本身的性能只能从有限的几个角度入手, 此处我们从回滚模式、执行模式和协调时间开销 3 个角度评估其性能.

a) 回滚模式: 回滚模式分为主动回滚和超时回滚, 主动回滚是指在准备阶段的任意时刻参与者可以发起回滚操作, 跨链操作执行进入提交阶段; 超时回滚是指, 参与者需要等待准备阶段设置的定时器超时后才能发起回滚操作.

b) 执行模式: 即各个阶段各区块链子操作间的执行模式, 包括串行执行与并行执行.

c) 协调时间开销: 即原子性保障方案中, 协调各部分工作所需要的时间开销, 包括但不限于协调者拆分子操作的时间开销、协调者对于参与方提交的签名等信息的确认时间.

2) 去中心化程度: 即协调者的去中心化程度, 用于描述方案抵御协调者单点故障与联合作恶的能力.

3) 智能合约支持: 即该方案的实现是否需要参与方双方所在区块链都支持智能合约, 用于评估该方案的适用范围.

4) 实现难度: 该指标意为该方案在跨链场景下的实现难度, 用以评价方案的可实施性.

5) 用户实时在线: 即用户是否需要实时在线并在区块链间同步信息、处理信息, 用于评估该方案对用户的要求高低.

6) 多参与方原子性保障: 即是否能保证多参与方跨链操作的原子性多, 分为不适用/适用/推测适用/不确定 3 种, 不确定即为文章没有明确表示是否适用, 但是本论文中也将给出自己的理解.

(2) 方案对比

● 安全性方面: HTLC[8]及其衍生出的 Zie 方案[80], 由于提交操作与回滚操作并不是完全互斥, 在洪泛攻击、审查攻击等场景下, 无法保证安全性, 进而无法保证方案原子性, 其他方案都可以有效保证安全性.

● 可终止性方面: Lipton 等人[81]提出的方案由于不提供回滚机制, 在参与者偏离协议时无法保证方案的可终止性.

下面是原子性保障机制扩展性要求的分析与评价.

● 去中心化程度方面: 基于公证人或用户协调的方案与基于区块链协调的方案相比, 去中心化程度更低, 更容易受协调者故障的影响, 协调者作恶破坏原子性的成本更低.

● 用户实时在线方面: HTLC 及 Zie 等人[80]提出的方案因为其自协调特性, 需要用户保证实时在线进行信息同步与协调处理, 对用户要求更高.

● 智能合约支持方面: 除 Zie 方案外, 其他方案都需要参与双方所在区块链都支持智能合约 (或脚本), 一定程度上限制了方案的实现.

● 实现难度方面, 基于单用户与公证人的方案依赖用户协调, 实现难度低; 基于区块链协调的方案需要在链上实现协调方法, 难度较前者更高. 多参与方原子性保障方面: 除 Zie 等人提出的方案外的其他各机制, 都可以有效扩展到多方资产交换的场景中, 其中 HTLC 作为最早的原子性保障方案, 最初仅实现了双方资产交换, 但是已有工作将其扩展至多方资产交换的场景. 而当一笔复杂的多方交换事务可以拆分成多笔智能合约链和非智能合约链的子操作时, 我们认为 Zie 等人提出的方案无法满足所有多方资产交换的需求, 只有在一个参与者在不支持智能合约的区块链上时才能实现多方资产交换.

● 性能方面: 1) 主动回滚相比超时回滚, 在参与者偏离协议等异常情况下可以更早地完成跨链操作执行, 性能表现更好. 2) 并行执行方案相比串行执行方案, 其原子性事务的执行耗时更短, 在越复杂、流程越长的跨链操作中性能表现更好. 3) 基于公证人协调或用户协调的方案与基于区块链协调的方案相比, 由于在链下完成协调功能, 其协调时间开销更低, 性能表现更好.

5 跨链整体解决方案

早期跨链解决方案主要集中于两个区块链交互的场景, 随着对跨链需求的深入认识, 研究焦点逐渐拓展到多区块链交互的场景. 针对这些场景, 已有多种跨链解决方案被提出, 根据是否需要借助其他区块链可以分为两大类: 一

类是基于直连的跨链解决方案, 其基本思想是有跨链需求的区块链直接实现区块链互操作; 一类是基于中继链的跨链方案, 其基本思想是有跨链需求的区块链借助其他区块链作为中继, 通过多跳跨链操作实现指定区块链互操作.

5.1 基于直连的跨链解决方案

直连跨链解决方案的基本思想是有跨链需求的区块链直接实现区块链互操作, 即在有跨链需求的区块链间解决上述 3 个跨链关键问题.

(1) BTC Relay

BTC Relay 被认为是最早的跨链方案[3], 旨在完善以太坊基础设施、帮助以太坊实现更大的创新, 其实现了比特币区块链与以太坊区块链的单向跨链操作——支持在以太坊上验证比特币交易. BTC Relay 在以太坊上部署可以接收并处理比特币区块头与交易的智能合约, 并通过 Relayer 将比特币区块头跨链传输至以太坊实现基于最长链规则的比特币轻客户端, 为比特币交易的链上验证提供基础. 用户将比特币交易传输至以太坊, 并附加比特币交易打包证明 (Merkle path), 实现以太坊对比特币交易的验证. BTC Relay 不解决跨链操作原子性保障问题.

(2) RSK

RSK 是锚定比特币区块链的一个开源智能合约平台, 其目标是将智能合约以可操作的形式带入比特币系统, 实现即时支付以及高扩展性, 从而为比特币生态系统增加价值和功能. 与 BTC Relay 实现的单向跨链操作不同的是, RSK 实现了与以太坊的双向跨链操作, 支持在 RSK 上根据比特币资产锁定交易生成资产以及在比特币上根据 RSK 资产锁定交易解锁资产[11]. 在此过程中需要在 RSK 上验证比特币交易、在比特币上验证 RSK 交易: 在比特币至 RSK 方向, 公证人联盟或者用户将比特币交易传输至 RSK 中, 并基于 RSK 链上维护的比特币轻客户端进行验证; 在 RSK 至比特币方向, RSK 交易并未传输至比特币链上, 而是由公证人联盟代替比特币接收、验证该信息, 并在比特币上签署对应解锁交易. RSK 的公证人联盟由 15 家公司组成, 在比特币参与 RSK 联合挖矿矿工比例足够时公证人联盟将切换为联合矿工. RSK 并没有考虑跨链操作原子性保障问题, 然而这对 RSK 是必要的, 目前 RSK 有可能面临转账不可终止的情况.

(3) HTLC

HTLC 是常用的跨链操作原子性保障技术[8], 但是作为面向跨链资产原子交换问题的完整解决方案, 其同样完成了跨链信息在链间的传输以及目的区块链对跨链信息的可信性验证. HTLC 实现了在目的区块链上验证特定的源区块链交易, 不过与其他方案不同, 特定的源区块链交易并没有直接被传输到目的区块链上, 仅有其包含的哈希锁密钥被用户传输到目的区块链上. 目的区块链、根据预先设置的哈希锁可以验证哈希锁密钥的正确性, 从而间接验证哈希锁密钥对应的特定源区块链交易有效. HTLC 基于参与者间的同步假设与各区块链上相同的哈希锁保证安全性, 基于差异化的时间锁, 保证跨链操作可终止, 然而由于 HTLC 的提交操作与回滚操作并不是完全互斥, 其在审查攻击、洪泛攻击等场景下无法保证跨链操作的原子性, 目前在实际应用中一般通过将时间锁设置得足够长减少审查攻击、洪泛攻击带来的影响.

(4) WeCross

WeCross 面向区块链互联互通问题, 旨在构建一套未来区块链互联基础设施[13], 虽然其支持多链跨链, 但是由于跨链交互不需要借助其他区块链, 本质上依旧是基于直连跨链架构的方案. WeCross 的核心组件是跨链路由, 每条参与跨链交互的区块链都有一个由部署该区块链的机构搭建的跨链路由, 区块链强信任其跨链路由. WeCross 基于跨链路由解决上述 3 个跨链关键问题: 通过跨链路由间的信息传输实现跨链信息传输, 通过在跨链路由内构建其他区块链的轻客户端实现对跨链信息的可信性验证, 通过跨链路由担任协调者实现基于可信公证人的原子性保障. 此外, WeCross 还集成了哈希时间锁技术 (HTLC) 解决跨链操作原子性保障问题.

5.2 基于中继链的跨链方案

一类是基于中继链的跨链方案, 其基本思想是有跨链需求的区块链借助其他区块链作为中继, 通过两跳跨链操作实现区块链互操作. 单个中继链的处理能力有限, 当存在大规模跨链需求时, 中继链可以进一步扩展成中继链网络, 有跨链需求的区块链借助中继链网络, 通过多跳跨链操作实现区块链互操作.

(1) Cosmos

Cosmos 针对区块链存在的扩展性、可用性以及独立性问题, 提出了构建区块链互联网的设想, 为此设计了一种区块链网络架构, 并提出了该架构下的跨链交互方案[15].

Cosmos 的区块链网络架构如图 8 所示, 独立区块链被称作分区 (zone), 连接分区的特殊分区被称作枢纽 (Hub), 分区借助枢纽实现跨链交互, 在大量分区存在交互需求时, 这种架构可以减少区块链之间的适配复杂度.

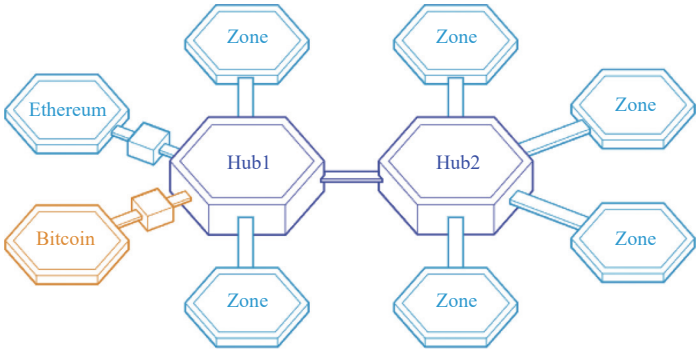


图 8 Cosmos 架构示意图

Cosmos 提出的 IBC 协议实现了相邻两个区块链间的信息传输与信任传递. 其中, 信息传输通过转发节点监听转发跨链信息实现, 信任传递通过在区块链上构建对端区块链轻客户端 (Client 模块) 实现. 尽管目前 IBC 协议仅适用于相邻两个区块链 (单跳), 但是未来将通过指定跨链信息传输的路径, 实现基于枢纽的多跳处理, 以支持区块链互联网的构建. Cosmos 的核心协议并不解决跨链操作的原子性保障问题, 而是将该问题交由具体应用解决.

在跨链交互方面, zone-zone 交互与 zone-Hub 交互并没有区别, Hub (中继链) 仅解决了适配复杂度的问题, 即当多个 zone 存在相互跨链需求时不需要两两适配, 仅需要每个 zone 与 Hub 适配.

(2) Polkadot

Polkadot 针对区块链在可扩展性、可伸缩性、安全性等方面普遍存在的问题, 设计了一种打破原有区块链共识架构中一致性与有效性紧密结合的网络架构, 旨在通过这种方法设计出一种完全可伸缩、可扩展的异构多链系统, 在保证最小化的安全性和传输功能的前提下将共识架构中的一致性和有效性分开[16].

Polkadot 的网络结构如图 9 所示, 由一条/多条中继链 (relay chain) 以及多条平行链 (parachain) 构成. 其中, 平行链负责具体业务的执行; 中继链负责与其直接连接的所有平行链的最终性共识. 为了在保证安全性的前提下实现中继链上复杂的操作, Polkadot 设计了 4 种角色共同维护网络: 收集者、渔夫、提名者以及验证者.

Polkadot 基于 XCMP 协议实现跨链信息传输, 并基于中继链对平行链区块最终有效性的验证与共识, 在共识层面完成对打包进目的平行链区块的跨链信息的可信性确认, 实现跨链信任传递. Polkadot 的核心协议并不解决原子性保障问题.

Polkadot 的中继链为跨链交互提供了信任传递功能, 平行链基于中继链对平行链区块最终有效性的验证与共识, 实现对对端链状态的确认.

(3) Interledger

Interledger 是 Ripple 面向交互区块链不存在可用中间人的跨链资产转移问题, 提出的不受区块链连接拓扑限制的安全支付协议[7]. 其核心思想如图 10 所示, 利用多跳同时在两条区块链上有账户的连接者连接跨链交易的发送者和接受者, 通过在源链、连接者所在区块链、以及目的链上的一组包括跨链交易发送者与连接者、连接者与连接者以及连接者与跨链交易接受者的交易, 实现借助一条或者多条区块链完成跨链支付. 虽然 Interledger 的目标是实现两个区块链间的交互, 但是实施过程中有多条区块链进行了跨链交互, 本质上属于基于中继链的跨链方案, 连接者所在区块链均为“中继链”.

聚焦到任意相邻区块链, 同时在两个区块链上拥有账户的连接者根据源区块链跨链请求, 在目的区块链上发

起对应交易, 完成单跳跨链操作. 在此过程中, 跨链信息并未传输到目的区块链并进行处理, 而是在链下由连接者代理接收并处理, 从而在应用层面达到跨链操作效果. 具体地, 连接者通过监听源区块链获取跨链请求, 通过接入源区块链实现对跨链请求的验证与确认. 由于连接者在目的区块链上的操作仅涉及其自身状态的更新, 因此连接者代理接收、处理跨链请求不需要目的区块链额外的授权与信任.

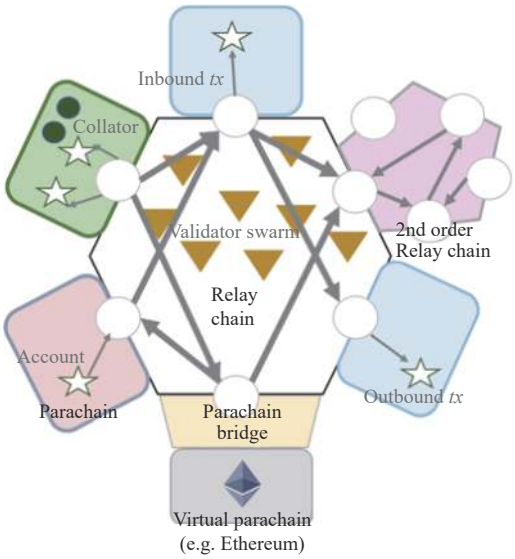


图 9 Polkado 络架构示意图[16]

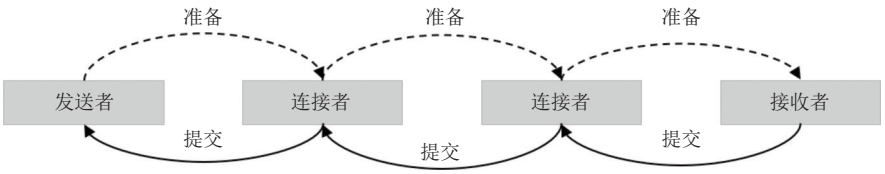


图 10 Interledger 多跳交易示意图

Interledger 包含通用模式和原子模式, 其中通用模式基于用户自协调保证跨链操作的原子性, 而原子模式基于一组公证人的协调保证跨链操作的原子性.

Interledger 的“中继链”与交互区块链在跨链交互为对等关系, 仅作为交互区块链打通跨链交互的通路.

(4) HyperService

HyperService 面向跨链合约调用问题, 提出了为异构区块链提供互操作性与可编程性的平台, 其具有 3 个核心组件 (如图 11)[85].

1) 全局状态区块链 (NSB): 独立于应用区块链的区块链, 保存其他区块链的轻量状态信息以及交易执行信息, 负责监管 Dapp 的正确执行.

2) 保险智能合约 (ISC 合约): 部署在 NSB 上的智能合约, 基于 NSB 存储的信息, 能够对 Dapp 执行进行正确的决策, 保证跨链合约的可结算性与金融原子性.

3) 可验证执行系统 (VES): 可验证执行系统 (VES) 处理用户提交的 HSL, 将其编译成可执行的 DAG, 并根据生成的 DAG 在 (NSB) 上部署保险智能合约 (ISC), 最后将生成的 DAG 与部署的 ISC 发送给用户验证与确认.

HyperService 由 VES 驱动跨链交易的执行, 并由 NSB 与 ISC 保证跨链交易的正确执行. 由于 VES 根据 DAG 在链下触发各个应用链上的操作, 属于链下触发模式 (具体定义见第 1.1 节), 区块链间没有直接的跨链交互, 因此不讨论应用链间对应 3 个关键问题的解决方案.

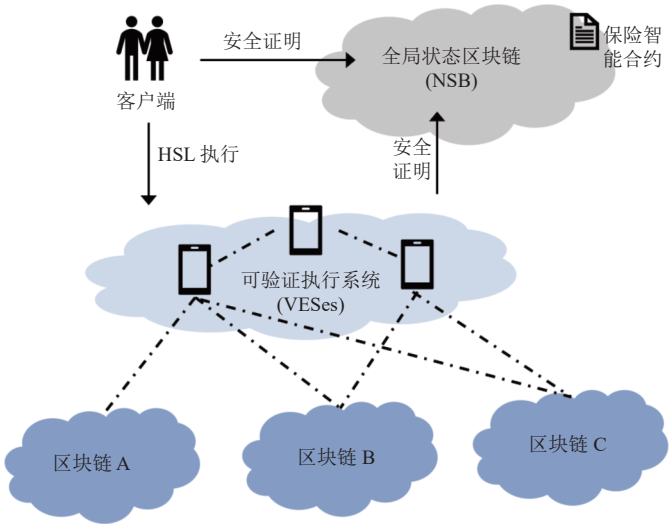


图 11 HyperService 架构图[85]

6 挑战与总结

6.1 跨链研究挑战

跨链技术已经取得了显著的进步, 但是它仍然是一个充满挑战的研究领域. 本文在此列出一些重要并且充满挑战性的问题.

(1) 跨链交互分层解耦: 跨链交互需要解决跨链信息传输、跨链信任传递与跨链操作原子性保障 3 个关键问题, 然而现有跨链方案信息传输与信任传递关键技术紧耦合, 跨链操作原子性保障与应用紧耦合, 限制了各类关键技术的独立发展, 制约了各类关键技术在功能、性能方面的优化, 增大了应用设计开发复杂度. 因此, 未来跨链研究需要解耦跨链交互, 针对各关键问题独立地深入研究解决方案, 实现功能完善与性能提升, 进一步可以通过通用层间接口的抽象与设计, 实现跨链交互分层协议栈.

(2) 可靠跨链传输机制: 跨链信息传输是跨链交互的基础, 现有跨链方案在传输层面仅能实现不可靠传输, 部分方案通过冗余的转发节点提升跨链传输的可靠性, 但是这种方法效果有限且会增大跨链开销. 可靠的跨链传输能有效支撑跨链其他关键技术研究, 有助于提升跨链服务质量, 并能满足未来更多跨链应用的需求. 因此, 未来需要研究容忍拜占庭节点、容忍崩溃等不同程度的可靠、低冗余的跨链传输机制.

(3) 轻量跨链信任传递技术: 现有跨链方案大多采用在链上构建对端链轻客户端的方法实现跨链信任传递, 然而这种方法链上开销与对端链成正比, 整体信任传递开销线形增长, 当跨链交易比例较小时, 每笔跨链交易平均信任传递开销较大. 未来可以根据跨链交易比例, 在现有区块链超轻量验证的基础上, 研究亚线形的、轻量的跨链信任传递技术.

(4) 高效原子性保障技术: 现有跨链方案中, 单跨链操作在多主体、长流程的场景下, 不同区块链上的处理串行执行, 导致单事务耗时长; 因区块链交易串行执行, 互不相关的多跨链操作在同一区块链内均串行执行, 导致区块链系统事务处理效率低, 未来可以从单跨链操作并行与多跨链操作并发等角度研究高效原子性保障技术.

(5) 跨链隐私与安全: 当前跨链研究聚焦于跨链方案的研究与实现, 对跨链技术打通区块链间壁垒后带来的跨链双花等安全性问题以及原有单链隐私保护、权限控制技术无法满足跨链需求的跨链隐私保护问题关注较少, 未来需要分析跨链技术带来的隐私保护问题与安全性问题, 并研究对应的方案.

6.2 总 结

跨链技术是打破当前区块链间数据孤岛现状的关键技术, 是当今区块链技术发展的迫切需求. 本文首先在区

分狭义与广义区块链互操作问题的基础上, 重新定义狭义区块链互操作, 并分析实现狭义区块链互操作需要解决的关键技术问题. 随后, 本文分别综述、讨论了各关键技术问题的研究现状, 包括跨链信息传输、跨链信任传递以及跨链操作原子性保证, 并进一步分析了当前具有代表性的跨链整体解决方案. 最后, 本文指出了几个值得进一步探索的研究方向. 总体而言, 跨链技术研究还在起步节点, 目前, 保证区块链可以相互操作尚有许多理论与技术问题亟待解决, 实现可以实际应用的、安全高效的区块链互操作则更是充满挑战.

References:

[1] Dabbagh M, Sookhak M, Safa NS. The evolution of blockchain: A bibliometric study. IEEE Access, 2019, 7: 19212–19221. [doi: 10.1109/ACCESS.2019.2895646]

[2] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system. 2008. https://bitcoin.org/bitcoin.pdf

[3] BTC Relay: Ethereum contract for Bitcoin SPV. 2017. https://github.com/ethereum/btcrelay

[4] Peace Relay. 2017. https://github.com/loiluu/peacerelay

[5] Frauenthaler P, Sigwart M, Spanring C, Sober M, Schulte S. ETH Relay: A cost-efficient relay for Ethereum-based blockchains. In: Proc. of the 2020 IEEE Int'l Conf. on Blockchain (Blockchain). Rhodes: IEEE, 2020. 204–213. [doi: 10.1109/Blockchain50366.2020.00032]

[6] Lerner SD. RSK white paper overview. 2015. https://bravenewcoin.com/assets/Whitepapers/RootstockWhitePaperv9-Overview.pdf

[7] Thomas S, Schwartz E. A protocol for interledger payments. 2015. https://interledger.org/interledger.pdf

[8] Herlihy M. Atomic cross-chain swaps. In: Proc. of the 2018 ACM Symp. on Principles of Distributed Computing. Egham: ACM, 2018. 245–254. [doi: 10.1145/3212734.3212736]

[9] HTLC implementation in the wallet. 2017. https://github.com/bitcoin/bips/blob/master/bip-0199.mediawiki

[10] Decred-compatible cross-chain atomic swapping. 2021. https://github.com/decred/atomicswap

[11] The Komodo Organization. BarterDEX—Atomic swap decentralized exchange of native coins. 2017. https://github.com/SuperNETorg/komodo/wiki/barterDEX-Whitepaper-v2

[12] Zyskind G, Kisagun C, Fromknecht C. Enigma catalyst: A machine-based investing platform and infrastructure for crypto-assets. 2018. https://assets.coingecko.com/paper_documents/documents/222/open-uri20180806-11-sc59sg.?1533563479

[13] WeCross. 2021. https://wecross.readthedocs.io/zh_CN/latest/

[14] BitXHub. 2023. https://bitxhub.cn

[15] Kwon J, Buchman E. Cosmos whitepaper. 2019. https://wikibitimg.fx994.com/attach/2020/12/16623142020/WBE16623142020_55300.pdf

[16] Wood G. Polkadot: Vision for a heterogeneous multi-chain framework. 2016. https://assets.polkadot.network/Polkadot-whitepaper.pdf

[17] Buterin V. Chain interoperability. 2016. https://r3.com/reports/chain-interoperability/

[18] Belchior R, Vasconcelos A, Guerreiro S, Correia M. A survey on blockchain interoperability: Past, present, and future trends. ACM Computing Surveys, 2021, 54(8): 168. [doi: 10.1145/3471140]

[19] Singh A, Click K, Parizi RM, Zhang Q, Dehghantanha A, Choo KKR. Sidechain technologies in blockchain networks: An examination and state-of-the-art review. Journal of Network and Computer Applications, 2020, 149: 102471. [doi: 10.1016/j.jnca.2019.102471]

[20] Johnson S, Robinson P, Brainard J. Sidechains and interoperability. arXiv:1903.04077, 2019.

[21] Zamyatin A, Al-Bassam M, Zindros D, Kokoris-Kogias E, Moreno-Sanchez P, Kiayias A, Knottenbelt WJ. SoK: Communication across distributed ledgers. In: Proc. of the 25th Int'l Conf. on Financial Cryptography and Data Security. Springer, 2021. 3–36. [doi: 10.1007/978-3-662-64331-0_1]

[22] Robinson P. Survey of crosschain communications protocols. Computer Networks, 2021, 200: 108488. [doi: 10.1016/j.comnet.2021.108488]

[23] Schulte S, Sigwart M, Frauenthaler P, Borkowski M. Towards blockchain interoperability. In: Proc. of the 2019 BPM Blockchain and CEE Forum on Business Process Management: Blockchain and Central and Eastern Europe Forum. Vienna: Springer, 2019. 3–10. [doi: 10.1007/978-3-030-30429-4_1]

[24] China Academy of Information and Communications Technology. White Paper for Blockchain Interoperability. Beijing: China Academy of Information and Communications Technology, 2020 (in Chinese). http://www.caict.ac.cn/kxyj/qwfb/bps/202012/P020201230759713827891.pdf

[25] Jin H, Dai XH, Xiao J. Towards a novel architecture for enabling interoperability amongst multiple blockchains. In: Proc. of the 38th IEEE Int'l Conf. on Distributed Computing Systems (ICDCS). Vienna: IEEE, 2018. 1203–1211. [doi: 10.1109/ICDCS.2018.00120]

[26] Lafourcade P, Lombard-Platet M. About blockchain interoperability. Information Processing Letters, 2020, 161: 105976. [doi: 10.1016/j.

ipl.2020.105976]
[27] WeBankBlockchain. WeCross: The blockchain interoperability platform white paper. 2020 (in Chinese). http://www.d-long.com/eWebEditor/uploadfile/20200416161636165570 31.pdf
[28] Chainmaker. 2022. https://docs.chainmaker.org.cn/index.html
[29] Goes C. The interblockchain communication protocol: An overview. arXiv:2006.15918, 2020.
[30] Zamyatin A, Harz D, Lind J, Panayiotou P, Gervais A, Knottenbelt W. XCLAIM: Trustless, interoperable, cryptocurrency-backed assets. In: Proc. of the 2019 IEEE Symp. on Security and Privacy (SP). San Francisco: IEEE, 2019. 193–210. [doi: 10.1109/SP.2019.00085]
[31] Dilley J, Poelstra A, Wilkins J, Piekarska M, Gorlick B, Friedenbach M. Strong federations: An interoperable blockchain solution to centralized third party risks. arXiv:1612.05491, 2016.
[32] Garoffolo A, Kaidalov D, Oliynykov R. Zendoo: A zk-SNARK verifiable cross-chain transfer protocol enabling decoupled and decentralized sidechains. In: Proc. of the 40th IEEE Int'l Conf. on Distributed Computing Systems (ICDCS). Singapore: IEEE, 2020. 1257–1262. [doi: 10.1109/ICDCS47774.2020.00161]
[33] Teutsch J, Straka M, Boneh D. Retrofitting a two-way peg between blockchains. arXiv:1908.03999, 2019.
[34] Kiayias A, Zindros D. Proof-of-work sidechains. In: Proc. of the 2019 Int'l Workshops on Financial Cryptography and Data Security. St. Kitts: Springer, 2019. 21–34. [doi: 10.1007/978-3-030-43725-1_3]
[35] Duffy M J. PlasmaChain, GameChain, SocialChain: The loom network universe expounded. 2018. https://medium.com/loom-network/plasmachain-gamechain-socialchain-the-loom-network-universe-expounded-5c672617a333
[36] Wang H, He D, Gao Y, Wang XY, Xu CC, Qiu WW, Yao YY, Wang Q. Research on data verification and exchange of heterogeneous blockchains for electricity application. Journal of Physics: Conf. Series, 2020, 1631(1): 012154. [doi: 10.1088/1742-6596/1631/1/012154]
[37] Web3 Foundation Research. Polkadot's Messaging Scheme. 2020. https://medium.com/web3foundation/polkadots-messaging-scheme-b1ec560908b7
[38] Xiao XT, Yu Z, Xie K, Guo SY, Xiong A, Yan Y. A multi-blockchain architecture supporting cross-blockchain communication. In: Proc. of the 6th Int'l Conf. on Artificial Intelligence and Security. Hohhot: Springer, 2020. 592–603. [doi: 10.1007/978-981-15-8086-4_56]
[39] Luo K, Yu W, Muhammad AH, Wang SY, Ling CG, Hu K. A multiple blockchains architecture on inter-blockchain communication. In: Proc. of the 2018 IEEE Int'l Conf. on Software Quality, Reliability and Security Companion (QRS-C). Lisbon: IEEE, 2018. 139–145. [doi: 10.1109/QRS-C.2018.00037]
[40] Belchior R, Vasconcelos A, Correia M, Hardjono T. Enabling cross-jurisdiction digital asset transfer. In: Proc. of the 2021 IEEE Int'l Conf. on Services Computing (SCC). Chicago: IEEE, 2021. 431–436. [doi: 10.1109/SCC53864.2021.00062]
[41] Belchior R, Vasconcelos A, Correia M, Hardjono T. Hermes: Fault-tolerant middleware for blockchain interoperability. Future Generation Computer Systems, 2022, 129: 236–251. [doi: 10.1016/j.future.2021.11.004]
[42] Hyperchain. BitXHub: The cross-chain technology platform white paper. 2022 (in Chinese). https://upload.hyperchain.cn/BitXHub 白皮书.pdf
[43] Sztorc P. Drivechain—The simple two way peg. 2015. https://www.truthcoin.info/blog/drivechain/#drivechain-a-simple-spv-proof
[44] Sergio DL. Drivechains, sidechains and hybrid 2-way peg designs. 2016. https://docs.rsk.co/Drivechains_Sidechains_and_Hybrid_2-way_peg_Designs_R9.pdf
[45] Intel Corporation. Intel® software guard extensions programming reference. 2014. https://www.intel.com/content/dam/develop/external/us/en/documents/329298-002-629101.pdf
[46] Robinson P, Ramesh R. General purpose atomic crosschain transactions. In: Proc. of the 3rd Conf. on Blockchain Research & Applications for Innovative Networks and Services (BRAINS). Paris: IEEE, 2021. 61–68. [doi: 10.1109/BRAINS52497.2021.9569837]
[47] Goldreich O, Oren Y. Definitions and properties of zero-knowledge proof systems. Journal of Cryptology, 1994, 7(1): 1–32. [doi: 10.1007/BF00195207]
[48] Westerkamp M, Eberhardt J. zkRelay: Facilitating sidechains using zkSNARK-based chain-relays. In: Proc. of the 2020 IEEE European Symp. on Security and Privacy Workshops (EuroS&PW). Genoa: IEEE, 2020. 378–386. [doi: 10.1109/EuroSPW51379.2020.00058]
[49] Buterin V. Ethereum whitepaper: A next-generation smart contract and decentralized application platform. 2014. https://ethereum.org/en/whitepaper/
[50] Merkle RC. A digital signature based on a conventional encryption function. In: Advances in Cryptology. Berlin: Springer, 1988. 369–378. [doi: 10.1007/3-540-48184-2_32]
[51] Kerala Blockchain Academy. Merkle Patricia trie in Ethereum: A silhouette. 2021. https://kbaiiitmk.medium.com/merkle-patricia-trie-in-ethereum-a-silhouette-c8d04155b490
[52] Xia Q, Dou WS, Guo KW, Liang G, Zuo C, Zhang FJ. Survey on blockchain consensus protocol. Ruan Jian Xue Bao/Journal of Software,

2021, 32(2): 277−299 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6150.htm [doi: 10.13328/j.cnki.jos.006150]

[53] Castro M, Liskov B. Practical Byzantine fault tolerance. In: Proc. of the 3rd Symp. on Operating Systems Design and Implementation. New Orleans: USENIX Association, 1999. 173–186.

[54] Eyal I, Gencer AE, Sirer EG, Van Renesse R. Bitcoin-NG: A scalable blockchain protocol. In: Proc. of the 13th USENIX Symp. on Networked Systems Design and Implementation. Santa Clara: USENIX Association, 2016. 45–59.

[55] Bentov I, Lee C, Mizrahi A, Rosenfeld M. Proof of activity: Extending Bitcoin's proof of work via proof of stake. ACM SIGMETRICS Performance Evaluation Review, 2014, 42(3): 34–37. [doi: 10.1145/2695533.2695545]

[56] Bentov I, Gabizon A, Mizrahi A. Cryptocurrencies without proof of work. In: Proc. of the 2016 Int'l Workshops on Financial Cryptography and Data Security. Christ Church: Springer, 2016. 142–157. [doi: 10.1007/978-3-662-53357-4_10]

[57] Gilad Y, Hemo R, Micali S, Vlachos G, Zeldovich N. Algorand: Scaling byzantine agreements for cryptocurrencies. In: Proc. of the 26th Symp. on Operating Systems Principles. Shanghai: ACM, 2017. 51–68. [doi: 10.1145/3132747.3132757]

[58] Kiayias A, Miller A, Zindros D. Non-interactive proofs of proof-of-work. In: Proc. of the 24th Int'l Conf. on Financial Cryptography and Data Security. Kota Kinabalu: Springer, 2020. 505–522. [doi: 10.1007/978-3-030-51280-4_27]

[59] Kiayias A, Lamprou N, Stouka AP. Proofs of proofs of work with sublinear complexity. In: Proc. of the 2016 Int'l Workshops on Financial Cryptography and Data Security. Christ Church: Springer, 2016. 61–78. [doi: 10.1007/978-3-662-53357-4_5]

[60] Bünz B, Kiffer L, Luu L, Zamani M. FlyClient: Super-light clients for cryptocurrencies. In: Proc. of the 2020 IEEE Symp. on Security and Privacy (SP). San Francisco: IEEE, 2020. 928–946. [doi: 10.1109/SP40000.2020.00049]

[61] zcash. 2023. https://github.com/zcash/zcash

[62] Westerkamp M. Verifiable smart contract portability. In: Proc. of the 2019 IEEE Int'l Conf. on Blockchain and Cryptocurrency (ICBC). Seoul: IEEE, 2019. 1–9. [doi: 10.1109/BLOC.2019.8751335]

[63] Fynn E, Bessani A, Pedone F. Smart contracts on the move. In: Proc. of the 50th Annual IEEE/IFIP Int'l Conf. on Dependable Systems and Networks (DSN). Valencia: IEEE, 2020. 233–244. [doi: 10.1109/DSN48063.2020.00040]

[64] Westerkamp M, Küpper A. SmartSync: Cross-blockchain smart contract interaction and synchronization. In: Proc. of the 2022 IEEE Int'l Conf. on Blockchain and Cryptocurrency (ICBC). Shanghai: IEEE, 2022. 1–9. [doi: 10.1109/ICBC54727.2022.9805524]

[65] Buterin V. The stateless client concept. 2017. https://ethresear.ch/t/the-stateless-client-concept/172

[66] Dryja T. Utreexo: A dynamic hash-based accumulator optimized for the Bitcoin UTXO set. 2019. https://eprint.iacr.org/2019/611

[67] Bailey B, Sankagiri S. Merkle trees optimized for stateless clients in Bitcoin. In: Proc. of the 2021 Int'l Conf. on Financial Cryptography and Data Security. Springer, 2021. 451–466. [doi: 10.1007/978-3-662-63958-0_35]

[68] Chepurnoy A, Papamanthou C, Srinivasan S, Zhang YP. Edrax: A cryptocurrency with stateless transaction validation. Cryptology ePrint Archive, 2018.

[69] Boneh D, Bünz B, Fisch B. Batching techniques for accumulators with applications to IOPs and stateless blockchains. In: Proc. of the 39th Annual Int'l Cryptology Conf. on Advances in Cryptology. Santa Barbara: Springer, 2019. 561–586. [doi: 10.1007/978-3-030-26948-7_20]

[70] Gorbunov S, Reyzin L, Wee H, Zhang ZF. Pointproofs: Aggregating proofs for multiple vector commitments. In: Proc. of the 2020 ACM SIGSAC Conf. on Computer and Communications Security. ACM, 2020. 2007–2023. [doi: 10.1145/3372297.3417244]

[71] Tomescu A, Abraham I, Buterin V, Drake J, Feist D, Khovratovich D. Aggregatable subvector commitments for stateless cryptocurrencies. In: Proc. of the 12th Int'l Conf. on Security and Cryptography for Networks. Amalfi: Springer, 2020. 45–64. [doi: 10.1007/978-3-030-57990-6_3]

[72] Bitcoin Wiki. Hashlock. 2019. https://en.bitcoin.it/wiki/Hashlock

[73] Bitcoin Wiki. Timelock. 2022. https://en.bitcoin.it/wiki/Timelock

[74] Belotti M, Moretti S, Potop-Butucaru M, Secci S. Game theoretical analysis of cross-chain swaps. In: Proc. of the 40th IEEE Int'l Conf. on Distributed Computing Systems (ICDCS). Singapore: IEEE, 2020. 485–495. [doi: 10.1109/ICDCS47774.2020.00060]

[75] Winzer F, Herd B, Faust S. Temporary censorship attacks in the presence of rational miners. In: Proc. of the 2019 IEEE European Symp. on Security and Privacy Workshops (EuroS&PW). Stockholm: IEEE, 2019. 357–366. [doi: 10.1109/EuroSPW.2019.00046]

[76] Tsabary I, Yechieli M, Manuskin A, Eyal I. MAD-HTLC: Because HTLC is crazy-cheap to attack. In: Proc. of the 2021 IEEE Symp. on Security and Privacy (SP). San Francisco: IEEE, 2021. 1230–1248. [doi: 10.1109/SP40001.2021.00080]

[77] Nadahalli T, Khabbazian M, Wattenhofer R. Timelocked bribing. In: Proc. of the 25th Int'l Conf. on Financial Cryptography and Data Security. Springer, 2021. 53–72. [doi: 10.1007/978-3-662-64322-8_3]

[78] Harris J, Zohar A. Flood & loot: A systemic attack on the lightning network. In: Proc. of the 2nd ACM Conf. on Advances in Financial Technologies. New York: ACM, 2020. 202–213. [doi: 10.1145/3419614.3423248]

[79] Zakhary V, Agrawal D, Abbadi AE. Atomic commitment across blockchains. Proc. of the VLDB Endowment, 2020, 13(9): 1319–1331. [doi: 10.14778/3397230.3397231]

[80] Zie JY, Deneuville JC, Briffaut J, Nguyen B. Extending atomic cross-chain swaps. In: Proc. of the 2019 Int'l Workshops on Data Privacy Management, Cryptocurrencies and Blockchain Technology. Luxembourg: Springer, 2019. 219–229. [doi: 10.1007/978-3-030-31500-9_14]

[81] Lipton A, Hardjono T. Blockchain intra- and interoperability. In: Babich V, Birge JR, Hilary G. Innovative Technology at the Interface of Finance and Operations: Vol. II. Cham: Springer, 2022. 1–30. [doi: 10.1007/978-3-030-81945-3_1]

[82] Gray J. The transaction concept: Virtues and limitations. In: Proc. of the 7th Int'l conf. on Very Large Data Bases. Cannes: VLDB Endowment, 1981. 144–154.

[83] Traiger IL, Gray J, Galtieri CA, Lindsay BG. Transactions and consistency in distributed database systems. ACM Trans. on Database Systems, 1982, 7(3): 323–342. [doi: 10.1145/319732.319734]

[84] Zhao DF, Li TL. Distributed cross-blockchain transactions. arXiv:2002.11771, 2020.

[85] Liu ZT, Xiang YX, Shi J, Gao P, Wang HY, Xiao XS, Wen BH, Hu YC. HyperService: Interoperability and programmability across heterogeneous blockchains. In: Proc. of the 2019 ACM SIGSAC Conf. on Computer and Communications Security. London: ACM, 2019. 549–566. [doi: 10.1145/3319535.3355503]

附中文参考文献:

[24] 中国信息通信研究院. 可信区块链推进计划: 区块链互操作白皮书. 北京: 中国信息通信研究院, 2020. http://www.caict.ac.cn/kxyj/qwfb/bps/202012/P020201230759713827891.pdf

[27] 微众银行区块链团队. WeCross技术白皮书: 区块链跨链协作平台. 2020. http://www.d-long.com/eWebEditor/uploadfile/20200416161636165570 31.pdf

[42] 趣链科技. BitXHub白皮书V2.0: 区块链跨链技术平台. 2022. https://upload.hyperchain.cn/BitXHub 白皮书.pdf

[52] 夏清, 窦文生, 郭凯文, 梁赓, 左春, 张凤军. 区块链共识协议综述. 软件学报, 2021, 32(2): 277–299. http://www.jos.org.cn/1000-9825/6150.htm [doi: 10.13328/j.cnki.jos.006150]

段田田(1996－), 女, 博士生, CCF 学生会员, 主要研究领域为区块链, 分布式系统.

李忠诚(1962－), 男, 博士, 研究员, 博士生导师, CCF 高级会员, 主要研究领域为计算机网络, 区块链.

张瀚文(1981－), 女, 博士, 副研究员, CCF 高级会员, 主要研究领域为计算机网络, 区块链.

张珺(1975－), 女, 博士, 副教授, CCF 高级会员, 主要研究领域为区块链, 下一代互联网, 信息安全.

李博(1996－), 男, 博士生, 主要研究领域为区块链, 数据定价与性能分析.

孙毅(1979－), 男, 博士, 研究员, 博士生导师, CCF 杰出会员, 主要研究领域为区块链, 分布式系统, 网络体系结构.

宋兆雄(1993－), 男, 工程师, CCF 专业会员, 主要研究领域为区块链, 分布式系统.