Ladder: A Convergence-based Structured DAG Blockchain for High Throughput and Low Latency

Dengcheng Hu, Jianrong Wang, Xiulong Liu, and Hao Xu, *Tianjin University;* Xujing Wu, *Jd.Com, Inc;* Muhammad Shahzad, *North Carolina State University;* Guyue Liu, *Peking University;* Keqiu Li, *Tianjin University*

https://www.usenix.org/conference/nsdi25/presentation/hu

This paper is included in the Proceedings of the 22nd USENIX Symposium on Networked Systems Design and Implementation.

April 28–30, 2025 • Philadelphia, PA, USA

978-1-939133-46-5

Open access to the Proceedings of the 22nd USENIX Symposium on Networked Systems Design and Implementation is sponsored by

جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology

Ladder: A Convergence-based Structured DAG Blockchain for High Throughput and Low Latency

Dengcheng Hu[1], Jianrong Wang[1], Xiulong Liu[1], Hao Xu[1], Xujing Wu[2],
Muhammad Shahzad[3], Guyue Liu[4], Keqiu Li[1]
[1]*Tianjin University* [2]*Jd.Com, Inc* [3]*North Carolina State University* [4]*Peking University*

Abstract

Recent literature proposes the use of Directed Acyclic Graphs (DAG) to enhance blockchain performance. However, current block-DAG designs face three important limitations when fully utilizing parallel block processing: high computational overhead due to costly block sorting, complex transaction confirmation process, and vulnerability to balance attacks when determining the pivot chain. To this end, we propose Ladder, a structured twin-chain DAG blockchain with a convergence mechanism that efficiently optimizes parallel block processing strategy and enhances overall performance and security. In each round, a designated convergence node generates a lower-chain block, sorting the forked blocks from the upper-chain, reducing computational overhead and simplifying transaction confirmation. To counter potential adversarial disruptions, a dynamic committee is selected to generate special blocks when faulty blocks are detected. We implemented and evaluated Ladder in a distributed network environment against several state-of-the-art methods. Our results show that Ladder achieves a 59.6% increase in throughput and a 20.9% reduction in latency.

1 Introduction

Background: Within blockchain networks, nodes process incoming transactions and collectively engage in a consensus protocol, assembling the transactions into a block that is securely linked to the distributed ledger using hash pointers. In traditional chain-based structures, if multiple blocks are generated by different nodes simultaneously, only one is accepted while the others are discarded, resulting in the wastage of the computational resources expended in producing those blocks [21]. In contrast, Directed Acyclic Graph (DAG)-based structures allow those blocks to become part of the blockchain such that all blocks are included when generating the ledger [3, 6, 17–19, 22, 24, 34]. Broadly, DAG-based structures can be classified into two types: **tx-DAG**, where nodes represent transactions, and **block-DAG**, where nodes represent blocks composed of multiple transactions [32]. This paper focuses on block-DAGs as they offer better compatibility with existing protocols and provide stronger scalability than tx-DAGs. Broader structures in block-DAG blockchains, while intuitively improving performance through parallel block generation, may also increase transaction ordering overhead, complicate confirmation processes, and exacerbate specific security risks. *Therefore, this paper explores how to fully leverage the advantages of the DAG by optimizing the parallel block processing strategy.*

Limitations of Prior Art: While several promising block-DAG based blockchain designs have been proposed [18, 19, 24–26], they have the following three important limitations when processing parallel blocks.

1) High Sorting Cost. Nodes need to independently sort blocks to maintain consistency when multiple valid blocks are generated in parallel. In approaches such as Conflux [19] and Inclusive [18], every node is required to recursively sort the newly generated blocks along with their predecessors, which leads to considerable redundant computations. As each node independently determines the order without a coordinated strategy, inconsistencies in block views arise, causing additional computational overhead as nodes attempt to reconcile these discrepancies. This lack of synchronization in the sorting process results in high sorting costs, leading to delays in block confirmation, especially under high concurrency.

2) Complex Confirmation Process. In DAG-based blockchains, the validity of individual transactions depends on the confirmation of their respective blocks, often requiring intricate inter-block validation processes. As multiple forks emerge, nodes need to independently verify and reference forked blocks, resulting in a buildup of cross-references that complicates the confirmation process. For example, in Spectre [24], the block-voting mechanism relies on each node incrementally accumulating support for previous blocks, which can be cumbersome as nodes attempt to reconcile conflicting views on the validity of numerous competing forks. Similarly, Conflux [19] utilizes forward references for faster pivot chain confirmation, but the absence of a coordinated approach to

managing these forks causes transactions in non-standard blocks to gain confirmation weight at a slower rate, further prolonging the finalization process. This lack of an efficient mechanism to centrally handle forks leads to a fragmented confirmation process, where the full potential of DAG's parallelism remains underutilized.

3) Susceptibility to Balance Attacks. DAG blockchains define rules to identify the pivot chain [18, 19, 26], allowing multiple valid blocks generated simultaneously to be added to the ledger without conflict. However, when multiple candidate pivot chains emerge, the lack of coordination leads to independent node decisions, hindering prompt convergence on a single pivot chain. This lack of synchronization allows adversaries to exploit decentralized decision-making to delay ledger updates and maintain parity between competing chains. For instance, while Conflux [19] employs a probabilistic method to address these issues, its independent chain weight evaluation at each node may introduce inconsistencies, which adversaries can exploit for balance attacks, delaying convergence to a single pivot chain. As a result, the absence of a mechanism to manage competing chains may lead to delays, threatening blockchain stability and reliability.

These limitations suggest that incorporating an effective convergence mechanism could better coordinate and optimize parallel block processing.

Proposed Approach: In this paper, we propose Ladder, a block-DAG-based blockchain that utilizes a convergence mechanism to achieve high transaction throughput and low confirmation latency, addressing the three key limitations of existing approaches. Ladder is composed of two chains: an upper-chain and a lower-chain, as shown in Figure 1. The upper-chain consists of blocks containing transactions, while the lower-chain coordinates the convergence of forked blocks in the upper-chain. In each round, nodes generate blocks for the upper-chain using Proof of Work (PoW). When multiple blocks are generated in a round, they form a block set. One block is selected as the standard block, while the rest are designated as forked blocks. The upper-chain continues from the standard block, as shown in Figure 1, but all forked blocks are retained to preserve the computational effort involved in their creation. A convergence node, responsible for each round, records the order of forked blocks in the lower-chain. This allows nodes to reconstruct the ledger efficiently by traversing both chains, ensuring a consistent view of the blockchain.

Technical Challenges: We identified two key challenges in the design of Ladder that required resolution. The first challenge is *how to use the lower-chain to effectively converge PoW-generated upper-chain blocks.* A node may lack sufficient computational power to complete the PoW in a timely manner to generate a lower-chain block. Such delays could impede Ladder's progress. To resolve this, we eliminate the PoW requirement for lower-chain blocks, enabling any node to generate a lower-chain block swiftly. To prevent malicious lower-chain blocks, the node that generated the previous round's standard upper-chain block is selected as the convergence node for the current round. This process ensures that no single node can dominate unless it controls a significant proportion of the computational power, which is highly unlikely.

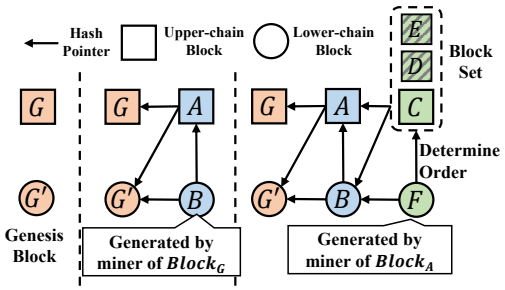


Figure 1: An introduction to the structure of Ladder.

The second challenge is *how to ensure system security by preventing adversaries from becoming convergence nodes.* An adversary could disrupt the blockchain by failing to produce or delaying the lower-chain block, hindering transaction confirmations and reducing throughput. Additionally, the adversary could ignore valid forked blocks or reference nonexistent blocks to disrupt the blockchain structure. To address this, Ladder introduces a super block in the lower-chain to override any faulty lower-chain block. If a fault is detected or a lower-chain block is not generated within a predefined duration, a committee forms and uses a Byzantine Fault Tolerance (BFT)-like consensus mechanism (*i.e.,* HotStuff) to generate a super block. BFT is a class of methods that ensures deterministic consensus within a fixed number of participants [16].

Key Advantages: In this paper, we propose a structured twin-chain DAG with a convergence mechanism for parallel block processing. The advantages of Ladder over the prior art are three-fold: i) compared to the block sorting by all nodes in [19, 24, 25], we introduce a mechanism that designates specific convergence nodes to handle upper-chain block sorting and record this information in lower-chain blocks, reducing sorting overhead and optimizing resource utilization; ii) compared to the complex transaction confirmation mechanism in [18, 19, 25, 26], our approach uses a convergence process to confirm transactions in unresolved upper-chain blocks, eliminating the need for intricate cross-referencing and weight accumulation, and thereby streamlining the transaction confirmation process; iii) compared to [18, 19, 25, 26], Ladder's structure inherently avoids additional computations for determining the pivot chain in the DAG topology, enhancing resistance to balance attacks and providing robust security guarantees against adversarial actions.

Summary of Experimental Results: We implemented Ladder and four state-of-the-art approaches: GHOST [26], Inclusive [18], Phantom [27], and Conflux [19]. Our experiments were conducted on a test bed of 80 servers deployed across a distributed network. Results show that Ladder outperformed prior approaches, achieving a 59.6% increase in throughput

Table 1: Comparison of block-DAG blockchains.

	Sorting by all nodes	Complex Confirmation logic	Balance attack resistance	Performance	
				Trans. txs / sec[1]	Confirmation time
GHOST	Yes	No	No	200	< 60min
Inclusive	Yes	Yes	No	350	< 1min
Spectre	Yes	Yes	-	-	< 1min
Phantom	Yes	Yes	No	40	< 1min
Conflux	Yes	Yes	No	2823	< 1min
OHIE	Yes	Yes	Yes	2513[2]	< 10min
Ladder	No	No	Yes	4506	< 1min

and a 20.9% reduction in transaction confirmation latency compared to the best-performing Conflux.

Limitation of Ladder: A limitation of PoW+BFT architecture is that while BFT consensus ensures security with fewer than $1/3$ malicious nodes, PoW's probabilistic nature does not guarantee this strict security threshold. For adversaries controlling less than 30% computational power, the committee can ensure with high probability that malicious nodes remain below $1/3$, satisfying BFT security requirements. However, the probability of more than $1/3$ of the committee being malicious in at least one of the R rounds is given by $1-(1-P)^R$, where P is the probability of selecting more than $1/3$ malicious nodes in a single BFT round. This probability increases with R, raising the risk of adversaries compromising the system by controlling over $1/3$ of the BFT committee, breaking consensus and disrupting system integrity. We analyze this limitation further in Sec. 4, where we study the effectiveness of increasing the BFT committee size on reducing the probability of malicious nodes exceeding $1/3$ of the committee.

2 Related Work

DAG for BFT Consensus: BFT consensus ensures agreement among distributed nodes with a bounded number of malicious participants and is more suited for permissioned networks [5, 16]. Several works have explored integrating DAG into BFT consensus, mainly in permissioned settings, such as Shoal [28], Bullshark [29], and Narwhal [8]. The fundamental difference between these solutions and Ladder is that Ladder is designed for permissionless blockchains. Ladder relies on PoW to ensure decentralization in trustless settings, while BFT is incorporated only as a lightweight fallback to handle adversarial scenarios.

DAG-based Structures: Prior work on utilizing DAGs in PoW-based blockchains can be broadly divided into two categories based on node representation: tx-based DAGs, where each node represents a single transaction [6, 17, 22], and block-based DAGs, where each node represents an entire block [18, 19, 24–26]. Tx-based DAGs enable decentralized validation but introduce challenges in transaction ordering and validation complexity. The independent nature of transactions increases communication overhead and complicates maintaining a consistent global transaction order. This is challenging in large-scale systems like Byteball [6], Tangle [22], and Nano [17], which face scalability and efficiency trade-offs. For block-based DAGs, a prominent block-based DAG design is GHOST [26], which extends Bitcoin's linear chain into a tree to enhance security. However, GHOST prioritizes security while offering limited improvements in throughput. Inclusive [18] extends GHOST by increasing block inclusion without full ledger ordering, limiting smart contract support. Similarly, Spectre [24] improves performance but sacrifices ledger ordering, restricting functionality. Phantom [25, 27] resolves the ordering issue but introduces a computationally expensive recursive sorting algorithm, hindering performance. Conflux [19] extends GHOST by leveraging a block-DAG structure to achieve high throughput. However, its reliance on probabilistic methods for balance attack defense leaves certain vulnerabilities unresolved.

Hierarchical and Parallel Structure: Meshcash [4] employs a hierarchical DAG, using block references as votes for confirmation. While enhancing security, this design disadvantages nodes with low computational power or poor network connectivity, leading to delayed confirmations and degraded performance. OHIE [37] improves throughput with a parallel chain architecture. However, each parallel chain in OHIE retains a linear blockchain structure, meaning block forks can still result in orphaned blocks. In contrast, Ladder utilizes a block-DAG structure to avoid valid orphaned blocks. Prism [2] adopts a decoupled design, using vote blocks to select a block for ledger ordering. However, vote blocks rely on PoW solely for voting, increasing resource consumption. Moreover, Prism's performance depends on near-ideal network conditions, such as a block propagation delay of less than one second. In contrast, Ladder packages transactions within vote blocks and does not require strict network assumptions, making it more adaptable to practical P2P networks.

Table 1 summarizes several related work, highlighting the limitations addressed by Ladder.

3 Detailed Design of Ladder System

In this section, we provide a detailed description of Ladder. We already provided an overview of Ladder in Sec. 1 under Proposed Approach. Here, we start with a system-level view of Ladder nodes and the system model. Based on this, we present various notations and terms that we use in this paper. Subsequently, we describe how nodes generate Ladder blocks and determine the pivot chain in the event of forking. We then explore various corner cases that may arise and how Ladder handles them. Finally, we end this section with a discussion about various performance-related aspects of Ladder.

3.1 System Model and Node Operations

We illustrate the system-level process of node participation in Ladder, as shown in Figure 2. Ladder operates under

[1] As measured in our experimental environment.

[2] We set the number of parallel chains to 40 in our experiments.

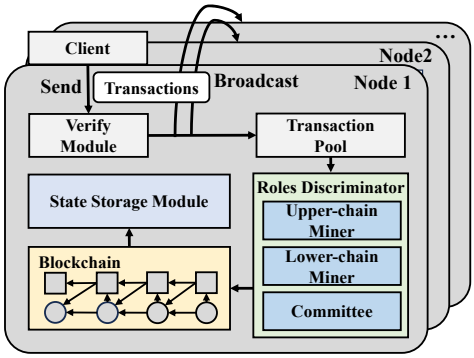


Figure 2: A system view of Ladder nodes.

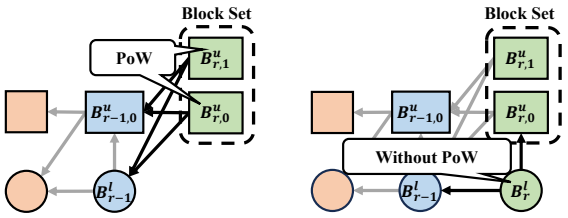
a δ-synchronous network assumption (following [21, 37]), where messages between honest nodes are delivered within a bounded time δ under normal conditions. The system tolerates temporary disruptions, such as message delays or node crashes, as long as the network eventually stabilizes. We assume that adversaries control at most p percent of the total computational power, where $p < 30\%$, ensuring the security of both upper-chain PoW and lower-chain BFT consensus with high probability. In Ladder, nodes participate as upper-chain miners, lower-chain miners, and/or BFT committee members, and are incentivized for their contributions to block generation. Upper-chain miners use PoW to generate blocks. Lower-chain miners handle transaction ordering and produce lower-chain blocks. In the event of waiting duration timeouts or faults, the BFT committee generates super blocks to ensure consensus continuity and maintain system integrity.

3.2 Model, Definitions, and Notations

We model Ladder using a DAG $\mathcal{G} = (\mathcal{B}, \mathcal{V}, \mathcal{E})$. Next, we describe in detail what $\mathcal{B}$, $\mathcal{V}$, and $\mathcal{E}$ represent.

Blocks $\mathcal{B}$: We represent the set of all valid blocks with $\mathcal{B}$. The set $\mathcal{B}$ consists of three types of blocks: B^u represents all valid blocks in the upper-chain. B^l represents all valid blocks in the lower-chain, and B^s represents all super blocks. Before we define these three types of blocks, we first introduce the concept of a *round*.

Round: A round in Ladder defines the position of a block within the upper- and lower-chains. Each round begins with the generation of the lower-chain block from the previous round and ends with the generation of the lower-chain block in the current round. During each round, one new block is added to the lower-chain and at least one block is added to the upper-chain. The upper-chain and lower-chain blocks generated in round r are denoted as B^u_r and B^l_r, respectively. Ladder does not require all honest nodes to have the same round number, as network delays may cause lower-chain blocks to be received at different times. The round number is simply used to track block order and maintain the local ledger. Even if nodes are temporarily in different rounds, the system ensures eventual consistency through block propagation and validation, within the bounds of the δ-synchronous network model.



(a) Upper-chain (b) Lower-chain

Figure 3: Normal operation of Ladder.

Upper-chain Blocks B^u: The upper-chain blocks in Ladder are generated by nodes using the PoW consensus mechanism. Despite being energy-intensive, PoW remains a cornerstone of blockchain security, and improving its performance and scalability is crucial for maintaining its relevance in decentralized and trustless environments. Upper-chain blocks are further categorized into standard and forked upper-chain blocks. Standard upper-chain blocks are those that become part of the upper-chain. We represent a standard upper-chain block in round r with $B^u_{r,0}$. $B^u_{r,0}$ contains two hash pointers. One pointer links to the standard upper-chain block $B^u_{r-1,0}$ of the preceding round, and the other links to the lower-chain block B^l_{r-1} of the preceding round. This process is shown in Figure 3(a). Forked upper-chain blocks are legitimate blocks generated by some nodes in the same round in which the standard upper-chain block is generated. We represent a forked upper-chain block in round r with $B^u_{r,i}$, where $i = 1, 2, 3, \ldots$ depending on how many forked upper-chain blocks were created during round r. A forked upper-chain block $B^u_{r,i}$ also contains two hash pointers. One pointer links to the standard upper-chain block $B^u_{r-1,0}$ of the preceding round, and the other to the lower-chain block B^l_{r-1} of the preceding round. The primary difference is that the upper-chain blocks in the next round will point to the standard upper-chain block of the current round, but not to forked blocks. In other words, the chain continues only from a standard upper-chain block, not from any forked upper-chain blocks. This process is shown in Figure 3(b).

Lower-chain Blocks B^l: Lower-chain block B^l_r in round r contains information about the standard upper-chain block, any forked blocks, and their respective sequence numbers. Lower-chain block B^l_r is generated in round r by the node that generated the standard upper-chain block $B^u_{r-1,0}$ in the previous round. The generation of lower-chain blocks is computationally inexpensive and does not require any PoW mechanisms. Lower-chain block B^l_r contains two hash pointers. One pointer links to the standard upper-chain block $B^u_{r,0}$ of the current round, and the other to the lower-chain block B^l_{r-1} of the preceding round. It also contains reference links to all forked upper-chain blocks in round r, facilitating the convergence of forks in the upper-chain.

Super Blocks B^s: A super block is added to the lower-chain when the referenced upper-chain blocks contain invalid trans-

actions from the current or previous rounds, or when a lower-chain block is not received within a predefined time duration despite the presence of valid upper-chain blocks. In Ladder, super blocks are generated through the HotStuff consensus mechanism [36]. Sec. 3.3 will provide further details on fault conditions and the generation process of the three types of blocks (upper-chain, lower-chain, and super).

Nodes $\mathcal{V}$: We represent the set of all nodes with $\mathcal{V}$ and any arbitrary node in $\mathcal{V}$ with V. The ratio of a single node V's computing power relative to the total computing power of all nodes is represented by $P(V)$, such that $\sum_{\forall V \in \mathcal{V}} P(V) = 1$.

Edges $\mathcal{E}$: We represent the set of link relationships among all blocks with $\mathcal{E}$. For example, if block $B^u_{r,0}$ points to block B^l_{r-1}, then $(B^u_{r,0}, B^l_{r-1}) \in \mathcal{E}$ represents a directed link from $B^u_{r,0}$ to B^l_{r-1}. More generally, expressing a link as (a,b), if $(a,b) \in \mathcal{E}$, a property of Ladder is that $a \neq b$ and $(b,a) \notin \mathcal{E}$.

Serial Numbers and Ledger Generation: To enable a node to generate the ledger from the blocks in the upper- and lower-chains of Ladder, we assign a serial number to each block. In any given round $r \geq 0$, the lower-chain block has the serial number $2r+1$ and the standard upper-chain block has the serial number $2r$. If round r has any forked upper-chain blocks, then the serial numbers of forked blocks are denoted by adding a subscript i to the value $2r$, where $i = 1, 2, 3, \ldots$. We represent the serial number of any block B with $O(B)$. For example, if B^u is a standard upper-chain block in round 3, then $O(B^u) = 6$. If B^u is the second forked upper-chain block in round 3, then $O(B^u) = 6_2$. Similarly, if B^l is a lower-chain block in round 3, then $O(B^l) = 7$. Suppose rounds 0, 1, 4, and 5 do not have any forked blocks but only standard upper-chain blocks. Suppose round 2 has one forked block, and round 3 has three forked blocks. To generate the ledger, the upper-chain blocks will be arranged in the following sequence: $0, 2, 4, 4_1, 6, 6_1, 6_2, 6_3, 8, 10, \ldots$. Transactions are inherently sequential, so while parallel block processing enables higher throughput, ordering the blocks ensures all transactions follow the correct sequence. This guarantees a totally ordered ledger, allowing Ladder to support not only simple transfer transactions but also complex smart contract executions.

3.3 Block Generation

Next, we describe the generation process for the different block types mentioned in Sec. 3.2.

Upper-chain Blocks: In each round r, each node attempts to find a nonce that meets the difficulty target required for block generation. Once a node finds a valid nonce, it broadcasts the resulting block to all nodes in the network. Round r concludes when the lower-chain block B^l_r is generated. Before the round concludes, it is common for multiple upper-chain blocks to be generated by different nodes. In such cases, one of these blocks is designated as the standard upper-chain block, while the others are classified as forked upper-chain blocks. If applicable, information about the standard and forked blocks in round r is stored in the corresponding lower-chain block B^l_r.

Lower-chain Blocks: Suppose we are currently in round r. This means that the standard upper-chain block and any forked upper-chain blocks from round $r-1$ have already been determined, with this information stored in the lower-chain block B^l_{r-1} of the previous round. In the current round r, the node whose block was designated as the standard upper-chain block in round $r-1$ is responsible for generating the lower-chain block. By generating the lower-chain block, Ladder achieves fast convergence of upper-chain blocks. Additionally, only the miner of the standard upper-chain block from round $r-1$ is required to generate the lower-chain block. The node collects upper-chain blocks for a predefined duration, checks for faults, discards faulty blocks, and selects the standard upper-chain block $B^u_{r,0}$ for round r according to the *Hardest Chain Principle*. This block must have hash pointers to the standard upper-chain block $B^u_{r-1,0}$ and lower-chain block B^l_{r-1}. It marks the remaining upper-chain blocks as forked and assigns them serial numbers $(2r)_1$, $(2r)_2$, and so on. It inserts this information into the lower-chain block B^l_r for round r, along with the hash pointers for $B^u_{r,0}$ and B^l_{r-1}. By producing lower-chain blocks, the convergence node exclusively records forked blocks and ensures a globally consistent transaction order. Unlike other solutions, Ladder optimizes block ordering while preserving decentralization, thereby enhancing both efficiency and finality. An upper-chain block in round r may reference $B^u_{r-1,0}$ and B^l_{r-1} but fail to reach the lower-chain block generator in time. Instead, it may arrive at the generator for round $r+e$ ($e \geq 1$) before its lower-chain block is generated. Consequently, this block will be sorted in round $r+e$, and it will be classified as a forked upper-chain block rather than a standard one.

Super Block: Suppose node V is in round r. It detects a fault if either of the following two situations occurs.

1. Node V receives a lower-chain block for round r, but the referenced upper-chain blocks contain transactions that conflict with those in the current or previous rounds.

2. Node V does not receive a lower-chain block for round r within a predefined time duration, despite receiving several valid upper-chain blocks.

To resolve such faults, Ladder employs the BFT consensus mechanism. Since lower-chain blocks must be unique, BFT consensus guarantees agreement on a single canonical lower-chain block among all nodes. As a deterministic consensus mechanism, BFT consensus has been widely used in various fields [1, 5, 7, 15, 30, 31, 35, 36]. The HotStuff consensus [36] is a BFT-based consensus executed by a committee comprising the nodes that generated the most recent n standard upper-chain blocks. HotStuff is selected for its simplicity, efficiency, and linear message complexity. Compared to other BFT mechanisms in some PoS proposals, HotStuff offers

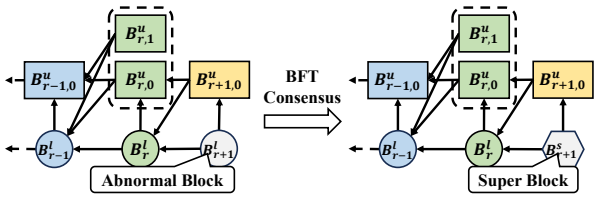


Figure 4: Example of the use of super block in Ladder.

superior scalability, making it suitable for super block generation. Upon committee formation, the leader is selected via VRF [12]. During Ladder's operation, if a node detects an issue, it notifies the committee members. The committee verifies the fault, discards conflicting upper-chain blocks, and reselects a standard upper-chain block based on the *Hardest Chain Principle*. The committee leader then generates a super block, replacing the lower-chain block, as shown by the hexagonal block B^s_{r+1} in Figure 4. This super block contains hash pointers to the previous round's lower-chain block and the current round's upper-chain block, along with references to any forked upper-chain blocks. Subsequent upper- and lower-chain blocks are generated through the standard process described earlier. Notably, a dynamically sized committee may cause inconsistencies in node perceptions, so we adopt a fixed size to ensure information consistency.

At the end of each round, nodes are rewarded based on their roles in that round, incentivizing sustained participation.

3.4 Hardest Chain Principle

Unlike Bitcoin, to enable faster transaction insertion on the ledger, a blockchain may produce blocks at a high rate. However, this can cause honest nodes to link blocks to different sub-chains based on their local views, fragmenting their computing power. An adversary can take advantage of this. It can concentrate all its computing power on a target sub-chain to attack it. To address this issue, GHOST [26] proposes the *heaviest chain principle*, where the pivot chain is determined by the number of sub-blocks linked to a block. However, GHOST is vulnerable to liveness attacks, where an adversary delays transaction confirmations by creating multiple forks. To mitigate this, Conflux [19] proposed GHAST, which introduces dynamic block weight (*e.g.,* special blocks with high weights) to mitigate liveness attacks. While GHAST improves resilience, it may not fully address balance attacks.

Ladder proposes the *Hardest Chain Principle*, which extends the *heaviest chain principle* with dynamic weight adjustment to support efficient parallel block processing. Unlike the *heaviest chain principle*, which relies on the number of sub-blocks, the *Hardest Chain Principle* selects the pivot chain based on the cumulative difficulty of blocks in a sub-tree. The difficulty of block B_i, denoted as $\mathcal{D}(B_i)$, is calculated by the number of leading zeros in its hash and the hashes of all blocks in its sub-tree, as defined by the following equation:

$$\mathcal{D}(B_i) := \sqrt{\mathcal{Z}(B_i)} + \sum_{B_j \in S_i} \mathcal{D}(B_j) \quad (1)$$

Here, $\mathcal{Z}(B_i)$ represents the number of leading zeros in the hash of block B_i, and S_i denotes the set of all legal blocks with hash pointers pointing to block B_i. Putting $\mathcal{Z}(B_i)$ under square root achieves two properties. First, in the case of a fork, the side with more legal blocks following the forked block has higher difficulty and is selected. This makes it easier to select a side. Second, if the number of legal successor blocks on both sides is equal, as long as the number of leading zeros is different in all the sub-blocks, the difficulty of the two sides will be unequal, enabling the selection of one of the sides. This significantly reduces the likelihood of a balance attack. If both sub-chains have m legal blocks after the fork position, it can be proven that the probability of the blocks having equal difficulty is bound by $P \leq \frac{1}{12}(1-\frac{1}{4^k})$, where k is the maximum number of leading zeros in the hash value.

We now prove that the side with more legal blocks following the forked block has higher difficulty. Consider a scenario where a fork leads to the formation of two sub-chains. Let sub-chain 1 have m blocks with a leading zeros each, and sub-chain 2 have $m-t$ blocks with b leading zeros, where $a < b$. In other words, each block in sub-chain 2 has more leading zeros than those in sub-chain 1. The generation probability of a block with x leading zeros is given by $P = \frac{1}{2^{x+1}}$. Thus, the generation probability of all m blocks in sub-chain 1 is $\prod_{i=1}^{m} P_i = \prod_{i=1}^{m} \frac{1}{2^{a+1}} = \frac{1}{2^{m(a+1)}}$. Similarly, the generation probability of sub-chain 2 is $\frac{1}{2^{(m-t)(b+1)}}$. If the generation probabilities of the two sub-chains are equal (*i.e.*, the computing power is equally divided between the two chains), then $b = \frac{ma+t}{m-t}$. To show that sub-chain 1 has greater weight than sub-chain 2, we must prove $\sum_{i=1}^{m} \sqrt{a} > \sum_{i=1}^{m-t} \sqrt{b}$. Alternatively, we need to demonstrate $m\sqrt{a} > (m-t)\sqrt{\frac{ma+t}{m-t}}$, which can be simplified to $m^2 a > (m-t)(ma+t)$, implying $ma > m-t$. The conclusion holds given that $m > 0$, $a > 1$, and $t > 1$. This proves that the side with a greater number of valid blocks following a given block must have a higher difficulty.

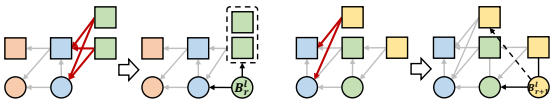
The *Hardest Chain Principle* can mitigate liveness attacks and balance attacks by selecting the chain with the highest cumulative difficulty, where a block's difficulty is determined by the number of leading zeros in its hash and the hashes of all blocks in its subtree. This approach ensures that an attacker must generate blocks with higher difficulty to influence the pivot chain, requiring substantial computational power and making such an attack economically infeasible. Furthermore, the *Hardest Chain Principle* can serve as an alternative to GHOST [26] and GHAST [20] in their respective systems.

3.5 Exception Handling in Block Generation

Exceptions in Upper-Chain Blocks: We categorize exceptions in the upper-chain into the following cases:

Invalid block: An invalid block is a block that contains duplicate or conflicting transactions either with itself or with previous upper-chain blocks. Its creator receives no reward.

Forks: Two types of forks may occur in the upper-chain:



(a) Tip fork in upper-chain (b) Chain fork in upper-chain

Figure 5: Exception cases in upper-chain.

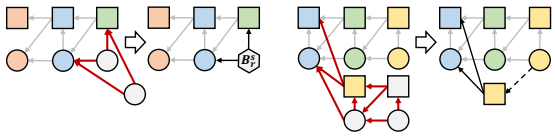
tip forks and chain forks. A tip fork occurs when multiple valid upper-chain blocks are generated in round r. In this case, the convergence node selects one block as the standard block and designates the others as forked blocks for round r. The convergence node then generates block B_r^l to include this information, as shown in Figure 5(a). A chain fork occurs when a valid upper-chain block is generated in round r but does not arrive in time at the node responsible for generating the lower-chain block for round r, instead reaching the node responsible for round $r+1$. In this case, the lower-chain block B_{r+1}^l contains a pointer to this upper-chain block as a forked block, as shown in Figure 5(b). Such an upper-chain block is never used as a standard upper-chain block.

Exceptions in Lower-Chain Blocks: We also categorize exceptions in the lower-chain into the following cases:

Block timeout: Block timeout occurs when a node belonging to the BFT committee does not receive a lower-chain block within a specified time. In such cases, the committee initiates HotStuff consensus and generates a super block.

Invalid block: Similar to invalid upper-chain blocks, a lower-chain block is considered invalid if it contains faulty information or is generated by an unauthorized node. Invalid blocks are ignored. However, if an invalid block is generated by the convergence node of the current round, the BFT consensus is triggered to generate a super block.

Fork: If a lower-chain block is not generated by the convergence node in any given round, it is simply ignored by the nodes. Lower-chain forks can also be classified into tip forks and chain forks. A tip fork occurs in the lower-chain when the node responsible for generating the lower-chain block malfunctions and generates more than one lower-chain block in a given round, as shown in Figure 6(a). This triggers the HotStuff consensus to generate a super block B_r^s, where the node that malfunctioned is not allowed to participate. A chain fork occurs in the lower-chain when different nodes accept different upper-chain blocks as standard blocks in a given round, and consequently accept lower-chain blocks from different nodes in the next round. This results in the propagation of two separate Ladders in the network, as shown in Figure 6(b). When eventually any given node detects the existence of multiple Ladders, it employs the *Hardest Chain Principle*, as discussed in Sec. 3.4, to select one Ladder and discard the rest. Any Ladder may contain super blocks in place of lower-chain blocks. In such cases, when applying the *Hardest Chain Principle*, the difficulty of the super block is quantified by the number of nodes participating in HotStuff.



(a) Tip fork in lower-chain (b) Chain fork in lower-chain

Figure 6: Exception cases in lower-chain.

Exceptions in Super Blocks: A super block is considered invalid if it contains conflicting information or fails committee validation due to adversarial behavior. Since the committee uses BFT to generate super blocks, such issues can be mitigated through BFT's internal mechanisms as long as the proportion of adversarial nodes remains below $1/3$. When an invalid super block is detected, the committee initiates a view change [36] and elects a new leader to restart consensus, ensuring super block reliability.

3.6 System Analysis

This section discusses Ladder's performance in throughput, latency, and scalability, as well as the bootstrapping process.

Throughput: Effective utilization of a node's computing power is key to improving system throughput. Nodes use PoW to generate blocks. Following the *Hardest Chain Principle*, they can become miners of standard upper-chain blocks. This grants them the role of convergence nodes, giving them the right to generate lower-chain blocks. Ladder improves upon GHOST and GHAST for pivot chain selection while maintaining comparable computational overhead, as the required information can be derived from block metadata. Convergence nodes determine block order, reducing network-wide coordination overhead while preserving throughput. The lower-chain allows nodes to package more transactions without PoW, increase rewards, and eliminate computational bottlenecks, improving efficiency.

Latency: In Ladder, the analysis of Bitcoin's longest chain principle and confirmation rule is still applicable for the *Hardest Chain Principle*. Consider a scenario where the adversary fully controls the computing power of malicious nodes. Let p and q represent the proportion of total computing power of honest and malicious nodes, respectively, where $p+q=1$. After a transaction is included in a block, a waiting period of z upper-chain blocks is necessary to prevent selfish mining and anomalies in block processing caused by the adversary. Recall that the explanation of lower-chain forking cases was presented in Sec. 3.5. The adversary's selfish mining behavior follows a Poisson distribution, where λ represents the expected number of upper-chain blocks secretly generated by the adversary during the waiting period of z upper-chain blocks. Since the number of generated upper-chain blocks is proportional to computing power, we have $\frac{\lambda}{z} = \frac{q}{p}$, or equivalently, $\lambda = \frac{zq}{p}$. Thus, the probability of the adversary producing k upper-chain blocks during the waiting period of z

upper-chain blocks can be calculated using the formula $\frac{\lambda^k e^{-\lambda}}{k!}$. In the case where the adversary generates k (where $k \leq z$) upper-chain blocks, the probability of catching up to the z revealed upper-chain blocks is $(\frac{q}{p})^{(z-k)}$. If $k > z$, the probability of catching up is 1. Consequently, the probability of being caught up by the adversary after waiting for z upper-chain blocks is given by $1 - \sum_{k=0}^{z} \frac{\lambda^k e^{-\lambda}}{k!}(1 - (\frac{q}{p})^{(z-k)})$.

Scalability: Ladder demonstrates excellent scalability, allowing nodes to join or leave the system freely without affecting the PoW of other nodes. Furthermore, as the number of nodes increases, the system's total computing power grows, thereby optimizing parallel block processing, improving throughput and reducing confirmation delays. In contrast, solutions like OHIE [37] extend the ledger to multiple parallel chains, which improves throughput. However, due to their chain-like structure, OHIE's parallel chains still face the issue of orphan blocks. A promising future direction is to combine these methods, utilizing DAG-based parallel chains with local consensus and lightweight global synchronization, which could enhance throughput while ensuring consistency.

Bootstrapping: Ladder requires an initial BFT committee to operate. However, relying on a global random selection process could render the system susceptible to Sybil attacks [9]. To address this, the system employs Nakamoto consensus during the bootstrap phase, then transitions to the Ladder protocol after k blocks have been generated [14].

4 Parameter Settings

BFT Committee Size: In Ladder, the presence of a BFT committee may introduce two security risks: selfish mining and adversaries exceeding $1/3$ of the committee. Therefore, the BFT committee size is critical to system security, as it determines which nodes are responsible for generating the super block.

To mitigate selfish mining, a larger committee size is advantageous, as it increases the difficulty of generating super blocks, making it harder for adversaries to withhold and release blocks to manipulate the chain order [10]. We show in Appendix A that the probability of a selfish mining attack occurring is bounded by $P = \frac{1}{2^{2\sqrt{k}}}$.

For adversaries exceeding $1/3$ of the committee, as noted in the Limitation of Ladder section (Sec. 1), Ladder requires malicious nodes within the committee remain below $1/3$. However, due to the probabilistic nature of PoW, this guarantee cannot be strictly ensured. Appendix B shows that this probability decreases exponentially as the committee size increases. Furthermore, over multiple rounds, the probability of at least one successful attack accumulates. In Table 2, we present the per-round probability of adversarial dominance and the number of rounds required for the cumulative probability of at least one successful attack to reach 99%. As the committee size grows, the probability of more than $1/3$ adversaries decreases, and the required number of rounds increases.

Table 2: Impact of Committee Size on Adversarial Risk

Committee Size	120	180	240	300
Probability	4.9×10^{-3}	1.8×10^{-4}	6×10^{-6}	1.96×10^{-7}
Rounds for 99%	9.2×10^{2}	2.6×10^{4}	7.7×10^{5}	2.4×10^{7}

For example, when the committee size is 300, the probability of selfish mining is approximately 4.67×10^{-11}, while the probability of more than $1/3$ adversaries in a single round is about 1.96×10^{-7}. This event requires roughly 2.4×10^{7} rounds to reach a cumulative probability of 99%.

Node Incentive Mechanism: As previously introduced, Ladder employs an incentive model to reward nodes for their participation. The block generation rewards in the upper- and lower-chain are similar and correlated with the number of valid transactions stored in the block. The super block generation reward is evenly distributed among all participating committee members in the BFT consensus algorithm. If a block is not selected as the standard upper-chain block but is selected as a forked upper-chain block, its reward is influenced by the difference in the number of rounds between its creation and its inclusion in the upper-chain. Specifically, this reference is added to a lower-chain block. To calculate the reward accurately, the reward should be multiplied by a block generation reward factor, which is less than 1. Let the upper-chain block B^u be linked to block B. The block generation reward factor is defined as $S = \frac{e^{l-d}}{1+e^{l-d}}$, where $d = R(B^u) - R(B)$ represents the difference in the number of rounds, and l is a manually set parameter that determines the rate of decrease of the block generation reward as the difference in rounds approaches l. For the lower-chain block generation node, the reward received for adding a reference to an upper-chain block should also be multiplied by the block generation reward factor. This incentivizes the lower-chain block generation node to promptly reference upper-chain blocks within lower-chain blocks. Both the block generation reward and the reward for adding a reference to an upper-chain block are stored in the lower-chain block for the respective round.

The incentive for the node to mine a single upper-chain block is calculated as $r_u = Block\ Reward + \sum_{Tx \in B} Txfee * (1 - \alpha)$. Similarly, for the convergence node, the incentive to generate a lower-chain block is determined by $r_l = Block\ Reward + \sum_{B \in round} \sum_{Tx \in B} Txfee * \alpha_i$. Next, we explain what α_i represents. The blocks in each round are generated by different nodes, and each node selects the transactions to include in its block from the transaction pool. Thus, it is possible that one or more transactions get included in multiple blocks generated by different nodes in any given round, resulting in duplicate transactions. α_i represents the proportion of transaction i among all duplicate transactions in a given round. Formally, let x_i represent the number of occurrences of transaction i that appears at least twice among all the blocks generated in a given round. Then $\alpha_i = \frac{x_i - 1}{\sum_{\forall i}(x_i - 1)}$.

To execute a double spending attack with V, the attack

involves generating n blocks during its phase, and the cost for each individual block generation is denoted as c. According to Hoeffding's inequality, take $\varepsilon = 1/2 - q$, $Pr[|A| \geq n/2] \leq \exp(-\frac{(1-2q)^2 n}{2}) = \exp(-\frac{(p-q)^2 n}{2})$. Then, to make the attack gain in the desired sense not to exceed the cost, $Pr[|A| \geq n/2] * V \leq \frac{n}{2}(c - r_d)$ is needed to obtain $V \leq \frac{n(c-r_d)}{2exp(\frac{(p-q)^2 n}{2})}$.

Waiting Time Duration: Following the generation of an upper-chain block in a specific round, the remaining nodes continue mining upper-chain blocks until they receive the lower-chain block from the designated convergence node. If the lower-chain block is not produced within the predefined waiting duration, the committee triggers the BFT consensus. Next we discuss how to set the waiting time duration.

Let W_i^j represent the waiting time for node j to receive a block with sequence number i, and T_i^j represent the timestamp when node j receives a block from node $V(i)$. We use $V(B)$ to represent the node that generated block B. $PT(V(i), j)$ indicates the time it takes for a block to propagate from node $V(i)$ to node j, and $MT(i)$ represents the mining time required by node $V(i)$ to generate a single block. Recall that the upper-chain block in round r is identified by the serial number $2r$, while the lower-chain block is denoted by the serial number $2r+1$. Assuming equal computing power, the mining time is the same for all nodes, while the propagation time varies. The waiting time W_i^j solely accounts for the propagation, excluding mining time, since any nonce can be used in the generation of lower-chain blocks. Thus, $W_i^j = PT(V(i), j)$. A node can estimate the mining and propagation times by analyzing the timestamps of the latest upper- and lower-chain blocks. Formally, this can be expressed as Eq.(2). Here, $MT(B^u_{r-1})$ is the mining time for the upper-chain block in round $r-1$, $PT(V(B^u_{r-1}), j)$ is the propagation time from the upper-chain block's generator to node j, and $PT(V(B^l_{r-1}), j)$ is the propagation time from the lower-chain block's generator to node j.

$$W^j_{2r+1} \approx \underbrace{(T^j_{2(r-1)} - T^j_{2(r-2)+1})}_{MT(B^u_{r-1})+PT(V(B^u_{r-1}),j)} - [\underbrace{(T^j_{2(r-2)} - T^j_{2(r-3)+1})}_{MT(B^u_{r-2})+PT(V(B^u_{r-2}),j)} - \underbrace{(T^j_{2(r-1)+1} - T^j_{2(r-1)})}_{PT(V(B^l_{r-1}),j)}] \quad (2)$$

The node responsible for generating the legal lower-chain block in round $r-1$ is the same as the node that generated the upper-chain block in round $r-2$ (denoted as $V(B^u_{r-2}) = V(B^l_{r-1})$), and the time for each node to generate a single block is equal ($MT(B^u_{r-1}) = MT(B^u_{r-2})$). Consequently, Eq.(2) can be simplified as $W^j_{2r+1} \approx PT(V(B^u_{r-1}), j)$. This means that the waiting time for node j to receive a block in the current round is equivalent to the propagation time it takes for a single block to propagate from the previous round's upper-chain block generator to node j.

The analysis presented above demonstrates that in Ladder, any given node can determine the waiting time for lower-chain blocks by utilizing the historical information it has collected regarding the arrival times of the standard block in round $r-1$ and lower-chain blocks in the next round r.

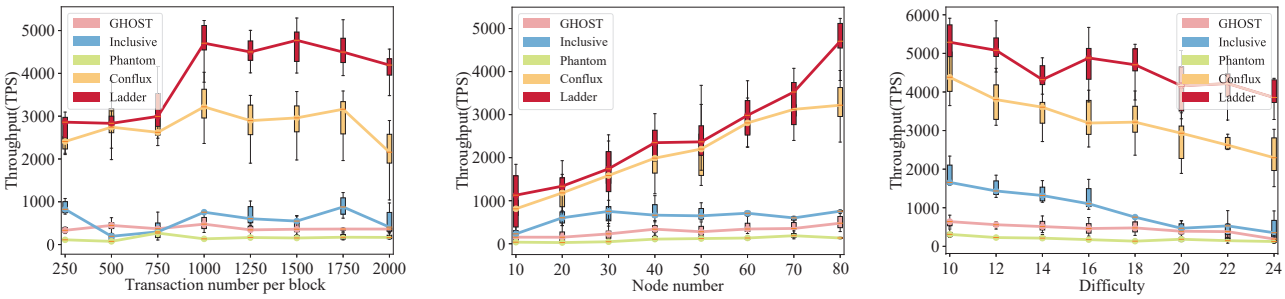
5 Evaluations

In this section, we present results from a comprehensive evaluation of Ladder and compare them with prior approaches.

Setup and Experiments: We implemented a prototype of Ladder and compared it with four recent approaches: GHOST [26], Inclusive [18], Phantom [27], and Conflux [19]. Experiments are conducted on an 80-node LAN testbed, with each machine equipped with an Intel(R) Core(TM) i5-4590 CPU @3.30 GHz and 8 GB of RAM. The inter-node communication latency is set to approximately 80-120 ms to align with real-world network conditions [33]. We use a P2P communication protocol where each node is connected to approximately 10 other nodes. Experiments are run in a controlled, adversary-free environment to observe Ladder's optimal performance without adversarial interference. To align with Conflux, which confirms a block after 6 subsequent blocks, Ladder also confirms a block after 6 subsequent upper-chain blocks. Unlike using a fixed block interval [19, 37], we employ PoW for block generation. In our experiments, the throughput results of previous approaches match their reported values under the following conditions: block difficulty requires 18 leading zeros, and each block contains 1000 randomly generated payment transactions (each approximately 300 bytes), as derived from our interpretation of the protocol specifications.

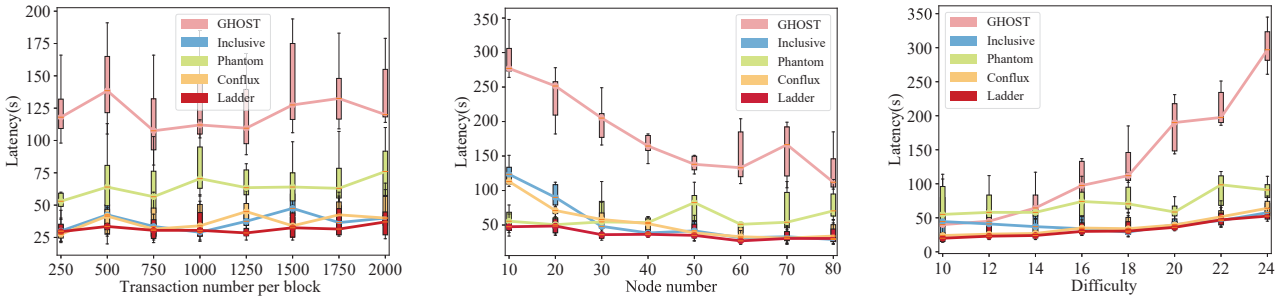
Metrics: We primarily evaluate the end-to-end performance of Ladder and prior approaches. Throughput is measured as the number of confirmed transactions per second (TPS), while confirmation latency is defined as the time from a transaction's submission to its block confirmation. All experiments are repeated 10 times. We conduct experiments over a 10-minute stable operation starting from the genesis block, following Conflux's methodology [19] to ensure a consistent evaluation period. Metrics are recorded every minute, yielding 10 data points per experiment and 100 in total across all runs. This ensures statistically meaningful measurements while accounting for PoW's inherent randomness in block generation intervals. Box plots are used to assess variance, preventing short-term fluctuations from distorting metric trends. To capture metric variations caused by PoW's stochastic nature, we use whiskers, which extend to the most extreme data points within $1.5 \times IQR$ from the first ($Q1$) and third ($Q3$) quartiles, minimizing the impact of outliers. Additionally, we use line plots with points representing the medians of the metrics.

Aggregate Observations: The results show that Ladder outperformed all prior approaches. Among prior approaches, Conflux performed the best, with a median throughput of 2823 TPS and a confirmation latency of 43 seconds. Compared to Conflux, Ladder showed a 59.6% improvement in



(a) Throughput with different block sizes (b) Throughput with different number of nodes (c) Throughput with different block difficulties

Figure 7: Throughput of Ladder and its comparison with prior approaches.



(a) Latency with different block sizes (b) Latency with different number of nodes (c) Latency with different block difficulties

Figure 8: Confirmation latency of Ladder and its comparison with prior approaches.

throughput and a 20.9% reduction in latency under the same settings. Ladder achieved a median throughput of 4506 TPS and a confirmation latency of 34 seconds. Next, we analyze the impact of block size, network size, and difficulty on the performance of Ladder and prior approaches.

Impact of Block Size: Figures 7(a) and 8(a) show the throughput and confirmation latency of various approaches across different block sizes under a difficulty level of 18. Results indicate that Ladder outperformed all prior approaches across the evaluated block sizes. For block size below 1000, Ladder exhibits lower throughput. As the block size increases from 1000 to 1750, throughput remains stable between 4000 and 5300 TPS. In this stage, the increased transaction capacity of larger blocks compensates for the growing propagation delay, maintaining stable throughput. However, at a block size of 2000, both Ladder and Conflux show a noticeable drop in throughput, likely due to higher network overhead and resource strain from larger block propagation. Throughput in Ladder fluctuates by around 1000 TPS, while latency varies by approximately 20 seconds. These fluctuations are primarily caused by PoW's randomness in block generation, with additional contributions from LAN congestion and scheduling delays affecting transaction propagation. Despite these fluctuations, Ladder maintains the highest efficiency among all tested approaches.

Impact of Network Sizes: Figures 7(b) and 8(b) show the throughput and confirmation latency of various approaches across different network sizes, with a fixed block size of 1000 and a difficulty level of 18. As the network size grows, throughput increases significantly, primarily due to the higher total computational power from more nodes participating in the PoW, resulting in more frequent block generation. Ladder's median throughput rises from 1043 TPS with 10 nodes to 4506 TPS with 80 nodes. Larger network sizes may introduce longer block propagation delays. However, confirmation latency decreases from a median of 47 seconds with 10 nodes to 34 seconds with 80 nodes, a reduction of 27.7%. The primary reason for this improvement is that increased computational power accelerates block generation, reducing the time required for transactions to be included in a block. This helps mitigate the impact of longer propagation delays by enabling faster confirmation through parallel processing.

Impact of Difficulty: Figures 7(c) and 8(c) show the throughput and confirmation latency at different block difficulty levels with a block size of 1000. Block difficulty significantly affects both throughput and confirmation latency. As the difficulty increases, throughput decreases, and confirmation latency increases across all prior approaches. Specifically, when the difficulty level is 10, the median throughput of Ladder reaches 5314 TPS, which is 27.2% higher than at level 24. This behavior occurs because higher difficulty reduces the block generation frequency, resulting in longer confirmation times. Despite the decline in performance with increasing difficulty, Ladder consistently outperforms all prior approaches.

Result Analysis: Ladder achieves higher throughput and lower latency across varying conditions due to several key

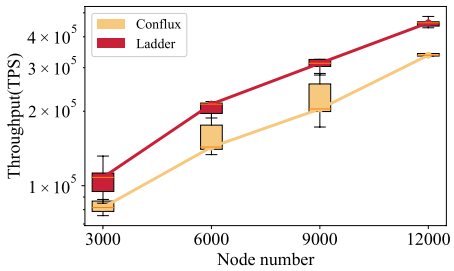


Figure 9: Throughput under large-scale node conditions.

factors. The convergence nodes streamline ledger consolidation by handling transaction ordering. This reduces the computational burden on other nodes, allowing them to focus on parallel block generation. Additionally, Ladder's convergence mechanism simplifies lower-chain block generation and transaction confirmation, thereby reducing overall system complexity. Even during anomalies that require committee intervention, upper-chain block production continues until a new lower-chain block is generated.

Large Scale Simulation: The simulation is conducted on a server with 128GB of memory, with time delays introduced to simulate the block generation and broadcasting processes. Block generation time is positively correlated with block difficulty, while broadcasting time is affected by both block size and the number of participating nodes. The experiments simulated networks of 3000, 6000, 9000, and 12000 nodes. As shown in Figure 9, the results indicate that Ladder's throughput improves by 34.2% compared to Conflux as the number of nodes increases to 12000.

Heterogeneous Nodes: We conduct experiments on Alibaba Cloud using a heterogeneous node setup of 80 virtual machines (VMs): 30 VMs with 4 cores and 8 GB RAM, 30 VMs with 4 cores and 16 GB RAM, and 20 VMs with 8 cores and 16 GB RAM. Due to the intensive hash computations required by PoW, CPU utilization on each VM remains high, ranging from 90% to nearly 100%. Memory usage averages 22.5% on 8 GB VMs and 13.25% on 16 GB VMs, primarily allocated for the transaction pool, blockchain state, and PoW computation caching. We evaluate system performance under varying bandwidth constraints of 10, 20, and 30 Mbps, with 1000 transactions per block and a difficulty level of 18. At 10 Mbps, throughput is 2011 TPS, indicating that bandwidth is the primary bottleneck. Increasing bandwidth to 20 Mbps raises throughput to 3986 TPS, approaching the theoretical maximum. At 30 Mbps, throughput reaches 4652 TPS, showing only marginal improvement over the 20 Mbps results, suggesting that PoW computation efficiency becomes the new bottleneck. These experiments show Ladder's ability to adapt to heterogeneous environments and optimize performance based on available resources and network conditions.

Impact of BFT Committee: We also perform latency testing on the committee in Ladder. Based on our analysis in BFT Committee Size section (Sec. 4), we choose a committee size of 300 nodes. Since committee consensus operates as an independent module, we conduct separate experiments simulating a scenario where nodes simultaneously receive two conflicting lower-chain blocks. We use 80 VMs to emulate a 300-node committee, including 99 Byzantine nodes. Under this setup, the average time cost for committee consensus is 3.25 seconds, which exceeds the lower-chain broadcast time. Although the committee introduces some performance overhead, it plays a crucial role in maintaining system security by mitigating the impact of malicious nodes and ensuring consensus integrity.

6 Security Analysis

6.1 Transaction Trustworthiness

Ladder consists of a growing structure, similar to chain growth in conventional blockchains, where upper and lower-chain blocks serve as the key elements binding the structure. Although the node whose block was selected as the standard upper-chain block in the previous round chooses the next standard block, it cannot always select its own block due to the competitive nature of upper-chain block generation. The probability of a node repeatedly generating blocks without faults is negligible, as it must also handle verification, sorting, and parallel processing of received upper-chain blocks. This makes it further improbable for a node to consistently generate lower-chain blocks in subsequent rounds and keep adding blocks to the upper-chain with invalid transactions. This establishes the trustworthiness of confirmed transactions.

6.2 Resistance Against Common Attacks

This subsection demonstrates how Ladder is resilient against some common attacks.

Sybil Attack [9]: In a Sybil attack, the attacker creates a large number of nodes to manipulate the consensus protocol. As Ladder relies on PoW rather than node count to determine if a given block can be added to the upper-chain, it is resilient against Sybil attacks. Furthermore, only nodes that have previously contributed a standard upper-chain block can generate lower-chain blocks or participate in HotStuff consensus for super block generation. This makes launching a successful Sybil attack on Ladder prohibitively costly.

Denial of Service (DoS) Attack: In PoW-based blockchains, block verification is fast, while block generation is slow and resource-intensive, providing inherent protection against DoS attacks. However, predictable leader election can expose specific nodes to targeted DoS attacks. To mitigate this, Ladder uses VRF for leader selection, reducing the likelihood of successful targeting by adversaries. Even if an attack occurs, other nodes can continue verification and mining without interruption, ensuring the system remains functional.

Double-Spend Attack: To perform a double-spend attack, the adversary cannot immediately engage in malicious activities but must first generate a block that is selected by the pivot chain node as an upper-chain block. The adversary must also

keep the pivot chain operating normally while mining in the next round to generate the lower-chain block. The limited computing power and the hardest chain principle make it nearly impossible for an attacker to execute a double-spend attack within any practically meaningful time frame (*e.g.*, one day). Attacks become highly improbable when adversary resources fall below a certain threshold in Ladder [23].

Eclipse attack [13]: Ladder relies on computing power to establish credibility. While the Eclipse attack impacts the HotStuff consensus component, other nodes that are not part of the BFT committee still generate blocks using PoW. This practically eliminates the impact of Eclipse attacks on the overall system performance.

6.3 Security and Availability

A blockchain network must ensure two key characteristics: security, which ensures that only valid transactions are included in the ledger, and availability, which guarantees that transactions are confirmed within a finite time. Prior work [11] has shown that the three properties of common-prefix, finality, and liveness are sufficient to achieve these characteristics. We define these three properties along with an additional one, and prove that Ladder satisfies both security and availability through lemmas and theorems. Theorems are proven in this section, while proofs of lemmas are provided in the Appendix.

Common-prefix: At any given time, if all nodes in the blockchain network have, with very high probability, the same knowledge of the verified portion of the ledger, the blockchain satisfies the common-prefix property.

Finality: If the verified portion of the ledger of any honest node does not change over time (such as in response to an attempted attack or due to intermittent inconsistencies among nodes), the blockchain ledger satisfies the finality property.

Liveness: If a valid transaction becomes part of the verified portion of the ledger within a predefined amount of time, the blockchain satisfies the liveness property.

l-balance: A sequence of $m \geq l$ nodes is said to be l-balanced when in any subsequence of l or more consecutive nodes, the honest nodes are in majority.

Lemma 1. *The probability that an adversary gets to cast more than $1/3$ of the votes in Ladder's BFT committee (which generates the super block) decreases exponentially with the committee size.*

Lemma 2. *Let k represent the number of blocks in the lower-chain. If the sequence of nodes that generated the given sequence of standard upper- and lower-chain blocks is $2k+2$-balanced, the blockchain ledger generated from the corresponding rounds satisfies the common-prefix property.*

Lemma 3. *If the sequence of nodes that generated the given sequence of standard upper-chain and lower-chain blocks is $2k$-balanced, the blockchain ledger generated from the corresponding rounds satisfies the finality property.*

Lemma 4. *If the sequence of nodes that generated the given sequence of standard upper-chain and lower-chain blocks is $2k$-balanced, the blockchain ledger generated from the corresponding rounds satisfies the liveness property.*

Lemma 5. *The sequence of nodes that generate the given sequence of standard upper-chain and lower-chain blocks is, with very high probability, $2k$-balanced.*

Theorem 1. *Any block in the lower-chain of Ladder is a valid block with very high probability.*

Proof of Theorem 1. For an adversary to generate a lower-chain block in a given round, the adversary must first gain the authority to generate the lower-chain block in that round, which can happen with the probability of just $\frac{q^3+q(1-q)}{q^3+1-q}$ (Appendix B). If an adversary is able to contribute a lower-chain block where it enters any malicious/incorrect information, that will be detected and a super block generation process will start.To succeed in the BFT consensus, the adversary must control more than $1/3$ of the committee. According to Lemma 1, the probability of this happening is extremely low. □

Theorem 2. *Ladder satisfies common prefix, finality, and liveness properties.*

Proof of Theorem 2. By Lemma 5, we know that the sequence of nodes that generated the given sequence of k lower-chain blocks in Ladder satisfies the $2k$-*balance*. Consequently, by Lemmas 2, 3, and 4, we conclude that the resulting blockchain ledger in Ladder satisfies the common prefix, finality, and liveness properties. □

These two theorems, together, prove that Ladder has both security and availability characteristics.

7 Conclusion

We propose Ladder, a structured twin-chain DAG blockchain that optimizes parallel block processing to achieve high throughput, low latency, and resilience against balance attacks. Ladder improves sorting efficiency by designating a convergence node per round, eliminates PoW for lower-chain blocks, and incentivizes node participation for enhanced performance and security. To counter adversarial disruptions, a committee is deployed to generate special blocks when faulty blocks are detected. Experiments on an 80-node network show that Ladder improves throughput by 59.6% and reduces latency by 20.9% compared to state-of-the-art methods.

Acknowledgements

We thank the anonymous NSDI reviewers and our shepherd Prof. Vincent Liu for their constructive feedback and suggestions. This work is supported by the National Natural Science Foundation of China under Grant No. 62032017. Xiulong Liu is the corresponding author: xiulong_liu@tju.edu.cn

References

[1] Yair Amir, Brian A. Coan, Jonathan Kirsch, and John Lane. Prime: Byzantine replication under attack. *IEEE Transactions on Dependable and Secure Computing*, 8(4):564–577, 2011.

[2] Vivek Kumar Bagaria, Sreeram Kannan, David Tse, Giulia Fanti, and Pramod Viswanath. Prism: Deconstructing the blockchain to approach physical limits. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 585–602. ACM, 2019.

[3] Leemon Baird. Hashgraph consensus: Fair, fast, Byzantine fault tolerance. *Swirlds Tech Report*, 2016.

[4] Iddo Bentov, Pavel Hubácek, Tal Moran, and Asaf Nadler. Tortoise and hares consensus: The meshcash framework for incentive-compatible, scalable cryptocurrencies. In *Cyber Security Cryptography and Machine Learning (CSCML)*, volume 12716, pages 114–127. Springer, 2021.

[5] Miguel Castro and Barbara Liskov. Practical Byzantine fault tolerance. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 173–186. USENIX, 1999.

[6] Anton Churyumov. Byteball: A decentralized system for storage and transfer of value, 2016. URL: https://byteball.org/Byteball.pdf.

[7] James A. Cowling, Daniel S. Myers, Barbara Liskov, Rodrigo Rodrigues, and Liuba Shrira. HQ replication: A hybrid quorum protocol for Byzantine fault tolerance. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 177–190. USENIX, 2006.

[8] George Danezis, Lefteris Kokoris-Kogias, Alberto Sonnino, and Alexander Spiegelman. Narwhal and tusk: A dag-based mempool and efficient BFT consensus. In *European Conference on Computer System (EuroSys)*, pages 34–50. ACM, 2022.

[9] John R. Douceur. The sybil attack. In *International Workshop on Peer-to-Peer Systems (IPTPS)*, volume 2429, pages 251–260. Springer, 2002.

[10] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. *Communications of the ACM*, 61(7):95–102, 2018.

[11] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The Bitcoin backbone protocol: Analysis and applications. *Journal of the ACM*, 71(4):1–49, 2024.

[12] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling Byzantine agreements for cryptocurrencies. In *Symposium on Operating Systems Principles (SOSP)*, pages 51–68. ACM, 2017.

[13] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. Eclipse attacks on Bitcoin's peer-to-peer network. In *USENIX Security Symposium (Security)*, pages 129–144. USENIX, 2015.

[14] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, Linus Gasser, and Bryan Ford. Enhancing Bitcoin security and performance with strong consistency via collective signing. In *USENIX Security Symposium (Security)*, pages 279–296. USENIX, 2016.

[15] Ramakrishna Kotla, Lorenzo Alvisi, Michael Dahlin, Allen Clement, and Edmund L. Wong. Zyzzyva: Speculative Byzantine fault tolerance. *ACM Transactions on Computer Systems*, 27(4):7:1–7:39, 2009.

[16] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine generals problem. *ACM Transactions on Programming Language and Systems*, 4(3):382–401, 1982.

[17] Colin LeMahieu. Nano: A feeless distributed cryptocurrency network, 2018. URL: https://nano.org/en/whitepaper.

[18] Yoad Lewenberg, Yonatan Sompolinsky, and Aviv Zohar. Inclusive block chain protocols. In *Financial Cryptography and Data Security (FC)*, pages 528–547. Springer, 2015.

[19] Chenxing Li, Peilun Li, Dong Zhou, Zhe Yang, Ming Wu, Guang Yang, Wei Xu, Fan Long, and Andrew Chi-Chih Yao. A decentralized blockchain with high throughput and fast confirmation. In *USENIX Annual Technical Conference (ATC)*, pages 515–528. USENIX, 2020.

[20] Chenxing Li, Fan Long, and Guang Yang. GHAST: Breaking confirmation delay barrier in Nakamoto consensus via adaptive weighted blocks. *arXiv preprint arXiv:2006.01072*, 2020.

[21] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. URL: https://bitcoin.org/bitcoin.pdf.

[22] Serguei Popov. The Tangle. *White paper*, 1(3):30, 2018.

[23] Meni Rosenfeld. Analysis of hashrate-based double spending. *arXiv preprint arXiv:1402.2009*, 2014.

[24] Yonatan Sompolinsky, Yoad Lewenberg, and Aviv Zohar. SPECTRE: A fast and scalable cryptocurrency protocol. *IACR Cryptol. ePrint Arch.*, page 1159, 2016.

[25] Yonatan Sompolinsky, Shai Wyborski, and Aviv Zohar. PHANTOM GHOSTDAG: A scalable generalization of nakamoto consensus: September 2, 2021. In *ACM Conference on Advances in Financial Technologies (AFT)*, pages 57–70. ACM, 2021.

[26] Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in Bitcoin. In *Financial Cryptography and Data Security (FC)*, pages 507–527. Springer, 2015.

[27] Yonatan Sompolinsky and Aviv Zohar. PHANTOM: A scalable blockdag protocol. *IACR Cryptol. ePrint Arch.*, page 104, 2018.

[28] Alexander Spiegelman, Balaji Arun, Rati Gelashvili, and Zekun Li. Shoal: Improving DAG-BFT latency and robustness. *arXiv preprint arXiv:2306.03058*, 2023.

[29] Alexander Spiegelman, Neil Giridharan, Alberto Sonnino, and Lefteris Kokoris-Kogias. Bullshark: DAG BFT protocols made practical. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 2705–2718. ACM, 2022.

[30] Giuliana Santos Veronese, Miguel Correia, Alysson Neves Bessani, and Lau Cheuk Lung. Spin one's wheels? Byzantine fault tolerance with a spinning primary. In *IEEE Symposium on Reliable Distributed Systems (SRDS)*, pages 135–144. IEEE, 2009.

[31] Giuliana Santos Veronese, Miguel Correia, Alysson Neves Bessani, Lau Cheuk Lung, and Paulo Veríssimo. Efficient Byzantine fault-tolerance. *IEEE Transactions on Computers*, 62(1):16–30, 2013.

[32] Qin Wang, Jiangshan Yu, Shiping Chen, and Yang Xiang. Sok: Dag-based blockchain systems. *ACM Computing Surveys*, 55(12):1–38, 2023.

[33] WonderNetwork. Global ping statistics: Ping times between wondernetwork servers, Apr. 2018. URL: https://wondernetwork.com/pings.

[34] Jie Xu, Qingyuan Xie, Sen Peng, Cong Wang, and Xiaohua Jia. Adaptchain: Adaptive scaling blockchain with transaction deduplication. *IEEE Transactions on Parallel and Distributed Systems*, 34(6):1909–1922, 2023.

[35] Lei Yang, Seo Jin Park, Mohammad Alizadeh, Sreeram Kannan, and David Tse. Dispersedledger: High-Throughput Byzantine consensus on variable bandwidth networks. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 493–512, 2022.

[36] Maofan Yin, Dahlia Malkhi, Michael K Reiter, Guy Golan Gueta, and Ittai Abraham. Hotstuff: BFT consensus with linearity and responsiveness. In *ACM Symposium on Principles of Distributed Computing (PODC)*, pages 347–356. ACM, 2019.

[37] Haifeng Yu, Ivica Nikolic, Ruomu Hou, and Prateek Saxena. OHIE: Blockchain scaling made simple. In *IEEE Symposium on Security and Privacy (SP)*, pages 90–105. IEEE, 2020.

A Selfish Mining Analysis

We analyze the probability of adversary-generated upper-chain blocks exceeding the difficulty of super blocks in the context of selfish mining. If the adversary generates at least m blocks with x leading zeros, it may exceed the difficulty of the k-member super block committee. The probability of generating these blocks, which would exceed the super blocks, can be calculated as $P = \frac{1}{2^{m(x+1)}}$. The difficulty of these upper-chain blocks is $\sum_{i=1}^{m} \sqrt{x} = m\sqrt{x}$, while the super blocks have a difficulty of $\sqrt{k}$. Therefore, the condition for the adversary to surpass the super block's difficulty is $\sqrt{k} \leq m\sqrt{x}$, which implies $x \geq k/m^2$. Substituting this into the probability expression for P, we can obtain that $P = \frac{1}{2^{m(x+1)}} \leq \frac{1}{2^{m(k/m^2+1)}} = \frac{1}{2^{k/m+m}} \leq \frac{1}{2^{2\sqrt{k}}}$.

B Proof of Lemmas

Proof of Lemma 1

Proof. Let $Pr(H|H)$ represent the probability that the honest lower-chain block generation node selects an honest node as the next round lower-chain block generation node. Let $Pr(A|H)$ represent the probability that the honest lower-chain block generation node selects an adversary node as the next round lower-chain block generation node. Let $Pr(H|A)$ represent the probability that the adversary lower-chain block generation node selects an honest node as the next round lower-chain block generation node. Finally, let $Pr(A|A)$ represent the probability that the adversary lower-chain block generation node selects an adversary node as the next round lower-chain block generation node.

If the block generation node in the current round is an honest node, the probability that the block generation node in the next round is an honest node or the adversary corresponds to the respective computing power share. Let q represent the fraction of the computational power contributed by the adversary to the blockchain network. Since there must be a next block generation node, we have $Pr(H|H)+Pr(A|H) = 1$, and $Pr(A|H) = q$, $Pr(H|H) = 1-q$.

If the block generation node in the current round is an adversary node, then if the adversary can compute a legal block within the waiting duration (see Waiting Time Duration in Sec. 4), then the block generation node in the next round will still be an adversary node with probability $Pr(A|A) = \frac{q}{q^2+1-q}$ If the adversary fails to compute a legal block and an honest node computes a legal block, then the block generation node in the next round will be an honest node with probability $Pr(H|A) = \frac{(1-q)^2}{q^2+1-q}$.

The selection of the lower-chain block generation node depends only on the current round's lower-chain block, independent of previous rounds. According to the Markov property, the steady-state probability distribution can be calculated with the calculated state transfer probability, and it can be solved that the probability of the next block generation node being an adversary in each round is less than $1/3$ when the q is less than 0.3.

As we know, the probabilities $Pr(H)$ and $Pr(A)$ can be expressed using the following equations:

$$Pr(H) = Pr(H|H)Pr(H) + Pr(H|A)Pr(A)$$

$$Pr(A) = Pr(A|H)Pr(H) + Pr(A|A)Pr(A)$$

By substituting the expressions for various terms, we can get

$$Pr(H) = \frac{(1-q)^2}{q^3+1-q}$$

and

$$Pr(A) = \frac{q^3+q(1-q)}{q^3+1-q}$$

Let the committee size $k = 3m$, then

$$\begin{aligned} Pr(|A| > 1/3) &= \sum_{i=m}^{3m} Pr(H)^{3m-i} Pr(A)^i \\ &< \sqrt{\frac{8m}{3\pi}} (\frac{16\sqrt{2}}{27})^m \end{aligned}$$

As m increases, the upper bound on the probability decreases exponentially, where $m \geq 3$. □

Proof of Lemma 2

Proof. Suppose the block sequence does not satisfy common-prefix property when the sequence of block generation nodes satisfies $2k$+2-balance. Let the first round be numbered 0. Let the two rounds when the ledger appears to have conflicting transactions/blocks be d and d', such that $d' - d \geq k$. Clearly, there exist at least $2k$+2 block generation nodes. The number of honest nodes $\leq d'$ in the ledger when honest nodes produce at most one block at each height. In the ledger, the number of adversary nodes $\geq (d-0) + (d'-d) = d'$, and the number of adversary nodes $\geq$ honest nodes in the ledger for the block sequence initial block to block d' block generation node sequence, which contradicts the $2k$+2-*balance* property. □

Proof of Lemma 3

Proof. Suppose the block sequence does not satisfy the **Finality** property when the given sequence of block generation nodes is $2k$-balanced. In other words, there exists a block b in round d that is part of ledger when ledger is generated in round $i > d+k$ but not a part of the ledger when ledger is generated in round $j > i$. At this point, for the last $\geq k+1$ blocks of the ledger, there is an inconsistent view, and that would require the block generation nodes to not be $2k$-balanced, contradicting the lemma statement. □

Proof of Lemma 4

Proof. After $2kt$ rounds, the height of the ledger increases by $\geq (k+1)t$, where $\leq (k-1)t$ is generated by the adversary.

Then $\geq (k+1)t-(k-1)t = 2t$ blocks are honest, and among them, $\geq 2t-k$ have been confirmed. This means that at least $2t-k$ honest blocks are added to the block sequence in every $2kt$ rounds.

Therefore, a transaction that is recognized by all honest nodes will definitely be added to the block sequence (ledger) and finally confirmed. □

Proof of Lemma 5

Proof. For a given sequence of length l, where the probability that the adversary accounts for more than 50% does not exceed $exp(-cl)$

By Chernoff's inequality $Pr[X \geq (1+\delta)\mu] < exp(-\frac{\mu\delta^2}{2})$, where $\mu = ql, \delta = \frac{1-2q}{2q}$, we obtain

$$\begin{aligned} Pr[X \geq (1+\delta)\mu] &= Pr[X \geq \frac{l}{2}] \\ &\leq exp(-\frac{l(1-2q)^2}{8q}) \\ &= exp(-cl), c > 0 \end{aligned}$$

For a sequence of block generation nodes of length L, any subsequence of length at least l has a probability of at most $\frac{(L-l+1)(L-l+2)}{2} exp(-cl) < L^2 exp(-cl)$ that the adversary controls more than 50%.

Then $Pr[Ladder's\ node\ sequence\ satisfies\ 2k\text{-}balance] > 1-\delta(l)$ and $\delta(l)$ decreases exponentially with l. □