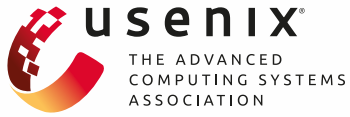

Monoxide: Scale out Blockchains with Asynchronous Consensus Zones

Jiaping Wang, *ICT/CAS, Sinovation AI Institute;* Hao Wang, *Ohio State University*

https://www.usenix.org/conference/nsdi19/presentation/wang-jiaping

This paper is included in the Proceedings of the 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI '19).

February 26–28, 2019 • Boston, MA, USA

ISBN 978-1-931971-49-2

Open access to the Proceedings of the 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI '19) is sponsored by

Monoxide: Scale Out Blockchain with Asynchronous Consensus Zones

Jiaping Wang
ICT/CAS & Sinovation AI Institute

Hao Wang
Ohio State University

Abstract

Cryptocurrencies have provided a promising infrastructure for pseudonymous online payments. However, low throughput has significantly hindered the scalability and usability of cryptocurrency systems for increasing numbers of users and transactions. Another obstacle to achieving scalability is the requirement for every node to duplicate the communication, storage, and state representation of the entire network.

In this paper, we introduce the Asynchronous Consensus Zones, which scales blockchain system linearly without compromising decentralization or security. We achieve this by running multiple independent and parallel instances of single-chain consensus systems termed as *zones*. The consensus happens independently within each zone with minimized communication, which partitions the workload of the entire network and ensures a moderate burden for each individual node as the network grows. We propose *eventual atomicity* to ensure transaction atomicity across zones, which achieves the efficient completion of transactions without the overhead of a two-phase commit protocol. Additionally, we propose *Chu-ko-nu mining* to ensure the effective mining power in each zone to be at the same level of the entire network, making an attack on any individual zone as hard as that on the full network. Our experimental results show the effectiveness of our work: on a testbed including 1,200 virtual machines worldwide to support 48,000 nodes, our system delivers 1,000× throughput and 2,000× capacity over the Bitcoin and Ethereum networks.

1 Introduction

Since the peer-to-peer electronic cash system [37] was published in 2008, decentralized consensus systems continuously enlarged its community and exerted bigger impacts on our society. However, low transaction confirming throughput measured as transaction-per-second (TPS) has significantly hindered the usability of such systems with increased amounts of users and transactions. Besides network latency, the root cause of the throughput issue is the sequential nature of block creation. In a blockchain, blocks are created sequentially with sufficient propagation time in between, which yields a fixed low TPS[1] regardless of how many full nodes and miners participate in the network.

Additionally, a consensus system can not scale out when every full node needs to duplicate the communication, storage, and state representation of the entire network, which are cases of Bitcoin and Ethereum. Even if a high throughput is achieved, workloads requiring fast communication, adequate storage and sufficient computing power will soon set a high barrier for full nodes to participate in, which in turn dramatically hinders decentralization in practice. Therefore, a scalable blockchain system needs to consider the scalability of its consensus protocol with the resource usage of communication, storage, computation and memory for state representation while preserving decentralization and security. Previously, the Ethereum community has investigated sharding in blockchain systems [39] (see Section 8 for more discussions). Motivation of the proposed method echoes many considerations discussed in this article regarding duplicated workload, low entry barrier, importance of efficient cross-shard transaction handling, and the security issue of diluted mining power.

A scalable Blockchain system is desired so that future applications at the Internet scale can be supported. VisaNet payment and clearance [46] takes roughly 4k TPS on average. Alipay mobile payment exceeded 256k TPS at peak traffic [3] in 2017. Rapid growth of DApps on blockchain [44] also exhibits huge demand for scalable Blockchain systems with high throughput and large capacity to support games and decentralized exchanges.

[1] Roughly 7 TPS for Bitcoin with 1MB block and 10-minutes intervals; and 15 TPS for Ethereum with roughly 32KB block and 15-seconds intervals.

Those demands motivate our work.

Asynchronous Consensus Zones is the main idea in this paper, which aims to design a scalable blockchain system without weakening decentralization or security. We scale out blockchain systems by partitioning and handling workloads in multiple independent and parallel instances, or *Consensus Zones*. The state of the entire network are partitioned by zones, and each zone is responsible for its own piece. The core data structures, such as blocks and transactions, are zone-specific, and are replicated and stored only within their own zones. Mining competition, chain growth, and transaction confirmation are carried out separately and asynchronously in each zone.

Consensus zones exhibit natural linear scalability for capacity by having the amount of storage, computation power, and state-representing memory proportional to the total number of zones. There are two major challenges to design such a blockchain system: (1) high throughput should be ensured in a scalable fashion when handling cross-zone transactions; (2) security should be reinforced as honest mining power diluted due to independent growths of chains in individual zones.

Cross-Zone Atomicity is crucial to the correctness and robustness of the blockchain system. In consensus zones, a transaction might involve multiple parties in different zones. It is challenging since state-updating of those parties occurs independently in different zones. Efficient handling of such cases is the key to the throughput scalability and the performance of the entire system. We propose *Eventual Atomicity* to ensure transaction atomicity across zones. With this technique, all operations will complete and will eventually achieve the correct end-state instead of serializing transactions like the two-phase commit protocol [27] does. Eventual atomicity allows interleaving of transactions in an asynchronous and lock-free manner to keep zones concurrent and fully utilized. Eventual atomicity decouples a cross-zone transaction into multiple steps (relay transactions), each involves a single zone. Those steps are relayed and executed across zones by miners. Our system doesn't guarantee miners to handle all relay transactions, just like transaction handling is not guaranteed in Bitcoin or Ethereum. Instead, miners in each zone are incentivized to complete each step by handling relay transaction and ensuring the atomicity eventually.

Effective Mining Power Amplification is introduced to reinforce the security for consensus zones. A blockchain system relies on the majority of mining power to outpace attackers. However, when the mining power is distributed to different zones, an attacker can gather the mining power toward a single zone and may easily exceed the 51% threshold within that zone. To address this problem, we propose *Chu-ko-nu*[2] *Mining* to ensure the per-zone security. With Chu-ko-nu mining, a miner is allowed to create multiple blocks in different zones by solving one proof-of-work puzzle. This greatly amplifies the effective mining power of honest miners who evenly distribute the mining power across zones. Such amplifications, on the other hand, doesn't apply to attackers because the amplified mining power is forced to be evenly distributed to multiple zones, which can not be gathered towards a single zone. In this way, the effective mining power in each zone will be at the same level of total physical mining power in the entire network, which makes attacking an individual zone as hard as attacking the entire network.

Contributions in this paper include the following:

1. A scale-out blockchain system that divides workloads of communication, computation, storage, and memory for state representation into independent and parallel zones. Our system keeps the burden of individual full nodes at a low level as the network grows.
2. An eventual atomicity technique for efficient handling of cross-zone transactions, ensuring correctness and robustness in zones that work asynchronously.
3. Chu-ko-nu mining, a novel proof-of-work scheme, to prevent lowering the attack bar when the mining power is dispersed into multiple zones.

To demonstrate the effectiveness of our system, we carry out a set of experiments on a testbed including 1,200 virtual machines worldwide with the historical data of ERC20 payments from the Ethereum network. In these experiments, our system has delivered $1,000\times$ throughput and $2,000\times$ capacity over Bitcoin and Ethereum networks.

2 Background

We now provide necessary background of this work including details of blockchain systems and a high-level comparison of two consensus mechanisms, i.e., Proof-of-Work (PoW) and Proof-of-Stake (PoS). Lastly, we discuss the differences between the UTXO and Account/Balance transaction models.

[2]Chu-ko-nu is a repeating crossbow shooting multiple arrows at once. It was invented by Zhuge Liang during the warring states period in ancient China.

2.1 Blockchain System

A blockchain system such as Bitcoin [37] and Ethereum [6] contains many compute nodes as miners and full nodes. Transactions are incremental updates of the state, which are confirmed and carried by blocks. Blocks are created by miners. The verification of a block is involved with a mathematical puzzle (also called the proof-of-work, PoW), which is moderately hard on the request side but easy to check for the network. Miners compete with each other, and the first miner who solves the puzzle will be given rewards and the one-time privilege for creating a new block. A newly-created block has to be sufficiently propagated among miners and full nodes before the next block can be created. Due to the network delay, other miners may still work on different blocks, and diverge the chain into different paths when appending blocks on the chain. The divergence of the chain is called a fork, and the blocks not in the main chain are called orphaned blocks. Enlarging the block size (higher propagation latency) or lessening the creation interval may lead to more orphaned blocks, and even prevent the system from converging to a single longest chain in extreme circumstances (e.g. orphan rate $> 50\%$). We will provide detailed discussions on related studies in Section 8.

We summarize other aspects that affect the overall performance of a blockchain system with the consensus protocol as below:

- *Consensus:* The sequential nature of block creation and confirmation required by the consensus protocol is the major challenge of scalability. This is bound by the throughput as analyzed above.
- *Communication:* Information, including unconfirmed transactions and newly-created blocks, needs to be exchanged between all miners and full nodes. It is bound by the local bandwidth.
- *Storage:* All accepted blocks of the chain need to be stored persistently in every miners and full nodes. These are bound by the local disk space.
- *Representation:* The global states of the entire network, e.g. per-address balance and smart contract state, are maintained by every miners and full nodes. These are bound by the size of the host memory.

A scalable design of blockchain has to take all four aspects into consideration.

2.2 PoW and PoS

As discussed above, PoW in predominant cryptocurrencies, including Bitcoin and Ethereum, requires miners to do a compute-intensive verification to maintain the consensus on each block. It yields huge electricity consumption, but it sets fundamental real-world values to the corresponding cryptocurrencies.

In contrast, PoS selects the creator of a block in a deterministic fashion that usually depends on the stake (wealth) of a node. Existing PoS systems adopt different methods to produce the randomness in the leader election to ensure decentralization and security [22, 7, 45, 18]. Although PoW is used in this paper, our technique is orthogonal to the actual consensus mechanism used per zone. Please refer to Section 8 for detailed discussions on state-of-the-art research of PoW and PoS.

2.3 UTXO and Account/Balance

There are two major types of transaction models in cryptocurrencies. The former is the Unspent Transaction Output (UTXO) model, where a transaction spends outputs from previous transactions and generates new outputs that can be spent in future transactions. In a UTXO-based system, a user or an account may have multiple UTXOs. When a user wants to spend money, she uses one or more UTXOs to cover the cost and may get some changes back as new UTXOs. This model is used by Bitcoin and many blockchain systems [11, 32, 23]. The latter is the account/balance model, which is similar to a bank account. Before approving a transaction, the bank needs to check if the account balance can cover the cost. This model is used by Ethereum and it is thought to be better than UTXO for supporting smart contracts [6].

Our system uses the account/balance model due to its simplicity since a transaction with an arbitrary amount can be performed with one sending account and one receiving account (instead of multiple UTXOs on both sides). Additionally, the balance can be extended to more complex state to support programable application logic. Another important benefit offered by the account/balance model is allowing transactions to carry incremental updates of states, as oppose to the UTXO transactions that can only carry full states. This makes a significant savings of transaction size for applications like non-fungible tokens (e.g., Ethereum's ERC-721 token), in which the state is a set of unique identifiers.

3 System Design

Before diving into details, let's first check the high-level architecture of our system for handling a payment that involves two users from different zones, i.e., zones A and B as shown in Figure 1. In that case, the withdraw operation ρ that only involves the state in zone A is handled by a miner in zone A. If the account balance satisfies the cost of this withdraw operation, the corresponding

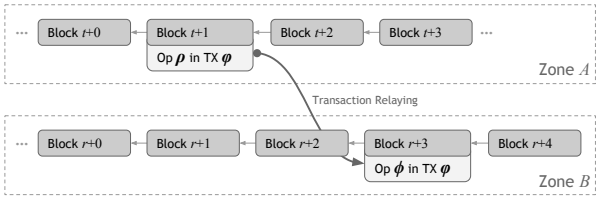

Figure 1: Message passing by relaying transaction across asynchronous zones.

block $t+1$ carrying the transaction (*initiative transaction*) will be created by the miner and only be appended to the chain of zone A. After that, a *relay transaction* carrying the deposit operation ϕ is composed in zone A and forwarded to zone B. The deposit operation ϕ that only involves the state in zone B can always be executed, regardless of the balance of the target account in zone B. Once the relay transaction is picked up by another miner in zone B, operation ϕ will be executed, concluding the complete of the payment transaction.

3.1 Partitioning and Naming

In our system, an account, or a user, is represented by its address, i.e., a fix-sized hash value of its public key. Our system uniformly partitions the space of user addresses into 2^k zones in a fixed and deterministic way: a zone is identified by its *sharding scale* k and *zone index* s ($s \in \{0,1,\ldots,2^k-1\}$).

Given a sharding scale k, the zone index of a user can be easily derived, i.e., by calculating the first k bits of its address. The zone index of an initiative transaction is determined by the payer's address, and the zone index of a relay transaction is determined by the payee's address. A block is specified with the $\langle s,k,h\rangle$, with h being its height of the chain. Sharding scale k defines the number of zones and in turn determines the throughput and the capability of the entire network. The following discussion assumes a fixed k for simplicity.

In our system, full nodes join *swarms* to broadcast new transactions and receive blocks from other full nodes, including miners. A swarm is a group of nodes that participate in the replication of the same data set. In Bitcoin or Ethereum, there is only one swarm and every full node replicates the same data set, including all blocks and transactions. In our system, multiple swarms are established for different purposes. A distributed hash table (DHT) is employed for swarm addressing and peer discovery. Details are described in section 7.

Our system has a *global swarm* joined by all full nodes for replicating the minimum common information of all zones. On the other hand, most communication occurs in *zone-specific swarms* with full nodes belonging to specific zones only. In each swarm, the participating full nodes are sparsely connected, and use the gossip protocol to broadcast messages. Similar to zones, zone-specific swarms are also identified by zone index s and sharding scale k.

3.2 Isolated Intra-Zone Workload

A full node, or a miner, will have a persistent identifier that is initialized randomly; it determines a particular zone the node should work on. With address space partitioning, a blockchain is established within each zone independently. A miner only competes on PoW with other miners in the same zone and confirms transactions from its own zone. Full nodes will ignore any blocks or transactions received that do not belong to their zone, although those are unlikely to be received.

Therefore, the computation and storage related to transaction validation and chain formation are independent and isolated between zones: (1) a miner is only responsible for mining transactions that happen within the zone in which it has chosen to participate, and (2) any full node only records the chain for balances of users in its own zone. As the entire network grows, more zones will be created, ensuring that the burden of computation and storage on an individual node is always at a reasonable level. A low barrier of joining and operating in the network for a full node is essential to maintaining decentralization and robustness of a blockchain system.

3.3 Minimized Cross-Zone Overhead

In a blockchain system, most communication is for replicating unconfirmed transactions and for broadcasting new blocks carrying confirmed ones. In our system, such communication is performed only among nodes within the zone. Our system maintains a distributed hash table (DHT) on each node. After getting the zone index s of an unconfirmed transaction or a forwarding block, our system selects out nodes having the same zone index as s based on the local DHT routing table, and it sends the transaction and block to these nodes following the gossip protocol [12] as is used in Bitcoin and Ethereum. This isolates most communication within each zone.

For cross-zone transactions, our system sends relay transactions only to destination zones instead of the whole network. Additionally, minimized data for chain forming excluding actual confirmed transactions are replicated across all zones. We will discuss this in the next section.

4 Efficient Cross-Zone Atomicity

We divide a block into two parts: a *chaining-block* for the chain formation and the PoW verification, and a *transaction-block* carrying actual confirmed transactions. As shown in Figure 2, a chaining-block, e.g., Θ^a, carries block metadata, including a PoW nonce, a pointer to the precursor block, a Merkle tree [34] root of the list of confirmed transactions, etc. In addition, a chaining-block provides the Merkle tree root for the list of all relay transactions originated from initiative transactions in this block, which is used for the validation of relay transactions in other zones. A transaction-block, e.g., Φ^a, that records the transaction list (same as those in existing systems like Bitcoin and Ethereum) is only replicated and stored by full nodes in the zone. Compared to the hundreds of kilo-bytes used for transactions, a chaining-block has a fix-sized data structure that takes roughly 100 bytes, introducing negligible overhead for both communication and storage.

For an initiative transaction with a withdraw operation ρ from payer a and a deposit operation ϕ to payee b, it can be immediately handled within the zone if a and b belong to the same zone. When a and b are from different zones, we introduce a dual-stage transaction handling mechanism by deriving and forwarding a relay transaction that carries the deposit operation to its destination zone. Figure 2 illustrates the process of cross-zone payment with the data structures processed.

Transaction Validation and Forwarding at Zone A

1. An unconfirmed transaction $\langle \rho, a, \phi, b \rangle$ is picked up by a miner in the payer a's zone, when the miner constructs a new block.
2. The initiative transaction is validated if the balance of a is not less than the transfer amount. If the balance is insufficient, the transaction will be marked as invalid, and be concluded and embedded in the block.
3. Otherwise, a chaining-block Θ^a and a transaction-block Φ^a are constructed. Φ^a has a list of validated transactions including the one from a to b.
4. The miner works on a PoW puzzle specific to the list of all confirmed transactions.
5. After the PoW puzzle is solved, immediately the chaining-block Θ^a is broadcast in the global swarm and the transaction block Φ^a is broadcast in a's zone-specific swarm.
6. Intra-zone transactions are executed and concluded.
7. The withdraw operations ρ in all cross-zone transactions are then executed.
8. Each cross-zone transaction derives an outbound relay transaction $\psi := \langle \phi, b, \gamma \rangle$, which will be sent to the destination zone, i.e., the payee b's zone B.

Relay Transaction Handling at Zone B

1. An inbound relay transaction $\psi := \langle \phi, b, \gamma \rangle$ is picked up by a miner in payee b's zone when constructing a new block.
2. The miner verifies the inbound relay transaction against its originate block Θ^a. Skip this if invalid.
3. The miner constructed a new chaining-block Θ^b and a new transaction-block Φ^b. Θ^b including the inbound relay transaction $\langle \phi, b, \gamma \rangle$.
4. The block Φ^b will be broadcast in b's zone after the PoW puzzle is solved.
5. The deposit operation ϕ is executed, concluding the transaction $\langle \rho, a, \phi, b \rangle$.

In Figure 2, the transaction-blocks are only propagated and stored in their own zone, i.e., Φ^a and Φ^b in zone A and zone B, respectively. The relay transactions, e.g., ψ , are generated at the originate zone, i.e., zone A, and only sent to the destination zone, i.e., zone B. The chaining-blocks, e.g., Θ^a and Θ^b, each of which has roughly 100 bytes, are replicated to all zones.

4.1 Verification

Transaction Verification

When a miner in zone B receives a relay transaction from zone A, it needs to verify this transaction in order to avoid the attack from a malicious peer. As shown in Figure 2, a forwarded transaction $\psi := \langle \phi, b, \gamma \rangle$ includes verification data γ, where

$$\gamma := \langle s, k, t, p, \{h_q\} \rangle, \tag{1}$$

position pointer p denotes the position in the list of outbound relay transactions in its originate block, Merkle tree path $\{h_q\}$ refers to hash values of all sibling nodes on the path from Merkle tree root to its entry; and zone index s, sharding scale k, and height t are used to identify its originate block.

The Merkle tree root will be recalculated using the Merkle tree path $\{h_q\}$ and the transaction $\langle \phi, b \rangle$ itself. It is verified if the recalculated Merkle tree root matches with that in its originate block Θ^a and Θ^a is on the chain in zone A. Note that the relationships to siblings (left or right) is not encoded in $\{h_q\}$, which is inferred from p instead.

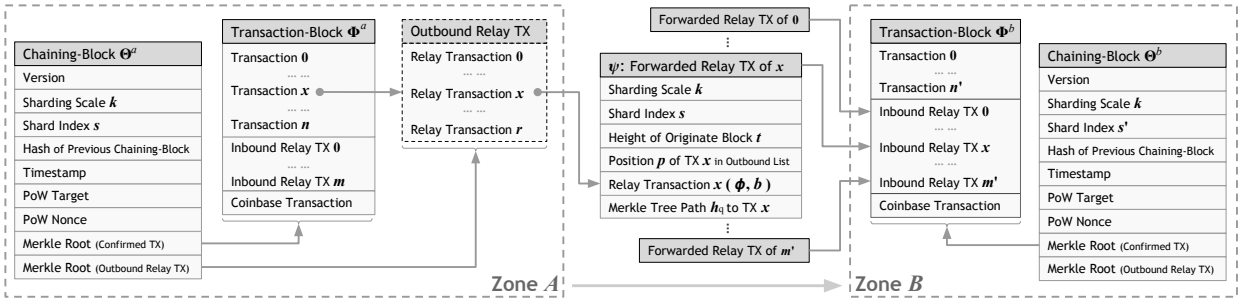

Figure 2: Data structure of the chaining-blocks and transaction-blocks. Outbound relay transactions are derived from confirmed transactions and forwarded with verification data ($\boldsymbol{h}_{\mathrm{q}}$ and etc.)

Block Verification

On receiving a block (pair of chaining-block and transaction-block) broadcasted by a miner, a full node needs to verify the block to defend malicious miners. In our system, a full node verifies three types of transactions. As shown in Figure 2, they are:

1. Confirmed initiative transactions in its own zone.
2. Inbound relay transactions previously forwarded from other zones.
3. Outbound relay transactions forwarded to other zones.

To save the storage space of full nodes, transactions of the first two types are actually embedded in the transaction-block; while outbound relay transactions are not, since they can be derived from the list of confirmed initiative transactions.

The confirmed transactions are verified against current user states. Any block containing illegal transactions or mismatched pairs of initiative/relay transactions will be rejected. The inbound relay transactions are verified against their originate blocks as described in section 4.1. This process also checks if all outbound relay transactions are created as expected, by double checking the Merkle tree root of the list of all outbound relay transactions. It is assumed that outbound relay transactions are ordered and precisely consistent with the order of confirmed initiative transactions. Note that only the Merkle tree root is embedded in the chaining-block, instead of the outbound relay transactions themselves.

4.2 Eventual Atomicity

A payment transaction involving withdraw and deposit operations, should be atomic to ensure correctness of the global ledger. In existing blockchain systems, e.g., RSCoin [11] and OmniLedger [23], the variants of two-phase commit (2PC) mechanism [36] are used to ensure the atomicity, with the known lock/unlock overhead.

In our system, for a cross-zone transaction, we allow the withdraw operation to execute first, interleaving with other transactions then the corresponding deposit operation to be settled later. What is achieved is that once the withdraw operation is confirmed, the deposit operation will be executed eventually. We call such an atomicity, **Eventual Atomicity**. We optimistically assume withdraw operations, carried by relay transactions, will be eventually picked as long as there are well-behaved miners that want to earn transaction fees. What our design ensures is that relay transactions will not be discriminated by sufficiently incentivizing with a fee split.

Theoretically, there could be bad-behaved miners creating empty blocks without confirming any transaction, neither for normal transactions and relay ones. In that case, throughput will be harmed and eventual atomicity will not be fulfilled until a well-behaved miner eventually gains the opportunity for block creation. In the history of the Bitcoin and Ethereum networks, creation of empty blocks is possible but rare [17, 33].

By default, the newly-derived outbound relay transactions are forwarded to their destination zones by the miner, who created the chaining-block and the transaction-block in the originate zone. Once the relay transaction gets replicated in the destination zone, it will never expire before being picked up by a miner, unless its initiative transaction is invalidated, e.g., being orphaned block due to the chain fork (we will discuss this case later). If a relay transaction is accidentally dropped, which is extremely unlikely, it can be reconstructed by any full node in the originate zone based on its originate block on the chain. No additional verification or consensus is required to restart the replication of the reconstructed relay transaction.

In the existing blockchain system, a payment transaction will be visible to its payee once it is packed in a block on the chain (first confirm). It will be secured after $n-1$ successive blocks appended (n-th confirm, $n=6$ in Bitcoin and 12 in Ethereum). In contrast, a cross-zone payment transaction in our system will be visible to

the payee, once its relay transaction is forwarded to the payee's zone, and its originate block becomes available. With the eventual atomicity, the transaction is considered as eventually secured, once its initiative transaction gets n-confirmed, and the relay transaction gets a first confirm. Since miners of these two zones are working independently, n-confirmation of the initiative transaction is overlapped with the forwarding and first confirmation of the relay transaction. Thus, eventual atomicity introduces no additional delay, which is also demonstrated in our experiments in section 7.3. Theoretically, additional latency may occur, when the relay transaction waits too long to be picked up by a miner. This can take even longer than n-confirmation of its initiative transaction.

4.3 Fork Resolution

In a proof-of-work consensus system, it is possible for different miners to create two, or even more, blocks at the height, which are *forks*. Eventually, after a successful fork resolution, only one block will be accepted among those and the rest will be discarded as *orphan blocks*. The longest-chain rule [37] is proposed to resolve forks in Bitcoin and GHOST protocol [42] is also employed in Ethereum. We use GHOST protocol in consensus zones, which is reliable even when the fork rate is high.

In each consensus zone, fork resolution is performed independently in exactly the same way as previous single-chain consensus systems. A block can be one of the three states:

1. **Available**: no fork, or the block is on the winning path.
2. **Unsolved**: the block is on one of the equally competing paths.
3. **Orphan**: the block is on the losing path.

Fork resolution is a continuous procedure as more blocks being created and appended. A previously available block may lately become orphan or unsolved, and vice versa.

With eventual atomicity, the consequence of fork resolution in one zone may affect the validity of relay transactions forwarded and confirmed in another zone. Verification of relay transaction relies on the proof related to its originate block. If the originate block is no longer available after fork resolution, the proof will be invalidated and in turn invalidates all relay transactions that originated from it.

Pre-Confirmed

An unconfirmed relay transaction will not be considered in new block creation until its originate block is in available state. Relay transactions will stay in the unconfirmed transaction set regardless of the validity of their originate block and wait for originate blocks become available.

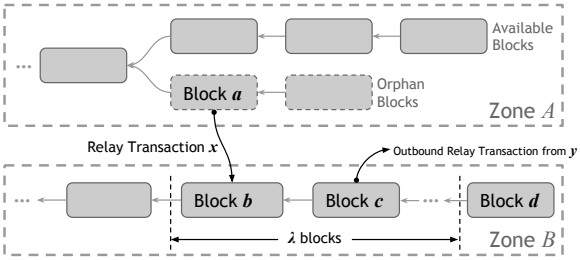

Figure 3: Relay transaction invalidated due to orphaned initiative block after fork resolution.

Post-Confirmed

As shown in figure 3 Zone A, a block a previously available lately become unsolved/orphan, and unfortunately, a relay transaction x originated from it has already been confirmed and embedded in a block b in zone B. In that case, block b will not be invalidated and just the relay transaction x will be invalidated. While, the ledger state in zone b is required to be rebuilt by executing all transactions of all historical blocks since the genesis block, skipping all invalidated relay transactions including x. In practice, state rebuilding is accelerated by state checkpoints so that only operations of recent blocks are re-executed.

Implicate Subsequent Transactions in Invalidating

After relay transaction x is invalidated, an even worse scenario is that a subsequent confirmed transaction become invalidated because it relies on the updated state by the relay transaction x. For example in Figure 3, the relay transaction x deposit u tokens in block b and the balance becomes $u_0 + u$. Lately a transaction y is confirmed in block c and it withdrawn v tokens ($u_0 < v \leq u_0 + u$).

If relay transaction x is invalidated and transaction y is a cross-zone transaction, block c will be invalidated and all subsequent blocks after that will be discarded as well. To avoid such a case, a miner will validate candidate transactions against a special state by delaying execution of inbound relay transaction for λ blocks. Thus, transaction y will be not be confirmed until block d, which makes such case unlikely since block a already received at least λ confirmations.

Let the latest block be b, the normal state $\mathbb{S}$ is built by executing all operations in all transactions from the genesis block to block b. For miners, an additional state $\mathbb{S}_\lambda$ is built by executing from the genesis block to block

$b-\lambda$ and then execute all operations except inbound relay transactions of blocks $b-\lambda+1$ to b. An unconfirmed cross-zone transaction must be validated against both states $\mathbb{S}$ and $\mathbb{S}_\lambda$ to be considered as candidate when creating block $b+1$.

5 Defense Per-Zone Security

In the early stage, miners are individuals and mining power is fragmented. The randomness of voluntary zone assignments based on the local identifier is safe, when no individual miner controls more than 50% mining power in any zone. Professional mining facilities will gradually dominate the mining power by owning great amount of mining power or aggregating mining power from individual miners, which delivers steady, frequent and divided mining rewards [28]. In our system with n zones, a rational mining facility will ideally distribute its total mining power in different zones to maximize the rewards (bias to zones with lower mining difficulty). Eventually, this makes the mining power of the entire network H converge to be evenly distributed across zones. Thus, per-zone mining power will be H/n. When a malicious mining facility gathers all its mining power T focuses on a single zone, the attack will success if $T > H/n \times 50\%$, which will be unacceptably low when with a large n.

To address this issue, we introduce a *Chu-ko-nu mining* mechanism that allows and encourages a miner to create multiple blocks in different zones with one PoW solution. This ensures the effective mining power in each zone is nearly equal to the total physical mining power in the entire network, raising the attack bar in each zone close to 50% when most miners participate in Chu-ko-nu mining. Also, Chu-ko-nu mining can save energy consumed in solving PoW by making it more productive of block creation. Like PoW mechanism itself, security based on Chu-ko-nu mining is driven by incentive.

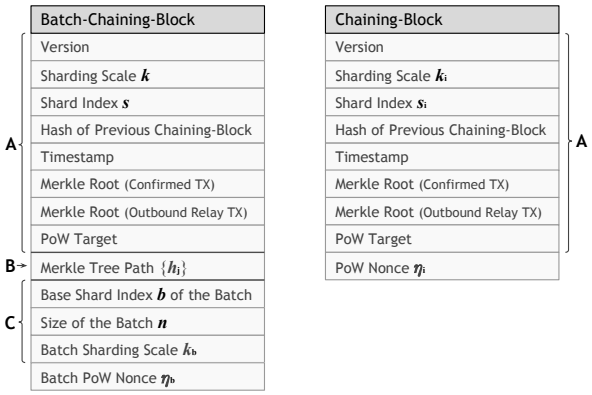

Figure 4: Comparison of a batch-chaining-block and a chaining-block

It will be taken down if the attacker controls more than 50% physical mining power of the entire network, which is also the case of PoW mechanism. While, the defense is all miners that acknowledge the incentive.

5.1 Chu-ko-nu Mining

Chu-ko-nu mining allows a miner use a single PoW solution to create multiple blocks at different zones simultaneously, but no more than one block per-zone. In such a case, a *batch-chaining-block* in Figure 4 will replace the chaining-block, as shown in Figure 2, and get replicated among all zones. Miners are allowed to perform Chu-ko-nu mining for n zones starting from zone index b or all zones ($n = 2^k, b = 0$) based on their capacities of IT resources.

A miner will perform the transaction validation for all involved n zones and collect n chaining-headers A_i, i.e., part **A** in Figure 4. Without Chu-ko-nu mining, as in Bitcoin or Ethereum, a miner is required to find n nonce η_i ($i \in [0, n-1]$) that each fulfills

$$\mathrm{hash}\left(\langle \mathrm{A}_i, \eta_i \rangle\right) < \tau, \tag{2}$$

in which τ denotes the PoW target (a big integer) [37] determining the mining difficulty, and the $<$ operator takes place by regarding the hash value as a big integer. With Chu-ko-nu mining, a miner only needs to find a single nonce η_b that fulfills

$$\mathrm{hash}\left(\langle h_0, \mathrm{C}, \eta_b \rangle\right) < \tau, \tag{3}$$

in which C is the configuration of the batch (part **C** in Figure 4) and h_0 denotes the root of Merkle tree Υ_b over the list of all chaining-headers in this batch:

$$\langle \mathrm{A}_0, \mathrm{A}_1, \ldots, \mathrm{A}_{n-1} \rangle . \tag{4}$$

Note that we assume the PoW targets are the same in all involved zones. We will discuss the case if they are not in the next section.

Once η_b is found, zone-specific batch-chaining-blocks will be composed and sent to corresponding zones. As illustrated in Figure 4, for each zone, a batch-chaining-block carries a chaining-header (A), its Merkle tree path $\{h_j\}$ (B), batch configuration (C), and the found batch PoW nonce η_b, which are the minimum pieces of information for recalculating equation 3 and for verifying the PoW. Our system doesn't explicitly record the relationships to siblings (left or right) in the Merkle tree path $\{h_j\}$. Instead, relationships are inferred from bits of the offset in the batch list ($s-b$). By this means, the zone index s is coupled with the offset in the block list (Equation 4) in the batch, which guarantees that a miner is able to create only one block for each involved zone in one batch.

In each zone, full nodes, as well as miners, treat batch-chaining-blocks and chaining-blocks equally when accepting a new block. They follow the same approach to detect and resolve forks as described in Section 4.3. A chain of a zone is allowed to contain batch-chaining-blocks and chaining-blocks at different block heights.

Chu-ko-nu mining shares similar spirit with merged mining [4] in practices by allowing creating multiple blocks with a single PoW solution. Merged mining is designed for different motivation and scenario, which is for protecting blockchains with small mining power. Chu-ko-nu mining is designed for reinforcing mining power distribution across zones with amplified effective mining power. Chu-ko-nu mining works with multiple chains with equal role and equal mining power, instead of a parent chain and an auxiliary chain. Chu-ko-nu mining also enforce one-block per-zone which leads to a different data structure and implementation.

5.2 Independent Validation in Zones

Chu-ko-nu mining solves the PoW for chaining-headers in batch, while the batch-chaining-block with solved nonce are zone-specific and sent separately. Per-zone batch-chaining-blocks will be validated and accepted independently in each zone. One block can be orphaned or even invalidated, but this will not affect the validation or the acceptance of others.

Independent validation also allows efficient handling of mixed PoW targets of zones in one batch. The construction of batch-chaining-blocks in this case is the same as described in section 5.1, but it allows different values of PoW targets in each chaining-header. While, in probing the batch PoW nonce η_b, it is possible that some blocks with high PoW targets (easier to fulfill) are fulfilled but others are not. Batch-chaining-blocks for these fulfilled zones will be composed and sent to their zones immediately, regardless PoW targets of other zones are not fulfilled (thus not be sent out). The fulfilled blocks in the list of chaining-headers (Equation 4) will be removed and replaced with their successive candidate blocks. The probing of the batch PoW nonce will switch to a new puzzle with updated h_0 in Equation 3. Since the event of finding a fulfilled nonce in different zones is independent and random, such switching will not reduce the mining efficiency.

5.3 Redistributed Mining Power

This section explains why Chu-ko-nu mining can make an attack on a single zone as hard as the attack on the entire network, which can set the attack bar at the same security level of Bitcoin and Ethereum.

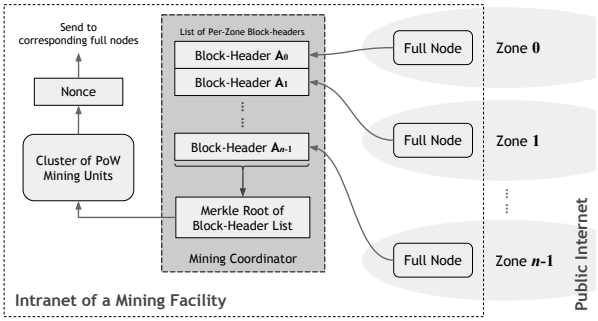

Figure 5: Internal architecture of a mining system

We use **hash rate** to describe the mining power, which is proportional to the speed of producing new blocks. In a network having 2^k zones, let m_p be the total physical hash rate of miners that participate in Chu-ko-nu mining and m_d be the total physical hash rate of miners that don't (as individual miners). The effective hash rate m_s distributed in each zone can be calculated as

$$m_s = \frac{m_d}{2^k} + m_p. \tag{5}$$

If a malicious node can obtain $> \frac{m_s}{2}$ hash rate in any zone, it can control this zone. Therefore, the attack bar in each zone, which can be calculated as the obtained hash rate of a malicious node divided by the total hash rate in the network, is

$$> \frac{m_s}{2 \cdot (m_d + m_p)} = 50\% - \frac{m_d \cdot (2^k - 1)/2^k}{2 \cdot (m_p + m_d)} \tag{6}$$

, which converges to 50% when the mining facility dominates the total hash rate. For example, if the mining facility contributes 99% hash rate in a 256-zone network, a successful attack requires 49.5% of total physical hash rate in the network. The current situation (September 2018) in Bitcoin network, 100% hash rate is contributed by mining facilities[5].

To maximize the incentive from a single PoW solution, professional mining facilities will participate in all zones and hopefully takes rewards from all zones. Chu-ko-nu mining actually amplifies miner's mining power, which is multiplied with the number of zones a miner participates in. The amplified mining power is evenly distributed to all involved zones. Such an effective amplification of mining power helps honest miners that distribute mining power to all zones but don't apply to attackers targeting on a single specific zone.

5.4 Scalable Mining System

A mining system of Bitcoin or Ethereum in a mining facility is a distributed system that consists of hundreds or even thousands PoW mining units, e.g,. GPU and Specialized Application-Specific Integrated Circuit (ASIC)

just focusing on hash probing, plus a few PCs working on actual transaction validation and block construction.

In our system, a mining system is desired to monitor multiple zones. The partitioning scheme in the network naturally provides a scalable solution for a mining facility observing a large number of zones. As illustrated in Figure 5, besides the cluster of PoW mining units, in addition, there is a cluster of full nodes for observing all involved zones, independently discovering new blocks, validating transaction, and constructing candidate block-headers A_i. Once A_i is updated in any zone i, new A_i will be sent to *Mining Coordinator* in Figure 5. Merkle tree of block-header list (Equation 4) will be recalculated by the mining coordinator and the updated Merkle tree root h_0 (Equation 3) will be broadcast to all PoW mining units. Such design provides an example and demonstrate how a scalable mining system can be implemented and operated for professional mining facility.

6 Discussions

6.1 Single Address Hotspot

It is possible that a single address is involved in a great number of transactions, e.g., a deposit address of a large cryptocurrency exchange. In the workload described in Section 7, the address 0x3f5CE5FBFe3E9af3971dD833D26bA9b5C936f0bE, which is one deposit address of Binance (a top cryptocurrency exchange), is the payee of more than 2% total transactions. Since a single address is the finest unit in our partitioning scheme, right now our system can not further partition such workload into multiple zones.

In practice, such a "single address hotspot" issue can be easily resolved with the co-design of applications at the upper layer. For example, an application or a user can allocate multiple deposit addresses in different zones for load balancing. An institutional operator announces a list of addresses for deposit, and the wallet application automatically chooses a random address, or even an intra-zone address if available, for coin transferring. As a result, the transaction throughput of multiple zones can be leveraged.

6.2 Incentives and Fees

We follow Bitcoin's incentive model by rewarding the miners with phase down coinbase in every zone, which end up with a fixed total supply of cryptocurrency. The transaction fee is a parameter set by whoever issue the transaction, which is usually based on the average amount of the fee in recent confirmed transactions. A rational miner prioritizes unconfirmed transactions based on the transaction fee.

We introduce fee splitting for cross-zone transactions, which incentivize both miners working on initial step and relayed step of transaction handling so that relayed transactions will be equally prioritized with transaction fees at similar levels. For a simple payment, the transaction fee can be equally split. For complex transaction with programmable transaction logic, the transaction splitting should be based on the evaluation of workloads in each step similar to the gas calculation in Ethereum.

We don't introduce additional fees for propagating cross-zone transactions. Optimistically, we expect such task will be done voluntarily by all full nodes and miners throughout the network as voluntarily propagating of intra-zone transactions works well in Bitcoin and Ethereum networks. Block creation with cross-zone transactions is a bit more costly than intra-zone transactions. We recommend cross-zone transaction issuer set double or even triple the amount of transaction fees.

We encourage Chu-ko-nu mining by providing equal coinbase reward for blocks generated using Chu-ko-nu or not. Thus, given fixed physical mining power, the profit of Chu-ko-nu mining is proportional to the number of zones participating. It is rational for professionally-operated miners to work on all zones simultaneously, which enhances the overall security of the network.

We do not extend to the quantitative analysis on the incentives and economics of the ecosystem, as it is beyond the scope of this paper.

6.3 Generalization beyond Payment

So far, our discussions are based on the withdraw-deposit paradigm that serves well payment-centric scenarios, e.g., Bitcoin. It is also desirable to extend the model for complex transaction logic and go beyond payment-centric applications. To this end, a main challenge is to correctly handle generalized relay transactions, and guarantee the atomicity. We extend the withdraw-deposit paradigm to a more general model with programmable transaction logic, which is presented in the appendix.

The proposed programmable transaction logic supports programmable issuing and transferring of cryptocurrencies as well as user-defined fungible and non-fungible tokens similar to ERC20/ERC721 tokens on Ethereum. It also supports more complicated applications like a domain registration system. Our method requires that a cross-zone transaction can be verified in a single zone and be completed by one-step irrevocable relay transactions, which don't support applications like many-to-many payments. It can be extended to revocable relay transactions and multi-step relaying, we leave those improvements to future works.

7 Experimental Results

Our system is implemented using C++. Cryptography is implemented based on Botan cryptography library (v1.11) [30] and Intel IPP Cryptography library (v7.1) [21]. We use RocksDB (v4.11) [16] to store archived blocks and transactions. We implement Mainline DHT [31] for P2P routing and swarm formation facilities peer discovery for the global swarm and per-zone swarms.

We evaluate the proposed system by playing back the complete historical ERC20 payments in Ethereum from the beginning up to the block height 5867279, which includes 16.5 million unique addresses and 75.8 million transactions. We deploy our system on a distributed environment that includes 1,200 virtual machines, each of which has 8 cores and 32GB memory. These machines are uniformly distributed in 15 Availability Zones for testing cross-country latency in the real world. In the test network, we restrict the end-to-end peek bandwidth to 30Mbps and the measured average end-to-end latency is 102.48 msec. In every zone, we set a 32KB limit for the block size and a target of 15-second block creation interval, which yields around 15.6 TPS for one-to-one token transfer. The average orphan rate in every zone is 8.3‰, which is independent of the number of zones. On such a testbed, we support 48,000 nodes of blockchain in our system.

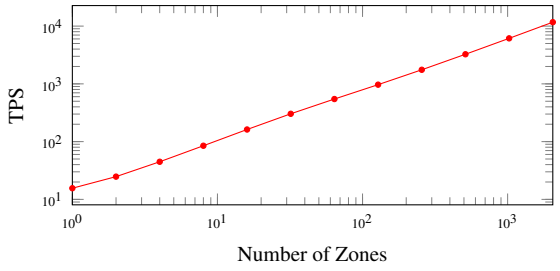

Figure 6: Linear scaling out with multiple zones.

7.1 Scalability

We first evaluate if our partitioning mechanism introduced in Section 3.1 can balance the number of payments between zones. We change the sharding scale k to generate different numbers of zones, i.e., setting k to 4, 5, 8 and getting 16, 32, 256 zones, respectively. Figure 7 illustrates transactions handled in each zone are balanced, with different sharding scale k. While, there exist single address hotspots, which can be further optimized as discussed in Section 6.1.

We evaluate the scalability by measuring the actual throughput of our system with a different number of zones. We fix the number of nodes in each zone, i.e., 24 nodes with 12 miners in average per zone, when increasing the number of zones. Figure 6 shows that the measured TPS scales out as the number of zones increase. The system exhibits the linear scalability, and achieves up to 11,694.89 TPS when there are 2,048 zones. The only exception occurs when increasing the number of zones from one to two. Due to the overhead of relaying transactions, the performance gain using two zones over one zone is 1.88×.

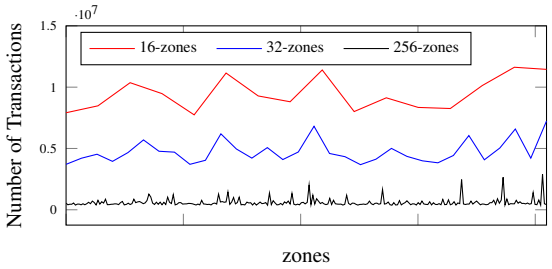

Figure 7: Transaction distribution across zones.

7.2 Overhead

Although the throughput of our system scales out linearly with the number of zones, relaying transactions across zones actually introduces the overhead, amplifying the number of transactions and total size of data to be stored and replicated. Figure 8 shows the percentage of cross-zone transactions grows when increasing the number of zones. Almost all transactions will lead to a relay transaction when there are more than 64 zones in our experiments. Such an amplification doubles the total number of transactions at most and don't weaken the scalability of throughput at all. As shown in Figure 6, throughput keep scaling out when there are more than 64 zones.

Figure 9 illustrates the amplified sizes of data replication and storage in the entire network. First, when increasing the number of zones, almost all transactions lead to a relay transaction, i.e., more than 64 zones in this case. Transactions are put into blocks and persistent storage in original zones; and relay transactions are those of destination zones, doubling the size of transaction-blocks. Second, compared to the transactions in a block, which takes hundreds of kilo-bytes, the chaining-block that has tens of bytes is much less significant. However,

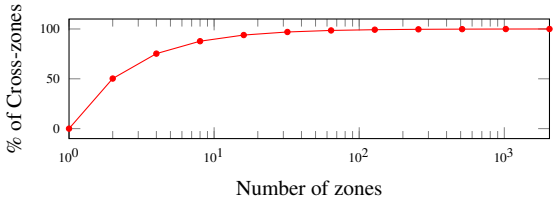

Figure 8: Percentage of cross-zone transactions, which approaches to 100%. Almost every original transaction produced a relay transaction.

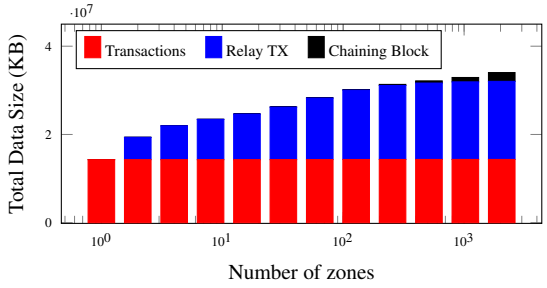

Figure 9: Sizes of the blockchain data in the entire network.

since the chaining-block is duplicated to all zones, their total size on storage in the entire network will be amplified with the number of zones. As shown in the figure, their total size is pushed up to tens of kilo-bytes with 2048 zones, 6.2% of the transaction size, which is acceptable in practice.

7.3 Confirmation Latency

The average number of connections per node affects the data propagation speed in the network, and in turn, the confirmation latency of blockchain. Figure 10 shows the cumulative distribution function of time elapsed and number of nodes reached. We configure the average number of connections per node to 16, 32, 64, 128, and observe that the fastest propagation speed is obtained at 128. In our experiments, each full node connects to 68.5 peers in average.

As presented in Section 4.2, a transaction is first confirmed when its block is replicated between nodes in the network; after n block, the transaction is secured. Bitcoin, which configures n to 6 with a 10-minute interval of block generation, requires 1 hour to secure a transaction. In our system, a relay transaction must be on-chain before the original transaction reaches n-confirms, so there is no additional confirmation latency. Figure 11 shows the first confirmation time of relay transactions. The latency is in a range of 13 to 21 seconds, when the number of zones is larger than 30 (almost all transactions have

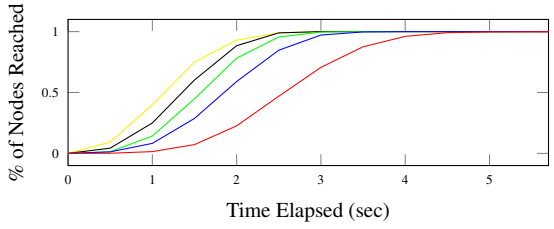

Figure 10: Transaction (<1KB) propagation speed with different average number of connections from each node to other peers. Fastest propagation shown here is 128-connected, slowest one is 8-connected. Ones in between are 64/32/16-connected.

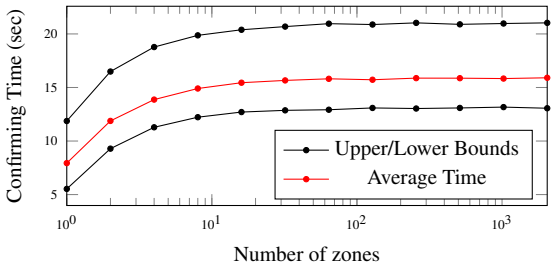

Figure 11: Average first confirming time of transactions.

relay transactions). Since we configure a 15-second interval of block generation, the original transaction is secured after 90 seconds ($n = 6$, same to Bitcoin), which is enough to cover the confirmation latency of relay transactions.

7.4 Throughput and Orphan Rate

We also evaluate the TPS and orphan rate of our system with different block sizes and block creation interval in Figure 12. In these experiments, we fix the number of zones to 256. As expected, enlarging block size or lessening block creation interval yields almost linear TPS in our system, since our system is neutral to actual configurations of Nakamoto consensus instances in each zone. However, a larger block size or a smaller block creation interval leads to a higher orphan rate, inducing blocks wasted. Such behavior matches well with the existing blockchain system. In a typical case, we configure a reasonable block size and a block creation interval, e.g., 32 KB and 15 seconds, for high TPS and low orphan rate.

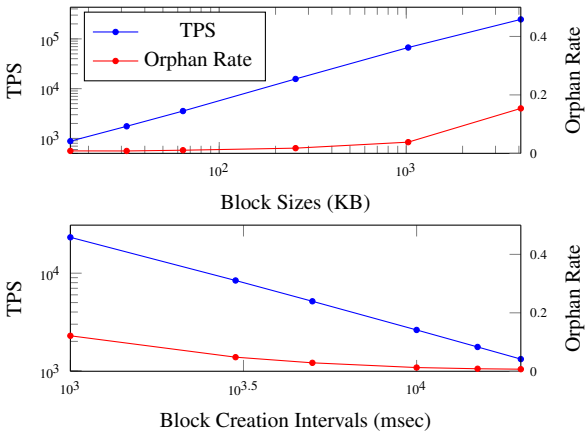

Figure 12: TPS and orphan rate with different block sizes (Upper) and with different block creation interval (Lower). (# zone = 256)

8 Related Work

Many research efforts have been put on improving PoW systems. Instead of the longest branch, GHOST protocol [42] chooses the block whose sub-tree contains most blocks as the main chain to prevent the attack by selfish mining at a fork. Bitcoin-NG [15] selects a leader in each epoch, and allows the leader to post multiple blocks, thus increasing the throughput. SPECTRE [41] and PHANTOM [43] increase Bitcoin's throughput by replacing the chain-based structure to the Directed Acyclic Graph (DAG)-based structure and merging blocks from different branches to the ledger. SPECTRE provides the partial order between DAG blocks, while PHANTOM can keep the total order. Conflux [29] is another DAG-based protocol. It constructs the consistent total order of transactions over DAG through introducing parent and reference edges and combining ghost for pivot chain selection. It allows blocks outside the pivot chain to be able to contribute to the overall throughput. These proposals can enhance Bitcoin security by eliminating the selfish mining and improve the throughput by merging blocks from different branches to the main chain. However, as discussed in Section 1, the performance of these single chain systems is bound to the available network bandwidth of full nodes. Thus, they cannot scale out to thousands of nodes. Moreover, due to the requirement of replicating and processing forked blocks in these systems, it is unclear how full nodes address and resolve the capacity issue on the memory and storage in a large scale.

PoS systems reach the consensus with the majority of stake share. Ouroboros [22] is a PoS system with enhanced persistence and liveness. It uses a coin-flipping protocol to produce the randomness in the leader election. However, it cannot avoid the targeted attack [18] with the assumption that most leaders are incorruptible in an epoch. Tendermint [26] and Casper the Friendly Finality Gadget [7] adopt the Byzantine Fault Tolerance (BFT) protocol to select the committees, reach the consensus, and tolerant against up to one third malicious nodes. However, Practical BFT (PBFT) protocol [8, 40] used in these proposals has significant communication cost and only scales to dozens of compute nodes. Furthermore, on a public blockchain that anyone can participate in, PBFT has the security issues. The permissionless characteristics of PBFT makes the blockchain under the risk of Sybil attack [13], where an adversary can create an arbitrary number of pseudonyms. And, the absence of public-verifiable and unbiased randomness makes the elected committees under the risk of targeted attack [18], once an adversary can predict their identities.

The recent work [1] has discussed the relationship between BFT protocols and blockchain consensus, and highlighted how to optimize BFT for blockchain. Zyzzyva [25] uses speculation to simplify BFT state machine replication and can reduce replication overhead significantly. SBFT [19] is a scalable, trust, and decentralized infrastructure of blockchain. It can handle hundreds of active replicas and support smart contracts of Ethereum. HoneyBadgerBFT [35] proposes an atomic broadcast protocol to optimize the communication complexity of BFT, making the asynchronous BFT to be able to support hundreds of nodes. ByzCoin [24] leverages the collective signing and optimizes the transaction commitment of the BFT-based blockchain. RandHound and RandHerd [45] provide public-verifiable, unpredictable, and unbiased randomness. Algorand [18] grows the blockchain in asynchronous rounds. In a round, each node computes a verifiable random function to determine if it is a committee member. Once a validator sends a message to prove its membership with its vote, Algorand replaces participants immediately. It can avoid the Sybil attack and targeted attack. However, these proposals are reported to have either the security issue or the performance issue. For example, Algorand is susceptible to bias in the randomness of the variable random function (VRF) invocation [20], while ByzCoin may not reach the agreement on the committee election, as reported in the hybrid consensus system [38].

Many distributed systems, e.g., Google Spanner [9] and Slicer [2], use the sharding protocols to scale out; while these solutions are centralized and cannot be used directly in the decentralized blockchain systems. Elastico [32] is a decentralized sharding protocol. In each consensus epoch, the participants can solve a PoW puzzle to join a consensus committee. The committee of each shard runs PBFT to reach the agreement on a set of transactions and sends the agreement to a final committee. The final committee generates the final values from the received agreements, and broadcasts to the network. First, Elastico doesn't ensure the transaction atomicity across shards. Second, in order to limit the overhead of running PBFT, it only supports a small number of committees, leading to a high failure probability [23]. Third, although each committee can verify transactions in their own shard, Elastico still broadcasts all blocks to all nodes and requires them to store the whole ledger. This leads to the capacity issue on full nodes inevitably. OmniLedger [23] is a distributed ledger based on a sharding protocol and tries to resolve the problems in Elastico. OmniLedger uses RandHoundm [45] to ensure the leader election bias-resistant and public-verifiable, and introduces Atomix, a two-phase atomic commit protocol, to guarantee the atomicity of cross-shard transactions. It only requires the validators to store the reference point of each shard, instead of the full transaction history, making full nodes more sustainable. The Ethereum community also introduces the beacon chain [14] to support the

sharding protocol. The beacon chain provides the distributed pseudorandomness for selecting committees of validators on each shard. Because the pseudorandomness randomness is susceptible to bias, the sharding-protocol based blockchain shouldn't assume a trusted randomness beacon. A recent study RapidChain [47] further optimizes these sharding protocols. It is resilient to Byzantine faults from up to a 1/4 fraction, e.g., in OmniLedger, to a 1/3 fraction of the participants. It also enhances the throughput via block pipelining and ensures the robustness of the setup for new participants to join the network. Compared to these systems, our work proposes the eventual atomicity to ensure the atomicity of cross-zone transactions without the lock/unlock overhead. Most importantly, a blockchain system that is partitioned to multiple zones/shards exposes a severe security problem, which is not addressed in these existing proposals. With partitioning, the honest majority of mining power or stake share or randomly selected committee is dispersed into individual zones/shards. This significantly reduces the size of honest majority on each zone/shard, thus dramatically lowering the attack bar on a specific zone/shard. Therefore, we introduce the Chu-ko-nu mining to ensure the security after partitioning, and make attacking any specific zone as difficultly as attacking the entire network.

In summary, the existing proposals targeting on the single chain and non-sharding solution [42, 41, 43, 15, 29, 24, 18] are bound to the network bandwidth and cannot scale out; while the sharding systems either do not support full sharding [32], or may lower the attack bar after sharding [23, 47]. To our best knowledge, Monoxide is the only scalable blockchain system implementing full sharding of the consensus protocol as well as the resource usage for scalability, preserving the guarantees of PoW for decentralization, and maintaining the same level of security of Bitcoin and Ethereum.

9 Conclusion

We proposed a scalable decentralized consensus system based on the blockchain mechanism. Without weakening decentralization and security, our technique offers a linear scale-out by partitioning the workload of all key components of a blockchain system including transaction broadcasting, mining competition, chain storage, transaction execution and state representation. We preserved the simplicity of the blockchain system and amplify its capacity by duplicating equal and asynchronous zones, which work independently with minimal coordination, in parallel. Additionally, Chu-ko-nu mining and eventual atomicity are key contributions proposed by our system, ensuring the efficiency and security of systems with thousands of independent zones. In our experiment, we demonstrate that our system delivers 1,000× throughput and 2,000× capacity over Bitcoin and Ethereum.

10 Acknowledge

We thank Heung-Yeung Shum (Microsoft), Lidong Zhou (Microsoft Research Asia), Xiaobin Zhang (PPTV), Hongbo Zhang (Bloomberg), Xiao Sophia Wang (Uber), Shuang Zhao (University of California, Irvine), Minghao Pan (AMD) and Shumo Chu (University of Washington) for their suggestions, challenges, and encouraging comments along the way. We also thank our shepherd, Vijay Chidambaram (University of Texas at Austin), and all anonymous reviewers for their helpful feedback.

References

[1] ABRAHAM, I., AND MALKHI, D. The blockchain consensus layer and bft. *Bulletin of the EATCS 123* (2017).

[2] ADYA, A., MYERS, D., HOWELL, J., ELSON, J., MEEK, C., KHEMANI, V., FULGER, S., GU, P., BHUVANAGIRI, L., HUNTER, J., PEON, R., KAI, L., SHRAER, A., MERCHANT, A., AND LEV-ARI, K. Slicer: Auto-sharding for datacenter applications. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation* (Berkeley, CA, USA, 2016), OSDI'16, USENIX Association, pp. 739–753.

[3] ALIPAY. World record!! we've processed 256,000 payment transactions per second (tps), 2017. https://twitter.com/alipay/status/929123909970153472.

[4] BITCOIN WIKI. Merged mining specification, 2015. https://en.bitcoin.it/wiki/Merged_mining_specification.

[5] BTC.COM. Pool distribution, 2018. https://btc.com/stats/pool.

[6] BUTERIN, V., AND ET AL. A next-generation smart contract and decentralized application platform, 2013. https://github.com/ethereum/wiki/wiki/White-Paper.

[7] BUTERIN, V., AND GRIFFITH, V. Casper the friendly finality gadget, 2017. https://arxiv.org/pdf/1710.09437.pdf.

[8] CASTRO, M., AND LISKOV, B. Practical byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst. 20*, 4 (Nov. 2002), 398–461.

[9] CORBETT, J. C., DEAN, J., EPSTEIN, M., FIKES, A., FROST, C., FURMAN, J. J., GHEMAWAT, S., GUBAREV, A., HEISER, C., HOCHSCHILD, P., HSIEH, W., KANTHAK, S., KOGAN, E., LI, H., LLOYD, A., MELNIK, S., MWAURA, D., NAGLE, D., QUINLAN, S., RAO, R., ROLIG, L., SAITO, Y., SZYMANIAK, M., TAYLOR, C., WANG, R., AND WOODFORD, D. Spanner: Google's globally-distributed database. In *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation* (Berkeley, CA, USA, 2012), OSDI'12, USENIX Association, pp. 251–264.

[10] CRYPTOKITTIES. Cryptokitties, 2017. https://www.cryptokitties.co.

[11] DANEZIS, G., AND MEIKLEJOHN, S. Centrally banked cryptocurrencies. In *Proceedings of the 23rd Annual Network and Distributed System Security Symposium* (2016), NDSS '16.

[12] DEMERS, A., GREENE, D., HAUSER, C., IRISH, W., LARSON, J., SHENKER, S., STURGIS, H., SWINEHART, D., AND TERRY, D. Epidemic algorithms for replicated database maintenance. In *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing* (New York, NY, USA, 1987), PODC '87, ACM, pp. 1–12.

[13] DOUCEUR, J. R. The sybil attack. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems* (London, UK, UK, 2002), IPTPS '01, Springer-Verlag, pp. 251–260.

[14] ETHEREUM. Ethereum 2.0 phase 0 – the beacon chain, 2019. https://github.com/ethereum/eth2.0-specs/blob/master/specs/core/0_beacon-chain.md.

[15] EYAL, I., GENCER, A. E., SIRER, E. G., AND VAN RENESSE, R. Bitcoin-ng: A scalable blockchain protocol. In *NSDI* (2016), pp. 45–59.

[16] FACEBOOK OPEN SOURCE. Rocksdb: A persistent key-value store, 2014. https://rocksdb.org/.

[17] GAUTHIER, P. Why do some bitcoin mining pools mine empty blocks?, 2016. https://bitcoinmagazine.com/articles/why-do-some-bitcoin-mining-pools-mine-empty-blocks-1468337739/.

[18] GILAD, Y., HEMO, R., MICALI, S., VLACHOS, G., AND ZELDOVICH, N. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles* (New York, NY, USA, 2017), SOSP '17, ACM, pp. 51–68.

[19] GUETA, G. G., ABRAHAM, I., GROSSMAN, S., MALKHI, D., PINKAS, B., REITER, M. K., SEREDINSCHI, D.-A., TAMIR, O., AND TOMESCU, A. Sbft: a scalable decentralized trust infrastructure for blockchains, 2018. https://arxiv.org/abs/1804.01626.

[20] HANKE, T., MOVAHEDI, M., AND WILLIAMS, D. Dfinity technology overview series, consensus system, 2018. https://arxiv.org/abs/1805.04548.

[21] INTEL. Developer Reference for Intel©Integrated Performance Primitives Cryptography 2018, 2018. https://software.intel.com/en-us/ipp-crypto-reference.

[22] KIAYIAS, A., RUSSELL, A., DAVID, B., AND OLIYNYKOV, R. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference* (2017), Springer, pp. 357–388.

[23] KOGIAS, E. K., JOVANOVIC, P., GASSER, L., GAILLY, N., SYTA, E., AND FORD, B. Omniledger: A secure, scale-out, decentralized ledger via sharding. In *Security and Privacy (SP), 2018 IEEE Symposium on* (2018), Ieee.

[24] KOKORIS-KOGIAS, E., JOVANOVIC, P., GAILLY, N., KHOFFI, I., GASSER, L., AND FORD, B. Enhancing bitcoin security and performance with strong consistency via collective signing. In *Proceedings of the 25th USENIX Conference on Security Symposium* (Berkeley, CA, USA, 2016), SEC'16, USENIX Association, pp. 279–296.

[25] KOTLA, R., ALVISI, L., DAHLIN, M., CLEMENT, A., AND WONG, E. Zyzzyva: Speculative byzantine fault tolerance. In *Proceedings of Twenty-first ACM SIGOPS Symposium on Operating Systems Principles* (New York, NY, USA, 2007), SOSP '07, ACM, pp. 45–58.

[26] KWON, J. Tendermint: Consensus without mining, 2014. `https://tendermint.com/static/docs/tendermint.pdf`.

[27] LAMPORT, L. How to make a multiprocessor computer that correctly executes multiprocess programs. *IEEE transactions on computers*, 9 (1979), 690–691.

[28] LEWENBERG, Y., BACHRACH, Y., SOMPOLINSKY, Y., ZOHAR, A., AND ROSENSCHEIN, J. S. Bitcoin mining pools: A cooperative game theoretic analysis. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems* (Richland, SC, 2015), AAMAS '15, pp. 919–927.

[29] LI, C., LI, P., XU, W., LONG, F., AND YAO, A. C.-C. Scaling nakamoto consensus to thousands of transactions per second. *arXiv preprint arXiv:1805.03870* (2018).

[30] LLOYD, J. Botan: Crypto and tls for c++11, 2015. `https://botan.randombit.net/`.

[31] LOEWENSTERN, A., AND NORBERG, A. Dht protocol (bep-0005), 2008. `http://www.bittorrent.org/beps/bep_0005.html`.

[32] LUU, L., NARAYANAN, V., ZHENG, C., BAWEJA, K., GILBERT, S., AND SAXENA, P. A secure sharding protocol for open blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (New York, NY, USA, 2016), CCS '16, ACM, pp. 17–30.

[33] MCFARLANE, G. Mining empty blocks is spiking on ethereum — that could be a problem, 2018. `https://ethereumworldnews.com/mining-empty-blocks-ethereum-could-be-a-problem/`.

[34] MERKLE, R. Method of providing digital signatures, 1979. `https://patents.google.com/patent/US4309569`.

[35] MILLER, A., XIA, Y., CROMAN, K., SHI, E., AND SONG, D. The honey badger of bft protocols. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (New York, NY, USA, 2016), CCS '16, ACM, pp. 31–42.

[36] MOHAN, C., LINDSAY, B., AND OBERMARCK, R. Transaction management in the r* distributed database management system. *ACM Trans. Database Syst. 11*, 4 (Dec. 1986), 378–396.

[37] NAKAMOTO, S. Bitcoin: A peer-to-peer electronic cash system, 2008. `http://bitcoin.org/bitcoin.pdf`.

[38] PASS, R., AND SHI, E. Hybrid consensus: Efficient consensus in the permissionless model. In *LIPIcs-Leibniz International Proceedings in Informatics* (2017), vol. 91, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[39] RAY, J. Sharding faqs, ethereum wiki, 2019. `https://github.com/ethereum/wiki/wiki/Sharding-FAQs`.

[40] SHI, R., AND WANG, Y. Cheap and available state machine replication. In *Proceedings of the 2016 USENIX Conference on Usenix Annual Technical Conference* (Berkeley, CA, USA, 2016), USENIX ATC '16, USENIX Association, pp. 265–279.

[41] SOMPOLINSKY, Y., LEWENBERG, Y., AND ZOHAR, A. Spectre: Serialization of proof-of-work events: Confirming transactions via recursive elections, 2016. `https://eprint.iacr.org/2016/1159.pdf`.

[42] SOMPOLINSKY, Y., AND ZOHAR, A. Secure high-rate transaction processing in bitcoin. In *International Conference on Financial Cryptography and Data Security* (2015), Springer, pp. 507–527.

[43] SOMPOLINSKY, Y., AND ZOHAR, A. Phantom, ghostdag: Two scalable blockdag protocols, 2018. https://eprint.iacr.org/2018/104.pdf.

[44] STATEOFTHEDAPPS.COM. Dapp statistics, 2019. https://www.stateofthedapps.com/stats.

[45] SYTA, E., JOVANOVIC, P., KOGIAS, E. K., GAILLY, N., GASSER, L., KHOFFI, I., FISCHER, M. J., AND FORD, B. Scalable bias-resistant distributed randomness. In *Security and Privacy (SP), 2017 IEEE Symposium on* (2017), Ieee, pp. 444–460.

[46] VISA. Visa acceptance for retailers, 2018. https://usa.visa.com/run-your-business/small-business-tools/retail.html.

[47] ZAMANI, M., MOVAHEDI, M., AND RAYKOVA, M. Rapidchain: Scaling blockchain via full sharding. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (New York, NY, USA, 2018), CCS '18, ACM, pp. 931–948.

Appendix

A Programmable Transaction

In Section 3, our discussions are based on a withdraw-deposit paradigm that serves well for payment-centric applications such as Bitcoin. However, it is desirable to have a transaction logic that is more complex and goes beyond payment-centric applications. To this end, the proposed system is required to correctly handle (generalized) relay transactions and guarantee eventual atomicity. We extend the withdraw-deposit paradigm to a more general model with programmable transactions logic.

A.1 World State

We extend per-user balance to per-user state with user-customizable data types and structures. The world state Ω only contains per-user states:

$$\Omega := \left\{ \psi_{\mu_i} \right\}, \tag{7}$$

where μ_i donates a user identified by its address. Each per-user state $\psi_\star$ can be simple integers (that describe account balance) or be arbitrarily complex data structures like strings, lists, sets or maps.

A.2 Transaction

We extend the fixed logic of withdraw and deposit to programmable logic with a few restrictions as follows. A transaction φ is an atomic updates over the state of users. It modifies states of one or multiple users with a condition ρ:

$$\varphi := \left\langle \rho\left(\psi_{\mu_c}\right), \left\{ \psi_{\mu_i} \leftarrow \phi_i(\psi_{\mu_i}; \psi_{\mu_c}) \right\}, \kappa \right\rangle, \tag{8}$$

where κ is the argument of the transaction available to condition and operation logic, ρ denotes the condition for validating the transaction (a binary function of state ψ_{μ_c}). $\{\phi_i\}$ contains operations that modify states of users $\{\psi_{\mu_i}\}$ if ρ is true. Note that we restrict the condition ρ to only be related to a single user μ_c, so that a transaction can be validated solely in the user μ_c's zone. If the condition ρ is true, all operations in $\{\phi_i\}$ are required to be executed successfully without failure or throwing exceptions. Each operation ϕ_i may have arbitrarily complex logic but restricted access pattern. Specifically, ϕ_i updates the state ψ_{μ_i} only with access to the previous state of ψ_{μ_i} and readonly access to ψ_{μ_c} which is involved in the condition ρ (and nothing else). Modifying ψ_{μ_c} is allowed if $\mu_c \in \{\mu_i\}$, which is a frequent case.

A.3 Operation

When a transaction is validated (i.e., ρ is true), operations $\{\phi_i\}$ modifying cross-zone users' states will be carried by relay transactions and executed in an asynchronous manner. As described in Section 4, relay transactions from different originate transactions might be interleaved and disordered during execution. To ensures a consistent end result without requiring serialization, we restrict all operations $\{\phi_i\}$ to be order-independent with any arguments κ_a and κ_b:

$$\forall \kappa_a, \kappa_b : \phi_i\left(\phi_i\left(\psi_i; \kappa_a\right); \kappa_b\right) \equiv \phi_i\left(\phi_i\left(\psi_i; \kappa_b\right); \kappa_a\right). \tag{9}$$

A.4 Smart Asset

The pre-user state ψ_{μ_i} in our system includes the balance of the native fungible token and a dictionary of states of the *smart asset* defined by all 3rd-parties.

$$\psi_{\mu_i} := \left\langle \gamma, \{ \upsilon \rightarrow \hat{\psi}_{\mu_i} \} \right\rangle \tag{10}$$

Native fungible token is the platform currency like Ether in Ethereum, which is consumed in the transaction handling as gas fee. It is issued as the coinbase reward when a miner find a new block. Smart asset are conceptually borrowed from smart contract in Ethereum.

The definition of a smart asset for standard tokens includes an issuing transaction $\breve{\varphi}$ and a payment transaction φ as in the equation (8). Issuing transaction $\breve{\varphi}$ is

designed for extending the coinbase logic for rewarding miners. It defines the 3rd-party mineable token. More types of transaction can be defined and invoked when issue a transaction.

Fungible Token

The state of a fungible token is a signed big integer representing the balance $\hat{\psi}_\mu := \beta_\mu$. An example of bitcoin-like issuing transaction can be:

$$\beta_{\mu_m} \leftarrow \beta_{\mu_m} + \left\lfloor \frac{50/n}{2^{\lfloor h_b/210000 \rfloor}} \right\rfloor, \quad h_b > 0 \tag{11}$$

, in which n is the number of zones, μ_m denotes the miner and h_b is the *asset block height*, the number of block appended since the block for deploying the specific smart asset.

Issuing transaction will also be invoked when it is deployed with h_b. For a pre-allocated 1 billion token, it can be:

$$\beta_{\mu_m} \leftarrow \lfloor 1000000000/n \rfloor, \quad h_b = 0 \tag{12}$$

Usually issuing transaction contains one unconditional operation and can be extended to multiple ones (e.g. given additional reward to certain users).

Payment transaction from μ_a to μ_b consists of a condition ρ and multiple operations:

$$\begin{aligned}
\rho: &\quad (\kappa_v \le \beta_{\mu_a}) \wedge (\kappa_e \le \gamma_{\mu_a}) && (13)\\
\{\phi_i\}: &\quad \beta_{\mu_a} \leftarrow \beta_{\mu_a} - \kappa_v && (14)\\
&\quad \gamma_{\mu_a} \leftarrow \gamma_{\mu_a} - \kappa_e && (15)\\
&\quad \beta_{\mu_b} \leftarrow \beta_{\mu_b} + \kappa_v, && (16)\\
&\quad \gamma_{\mu_m} \leftarrow \gamma_{\mu_m} + \kappa_e && (17)
\end{aligned}$$

, in which κ_v is a positive big integer indicating the amount of the transfer and κ_e is a non-negative big integer denoting the transaction fee charged from the platform currency.

Non-Fungible Token

The state of a non-fungible token is an set, $\hat{\psi}_\mu := \{\tau_i\}$ indicates the ownership of each non-fungible token. Non-fungible tokens usually don't need any issuing transaction, which can not be mined.

Payment transaction from μ_a to μ_b consists of a condition ρ and multiple operations:

$$\begin{aligned}
\rho: &\quad (\kappa_\tau \in \hat{\psi}_{\mu_a}) \wedge (\kappa_e \le \gamma_{\mu_a}) && (18)\\
\{\phi_i\}: &\quad \hat{\psi}_{\mu_a} \leftarrow \hat{\psi}_{\mu_a} - \{\kappa_\tau\} && (19)\\
&\quad \gamma_{\mu_a} \leftarrow \gamma_{\mu_a} - \kappa_e && (20)\\
&\quad \hat{\psi}_{\mu_b} \leftarrow \hat{\psi}_{\mu_b} \cup \{\kappa_\tau\} && (21)\\
&\quad \gamma_{\mu_m} \leftarrow \gamma_{\mu_m} + \kappa_e && (22)
\end{aligned}$$

, in which κ_τ is the non-fungible token to be transferred.

A customized transaction can be defined and allow invocation only by a hard-coded issuer μ_u (e.g., the game developer [10]). The example issues and releases a non-fungible token κ_τ to a specific user κ_μ.

$$\begin{aligned}
\rho: &\quad (\mu_a = \mu_u) \wedge (\kappa_e \le \gamma_{\mu_u}) && (23)\\
\{\phi_i\}: &\quad \gamma_{\mu_u} \leftarrow \gamma_{\mu_u} - \kappa_e && (24)\\
&\quad \hat{\psi}_\mu \leftarrow \hat{\psi}_\mu \cup \{\kappa_\tau\} && (25)\\
&\quad \gamma_{\mu_m} \leftarrow \gamma_{\mu_m} + \kappa_e && (26)
\end{aligned}$$