W3Chain: A Layer2 Blockchain Defeating the Scalability Trilemma

Miaoyong Xu*, Qing Wang*, Haohan Sun*, Jianru Lin†, Huawei Huang†*
*School of Computer Science and Engineering, Sun Yat-Sen University, China.
†School of Software Engineering, Sun Yat-Sen University, China.
Corresponding author: Huawei Huang. Email: huanghw28@mail.sysu.edu.cn

***Abstract*—Scalability trilemma is a classical research topic in the area of blockchains. To defeat such trilemma, many previous solutions have been proposed. However, none of those previous solutions can break such scalability trilemma. In this paper, we present a new Layer2 blockchain called W3Chain, which is promising to deliver high *transactions per second* (TPS) while defeating the scalability trilemma of a public blockchain. To enable the claimed performance, we particularly design our W3Chain by decoupling the correctness of the blockchain into two parts and adopting several crucial technical issues such as the reconfiguration of committees, the design of query APIs, and the handling of cross-shard transactions. We also propose a Time-Beacon Chain (TBChain) to record pivotal data of W3Chain. To show the correctness and safety features, we rigorously analyze multiple properties of W3Chain, including decentralization, scalability, and security under typical attacks. Finally, we conduct extensive simulations using Ethereum's historical transactions to examine the proposed W3Chain. The evaluation results show that our W3Chain can achieve a TPS as high as 10K+, and much lower transaction confirmation latency compared with Ethereum.**

***Index Terms*—Scalability, Blockchain, Sharding**

I. Introduction

Blockchain's scalability trilemma is a classic research question faced by researchers in the area of parallel and distributed computing. Various solutions have been proposed to improve the three most important metrics, i.e., scalability, security, and decentralization. However, due to the well-known scalability trilemma [1], at most two of the three metrics can be satisfied in any settings of a blockchain system, simultaneously.

In this paper, we revisit such a topic in the context of sharding blockchains. Particularly, we try to defeat the scalability trilemma of a blockchain by proposing a new solution named *W3Chain*. Technically, such W3Chain is a reliable public blockchain with high throughput. We present our design of the proposed W3Chain systematically. Basically, the proposed W3Chain works on top of a *Layer1* blockchain. In our design, W3Chain consists of several committees and blockchain shards. In each committee, blockchain nodes execute transactions (TXs) using intra-committee consensus. Each blockchain shard is exploited to store states of blockchain accounts and historical blocks. Thus, multiple parallel committees and shards can largely improve the *transaction per second* (TPS) of the holistic blockchain.

To enable the functionality of the proposed W3Chain, we also propose and implement several crucial technical solutions, including the *reconfiguration of committees*, the implementation of *query APIs*, and the storage of chain data. To support the logical correctness of transactions submitted by users, our W3Chain also needs to handle the cross-shard TXs while guaranteeing the *atomicity* [2] of cross-shard TXs.

In addition, to guarantee the security of W3Chain, we also implement a time-beacon chain (shortened as *TBChain*) by invoking a trusted Layer1 blockchain. Such a TBChain records the pivotal data of W3Chain, including the state trie root of W3Chain, the hash root of the transaction trie, and the hash values of W3Chain's blocks.

Our work presented in this paper includes the following contributions.

- We propose W3Chain, which is promising to defeat the scalability trilemma of a public blockchain.
- To prove the correctness of W3Chain, we then rigorously analyze its properties such as decentralization, scalability, and security under typical attacks.
- Finally, we evaluate the proposed W3Chain using the real-world Ethereum's historical TXs. The evaluation results show that W3Chain can achieve a high TPS, and low transaction's confirmation latency.

The remaining of this paper is organized as follows. Section II introduces some preliminaries. Section III describes the protocol design. Section IV depicts the handling of cross-shard TXs. Section V analyzes the security issues and other properties of W3Chain. Section VI demonstrates the performance simulation results. Section VII reviews state-of-the-art studies. Finally, Section VIII concludes this paper.

II. Preliminaries

A. Scalability Trilemma

Blockchain is essentially a technology that ensures the correctness of transaction data without the endorsement of a centralized institution. In a blockchain system, a transaction will be first proposed by a client, then verified, executed, and committed (i.e., the transaction, along with other transactions, is packed into an on-chain block) by the majority of nodes in the entire network, which reaches an agreement on the transaction data through a consensus algorithm, and jointly endorse the correctness of the transaction data. The form of endorsement varies in different consensus protocols, e.g., PoW [3] uses cumulative workload, and PoS [4] uses the amount of

staked tokens. The **security** of the blockchain refers to the joint endorsement strength of honest nodes in the entire system. The more honest nodes participating in the consensus, the higher the security of the blockchain. The **decentralization** of the blockchain means that each node participating in the consensus can independently verify and execute transactions without relying on other nodes. A blockchain with good decentralization should have low resource requirements for nodes, otherwise will prevent normal users from entering, making itself actually controlled by centralized institutions. The **scalability** of the blockchain refers to the fact that the system can process more transactions than a single decentralized node can handle. By increasing the number of nodes, the transaction processing capability of the system can be improved. Existing blockchains cannot satisfy security, decentralization, and scalability at the same time, and increasing scalability will reduce the decentralization or security of the blockchain, or both. Vitalik Butterin calls this *the Scalability Trilemma* [1].

B. Merkle Patricia Trie

Merkle Patricia Trie (MPT) is a combination of Merkle tree and prefix trie, first proposed in Ethereum [5] to store key-value pairs. Similar to the prefix trie, keys are segmented and stored on the path from the root node to the leaf nodes. In order to improve the storage and operation efficiency, MPT optimizes the prefix trie and merges those consecutive branch nodes which have no forks.

MPT also has the characteristics of a merkle tree, where each node stores the hash of its child nodes. Merkle proof is a method to quickly verify whether a key-value pair is included in an MPT. A merkle proof contains the node information on the path from the leaf node where the key-value pair is located to the root node. If the hash obtained by hashing the nodes in the merkle proof from bottom to top is the same as the known hash of the MPT root, it means that the key-value pair does exist in the MPT. It only takes O(log_n) complexity when verifying the inclusion of a key-value pair in a MPT, where n represents the total number of pairs.

As a data structure, MPT can be used to organize all accounts on a blockchain. The address, balance and other information of each account will be stored on a MPT object as a leaf node. Such an MPT is generally called a *state trie*. Similarly, the MPT that organizes the transactions packed in a block is called a *transaction trie*.

C. Verifiable Random Function

In cryptography, a verifiable random function (VRF) is a public-key pseudorandom function that provides a proof that its output was calculated correctly, first proposed by Micali et al. [6]. Given a determined input value, the owner of a secret key can compute the function value and provide the associated proof. Others can check that this output value was indeed calculated correctly using the proof and the owner's public key, yet the secret key remains unknown.

D. Sharding and Reconfiguration

The main idea of sharding is to divide the nodes in the network into different parts, each part is called a committee (or a shard) and processes a part of the transactions in the system. State sharding is the most thorough and difficult to implement of all sharding solutions, which partitions the states of all accounts and historical transaction data into different committees.

Since different committees process different groups of transactions in parallel, the scalability of sharding blockchains can be greatly improved in theory. In a well-designed sharding system, the improvement of transaction throughput will increase as the number of shards increases.

However, since each committee only contains a part of the entire network, malicious nodes can concentrate on attacking a certain committee. Compared with attacking the entire network, attacking a single committee requires less resources , thus more likely to succeed. Some sharded blockchains [7]–[10]employ periodic committee reconfigurations to avoid attacks on a single committee. Committee reconfiguration refers to reshuffling nodes into different committees, which includes the generation of global random numbers, node discovery, and each node synchronizing the data of its new committee. Among them, it will take a long time for each node to synchronize the data of the new committee, making frequent reconfiguration difficult to achieve.

III. System Design

A. Decoupling Blockchain Correctness

As mentioned in Section II, the security of the blockchain means that there are enough nodes to endorse the correctness of the transaction data. In W3Chain, we partition the correctness of transaction data into two parts, with the aim of separating the endorsements of the two parts of correctness. We partition the correctness of transaction data into **block correctness** and **chain consistency**. The block correctness refers to the correctness of the internal data of a block. For each transfer transaction in the block, it is necessary to verify that the payer has signed, the payer's balance is sufficient, and there is no double-spending transaction. Chain consistency means that there is no double-spending attack caused by chain forks. The verification of the block correctness only needs to know the account state related to the current block, while the verification of the chain consistency needs to know the information of the historical block.

Existing blockchains [8], [9] that implement state sharding do not have the correctness of blockchain partitioned. Subject to this, each committee needs to store all the data of the shard, which makes the reconfiguration cost of the committees high and non-negligible.

In W3Chain, we partition the correctness of each shard chain, and move the responsibility of chain consistency guarantee to the chain in Layer1, so that the committee only need to guarantee the block correctness. As a result, the committee of each shard does not need to store the full data of its shard,

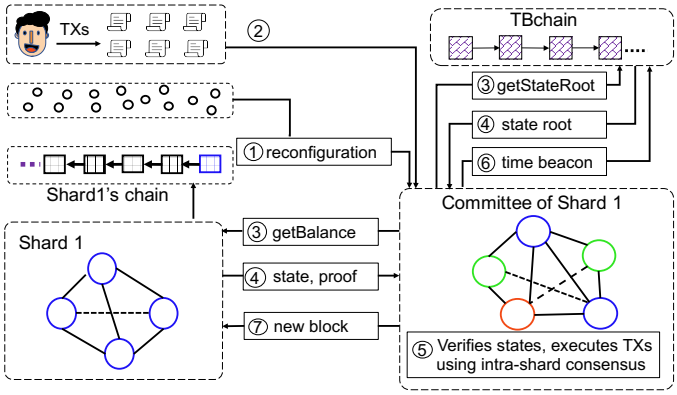

Fig. 1. Overview of the proposed W3Chain.

and can achieve fast reconfiguration. We leave the details of the analysis to Section V, and first explain the system design.

B. Critical Roles of our System

In order to explain our system more clearly and avoid confusion for the readers, we first explain several key roles in our system.

Shards And Committees: W3Chain adopts state sharding, which divides the nodes in the network into different groups, and each group is called a shard. The mapping of node ID to shard ID is fixed, and each node only stores and maintains the state of the shard to which it belongs. However, unlike other state sharding blockchains [8], [9], [11], it is not the duty of the nodes that make up the shard to generate blocks (as it will possibly result in a single shard being less secure). Instead, nodes are randomly selected from the entire network to form committees, and each committee is responsible for the generation of blocks in a shard. The committee does not store the historical state of the corresponding shard, and only needs to obtain a small part of the latest state when processing transactions, which can be obtained by initiating a query to the shard. The mapping of node ID to committee ID is dynamic, i.e., W3Chain reshuffles committees at every block height using verifiable random function, making the mapping of node to committee unpredictable.

Time Beacon Chain: We call the Layer1 blockchain running in parallel with W3Chain the Time Beacon Chain (TBChain), on which deploys a W3Chain-related smart contract. When a committee in W3Chain produces a block, the smart contract is called and a tuple containing the most basic information of the block is stored.

Every node on W3Chain can query the contract for the tuple corresponding to any height of any shard, and we call such a tuple a time beacon.

Client: The main function of a client is to broadcast the user's transactions to the network and return the system's processing results to the user. A client can also be a node participating in the consensus at the same time.

C. System Overview

We define a fixed period of the running time of the system as an epoch, and each epoch is divided into several smaller units called slots. Fig.1 shows the execution process of a shard and related roles in a slot. The roles that appear in Fig.1 are Client, Shard 1, Committee 1 corresponding to Shard 1, and TBChain. We break it down into four stages:

Reconfiguration And Broadcast Stage: In the initial stage of a slot, nodes of the entire network are re-divided into different committees through a verifiable random function. Then clients broadcast transactions to the whole network, among which Committee 1 filters out the transactions of its corresponding shard.

Query Stage: Committee 1 requests the data needed to verify the current transactions from Shard 1 and TBChain through API:`getBalance` and API:`getStateRoot` respectively. Specifically, Committee 1 requests Shard 1 for the latest state of the accounts related to the current processing transactions, and requests the root of Shard 1's latest state trie from TBChain. When Shard 1 receives the request, it obtains the corresponding account states from the state trie stored locally, each together with a merkle proof that proves the legitimacy of the state, and sends them to Committee 1. At the same time, TBChain sends the corresponding state trie root to Committee 1.

Verification And Execution Stage: When Committee 1 receives the account states provided by Shard 1, it verifies the correctness of every state in combination with the merkle proof attached to the state and the state trie root returned by TBChain. At the same time, Committee 1 verifies the signature of each transaction. After verification, Committee 1 executes the transactions and packs them into a block.

Storage Stage: When a block is successfully generated through the internal consensus of the committee, Committee 1 sends the block to Shard 1 for committing to the chain, and records a time beacon to the smart contract on TBChain. The time beacon contains four pieces of information: the block producer, the block hash, the new state trie root, and the transaction trie root of the block.

In the following, we elaborate two critical parts: Reconfiguration of Committees in W3Chain and Query APIs.

D. Reconfigurations of Committees in W3Chain

For security, it should be ensured that when doing committee reconfigurations, the mapping of node ID to committee ID cannot be predicted in advance. A feasible solution is to apply the VRF.

At the beginning of a slot, each node takes the hash of the latest block on TBChain as input, and get its output by running VRF. Then it maps the output to an integer with a mapping function which is unified of the whole network, and finally, it let the integer modulo the total number of shards, which returns a integer denoting the shard to which the node belongs in current slot.

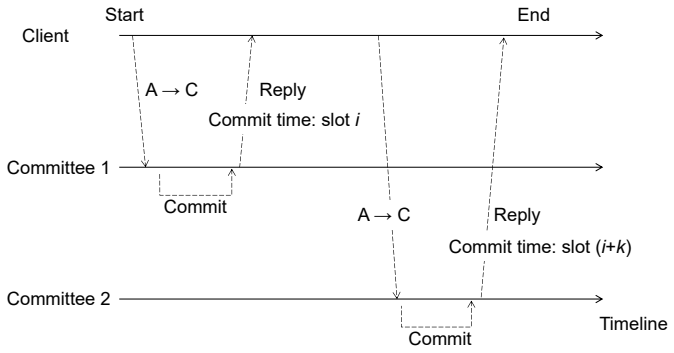

Fig. 2. Successful execution process of a cross-shard transaction.

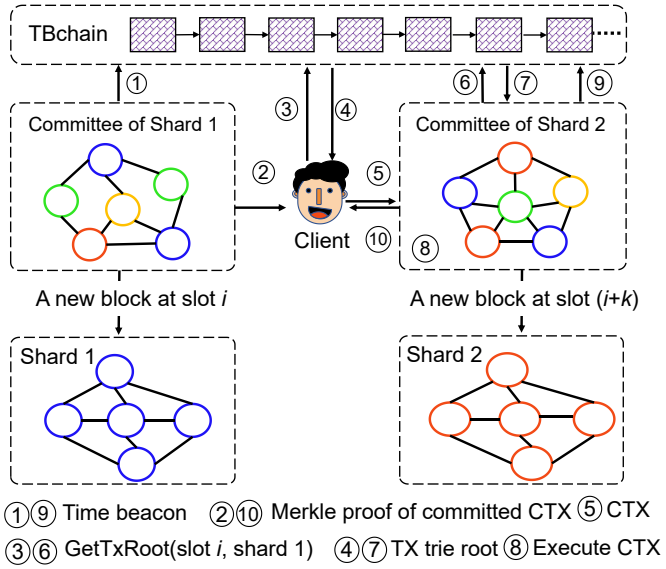

Fig. 3. Verification of cross-shard transactions.

E. Query APIs

While a committee is processing the transactions of a shard, a node in the committee may have the state of the shard (it happens to be the node that constitutes the shard), and can directly obtain the state of the relevant accounts from the local. Otherwise, the states need to be obtained from the shard through the network. We encapsulate the acquisition of an account state in these two cases into an API `getBalance(accountAddress)`, of which the parameter is the requested account address. By calling this API, A node can obtain the latest state of any account in the entire network and the corresponding merkle proof. In the same way, we also encapsulate the acquisition of information stored on TBChain into APIs: `getStateRoot(shardID, height)`, `getTxRoot(shardID, height)`, and `getBlockHash(shardID, height)`, where `shardID` indicates which shard is requested, and `height` indicates the height of the requested block. If a node has cached the information, it can be directly read from the local, otherwise, it is obtained from TBChain.

IV. Handling Cross-shard TXs

When the sending and receiving accounts of a transaction are not in the same shard, it is called a cross-shard transaction, and needs to be verified and executed in the sender's shard and the receiver's shard respectively. Cross-shard transactions are common in sharding blockchains. In this section, we introduce the cross-shard transaction processing mechanism of W3Chain, including the process of successful transaction execution and the rollback method when transaction execution fails.

A. Process of Successful Execution

Assuming that account A is stored in shard 1, and account C is stored in shard 2, then the successful execution process of a cross-shard transaction (denoted as *CTX*) for a transfer from A to C is shown in Fig. 2.

A *CTX* is first initiated by a client, and goes through the network to committee 1. Committee 1 verifies the *CTX*, executes the deduction operation from A. After committing the *CTX* on chain, it replies to the client. Then the client broadcasts the *CTX* again, which reaches committee 2 through the network. Committee 2 executes the deposit operation to C, commits the *CTX* and replies to the client. Finally, The client submits a success message to the user, and the process ends.

Multi-step verification is involved in this process for security, and TBChain is used to assist verification, as shown in Fig. 3. The first is the verification of the *CTX* by committee 1. This process is the same as the normal intra-shard transactions, so it is not shown in Fig.3. We start directly with the *CTX* verified and executed at committee 1.

- When this *CTX* is executed and committed by committee 1, committee 1 stores the new block in shard 1, stores the block's beacon on TBChain, and then replies the client a message with merkle proof that the *CTX* is on the transaction trie.
- When the client receives the reply, it queries TBChain for the transaction trie root of the corresponding shard and block height to confirm that the *CTX* has been committed by committee 1. After that, it broadcast the *CTX* to the whole network again, along with the merkle proof.
- Committee 2 receives the *CTX* and verifies it with the transaction trie root on TBChain. After the verification, committee 2 executes the *CTX*, saves the time beacon on the TBChain, and replies the proof to the client.
- When the client receives the reply from committee 2, it verifies the proof (again with the aid of TBChain). Assume that the *CTX* is committed in committee 1 in slot i, and is committed in committee 2 in slot $(i+k)$. If k is less than a predefined integer, then the client will consider the transaction to be processed timely and returns a success message to the user.

B. Fail and Roll back

The atomicity of a transaction, in layman's terms, means that the two operations of the sender's deduction and the

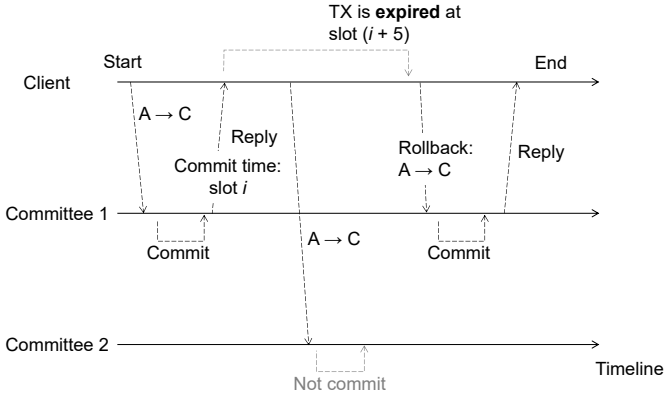

Fig. 4. Handling a cross-shard transaction under the *Fail and Roll back* case.

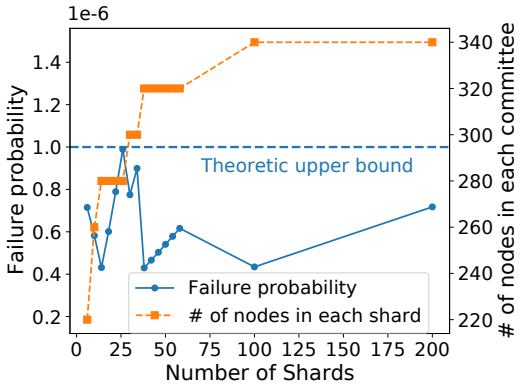

Fig. 5. The probability that at least one committee fails.

receiver's deposit must be bound to each other, either both operations are executed, or neither. In the example of cross-shard transaction execution in Section IV-A, if committee 1 has executed the *CTX* (i.e., deducts money from account A), whereas committee 2 does not execute the *CTX* in the end (i.e., deposit to account C), the atomicity of the transaction will be destroyed.

Monoxide [11] proposes *eventual atomicity*, which is based on an optimistic assumption, i.e., even if committee 2 does not execute the *CTX* in a short period of time, by re-sending relay transactions or using other measures, the *CTX* can be finally executed by committee 2. Compared with the eventual atomicity of Monoxide, we make the transaction atomicity more complete by introducing a rollback mechanism for cross-shard transactions.

In W3Chain, when a client sends the *CTX* to committee 2, if the client still does not receive a reply from committee 2 after waiting for a specified number of slots, e.g., 5 slots, this transaction is considered expired. Then, a rollback transaction is sent to committee 1, as shown in Fig. 4.

A rollback transaction is a special transaction. Committee 1 needs to verify that the *CTX* is not committed in committee 2. Committee 1 issues a query `getTxNotExistProof (ctx)` to shard 2. When shard 2 receives the query, it returns the Merkle proof that the *CTX* is not committed in any of the blocks from block (i + 1) to block (i + 5), or otherwise, the Merkle proof that the *CTX* is committed in one of these blocks.

If the *CTX* is indeed not committed on shard 2, committee 1 will execute the rollback transaction with a higher priority than normal transactions, and send the merkle proof of the rollback transaction committed to the client. Otherwise, committee 1 will reply with a message to the client showing that the rollback transaction failed, with the proof provided by shard 2.

To prove that a transaction is not included in a block, using the characteristics of MPT, shard 2 only needs to prove that the leaf node corresponding to the key of the transaction in the transaction trie does not exist (i.e., the value of the leaf node is `null`), or the key corresponding to an existed leaf node has the same prefix of the transaction's key. Each of these two proofs can be expressed as Merkle proof. This manner is called *proof of exclusion* in [12].

V. Property Analysis

A. Security

By using VRF in each slot to reconfigure the committees, we keep the security of each committee at a high level. To measure this security precisely, we calculate the probability that no committee in the system fails under a certain configuration.

Assuming that the maximum ratio of malicious nodes that a committee can tolerate is r, the ratio of malicious nodes to all nodes is α, the number of shards is S (i.e., the number of committees is S), and the size of each shard is n. Then according to [13], [14], the probability that at least one committee fails is

$$p = 1 - \frac{\left[x^{\alpha n S}\right]\left(\sum_{i=0}^{rn-1}\binom{n}{i}x^i\right)^S}{\binom{nS}{\alpha nS}} \tag{1}$$

We set r to 1/3 and α to 20%, and obtained the change of p under different S and corresponding n, as shown in Fig.5. We consider each committee to be safe when the probability p of at least one committee failure is less than 1e-6. It can be seen from Fig. 5 that as the number of shards increases, the security of each committee can be maintained at a high level by only slightly increasing the number of nodes in it. In addition, when α increases, the security of each committee can also be guaranteed by increasing the number of nodes in it.

Since each committee is sufficiently secure in each slot, the block correctness (see definition in Section III-A) of each shard chain can be guaranteed. In other words, under the premise that the data obtained by a committee from any shard and the data obtained from TBChain are correct, the committee must be able to execute and pack transactions through the correct consensus process.

Then we choose a public chain with high reliability (e.g., Ethereum [5]) as the TBChain, by which the correctness of TBChain data can be guaranteed. By storing time beacons,

TBChain records the block information in each slot of each shard in W3Chain. When a committee obtains the account states and the corresponding merkle proofs from a shard, it will verify them through the state trie root recorded on TBChain. Only when the verification is correct can the legitimacy of the account states returned by the shard be proved. Similarly, nodes in the committee will verify that the block is committed by querying the block hash stored on TBChain, and verify that the transaction is committed through the transaction trie root.

Through the auxiliary verification of TBChain, the correctness of the data obtained from the shards can be ensured. Malicious shards cannot deceive the committee by falsifying the state, nor can they perform double-spending attacks through chain forks. That is to say, the chain consistency (see definition in Section III-A) of each shard chain is guaranteed.

To sum up, the security of each shard chain of W3Chain is jointly guaranteed by the overall security of W3Chain and the security of TBChain. When a highly reliable public chain (e.g., Ethereum) is selected as TBChain, the security of each shard chain is comparable to the security of W3Chain as a whole.

B. Decentralization

Monoxide [11] ensures that the computing power of each shard is equivalent to the computing power of the entire network through Chu-ko-nu mining, thereby ensuring that the security of each shard chain is equivalent to the entire network. However, decentralization is compromised as each node needs to store, verify, and execute transactions for all shards.

In W3Chain, the nodes in a shard only need to store data (e.g., state trie, blocks) of the shard, while the committees are stateless, i.e., the nodes in a committee only need to query the corresponding shard to obtain the latest account states required to process the current transactions. Therefore, assuming that the number of shards is n, and the computational complexity of verifying and executing a block is $O(c)$, then within a slot, the computational complexity of each node in W3Chain and Monoxide is $O(c)$ and $O(nc)$, respectively. In terms of storage, assuming that storing a shard of data requires $O(s)$ capacity, the storage overhead of each node in W3Chain and Monoxide is $O(s)$ and $O(ns)$, respectively. W3Chain has some more query overhead than Monoxide, i.e., in every slot, a committee queries the shard for the partial state of the shard, and queries TBChain for the time beacon. The overhead of this part of the query is about $O(c+1)$, so the computational complexity of each node of W3Chain is still $O(c)$.

To sum up, compared with Monoxide, the node computing overhead and storage overhead of W3Chain do not increase linearly with the number of shards, meaning that decentralization is indeed achieved.

C. Scalability

W3Chain implements state sharding. On one hand, from Section V-A, we can see that in the existing design, the security of a single shard chain will not decrease as the number of shards increases. Besides, Section V-B shows that the overload of each node in W3Chain will not increase as the number of shards increases. Therefore, w3 can improve system throughput by increasing the number of shards without compromising security and scalability, breaking the Scalability Trilemma.

D. Defense Against the Sybil Attack

This subsection introduces W3Chain's defense against the Sybil attack [15]. If a certain threshold is not set when selecting nodes to form a committee, e.g., proof of work, malicious nodes can easily launch a Sybil attack. In a Sybil attack, a malicious node can pretend to be multiple nodes by creating multiple accounts, in an aim to increase their probability of being elected to the committee. In order to prevent the Sybil attack, we have introduced a *Proof of Balance* mechanism, in which any node participating in the committee election needs to prove that its balance is greater than a certain threshold. A node that meets the condition only needs to provide the merkle proof corresponding to its account state.

VI. System Simulation and Experiment

A. Settings

To evaluate the performance of W3Chain, we implement a simulator using Go. The simulator implements blockchain sharding and the cross-shard transaction mechanism proposed in this paper. We collected Ethereum's historical blocks, which include 1,536,000 token-transfer transactions occurred from June 7, 2022 to June 14, 2022. We then replay those transactions on our simulator. Specifically, we first read all the accounts associated to these transactions, and divide them into different shards according to the last few bits of their account addresses. Then, we inject a certain number of transactions per second into the simulator. The simulator allocates these transactions to different committees according to the sending account's address. When the transaction injection is over, we read the corresponding transaction execution results from simulator's log files.

B. Throughput and Latency

In order to set reasonable parameters for both block size and block interval, we investigated the settings of several classic sharding systems, as shown in Table I. Accordingly, we set the block generation interval of each shard chain in our simulator to 5 seconds, the block capacity to 500 transactions (shortened as TX), and thus the maximum supported throughput of each shard is 100 TX/s. We then execute our simulator by varying the number of shards (i.e., committees). The results of throughput and latency are shown in Fig 6 and Fig 7, respectively.

Throughput is measured by *transactions per second* (TPS), which can be obversed from Fig 6. We see that as the number of shards increases, the throughput also increases linearly. When the number of shards is 256, the throughput reaches 10948, and the average throughput of each shard chain is 42 TX/s. This TPS is less than a half of the expected maximum throughput. Reasons are two folded. First, as the number of

TABLE I
THE SETTINGS OF SEVERAL CLASSIC SHARDING SYSTEMS.

System	Monoxide	RapidChain	OmniLedger[1]
TPS per shard	15∼20	300∼400	400∼500
Block interval	15s	<10s	5∼10s

[1] The tolerance for malicious nodes is 12.5%.

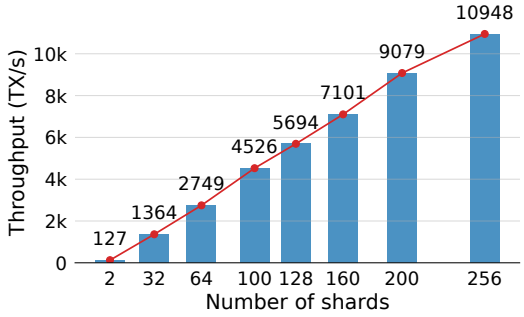

Fig. 6. Linearly increasing throughput versus the number of shards.

shards increases, the proportion of cross-shard transactions increases dramatically [11]. A cross-shard transaction needs to be committed in two shards, thus occupying twice of block space of an intra-shard transaction. Second, as the number of transactions initiated by different accounts varies, shards with many inactive accounts may not have enough transactions to be executed.

Fig 7 shows the distribution of transaction latency versus different numbers of shards. Compared with the real situation, we made the following simplifications in the simulator:

- We ignore the network propagation time of the transaction from the client to the committees.
- We assume that when the client is processing a cross-shard transaction, it will respond immediately as long as it receives a reply from the source committee, and immediately send the transaction to the target committee when verification is correct (see the cross-shard transaction processing process in IV). In addition, once the client finds that a cross-shard transaction has expired (i.e., the transaction is not committed on the target shard timely), it will immediately send a rollback transaction to the source committee.
- We assume that TBChain stores each beacon of W3Chain and replies to query requests in time.

Ideally, the delay between a client and a committee is milliseconds, and the client and TBChain will respond in a timely manner. In this case, the actual transaction latency will not be much higher than our simulation results.

We also calculated the average transaction latency, which has the same observations as shown in Fig. 7. When the number of shards increases from 2 to 32, the average latency increases. That is because that the proportion of cross-shard transactions increases. Cross-shard transactions have higher

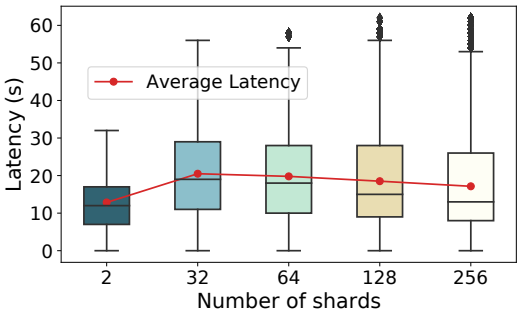

Fig. 7. Distribution of transaction confirmation latency under different numbers of shards.

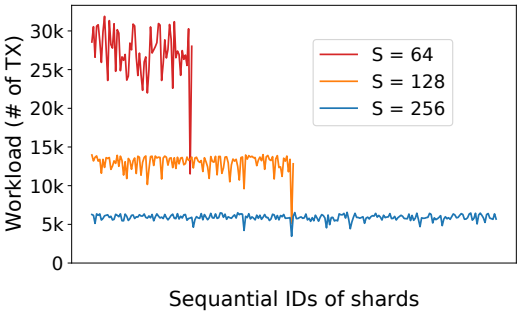

Fig. 8. Distribution of transaction workloads across shards.

average latency than intra-shard ones. When the number of shards keeps larger than 32, the average delay decreases slowly. The possible reason is described as follows. When the number of shards increase, the load of each shard (see Fig 8) tends to become more balanced, and thus transactions' queueing delay becomes shorter accordingly.

C. Workload

We inject the same batch of transactions into our simulator while varying the number of shards. We then measure the workload of each shard, i.e., the number of committed transactions (for each cross-shard transaction committed, the workloads of the source shard and the target shard both increase by 1), as shown in Fig 8. When the number of shards increases, the workload of each shard is observed decreased. This means that the workload of each node composing the shards and corresponding committees also decreases. Monoxide [11] requires each full node to process transactions of multiple shards at the same time, so even if the workload of each shard decreases as the number of shards increases, the workload of each node that makes up the shards will not decrease. Compared with Monoxide's result, W3Chain has achieved the property of decentralization.

D. Proportion of Rollback Transactions

Fig 9 shows the proportion of transactions that were rolled back because of failing to commit timely in the target shard. We conducted this group of experiments under different numbers of shards and different expiration time threshold (denoted

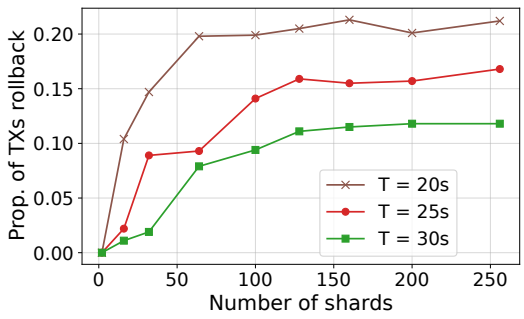

Fig. 9. Proportion of transactions that were rolled back because of failing to commit timely in the target shard. T indicates the expiration time of a CTX.

as T). When the number of shards is the same, a larger T indicates longer time for cross-shard transactions to be committed in the target shard. As a result, the proportion of failed TXs decreases. When T is fixed, as the number of shards increases, the proportion first increases and then converges. This result very possibly follows the changing proportion of cross-shard transactions.

VII. Related Work

A. Sharding Solutions

A great number of sharding solutions have been proposed to solve the scalability trilemma of blockchain. Elastico [7] is the first blockchain protocol to implement sharding, where each shard processes a portion of transactions in parallel, and system throughput scales out linearly with the number of shards. Then Kokoris *et al.* [8] proposed a state sharding protocol named OmniLedger, in which each shard only needs to store a part of the system's state, aiming to improve decentralization. Facing the high overhead of shard reconfiguration in Elastico and OmniLedger, Zamani et al. [9] proposed Rapidchain, aiming to address this issue using the Bounded Cuckoo Rule. Wang *et al.* [11] proposed Monoxide, which realizes an account-based sharding blockchain. By introducing Chu-ko-nu mining, the security of each shard is proved to be equalized to the security level of the entire network. Based on previous work, Hong *et al.* [16] and Huang *et al.* [2] focused on cross-shard transaction processing and load balancing in sharded blockchains, respectively. Recently, the Ethereum team is working on Danksharding [17], which is combined with layer2 rollups technologies.

Although the previous studies provide a lot of inspiration, they have not fully solved the problem of blockchain scalability. In [2], [7]–[9], [16], due to the large overhead of shard reconfiguration, the interval between two shard configurations might be too long, and malicious nodes can concentrate on attacking a certain shard within this interval, causing low security. The authors of Monoxide [11] found and solved this problem but at the cost of damaging decentralization, since every node in the system needs to store and execute transactions of many shards. Thus, their solution brings a lot of workload to shard nodes. Danksharding [17] is essentially a rollup-based data compression technology but having a limited scalability.

B. Various Scaling-out Solutions

In addition to the sharding technique, a couple of other solutions have been proposed to improve the scalability of blockchain [18]. Those previous studies can be classified into two categories: *Layer1* and *Layer2* solutions. Layer1 solutions focus on consensus, peer-to-peer networks, and the data structure of blockchain, improving the scalability by making on-chain modifications. For example, Layer1 blockchain techniques include sharding [2], [16], [19], Directed Acyclic Graph (DAG) [20], and new proof-based consensus mechanisms [21]. In contrast, Layer2 solutions try to scale out blockchain using off-chain methods, including off-chain channel [22], [23], side-chain [24], cross-chain protocols [25]–[29], and Rollups [30], [31].

Among these solutions, the scalability trilemma of the blockchain has not been broken, i.e., the scalability can only be improved to a limited level given that the security and decentralization are not excessively decreased. Off-chain channels do not support smart contracts. The security of the side chain is highly dependent on that of the main chain, and it has data availability issues and is vulnerable to malicious attacks. Cross-chain technologies will lead to the dispersion of resources, reducing the security and decentralization of each single chain. Rollups technology is essentially a data compression technique that improves the scalability of Layer1 blockchains to a limited extent.

Compared with those existing solutions, our proposed W3Chain defeats the scalability trilemma through a Layer2 sharding design. In particular, we ensure the security of W3Chain using a time-beacon chain.

VIII. Conclusion and Future Work

W3Chain is a Layer2 blockchain proposed to defeat the scalability trilemma of the traditional blockchain. To this end, W3Chain decouples the correctness of the blockchain and hands over the guarantee of chain consistency to Layer1. In addition, W3Chain ensures the block correctness of each shard chain through the committee's reorganization in each slot. In particular, W3Chain guarantees the atomicity of cross-shard transactions using the rollback design. Theoretical analysis and simulation results show that W3Chain can achieve high scalability and decentralization while ensuring high security.

ACKNOWLEDGMENT

This Work is supported by National Key R&D Program of China (No. 2022YFB2702304), and National Natural Science Foundation of China (62272496).

REFERENCES

[1] "Why sharding is great: demystifying the technical properties," https://vitalik.ca/general/2021/04/07/sharding.html, 2021.

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

[3] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Tech. Rep., 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[4] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Annual international cryptology conference*. Springer, 2017, pp. 357–388.

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

[6] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *40th annual symposium on foundations of computer science (cat. No. 99CB37039)*. IEEE, 1999, pp. 120–130.

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

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

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

[10] Y. Liu, Y. Xia, J. Liu, and Y. Hei, "A secure and decentralized reconfiguration protocol for sharding blockchains," in *2021 7th IEEE Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*. IEEE, 2021, pp. 111–116.

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

[12] Z. Gao, Y. Hu, and Q. Wu, "Jellyfish merkle tree," 2021.

[13] A. Hafid, A. S. Hafid, and M. Samih, "A tractable probabilistic approach to analyze sybil attacks in sharding-based blockchain protocols," *IEEE Transactions on Emerging Topics in Computing*, 2022.

[14] M. Zochowski, "Sharding: How many shards are safe?" https://medium.com/logos-network/sharding-how-many-shards-are-safe-bc361c487083, 2018.

[15] J. R. Douceur, "The sybil attack," in *International workshop on peer-to-peer systems*. Springer, 2002, pp. 251–260.

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

[17] "The hitchhiker's guide to ethereum," https://members.delphidigital.io/reports/the-hitchhikers-guide-to-ethereum, 2022.

[18] Q. Zhou, H. Huang, Z. Zheng, and J. Bian, "Solutions to scalability of blockchain: A survey," *IEEE Access*, vol. 8, pp. 16 440–16 455, 2020.

[19] J. Zhang, Z. Hong, X. Qiu, Y. Zhan, and W. Chen, "Skychain: A deep reinforcement learning-empowered dynamic blockchain sharding system," in *Proc. of 49th International Conference on Parallel Processing (ICPP'20)*, 2020, pp. 1–11.

[20] J. Xiao, S. Zhang, Z. Zhang, B. Li, X. Dai, and H. Jin, "Nezha: Exploiting Concurrency for Transaction Processing in DAG-based Blockchains," in *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2022, pp. 269–279.

[21] "Solana," 2022. [Online]. Available: http://solana.io/

[22] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," 2016.

[23] "The raiden network," https://raiden.network/, 2019.

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

[25] J. Kwon and E. Buchman, "Cosmos whitepaper," *A Netw. Distrib. Ledgers*, 2019.

[26] "Polygon," https://polygon.technology/solutions/polygon-pos, 2017.

[27] "Skale," https://skale.space/, 2018.

[28] "gnosischain," https://www.gnosischain.com/, 2019.

[29] "Loomx," https://loomx.io/, 2018.

[30] "Zk-rollups," https://docs.ethhub.io/ethereum-roadmap/layer-2-scaling/zk-rollups/, 2021.

[31] "Optimistic rollups," https://docs.ethhub.io/ethereum-roadmap/layer-2-scaling/optimistic_rollups/, 2021.