DOI: https://doi.org/10.15276/hait.08.2025.5
UDC 004.922

Virtually unlimited sharding for scalable distributed ledgers

Sergii S. Grybniak[1)]
ORCID: https://orcid.org/0000-0001-6817-8057; s.s.grybniak@op.edu.ua. Scopus Author ID: 57962557300
Yevhen Yu. Leonchyk[2)]
ORCID: https://orcid.org/0000-0003-1494-0741; leonchyk@onu.edu.ua. Scopus Author ID: 57192064365
Igor Ye. Mazurok[3)]
ORCID: https://orcid.org/0000-0002-6658-5262; mazurok@onu.edu.ua. Scopus Author ID: 57210121184
Oleksandr S. Nashyvan[1)]
ORCID: https://orcid.org/0000-0001-8281-4849; o.nashyvan@op.edu.ua. Scopus Author ID: 57963260000
Ruslan V. Shanin[2)]
ORCID: http://orcid.org/0000-0002-4414-1126; ruslanshanin@onu.edu.ua. Scopus Author ID: 55983005400
Alisa Yu. Vorokhta[4)]
ORCID: https://orcid.org/0000-0002-2790-1517; alisa.vorokhta@uni.lu. Scopus Author ID: 59184524100
[1)] Odesa National Polytechnic University, 1, Shevchenko Ave. Odessa, 65044, Ukraine
[2)] Odesa I. I. Mechnikov National University, 2, Dvoryanskaya Str. Odessa, 65082, Ukraine
[3)] Waterfall DAO, Zug, Switzerland
[4)] University of Luxembourg, 2, Ave. de l'Université Esch-sur-Alzette, 4365, Luxembourg

ABSTRACT

This paper presents an approach to improving the scalability of the decentralized smart contract platform Waterfall, based on the concept of hierarchical fractal sharding. Although distributed ledger technology holds significant promise for building secure and transparent digital ecosystems, its widespread adoption remains limited by scalability issues. A key challenge lies in the inability to proportionally increase transaction throughput with the growing number of participants without undermining either decentralization or security. The proposed solution reduces both computational and communication loads by distributing transactions, smart contracts, and network state across a system of recursively structured shards. Each shard operates as an independently validated subnetwork organized as a directed acyclic graph structure that supports asynchronous execution and consensus. This design enables the participation of low-power nodes, enhances load balancing, and achieves scalability not only at the level of the entire network but also within its internal components. The study details the mechanisms for shard formation and merging, transaction routing strategies, and dynamic placement of smart contracts. In addition, a probabilistic model is introduced to evaluate the risk of malicious capture of individual shards, and guidelines are provided for choosing safe shard sizes under various threat assumptions. While the proposed architecture is designed specifically for the Waterfall platform, its core principles and several of its methods may be adapted to other distributed ledger systems, including but not limited to blockchain-based platforms, particularly those employing modular or directed acyclic graph-structured architectures.

Keywords: Fractal sharding; smart contracts; blockchain scalability; distributed ledger; hierarchical architecture

For citation: **Grybniak S. S., Leonchyk Ye. Yu., Mazurok I. Ye., Nashyvan O. S., Shanin R. V., Vorokhta A. Yu.** “**Virtually unlimited sharding for scalable distributed ledgers**”. ***Herald of Advanced Information Technology.*** **2025; Vol. 8 No.1: 67–86. DOI: https://doi.org/10.15276/hait.08.2025.5**

INTRODUCTION

Distributed Ledger Technology (DLT) has emerged as a promising innovation that has the potential to transform various industries [39]. DLT is a decentralized system that enables secure and transparent transactions without the need for intermediaries such as banks or government entities. The technology is based on a distributed database that stores information across a network of nodes, where each node has a copy of the database.

This feature ensures that no single entity controls the data, and any changes to the database are validated through network consensus. One of the most prominent types of DLT is blockchain, which uses cryptographic algorithms to ensure data integrity and security [47]. Blockchain technology has gained traction in recent years, with many industries exploring its applications — particularly in Decentralized Finance (DeFi) [35].

Vitalik Buterin formulated [20] the so-called blockchain trilemma: of the three core attributes – decentralization, security, and scalability – a blockchain can typically achieve only two simultaneously. Since decentralization is intrinsic to DLT, and adequate security is essential for any practical deployment, scalability is often compromised. For example, the average transaction throughput of Bitcoin and Ethereum is currently limited to 11 and 63 transactions per second (TPS), respectively [7], whereas centralized payment systems such as VISA routinely process thousands of TPS (e.g., about 7,400 TPS on average in 2024 [43]).

This limitation stems from consensus protocols that require every node to validate each transaction, leading to significant processing bottlenecks [41]. As DLT adoption grows, solving the scalability challenge becomes increasingly critical. While numerous approaches have been proposed, each with its own strengths and trade-offs, achieving a practical balance between scalability, security, and decentralization remains an open problem.

Although scalability is widely recognized as a key challenge, the core issue lies in the **inability of current DLT systems to scale performance proportionally with network size**. In traditional blockchain and directed acyclic graph (DAG) based architectures, all nodes must execute every smart contract and store the complete ledger and state. As a result, increasing the number of nodes leads to linear or super-linear growth in computational and storage requirements — without any corresponding increase in transaction throughput.

The problem addressed in this work is how to enable a distributed ledger platform to scale its processing throughput and reduce resource consumption as the number of participating nodes increases. This includes minimizing redundant execution of transactions and unnecessary replication of data across the network, while preserving security and decentralization.

This work addresses this inefficiency by proposing a **novel hierarchical sharding technique** designed to enable **proportional performance scaling**. The approach involves distributing transaction execution and smart contract processing among multiple shards, and decentralizing storage of ledger and state data using an **optimal replication coefficient**. We implement this approach on the Waterfall platform, which combines a DAG-based ledger with a Proof-of-Stake coordination layer, and show that our method achieves scalable, efficient, and decentralized transaction processing.

RELATED WORKS

Various solutions have been proposed to address the scalability issue, including off-chain payment channels, diverse consensus algorithm optimizations, numerous sharding approaches, etc. Off-chain solutions such as Lightning [33] and Plasma [32] allow transactions to be processed outside the main blockchain, reducing congestion on the main chain. Consensus algorithm optimizations and new types of protocols, e.g. Proof-of-Stake [30], increase the speed of transaction processing while also reducing the node load and overall energy consumption.

Sharding (vertical and horizontal) is a well-established technique in database management systems. It entails splitting a ledger to solve the problem of scaling. In decentralized systems, sharding involves partitioning a set of nodes into groups (so-called shards) with or without appropriate ledger partitioning [9, 17], [46]. Each of the shards can handle a subset of transactions, thereby increasing the overall transaction throughput. A detailed review of sharding-based scaling methods is presented in e.g. [45]. There are a few main objects that can be split: network actors (validators, wallets, etc.), transactions, and the network state. If only one set of nodes is partitioned, this is done to speed up consensus and reduce the amount of associated communication. If the sharding of nodes is performed simultaneously with the sharding of the state and the ledger (each shard leads its own portion of data), then with a significant reduction of network load, we get the problem of shard matching coordinating, which can be solved synchronously or asynchronously. In the first case, the validators of both shards work together, and in the second case, transactions are separately executed in the shards that they affect, but a confirmation mechanism is required. Most existing cross-shard transaction processing solutions are based on the two-phase commit (2PC) protocol, which contains a prepare phase and a commit phase [10, 25]. In addition, the implementation of cross-shard execution of smart contracts is a particular challenge that is presently under active investigation, especially for the Ethereum Virtual Machine case (e.g. [8, 34]).

The sharding issue remains a complex and ongoing problem of DLT, requiring substantial scientific efforts from the community of researchers, and at present, proposed approaches have substantial concessions. For example, Ethereum 2.0 introduces significant simplifications in its sharding design, so-called Danksharding, compared to its previous approach of splitting only the ledger containing ordinary transactions [12]. The Open Network (TON) presents a multichain system supporting both homogeneous and heterogeneous shards with fastforwarding messages between them [11]. Its distinctive feature is the automatic change in the number of shards, depending on the network load. However, finalized blocks are not immutable and can be further reorganized. In addition, a modified Byzantine fault tolerance consensus protocol [28] used by TON can effectively handle relatively small (not more than a few hundred) numbers of validators that impose certain restrictions on the system's decentralization. A similar dynamic approach for shard reorganizing (Adaptive State Sharding) is presented by MultiverseX [29]. It improves overall security, in particular, preventing conspiracy in a shard and other attacks associated with a relatively small number of validators. However, shard reorganizing demand itself creates additional network load and demands extra communication overheads. This limitation becomes more and more significant as both network validators and wallets grow. Also, some platforms propose only computational scalability, e.g. Venom [42] and Everscale [14] divide the execution of smart contracts into threads that are processed by different groups of validators in parallel.

Another approach to the implementation of scaling is the creation of so-called sidechains, L2s, etc. [1, 13, 36] with their own digital assets (including their own coins), different formats of transactions, network protocols, architecture, etc. Each sidechain is attached to its main blockchain and operates parallel to it. At the moment, one of the most popular Bitcoin-based sidechains is the Liquid Network [24] and Polygon is a popular Ethereum-based sidechain [31].

This approach enables the transfer of assets from the main blockchain to the sidechain, where they can be processed in a more efficient and flexible manner. However, despite their potential benefits, there are still several challenges that need to be addressed before sidechains can be widely adopted. One of the main goals is to ensure the security and integrity of the sidechain, as any vulnerabilities or weaknesses in the sidechain can potentially compromise the entire blockchain system. Another problem is ensuring the interoperability between different sidechains and the main blockchain, which requires the development of robust protocols and standards.

One of the most promising and actively developing approaches for providing a connection between sidechains and the main network is Zero Knowledge (ZK) rollups [40], which can also improve the security and finality of transactions, as they do not rely on fraud proofs or challenge periods that are used by other rollup variants. This technique aggregates transactions into batches and generates ZK proofs for each batch. ZK rollups can reduce the amount of data that needs to be stored on the main blockchain, as well as the cost of validating transactions. In addition, ZK proofs can be used to shard smart contracts [27], ensure data availability in shards [12], and mitigate other blockchain security and scalability issues [38].

The evolution of the sidechain mechanism is fractal scaling that allows the creation of various subnetworks with unique settings. For example, the Ethereum sidechain StarkWare [22] uses its own sidechains (sub-sidechain) to implement customizable functionality. For the same purposes, in Kaspa [23], some nodes can be grouped into application-specific clusters with specific rules, but such partial nodes cannot produce blocks because they do not have complete information in contrast to full nodes. Currently, the development of scalable networks with custom features has gained significant traction and it is considered one of the crucial factors for the mass adoption of DLTs in enterprise-class applications.

Other notable projects include NEAR network protocol [37], which implements dynamic state sharding through its Nightshade architecture, enabling asynchronous cross-shard execution with delayed finality; and Internet Computer DFinity project [6], which introduces a canister-based model for parallel computation and storage across subnets, effectively acting as an application-oriented form of sharding. These platforms offer innovative approaches to scalability and decentralization, and are included in the comparative analysis in the table below.

To enhance clarity and facilitate comparison, we summarize in the table below the key characteristics of several prominent distributed ledger platforms that employ sharding. Each platform is evaluated according to its sharding type, ability to support cross-shard transactions, scalability, decentralization, and known limitations. This comparative overview contextualizes our approach and highlights its advantages over existing solutions.

The Waterfall platform introduces a fractal hierarchical sharding model that addresses the limitations identified in other systems, particularly by offering virtually unlimited scalability, dynamic reconfiguration, and fully integrated support for cross-shard transactions without relying on centralized coordinators.

Table 1. **Comparative Analysis of Sharding Approaches**

Platform	Sharding Type	Cross-shard Tx Support	Dynamic Scaling	Decentralization	Key Limitations
Ethereum 2.0	Static execution shards	Limited (via Beacon Chain)	No	High	Still under full deployment, limited shard communication
NEAR	Dynamic state sharding	Supported with asynchronous execution	Yes	Medium	Complex routing and state synchronization
Polkadot	Relay-chain parachains	Via relay-chain	Limited	Medium-High	Coordination overhead, limited parallelism
DFinity	Canister-based sharding	Yes (via routing)	Moderate	High	Protocol complexity, high validator requirement
Avalanche	Subnetworks	Yes (via Avalanche Warp Messaging)	Yes (independent subnets)	Medium	Subnet isolation, requires bridging
Waterfall (proposed)	Hierarchical fractal sharding	Fully supported via DAG and routing	Virtually unlimited	High	Currently in testnet phase, tokenomics under design

Source: **compiled by the authors**

WATERFALL PLATFORM OVERVIEW

The scalability approach proposed in this work is designed specifically for the Waterfall platform. Therefore, before introducing the sharding model itself, it is essential to present the key components and operational principles of the platform. This section provides the necessary architectural context for the subsequent sections, where the sharding solution will be developed and integrated.

The Waterfall distributed protocol is based on Directed Acyclic Graph (DAG) technology, with the Salto Collores consensus [18] supported by the Salto Collazo tokenomic model [16, 19] involving the participation of millions of nodes. The efficiency of this platform is dependent on successful collaboration between the Coordinating network and shard networks, which work in parallel to achieve a high transaction throughput [15]. Every system Worker has two essential components: the Coordinator and the Validator, both playing key roles within their respective networks (Fig. 1). Also, there are Light Workers on the Waterfall platform [3] that do not store the ledger, but store the entire network state. The implementation of the Waterfall platform provides for the possibility of deploying several autonomous Workers on each node, with a common ledger and a pool of transactions. In this case, the first of them is considered the organizer of such a node.

In a shard network, all received transactions are first added to its DAG-based ledger and are applied to alter the network state only after they are finalized [5]. A registry of Validators is responsible for assigning block producers in each slot at the start of each epoch. The Coordinating network handles the crucial tasks of linearizing (ordering) and finalizing shard ledgers, thereby enhancing security and synchronization across the platform. This network also holds information about the approved blocks generated on shard networks.

The prime goal of the Waterfall platform is to provide an efficient and favorable ecosystem for the development of various decentralized applications (DApps) using smart contracts and tokens. There are embedded tokens (including NFTs) in this network that do not demand special smart contracts to release and maintain them, but are carried out with ordinary transactions that significantly reduce overhead charges. In addition, this makes their usage more accessible to a wide range of users.

With Waterfall, a specially developed subnetwork technology ensures scalability in terms of transaction processing capacity within the shard networks; in other words, it provides horizontal sharding [2, 4]. A transaction pool, the set of valid pending transactions to be recorded in blocks, is split between network Validators by applying a hierarchical and graph-based clustering algorithm.

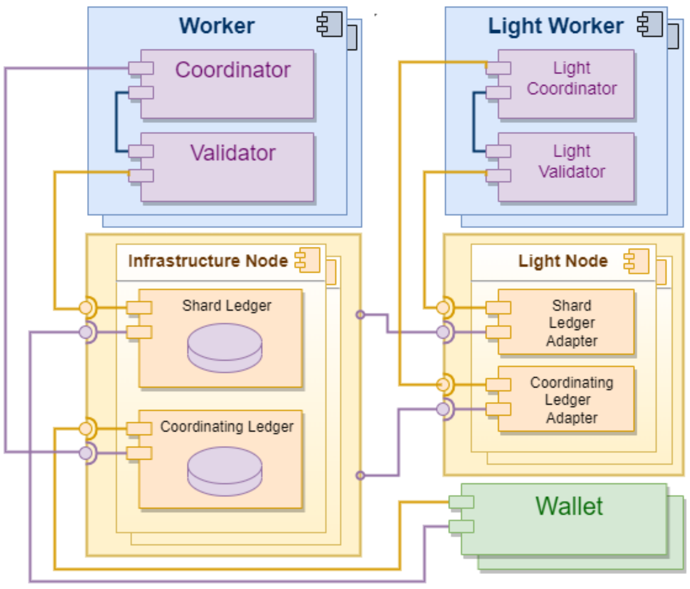


Fig. 1. **Architecture of Waterfall's nodes and Workers**
Source: **compiled by the authors**

However, all Validators of a shard network have the same ledger and the same general state of processing of all finalized blocks. Thus, as the number of Validators and transactions grows, the number of blocks in each slot will also increase. Although subnetworks parallelize the validation of transactions and their inclusion in blocks, further processing of blocks to include them in the ledger and change the current state is performed by the nodes of all subnetworks.

Hence, the main tasks to be solved by sharding are to reduce the node load on:
1) execution of finalized transactions;
2) execution of smart contracts;
3) storing the network state and the ledger;
4) transmission of network traffic.

This article describes the first of two stages of the Waterfall virtually unlimited sharding.

1. Homogeneous shards consist of approximately the same number of Workers and distribute the total workload into approximately equal parts. They are created in such a way as to minimize the number of cross-shard transactions. Our goal is to group network users (network wallets and smart contracts) so that they communicate mostly within shards, since the intensity of cross-shard activity reduces the efficiency of the entire network.

2. Heterogeneous shards are built according to their principles and can even be developed by a third party interacting with others based on common requirements and an interface. For example, there may be dedicated shards for decentralized finance, web3 games, e-voting services, electronic medical screening systems, etc. However, dividing users into shards that are relatively closed in terms of activities not only reduces the overall load on network nodes, but also opens up new opportunities for intra-shard interactions.

Currently, the main public network of the Waterfall [44] project operates with over 42,000 active validators and demonstrates a transaction throughput exceeding 12,000 transactions per second under real-world conditions.

This architectural overview serves as the foundation for the sharding model proposed in this work. Since the solution is tightly integrated with the internal mechanisms of the Waterfall platform, understanding these components is essential. Nevertheless, the general structure of our model is sufficiently abstract to allow potential adaptation to other platforms that share architectural similarities, particularly those with modular consensus or DAG-based data structures.

SHARD DESIGN

Shard Ledger. In broad terms, the shard ledger is a heterogeneous DAG. Its vertices are blocks, with some containing transactions and others serving as blocks in the chain to achieve consensus. In addition, this DAG has an initial genesis block that establishes the network's rules. In practical terms, the Waterfall DAG is implemented as two separate entities with different genesis blocks, block structures, and, formally, block creators: Coordinators and Validators (Fig. 2).

.

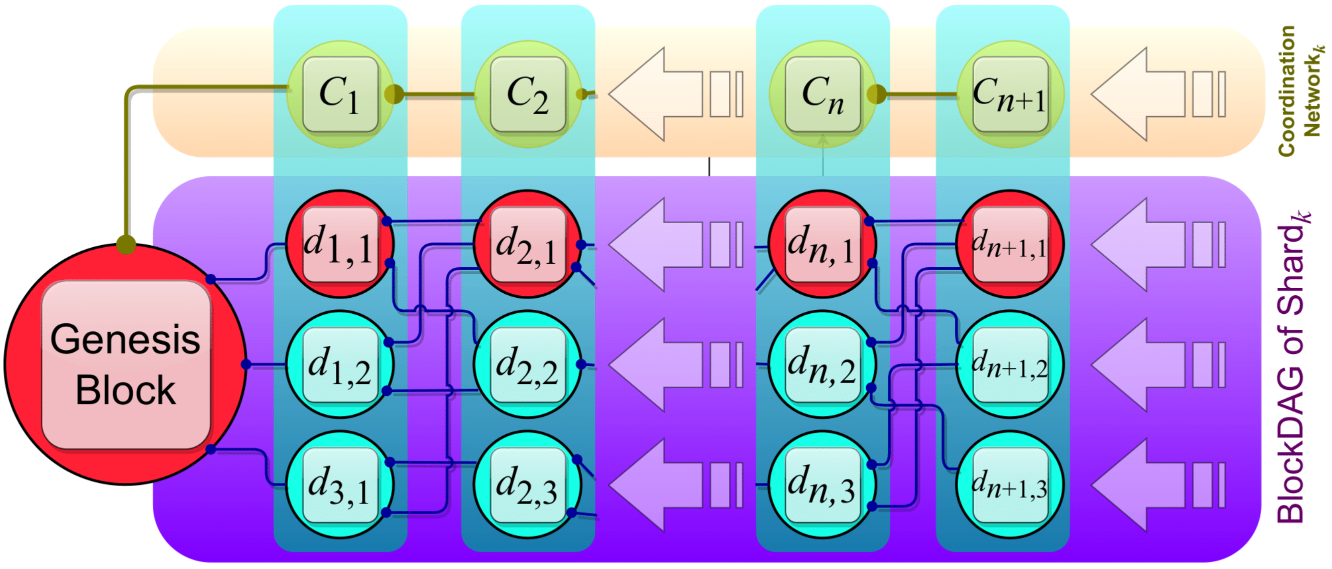


Fig. 2. **Structure of a shard ledger**
Source: **compiled by the authors**

As we move towards sharding, we preserve this outlined structure for each shard. Further, the initial blocks of a new shard will also, if needed, refer to the blocks of the shard(s) that created it.

Procedures for Splitting and Merging. First, we will describe the procedures for splitting and merging shards, and then we will consider when the network performs these actions. Initially, the Waterfall network operates without shards. Consequently, when the network is first split into shards, the Root Network and two additional shard-descendants are created (Fig. 3). The Root Network continues the initial network and is designed to work only with inherent coins. All smart contracts and tokens are distributed among the shard-descendants, which are subsequently responsible for working with smart contracts and tokens.

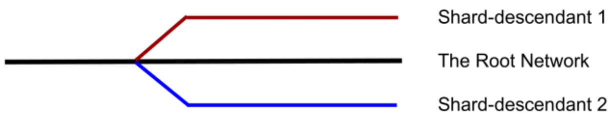


Fig. 3. **The first splitting**
Source: **compiled by the authors**

The procedure for creating these shards is as follows.

1. Two blocks are published in the blockDAG network, which will become the genesis blocks for shards. These blocks contain the information necessary for the operation of the corresponding shards.

2. Split networks begin their work epoch after the finalization of the genesis blocks of new shards.

3. Once sharding begins, smart contract calls are published to the corresponding shards, and the initial network becomes the Root Network.

4. Until the part of the Root Network blockDAG that contains calls to smart contracts is finalized, the states of the token shards cannot be finalized.

5. The Worker-organizer of the node not only remains to work in the Root Network, but is also assigned to one of the two newly formed shards, along with the rest of their Workers-nodemates. Therefore, each node is represented in the Root Network by only one of its Workers-organizers, and all of its Workers in one of the shard-descendants.

The Root Network performs the following functions:

1) conducts all operations with the main coin of the network;

2) keeps records of shards, splits, and merges them;

3) is responsible for rewarding Workers for work in shards;

4) is responsible for on/off-boarding of Workers in all shards;

5) is responsible for registering new smart contracts and embedded tokens.

All coin accounts are processed in the Root Network. Otherwise, nearly all transfers of coins and calls of smart contracts would be intershard, increasing network workload. At the same time, other shards have the same structure as the original network and additionally process a certain number of smart contracts. Concurrently, data regarding the average computational complexity during network operation (represented as the average gas spent per second over the last reporting period) is published and signed by all shard participants, to monitor the need to combine or merge shards.

Now we will outline the process of splitting an arbitrary shard into two subshards. Let us consider shard A, which we split into subshards B and C (Fig. 4).

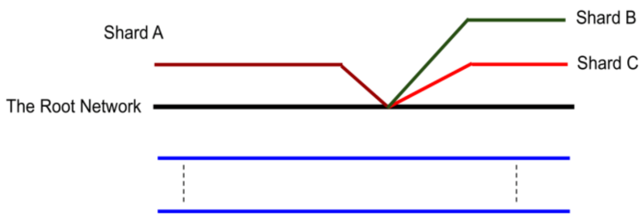


Fig. 4. **Splitting into two subshards**
***Source:* compiled by the authors**

The procedure for this split is as follows.

1. In the Root Network, blocks are published that will become the genesis for shards B and C.
2. Similar to the initial sharding procedure, split networks begin their work one epoch after the finalization of the genesis blocks of new shards.
3. Once shards B and C are operational, smart contract calls are no longer published to shard A, but only to the corresponding new shards.
4. Blocks in shards B and C reference the genesis blocks of those shards and the tips (blocks not referenced by other blocks) of shard A.
5. The non-finalized part of shard A is finalized in both shard B and C separately, but at the same time, each of the shards finalizes and executes only transactions of its respective smart contracts.

Finally, let us describe the procedure for merging shards (Fig. 5).

1. In the Root Network, a block is published that will become the genesis for the new shard.
2. Similar to the procedure for the initial splitting into shards, a new shard begins its work an epoch after the finalization of its genesis block.
3. Until this time, the nodes of the merged shards are synchronized.
4. In addition to the rules for constructing a blockDAG, the blocks of the new shard refer to the block tips of the shards that have merged.
5. The ordering of unfinalized blocks from old shards is determined in the Coordinating Network of the new shard, taking into account references to the tips of old shards.

Note that during both the splitting and merging processes, the resulting shards get a number through auto-incrementation. For instance, if at the current moment the last assigned number is N, then when any shard is split, the resulting subshards will be numbered as $N + 1$ and $N + 2$. Similarly, when shards are merged, the resulting combined shard will receive the number $N + 1$.

Decision on Splitting and Merging. A change in the number of shards is usually tied to a change in the number of transactions over a certain period (e.g. [11, 14], [42]) and the number of Validators (e.g. [29]).

Let us denote by G the average gas consumption per second. This value is an integral characteristic of the load on the Worker (its Validator component), since it determines the volume of calculations performed. Also, gas consumption indirectly characterizes the average number of transactions, the volume of transmitted messages (traffic), and the increase in the size of the state and ledger. The minimum recommended system requirements for the node allow processing up to G_{sup} per second.

Next, we set the parameters α_{min} and α_{max},

$$0 < \alpha_{min} < \alpha_{max} < 1.$$

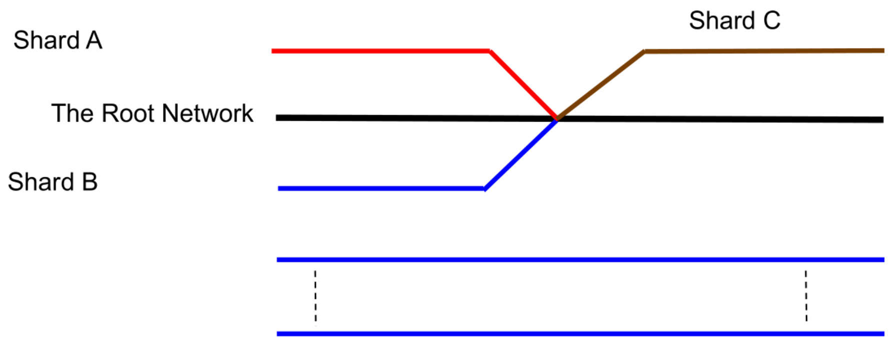


Fig. 5. **Merging of two shards**
***Source:* compiled by the authors**

Let us also denote $G_{max} = \alpha_{max}G_{sup}$, $G_{min} = \alpha_{min}G_{sup}$. In addition, let us denote by W the number of Workers in the shard, by W_{min} the minimum and by W_{max} the maximum number of Workers in the shard and assume that

$$2G_{min} < G_{max},\ 2W_{min} < W_{pre},\ 2W_{pre} < W_{max},$$

with some W_{pre}. Then the shard is split into two subshards if

$$W > W_{max} \text{ or } G > G_{max} \text{ or } W > W_{pre}.$$

We do not split a shard into two shards in the case when $G > G_{max}$ and $W \leqslant W_{pre}$ since in this case the resulting shards will have too few Workers. Likewise, if

$$W < W_{min} \text{ or } G < G_{min} \text{ and } W < W_{pre},$$

then the shards are merged. Fig. 6 graphically shows the rule for making decisions about splitting or merging shards.

Distribution of Workers and Smart Contracts. Let us now consider issues related to the distribution of Workers and smart contracts among shards. Their registration takes place in the Main Shard. The user contributes a fixed Worker's stake or sends a smart contract deployment transaction with the corresponding fee in intrinsic coins. When creating a new node, its Worker-organizer is defined in both the Root Shard and the shard-descendant. Subsequent Workers of this node are placed together with the organizer in the same shard-descendant.

During the onboarding of a Worker-organizer, the probability of it getting into some shard is inversely proportional to the number of Workers already existing there. Thus, on the one hand, uniform filling of shards is achieved, and on the other hand, the randomness of the distribution does not allow Workers to get into a pre-selected shard, which mitigates any increase in the share of colluded malicious users.

Let N shards be registered in the system at a given time to process smart contracts or embedded tokens, one of which can be allocated a Worker-organizer of a node, with $n_k\ (k = 1, \ldots, N)$ Workers.

Let us set the probability of a new Worker getting into i-th shard

$$p_i = \frac{1}{n_i}\left(\sum_{k=1}^{N} n_k^{-1}\right)^{-1},\ i = 1, \ldots, N.$$

Next, based on the hash of the public key of Worker and the hash of the block in which it was registered, we calculate the value A in the range from 0 (not inclusive) to 1 and recurrent with a0 = 0 we obtain a partition of the segment [0; 1]:

$$a_k = a_{k-1} + p_k,\ k = 1,\ \ldots,\ N.$$

Then the shard number m, into which this Worker will be distributed, is determined from the condition

$$a_{m-1} < A \leq a_m.$$

Let us now describe the procedure for distributing Workers among shards in the case of shard splitting. Let X_W be an ordered list of shard nodes. When dividing this shard into two subshards, the two nodes with the largest number of Workers are assigned to different subshards. Next, we select the 1st node from the remaining part of the X_W and place it in a subshard with the currently smaller number of Workers (or in the 1-st if they are equal). Repeat the last step until the end of the list X_W. Moreover, even nodes that are not members of a given shard can independently determine the composition of new subshards.

Let us now move on to the issue of the distribution of smart contracts among shards during their deployments. In this case, we will take the computational load as a basis. A smart contract or embedded token is randomly distributed into one of the existing shards based on the hash of the transaction of its creation, and the hash of the block in which it is published. The probability of getting into the i-th shard is inversely proportional to its total gas consumption G_i for the last reporting period.

The selection algorithm is similar to that described above with

$$p_i = \frac{1}{G_i}\left(\sum_{k=1}^{N} G_k^{-1}\right)^{-1},\ i = 1, \ldots, N.$$

In addition, if the user has previously placed any contract or token, then he can indicate by himself that the new one should be placed in the same shard, and in the future when this shard is divided, both will always remain in the same subshard. For example, the user first created a token T_1, and then, when creating another token T_2, specified that it should be "connected" to T_1. Consequently, both tokens will always be processed in the same shard. This is done for the convenience of DApp developers and to reduce intershard interactions, since the operation of a DApp may require several smart contracts interacting with each other.

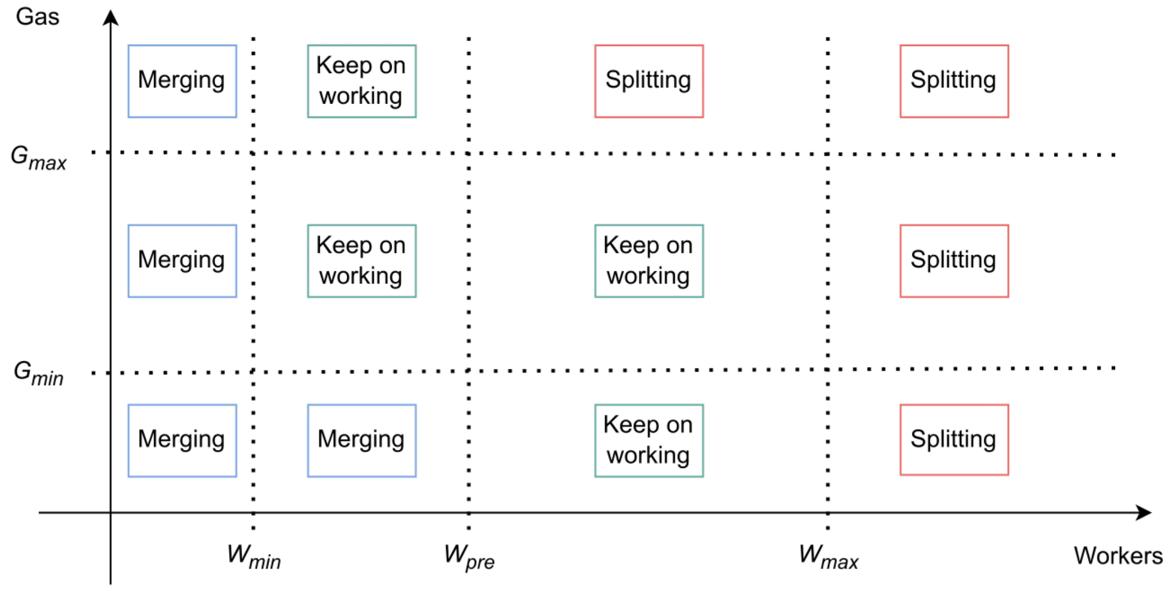


Fig. 6. **Deciding on splitting or merging shards**
Source: **compiled by the authors**

Regarding the distribution of smart contracts among shards during a shard split, it follows a process similar to the distribution of Workers. Specifically, an ordered list consisting of both individual smart contracts alongside embedded tokens and their "related" groups within a shard is split into segments using a similar algorithm, taking into account their gas consumption. These groups are then entirely moved to one of the subshards, ensuring the continuity of their intragroup interactions.

Hierarchical Structure. In the future, with a significant increase in the number of subshards of the Root Network, the split will occur with the addition of new levels using the so-called fractal approach. Let us explain the procedure for creating a new level using the following example. Let's assume that shard 3 is to be split into two shards, 4 and 5. This can be done as described above, forming two new subshards instead of one shard 2 (Fig. 7, the left panel). However, adding new shards increases the load on the Root Network and therefore this approach has limitations. At that point, shard 2 itself begins to work similarly to the Root Network, and shards 4 and 5 become its descendants (Fig. 7, the right panel).

When splitting a shard to create a new level, only the node's Worker-organizer remains in the shard-ancestor, while the rest, along with the organizer, are distributed among one of the shard-descendants. For example, in the scenario depicted on the right panel in Fig. 7, there are four possible situations: the Worker-organizer operates in one of the chains of shards 0-1, 0-2, 0-3-4, or 0-3-5, while all its nodemates are in only 1, 2, 4, or 5, respectively.

In this example, when on-boarding a new node, as described above, its Worker-organizer can also be distributed among one of the 4 chains of shards with a probability inversely proportional to the number of existing Workers in shards 1, 2, 4, and 5.

Therefore, the fractal structure provides exponential growth in the number of supported shards that allows for virtually unlimited sharding. An example of the resulting structure can be seen in Fig. 8. Note that each line in Fig. 8 is a shard ledger represented in Fig. 2. The definition of specific conditions under which separation occurs with a transition to a new tier will be determined after the implementation of the first stage/level. In addition, such a hierarchical system facilitates the creation of heterogeneous shards. In this case, a shard with a specific specification and/or theme is connected directly to the Root Network and, in case of its splitting, new subshards are automatically created at the lower level, without increasing the overall load on the Root Network.

TRANSACTION ROUTING

When partitioning a large number of nodes into shards, a challenge emerges as transactions can enter the network through any node but are processed by only a specific subset of nodes. Naturally, the problem of optimal routing of a transaction arises.

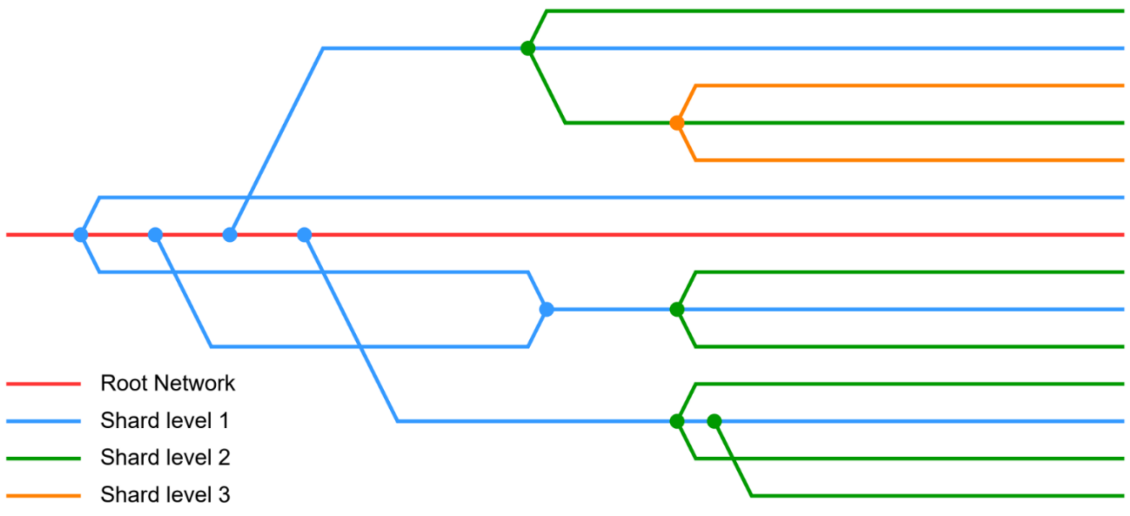


Fig. 7. **Two ways of creating new shards**
***Source:* compiled by the authors**

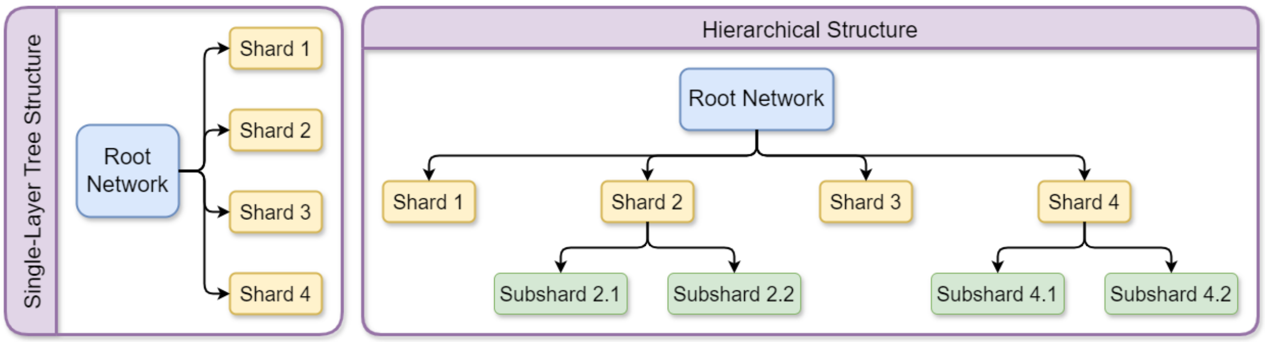


Fig. 8. **Structure of the Waterfall's network ledger**
***Source:* compiled by the authors**

This involves directing it through an arbitrary node to reach the maximum possible number of nodes within the target shard (ideally all) while minimizing its impact on extraneous nodes. In Fig. 9, we observe an illustration of the optimal method for propagating a transaction in such a scenario.

The construction of such an optimal route can only be assured at any point in time in problems with complete information, when each node knows the current neighbors of all other nodes, i.e. the complete graph of connections is known. In real applications, a node typically has awareness limited to a specific (usually fixed) number of nodes (see, for example, the Kademlia protocol [26]) and does not know the neighbors of other nodes. However, the protocol guarantees finding the required node in a limited number of requests, which means that the system can adapt to the statistical characteristics of requests and automatically solve the problem of acceptable routing with a small number of shards. Essentially, upon receiving a "foreign" transaction, each node searches in its directory for a certain number of nodes of the target shard. If the required number of nodes is found, the transaction is transferred. Otherwise, if such nodes are not found, a required number of random nodes of the target shard is selected, and requests are sent to neighbors to search for them. Since the query results are hashed, further access to this shard is not a problem for a certain period.

Nevertheless, this strategy falls short in addressing the issue when there are a large number of shards. The constrained memory capacity for retaining neighbors leads to the constant deletion of recently found nodes of the subsequent shard. The hierarchical organization of shards (vertical sharding) solves this problem, under the condition that there are approximately 10 shards at one level, since the nodes will remember 4-5 nodes from each shard.

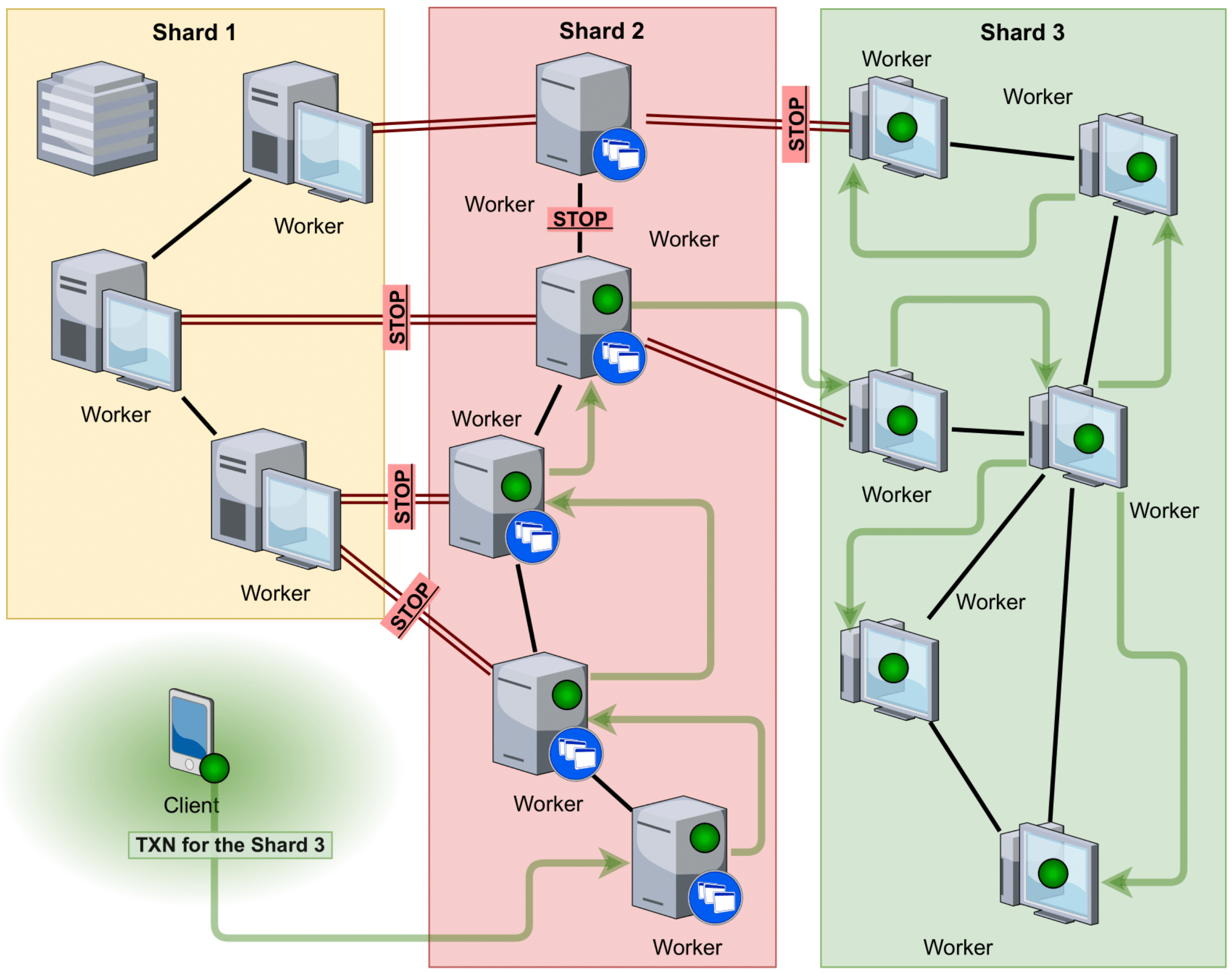


Fig. 9. **The ideal routing of transactions**
Source: **compiled by the authors**

In addition, the complexity of this issue is further compounded by its massiveness. Considering that wallets can connect to arbitrary nodes (as is commonly the case), it results in the majority of transactions initially entering the network through nodes that are not part of the shard responsible for processing them.

Thus, the problem primarily comes down to ensuring that a sufficient number of nodes within the target shard receive the transaction.

To achieve this, it is necessary to solve the problem of intershard node search, which is divided into the following parts:

1) search for the shard in the shard hierarchy;

2) search for the public key of the Workers of the target shard;

3) search for the Worker's address using its public key.

The first problem can be solved by domain identification of shards within a hierarchical sharding framework. In the shard identifier, we will indicate the shard number and all its shard-ancestors. Then the path from the Root Network to the target shard will be obvious.

To address the second issue, it will be necessary for Workers to be aware of the public keys of:

1) all Workers within their own shard;

2) several Workers from each descendant shard (if any);

3) several Workers from the ancestor shard (if any).

Therefore, by combining these requirements with domain identification of shards, we will always know a few nodes to refer to for continued searching through the shard tree.

The third problem, taking into account the reservations made above, is solved by Kademlia. In this case, it is necessary to supplement the Kademlia address table with the identifiers of the shards to which they belong. This approach allows us to completely solve the problem of searching for nodes, providing instantaneous access to recently requested nodes.

SHARD TAKEOVER ATTACK

The potential for a shard takeover attack arises when a well-resourced adversary compromises the consensus mechanism by filling the shard with its Workers [21]. Once the adversary achieves control over one-third of the Workers, it possesses the capability to halt the shard. If control extends to two-thirds of the Workers, it can manipulate transactions within the shard, including double-spend and censorship. Moreover, the adversary can also disrupt the communication and coordination among different shards, potentially propagating the attack to other shards. Therefore, sharded networks must design effective mechanisms to mitigate these attacks, such as randomizing allocation and balancing shard size. In particular, based on the results obtained in this section, the value W_{min} considered above can be determined.

Case of Shard Splitting. Let's consider a scenario where a shard needs to be divided into two separate shards. According to the consensus protocol, the share of faulty Workers, denoted as *f*, should not exceed one-third of their total number [5]. However, ensuring this condition in the original shard with $f > 1/6$ does not automatically guarantee that in each of the formed subshards the share of faulty nodes will also be less than one-third. We will calculate the probability that, for a given *f*, at least one subshard will consist of one-third or more faulty nodes.

Let us introduce the following notation:
– f – share of faulty nodes in the original shard;
– $N_0 = N_1 = N/2$ – number of nodes in each subshard;
– $N = N_0 + N_1$ – number of nodes in the original shard;
– F_0 – number of faulty nodes in the first subshard;
– F_1 – number of faulty nodes in the second subshard;
– $F = F_0 + F_1 = fN$ – number of faulty nodes in the original shard.

Then, we can write the condition of faultless shard splitting as follows

$$\forall i \in \{0,1\}\ F_i < \frac{1}{3}N_i = \frac{1}{6}N.$$

Using the equality $fN = F_0 + F_1$, we obtain that this condition is equivalent to the following

$$\forall i \in \{0,1\}\ N\left(f - \frac{1}{6}\right) < \frac{1}{3}N_i = \frac{1}{6}N.$$

Therefore, we need to consider further only the value of $1/6 < f < 1/3$ for which, after division, the shards may turn out to be both fair and foul. The options for distributing faulty nodes across subshards will be $2^F = 2^{Nf}$. Of this number, these outcomes for which the condition above is met will be successful.

Their quantity can be found using the formula:

$$n_t = \sum_{i=N(f-1/6)+1}^{N/6-1} C_{Nf}^{i}.$$

This allows us to assess the likelihood of favorable and unfavorable outcomes:

$$p_f = 1 - 2^{-Nf} \sum_{i=N(f-1/6)+1}^{N/6-1} C_{Nf}^{i},$$

$$p_t = 2^{-Nf} \sum_{i=N(f-1/6)+1}^{N/6-1} C_{Nf}^{i}.$$

Where p_f is the probability that at least one of the resulting subshards becomes faulty after the split, p_t is the probability that all resulting subshards remain correct (i.e., free of majority-faulty nodes)

Let's examine the graphs of this function (Fig. 10). We will plot the probability of at least one of the resulting subshards being faulty on the vertical axis. On the horizontal axis, we will depict the number of nodes in the original shard and the share of faulty ones among them.

As we see, as the share of faulty nodes approaches 1/3, the probability of a successful shard split decreases noticeably. Of practical interest is mainly the situation with the share of faulty nodes less than 20 % and several hundred nodes in the target shard. In this case, the probability of success can be considered acceptable. For large shards with several thousand nodes, a share of faulty ones of up to 25 % can be considered acceptable. If the share of faulty nodes approaches 30%, one can count on the successful division of the shard only with the number of nodes measured in tens of thousands.

Case of Shard Onboarding. Let's consider the problem of estimating the probability p of the formation of a faulty shard of a given size, which is composed of Workers with a given probability of faultiness. Suppose the probability of faultiness for any randomly chosen Worker is p_{faulty}. (uniform across all Workers), and the shard should consist of n Workers. Let's calculate the probability that the share of faulty Workers, where A is the number of faulty ones, will exceed a certain threshold value q = 1/3 at which the shard is considered to be faulty and cannot fully work.

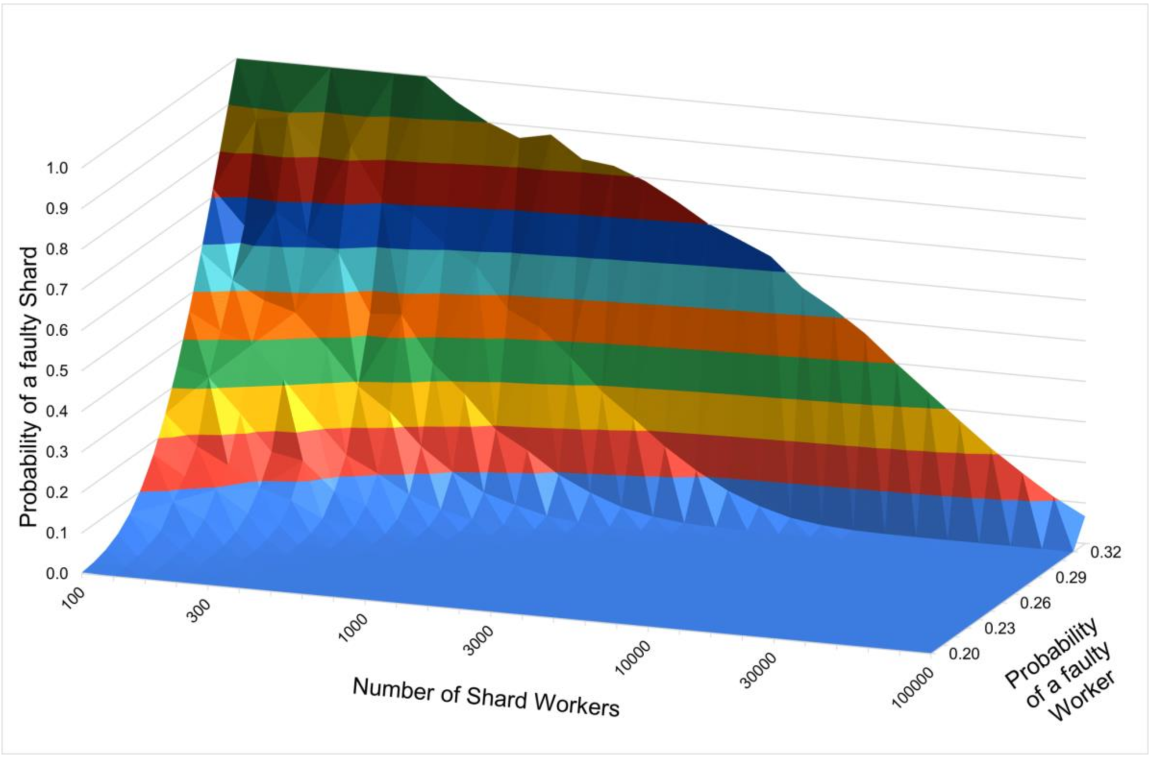


Fig. 10. **The faulty subshard probability**
Source: **compiled by the authors**

Let $q_{faulty} = 1 - p_{faulty}$. Without loss of generality, we can assume that the number of Workers will be selected for which the inequality is satisfied: $n \cdot p_{faulty} \cdot q_{faulty} > 10$.

Then we can use Laplace's integral theorem:

$$p = \mathrm{P}(n/3 \le A) = \Phi(\infty) - \Phi\left(\frac{n/3 - n \cdot p_{faulty}}{\sqrt{n \cdot p_{faulty} \cdot q_{faulty}}}\right) =$$

$$= \frac{1}{2} - \Phi\left(\sqrt{n}\,\frac{1/3 - p_{faulty}}{\sqrt{p_{faulty} \cdot q_{faulty}}}\right).$$

Fig. 11 illustrates that it is feasible to establish a properly functioning shard with 100 or more Workers if the probability of incoming faulty nodes is below 1/3. Obviously, if the probability of new nodes being faulty is 1/3, then the probability of creating a functional shard will be equal to 1/2 for any sufficiently large shard. Conversely, computations indicate that creating a functional shard with a size of 100 or more Workers becomes nearly impossible when the probability of incoming faulty nodes is around 1/2.

Let's imagine that we are ready to accept some very insignificant probability of a faulty shard occurring. Then Fig. 12 will allow one to choose the appropriate shard size for any probability of faulty nodes appearing (but no more than 1/3). For example, if the estimated probability of a faulty Worker appearing is 0.25 and we are willing to put up with the probability of a faulty shard appearing no more than 10−15, then we will have to create shards of at least 1,000 Workers in size, which is quite acceptable. A smaller number of Workers is also not justified from the point of view of the tokenomics of the platform [19], since in this case, a relatively small total stake reduces the level of shard security.

Additional Security Considerations. While this work focuses on the scalability aspects of sharding within the Waterfall platform, it is important to acknowledge other potential vectors of attack that may affect the security of the system.

In addition to shard takeover threats, the following scenarios are worth considering:

- Cross-shard replay attacks, where the same transaction may be reused maliciously in different shards without proper validation safeguards.
- Denial of service (DoS) by flooding multiple shards with high-frequency intershard transactions, aiming to exhaust network or computational resources.

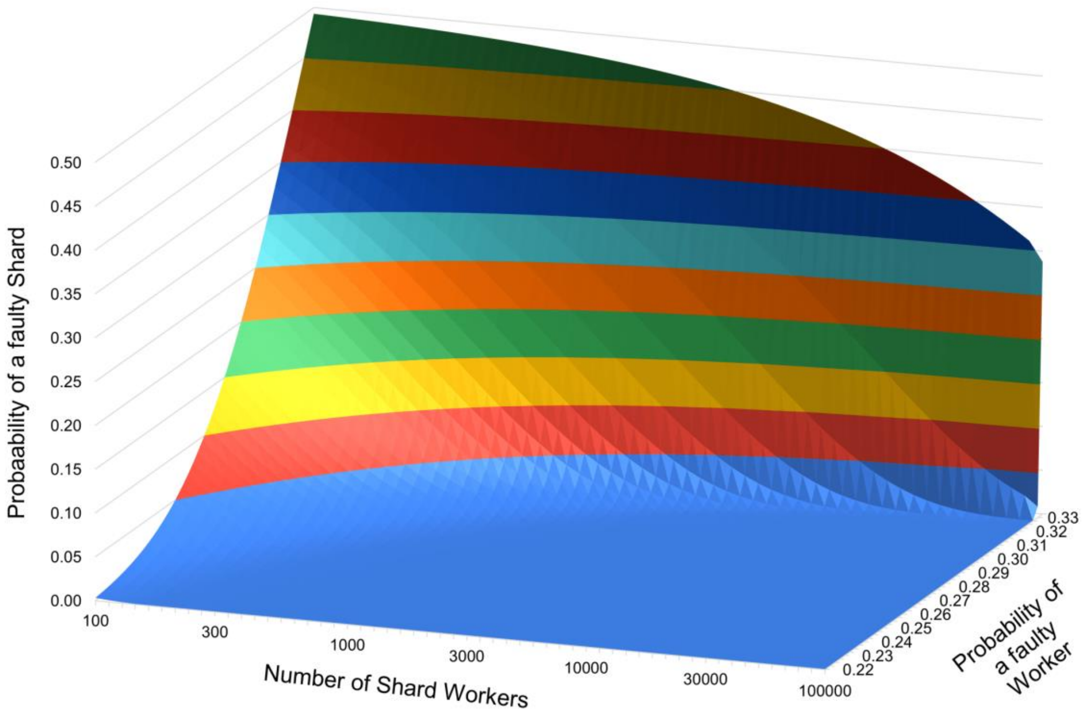


Fig. 11. **The faulty shard probability**
***Source:* compiled by the authors**

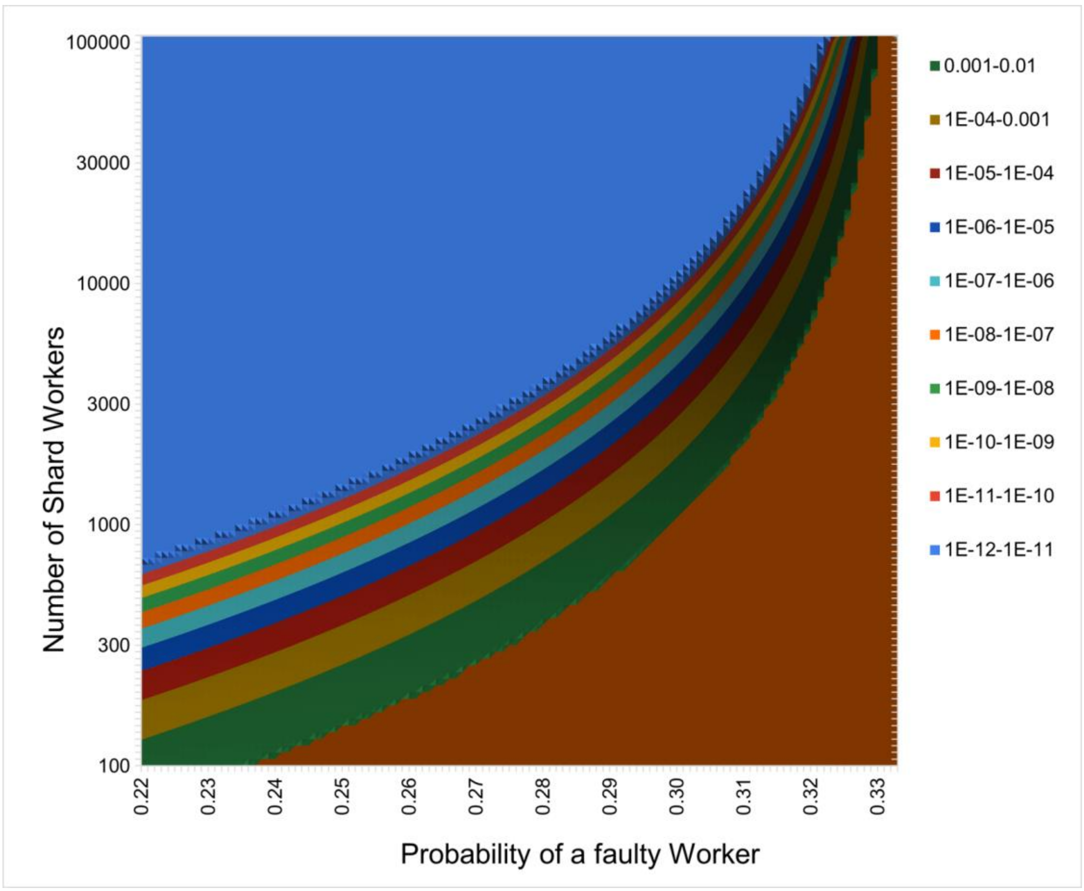


Fig. 12. **Probability of occurrence of a faulty shard**
***Source:* compiled by the authors**

- Manipulation of shard assignment, especially during node onboarding or smart contract deployment, by adversaries attempting to concentrate influence within specific shards.
- Inconsistent state propagation, which may arise from latency in intershard communication and lead to temporary disagreements in network state.

These aspects are part of a broader security framework that is being developed in parallel and will be discussed in a dedicated publication. The present paper is limited to scalability-centric considerations but aims to remain compatible with robust security assumptions.

CONCLUSION

Distributed Ledger Technologies (DLTs) offer transformative potential across a wide range of industries. However, they continue to face fundamental architectural challenges, particularly in achieving scalability without compromising decentralization and security – the so-called blockchain trilemma. In practice, this often results in a trade-off between scalability and decentralization.

In this work, we proposed and detailed a hierarchical sharding architecture tailored for the Waterfall platform, a blockDAG-based DLT with a hybrid PoS coordination mechanism. The proposed sharding model reduces the overall system load by distributing transaction processing, smart contract execution, ledger storage, and network traffic across dynamically managed shards. This design builds upon previously implemented subnetwork and Light Worker mechanisms, creating a synergistic ecosystem capable of scalable and sustainable growth.

The practical viability of our approach has been demonstrated in the Waterfall public mainnet. The Main Shard successfully sustained a transaction throughput of 12,693 transactions per second (TPS) under synthetic load conditions without performance degradation. Simulation experiments confirmed the correctness of shard splitting and merging procedures, ensured consistent state convergence, and validated our transaction routing strategy based on adaptive memory and probabilistic node discovery. Furthermore, the probabilistic algorithms for allocating Workers and smart contracts were shown to achieve near-uniform distribution and resist adversarial manipulation.

Security aspects were also considered. We provided a quantitative analysis of shard takeover attacks, determining thresholds for the number of malicious nodes that could compromise a shard. Based on this, we proposed safe shard sizing parameters, taking into account acceptable risk probabilities and validator distributions. For instance, assuming that no more than 25 % of Workers may behave maliciously, our model indicates that a shard size of approximately 1,000 nodes is required to reduce the probability of a successful takeover attack below 10^{-6}. The corresponding analysis, including the formal model and simulation results, is presented in Section **“Shard Takeover Attack”** and illustrated in Fig. 7.

The proposed architecture enables the formation of a theoretically unlimited number of heterogeneous shards, including application-specific shards with custom logic and governance models. This flexibility makes the Waterfall platform highly adaptable to enterprise and sectoral needs, paving the way for decentralized infrastructures in finance, gaming, healthcare, and beyond.

Looking forward, the high-level sharding framework presented here must be implemented and fine-tuned at the network protocol level. Additionally, intra- and inter-shard interactions should be governed by the platform’s tokenomics model, ensuring appropriate incentives, economic sustainability, and operational security.

In addition to practical implementation results, this work offers a set of novel scientific contributions that advance the current state of distributed ledger scalability. The main innovations introduced in this research include.

1. *Fractal hierarchical sharding model.* We introduce a multi-level sharding architecture with fractal structure, enabling theoretically unlimited scalability without overloading the coordination layer. Unlike traditional flat sharding schemes, each shard can act as a root for its own descendants, maintaining decentralization and minimizing overhead.
2. *Probabilistic self-balancing allocation algorithm.* A novel method for assigning Workers and smart contracts to shards is proposed. It ensures statistical load balancing based on inverse-proportional distribution (to shard size or gas usage) and provides resistance to targeted collusion attacks by randomizing placement.
3. *Integration of Light Workers with subnetworked DAG shards.* The paper extends the Waterfall platform architecture by combining DAG-based shard ledgers with lightweight validator nodes, allowing nodes with limited computational resources to participate fully in transaction validation without storing the entire ledger.

4. *Quantitative shard takeover probability modeling.* We present a mathematical model to estimate the probability of forming a faulty shard during splitting or onboarding, based on the assumed share of malicious nodes. This yields concrete recommendations for minimum shard size to maintain consensus integrity under probabilistic assumptions.

5. *Efficient transaction routing in hierarchical shard trees.* A transaction routing mechanism is designed for large-scale sharded systems with limited peer knowledge. It combines hierarchical shard identifiers with enhanced Kademlia-based node discovery, ensuring reliable delivery even with many shards and partial topology visibility.

These contributions form the foundation for a scalable, secure, and adaptable DLT architecture applicable to a broad range of decentralized systems and enterprise-grade blockchain infrastructures.

REFERENCES

1. Abbas, H., Caprolu, M. & Di Pietro, R. "Analysis of polkadot: architecture, internals, and contradictions". *IEEE International Conference on Blockchain (Blockchain)*. Espoo, Finland. 2022. p. 61–70, https://www.scopus.com/record/display.uri?eid=2-s2.0-85139963560&origin=recordpage. DOI: https://doi.org/10.1109/Blockchain55522.2022.00019.

2. Antonenko, O., Grybniak, S., Guzey, D., Nashyvan, O. & Shanin, R. "Subnetworks in BlockDAG" . *IEEE 1st Global Emerging Technology Blockchain Forum: Blockchain & Beyond (iGETblockchain)*. Irvine, USA. 2022. p. 1–6, https://www.scopus.com/record/display.uri?eid=2-s2.0-85153859577&origin=resultslist. DOI: https://doi.org/10.1109/iGETblockchain56591.2022.10087101.

3. Antonenko, O., Grybniak, S., Guzey, D., Nashyvan, O. & Shanin, R. "Light Workers in Waterfall". *IEEE 1st Ukrainian Distributed Ledger Technology Forum (UADLTF)*. 2023, https://www.scopus.com/record/display.uri?eid=2-s2.0-85196748086&origin=resultslist. DOI: https://doi.org/10.1109/UADLTF61495.2023.10548294.

4. Antonenko, O., Grybniak, S., Guzey, D., Nashyvan, O. & Shanin, R. "Subnetworks in BlockDAG." *ACM Distrib. Ledger Technol: Research and Practice.* 2023; 3 (2): 1–23, https://www.scopus.com/record/display.uri?eid=2-s2.0-85153859577&origin=resultslist. DOI: https://doi.org/10.1145/3627540.

5. Antonenko, O., Grybniak, S., Guzey, D., Nashyvan, O. & Shanin, R. "Waterfall: Salto Collores. BFT based PoS on blockDAG" . *IEEE 1st Ukrainian Distributed Ledger Technology Forum (UADLTF)*. 2023, https://www.scopus.com/record/display.uri?eid=2-s2.0-85196756787&origin=resultslist. DOI: https://doi.org/10.1109/UADLTF61495.2023.10548638.

6. Assmann, B. & Burri, S. J. "Advancing Blockchain Scalability: A linear optimization framework for diversified node allocation in shards" *arXiv*. 2024. DOI: https://doi.org/10.48550/arXiv.2405.05245.

7. "What is transactions per second (TPS)?" – Available from: https://chainspect.app/blog/transactions-per-second-tps. – [Accessed: Dec. 2024].

8. Cherniaeva, A., Nikolaev, M. & Komarov, M. "=nil;'s zkEVM1: A Secure Updatable Type-1 zkEVM". 2023. – Available from: https://cms.nil.foundation/uploads/zk_EVM_1_7d6f8caa16.pdf. – [Accessed: Dec. 2024].

9. Dang, H., Dinh, T., Loghin, D., Chang, E.-C., Lin, Q. & Ooi, B. "Towards scaling Blockchain systems via Sharding". *Proceedings of the 2019 International Conference on Management of Data (Amsterdam, Netherlands) (SIGMOD '19)*. New York, USA. 2019. p. 123–140. DOI: https://doi.org/10.1145/3299869.3319889.

10. Das, S., Krishnan, V. & Ren, L. "Efficient cross-shard transaction execution in Sharded Blockchains". *ArXiv*. 2020. DOI: https://doi.org/10.48550/arXiv.2007.14521.

11. Durov, N. "The Open Network." 2021.– Available from: https://ton.org/whitepaper.pdf – [Accessed: Dec. 2024].

12. "Danksharding". 2023. – Available from: https://ethereum.org/en/roadmap/danksharding – [Accessed: Dec. 2024].

13. "Sidechains". 2023.– Available from: https://ethereum.org/en/developers/docs/scaling/sidechains – [Accessed: Dec. 2024].

14. Goroshevsky, M. "Everscale Whitepaper". 2021.– Available from: https://everscale.network/files/ Everscale_Whitepaper.pdf – [Accessed: Dec. 2024].

15. Grybniak, S., Dmytryshyn, D., Leonchyk, Y., Mazurok, I., Nashyvan, O. & Shanin, R. “Waterfall: A scalable distributed ledger technology”. *IEEE 1st Global Emerging Technology Blockchain Forum: Blockchain & Beyond (iGETblockchain).* Irvine, USA. 2022. p. 1–6, https://www.scopus.com/record/display.uri?eid=2-s2.0-85153866003&origin=resultslist. DOI: https://doi.org/10.1109/iGETblockchain56591.2022.10087112.

16. Grybniak, S., Leonchyk, Y., Masalskyi, R., Mazurok, I. & Nashyvan, O. “Waterfall: Salto Collazo. Tokenomics”. *IEEE International Conference on Blockchain, Smart Healthcare and Emerging Technologies (SmartBlock4Health).* Bucharest, Romania. 2022. p. 1–6, https://www.scopus.com/record/display.uri?eid=2-s2.0-85148599293&origin=resultslist. DOI: https://doi.org/10.1109/SmartBlock4Health56071.2022.10034521.

17. Grybniak, S., Leonchyk, Y., Masalskyi, R., Mazurok, I., Nashyvan, O. & Shanin, R. “Decentralized platforms: Goals, challenges, and solutions.” *IEEE 7th Forum on Research and Technologies for Society and Industry Innovation (RTSI).* Paris, France. 2022. p. 62–67. DOI: https://doi.org/10.1109/RTSI55261.2022.9905225.

18. Antonenko, O., Grybniak, S., Guzey, D., Nashyvan, O. & Shanin, R. “Waterfall: Salto Collores. BFT Based PoS on BlockDAG”. *IEEE 1st Ukrainian Distributed Ledger Technology Forum (UADLTF).* 2023. DOI: https://doi.org/10.1109/UADLTF61495.2023.10548638.

19. Grybniak, S., Leonchyk, Y., Mazurok, I., Nashyvan, O. & Vorokhta, A. “Waterfall: Salto Collazo. High-Level Design of Tokenomics”. *Advances in Science, Technology and Engineering Systems Journal.* 2023; 8 (3): 231–243. DOI: https://doi.org/10.25046/aj080326.

20. Hafid, A., Hafid, A. & Samih, M. 2020. “Scaling Blockchains: A Comprehensive Survey”. *IEEE Access*. 2020; 8: 125244–125262, https://www.scopus.com/record/display.uri?eid=2-s2.0-85088708743&origin=resultslist. DOI: https://doi.org/10.1109/ACCESS.2020.3007251.

21. Han, R., Yu, J. & Zhang, R. “Analysing and Improving Shard Allocation Protocols for Sharded Blockchains”. *Proceedings of the 4th ACM Conference on Advances in Financial Technologies (Cambridge, MA, USA) (AFT’22)*. New York, USA: 2023. p. 198–216. DOI: https://doi.org/10.1145/3558535.3559783.

22.Kaempfer, G. “Fractal Scaling: From L2 to L3”. *Medium, StarkWare*. 2021. – Available from: https://medium.com/starkware/fractal-scaling-from-l2-to-l3-7fe238ecfb4f. – [Accessed: Dec. 2024].

23.“Subnetworks.” 2021. – Available from: https://kaspa.gitbook.io/kaspa/archive/archive/components/kaspad-full-node/reference/subnetworks-1. – [Accessed: Dec. 2024].

24. “The Liquid Network.” 2024. – Available from: https://liquid.net – [Accessed: Dec. 2024].

25. Liu, Y., Liu, J., Yin, J., Li, G., Yu, H. & Wu, Q. “Cross-shard Transaction Processing in Sharding Blockchains”. *Algorithms and Architectures for Parallel Processing*. Meikang Qiu (Ed.); Springer. 2020.

26. Maymounkov, P. & Mazieres, D. “Kademlia: A Peer-to-Peer Information System Based on the XOR Metric.” *In Peer-to-Peer Systems. IPTPS 2002. Lecture Notes in Computer Science*. Springer, Berlin, Heidelberg, Cambridge, USA. 2002; 2429: 53–65, https://www.scopus.com/record/display.uri?eid=2-s2.0-84947235017&origin=resultslist. DOI: https://doi.org/10.1007/3-540-45748-8_5.

27. Mazurok, I., Leonchyk, Y., Antonenko, O. & Volkov, K. “Smart contract sharding with proof of execution”. *Applied Aspects of Information Technology*. 2021; 4 (3): 271–281. DOI: https://doi.org/10.15276/aait.03.2021.6.

28. Miller, A., Xia, Y., Croman, K., Shi, E. & Song, D. “The Honey Badger of BFT Protocols”. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (Vienna, Austria) (CCS ’16).* New York, USA; 2016. p. 31–42. https://www.scopus.com/record/display.uri?eid=2-s2.0-84995495375&origin=resultslist. DOI: https://doi.org/10.1145/2976749.2978399.

29. “A Highly Scalable Public Blockchain via Adaptive State Sharding and Secure Proof of Stake”. 2019. – Available from: https://files.multiversx.com/multiversx-whitepaper.pdf – [Accessed: Dec. 2024].

30. Nguyen, C., Hoang, D., Nguyen, D., Niyato, D., Nguyen, H. & Dutkiewicz, E. “Proof-of-Stake Consensus Mechanisms for Future Blockchain Networks: Fundamentals, Applications and Opportunities”. *IEEE Access.* 2019; 7: 85727–85745, https://www.scopus.com/record/display.uri?eid=2-s2.0-85068826978&origin=resultslist. DOI: https://doi.org/10.1109/ACCESS.2019.2925010.

31. “Polygon Technology.” 2024. – Available from: https://polygon.technology – [Accessed: Mar. 2024].

32. Poon, J. & Buterin, V. “Plasma: Scalable Autonomous Smart Contracts”. 2017. – Available from: https://plasma.io/plasma-deprecated.pdf – [Accessed: Dec. 2024].

33. Poon, J. & Dryja, T. “The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments”. 2016.

– Available from: https://lightning.network/lightning-network-paper.pdf. – [Accessed: Dec. 2024].

34. “PREDA: A Programming Model to Scale out Smart Contracts”. 2023. – Available from: https://www.preda- lang.org/pdf/preda-model-sole.pdf – [Accessed: Dec. 2024].

35. Selkis, R. “A Messari report: Crypto theses for 2023. Technical Report”. *Messari, Inc.* 2023. – Available from: https://resources.messari.io/pdf/messari-report-crypto-theses-for-2023.pdf – [Accessed: Mar. 2024].

36. Singh, A., Click, K., Parizi, R., Zhang, Q., Dehghantanha, A. & Choo, K.-K. “Sidechain technologies in blockchain networks: An examination and state-of-the-art review”. *Journal of Network and Computer Applications*. 2020; 149: 102471, https://www.scopus.com/record/display.uri?eid=2-s2.0-85074299336&origin=resultslist. DOI: https://doi.org/10.1016/j.jnca.2019.102471.

37. Skidanov, A., Polosukhin, I. & Wang, B. “Nightshade: Near Protocol Sharding Design 2.0”. 2024. – Available from: https://discovery-domain.org/papers/nightshade.pdf. – [Accessed: Dec. 2024].

38. Sun, X., Yu, F., Zhang, P., Sun, Z., Xie, W. & Peng, X. “A Survey on Zero-Knowledge Proof in Blockchain”. *IEEE Network*. 2021; 35 (4): 198–205. DOI: https://doi.org/10.1109/MNET.011.2000473.

39. Sunyaev, A. “Distributed Ledger Technology”. *Springer International Publishing*. Cham: 2020 p. 265–299. DOI: https://doi.org/10.1007/978-3-030-34957-8_9.

40. Thibault, L. Sarry, T. & Hafid, A. “Blockchain Scaling Using Rollups: A Comprehensive Survey”. *IEEE Access*. 2022; 10: 93039–93054, https://www.scopus.com/record/display.uri?eid=2-s2.0-85136667976&origin=resultslist. DOI: https://doi.org/10.1109/ACCESS.2022.3200051.

41. Steen, M., Chien, A. & Eugster, P. “The Difficulty in Scaling Blockchains: A Simple Explanation”. *ArXiv*. 2021. DOI: https://doi.org/10.48550/arXiv.2103.01487.

42. “Architecture.” 2023. – Available from: https://docs.venom.foundation/learn/architecture/ – [Accessed: Dec. 2024].

43. “Annual Report 2024.” 2025. – Available from: https://s29.q4cdn.com/385744025/ files/doc_downloads/2024/Visa-Fiscal-2024-Annual-Report.pdf. – [Accessed: Dec. 2024].

44.“Testnet 8.” 2024. – Available from: https://stats.waterfall.network – [Accessed: Dec. 2024].

45.Yu, G., Wang, X., Yu, K., Ni, W., Zhang, A. & Liu, R. “Survey: Sharding in Blockchains”. *IEEE Access*. 2020; 8: 14155–14181. DOI: https://doi.org/10.1109/ACCESS.2020.2965147

46. Zamani, M., Movahedi, M. & Raykova, M. “RapidChain: Scaling Blockchain via Full Sharding”. *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (Toronto, Canada) (CCS ’18)*. New York, USA: 2018. p. 931–948, https://www.scopus.com/record/display.uri?eid=2-s2.0-85056891496&origin=resultslist. DOI: https://doi.org/10.1145/3243734.3243853.

47. Zheng, Z., Xie, S., Dai, H.-N., Chen, X. & Wang, H. “Blockchain challenges and opportunities: a survey”. *International Journal of Web and Grid Services*. 2018; 14 (4): 352–375. DOI: https://doi.org/10.1504/IJWGS.2018.095647.

Conflicts of Interest: The authors declare that they have no conflict of interest regarding this study, including financial, personal, authorship or other, which could influence the research and its results presented in this article

Author Igor Ye. Mazurok is a member of the Editorial Board of this journal. This role had no influence on the peer review process or editorial decision regarding this manuscript

Received 02.02.2025
Received after revision 19.03.2025
Accepted 22.03.2025

DOI: https://doi.org/10.15276/hait.08.2025.5 УДК 004.922

Віртуально необмежений шардинг для масштабованих розподілених реєстрів

Грибняк Сергій Сергійович[1)]
ORCID: https://orcid.org/0000-0001-6817-8057; s.s.grybniak@op.edu.ua. Scopus Author ID: 57962557300

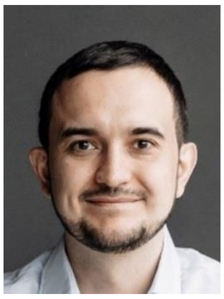
Леончик Євген Юрійович[2)]
ORCID: https://orcid.org/0000-0003-1494-0741; leonchyk@onu.edu.ua. Scopus Author ID: 57192064365
Мазурок Ігор Євгенович[3)]
ORCID: https://orcid.org/0000-0002-6658-5262; mazurok@onu.edu.ua. Scopus Author ID: 57210121184
Нашиван Олександр Сергійович[2)]
ORCID: https://orcid.org/0000-0001-8281-4849; o.nashyvan@op.edu.ua. Scopus Author ID: 57963260000
Шанін Руслан Васильович[2)]
ORCID: http://orcid.org/0000-0002-4414-1126; ruslanshanin@onu.edu.ua. Scopus Author ID: 55983005400
Ворохта Аліса Юріївна[4)]
ORCID: https://orcid.org/0000-0002-2790-1517; alisa.vorokhta@uni.lu. Scopus Author ID: 59184524100
[1)] Національний університет «Одеська політехніка», пр. Шевченка, 1. Одеса, 65044, Україна
[2)] Одеський національний університет ім. І. І.Мечнікова, вул. Дворянська, 2. Одеса, 65082, Україна
[3)] Waterfall DAO, Цуг, Швейцарія
[4)] Люксембурзький університет, пр. де л'Юніверсіте, 2. Еш-сюр-Альзетт, 4365, Люксембург

АНОТАЦІЯ

У роботі представлено підхід до масштабування децентралізованої платформи смарт-контрактів Waterfall, заснований на концепції ієрархічного фрактального шардингу. Незважаючи на потенціал технології розподіленого реєстру, її широке впровадження стримується проблемами масштабованості — зокрема, неможливістю пропорційно збільшувати пропускну здатність мережі із зростанням кількості учасників без шкоди для безпеки або децентралізації. Запропонована архітектура зменшує обчислювальне та мережеве навантаження шляхом розподілу транзакцій, смарт-контрактів та станів між фрактально організованими шардами, кожен з яких функціонує як орієнтований ациклічний граф. Це дозволяє залучати вузли з обмеженими ресурсами та досягати масштабованості не лише на рівні всієї системи, але й у її компонентах. У роботі описано механізми поділу та злиття шардів, маршрутизації транзакцій, динамічного розміщення смарт-контрактів, а також імовірнісну модель для оцінки ризику атаки на окремий шард. Проведено моделювання та представлено рекомендації щодо параметрів безпечного розміру шардів. Хоча розробка здійснювалась спеціально для платформи Waterfall, загальна концепція фрактального ієрархічного шардингу, а також її окремі компоненти, можуть бути адаптовані до інших блокчейн-систем, зокрема з модульною архітектурою або архітектурою, побудованою на основі орієнтованого ациклічного графа.

Ключові слова: фрактальний шардинг; смарт-контракти; технологія розподіленого реєстру; масштабованість

ABOUT THE AUTHORS



Sergii S. Grybniak - Ph.D in Computer Science, Department of Applied Mathematics and Information Technologies, Odesa National Polytechnic University. 1, Shevchenko Ave, Odesa, 65044, Ukraine
ORCID: https://orcid.org/0000-0001-6817-8057; s.s.grybniak@op.edu.ua. Scopus Author ID: 57962557300
Research field: Blockchain and directed acyclic graph technologies, distributed ledger technologies, data science, decentralized systems design and governance models

Грибняк Сергій Сергійович - доктор філософії з комп'ютерних наук, Національний університет «Одеська політехніка», пр. Шевченка, 1. Одеса, 65082, Україна

Yevhen Y. Leonchyk - (†2025) was a PhD in Physics and Mathematics, Associate Professor of Department of Mathematical Analysis. Odesa I. I. Mechnikov National University. 2, Dvoryanskaya Str. Odesa, 65082, Ukraine.
His contributions to the field of mathematical modeling, environmental and economic complex systems, and Distributed Ledger Technology were significant. Sadly, he passed away in 2025. This work is published in honor of his scientific achievements and contributions to the field.
ORCID: https://orcid.org/0000-0003-1494-0741; leonchik@ukr.net, Scopus ID: 57192064365
Research field: Mathematical modeling of computer, environmental and economic complex systems, blockchain technology

Леончик Євген Юрійович - (†2025) доктор філософії з фізико-математичних наук, доцент кафедри Математичного аналізу. Одеський національний університет ім. І. І. Мечникова, вул. Дворянська, 2. Одеса, 65082, Україна

Його внесок у розвиток математичного моделювання, дослідження екологічних та економічних комплексних систем, а також технології розподілених реєстрів (DLT) був значним. На жаль, він пішов із життя у 2025 році. Ця робота публікується на знак пошани до його наукових здобутків та внеску в розвиток галузі

Igor Y. Mazurok - PhD in Engineering Sciences, Senior Researcher, Waterfall DAO, Zug, Switzerland
ORCID: https://orcid.org/0000-0002-6658-5262; igor@mazurok.com, Scopus ID: 57210121184
Research field: Distributed computing, decentralized system design and modeling, artificial intelligence

Мазурок Ігор Євгенович - доктор філософії з технічних наук, старший науковий співробітник, Waterfall DAO, Цуг, Швейцарія

Oleksandr S. Nashyvan - Master of Software for Automated Systems. Odesa National Polytechnic University. 1, Shevchenko Ave. Odesa, 65044, Ukraine.
ORCID: https://orcid.org/0000-0001-8281-4849; o.nashyvan@op.edu.ua. Scopus Author ID: 57963260000
Research field: Software development, decentralized systems design, blockchain and directed acyclic graph technologies

Нашиван Олександр Сергійович - магістр программного обеспечения для для автоматизованих систем. Одеський національний політехнічний університет, пр. Шевченка, 1. Одеса, 65082, Україна

Ruslan V. Shanin - PhD in Physics and Mathematics, Associate Professor, Department of Mathematical Analysis. Odesa I. I. Mechnikov National University. 2, Dvoryanskaya Str. Odesa, 65082, Ukraine
ORCID: http://orcid.org/0000-0002-4414-1126; ruslanshanin@onu.edu.ua. Scopus Author ID: 55983005400
Research field: Real Functions, Harmonic Analysis on Euclidean spaces, Function Spaces Arising in Harmonic Analysis, Blockchain and Directed Acyclic Graph Technologies, Distributed Ledger Technologies, Decentralized Systems Design

Шанін Руслан Васильович - д-р філософії з фізико-математичних наук, доцент кафедри Математичного аналізу. Одеський національний університет ім. І. І. Мечникова, вул. Дворянська, 2. Одеса, 65082, Україна

Alisa Y. Vorokhta - PhD student in Computer Science, Interdisciplinary Centre for Security, Reliability, and Trust. University of Luxembourg, 2, Ave. de l'Université Esch-sur-Alzette, 4365, Luxembourg
ORCID: https://orcid.org/0000-0002-2790-1517; alisa.vorokhta@uni.lu. Scopus Author ID: 59184524100
Research field: Optimization Algorithms, High-Performance Computing, Data Science, Blockchain and Directed Acyclic Graph Technologies

Ворохта Аліса Юріївна - аспірант з Комп'ютерних наук, Міждисциплінарний центр безпеки, надійності та довіри. Люксембурзький університет, пр. де л'Універсіте, 2. Еш-сюр-Альзетт, 4365, Люксембург