UTS UNIVERSITY OF TECHNOLOGY SYDNEY

Improving Efficiency and Security of Sharded Blockchain

by Zixu Zhang

Thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

under the supervision of Xu Wang, Ren Ping Liu, Guangsheng Yu

University of Technology Sydney
Faculty of Engineering and IT

January, 2025

CERTIFICATE OF ORIGINAL AUTHORSHIP

I, Zixu Zhang, declare that this thesis is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the School of Electrical and Data Engineering at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Signature: Production Note: Signature removed prior to publication.

Date: January 2, 2025

Abstract

Blockchain technology has significantly evolved, notably through the adoption of sharding techniques to enhance scalability and manage growing transaction demands efficiently. Sharding divides the blockchain network into smaller, more manageable segments, or shards, which operate parallel transactions, thereby boosting throughput. However, conventional sharding solutions often encounter challenges related to node distribution and transaction efficiency, particularly when managing cross-shard transactions which can significantly degrade network performance and increase transaction costs.

In this thesis, Blockchain Sharding bottlenecks are addressed by introducing novel methods to optimize sharding schemes for enhanced efficiency and security. Firstly, a community detection-based sharding scheme are validated using over one million public Ethereum transactions. This scheme dramatically decreases the ratio of cross-shard transactions from 80% to 20%, significantly lower than that of baseline random sharding methods. By clustering transactional communities, community detection-based sharding approach promotes intra-shard transactions, which substantially reduces transaction fees by 50%, thereby enhancing blockchain scalability and making the ecosystem more budget friendly. Furthermore, a novel Trust-based and Deep Reinforcement Learning-driven (TbDd) framework are introduced to mitigate collusion risks and dynamically adjust node allocation, thus enhancing throughput while maintaining network security. The TbDd framework features a comprehensive trust evaluation system that identifies node behaviors and performs targeted resharding to address potential threats. Designed to increase tolerance for dishonest nodes, optimize node movement frequency, and ensure equitable distribution across shards, this framework effectively balances sharding risks. Extensive testing shows that TbDd surpasses traditional random, community, and trust-based methods in maintaining shard risk equilibrium and reducing cross-shard transactions. Lastly, an advanced Overlapping Sharding with xPBFT Consensus Mechanism are proposed, which simplifies cross-

shard transactions by treating them as intra-shard processes. This approach reduces latency by up to 40% and strengthens security, offering a scalable solution for decentralized applications. These contributions advance the efficiency, security, and scalability of sharded blockchain systems, providing a robust foundation for future developments in blockchain technology.

Acknowledgements

I would like to extend my deepest gratitude to my supervisor, Dr. Xu Wang, as well as Prof. Ren Ping Liu and Dr. Wei Ni, for their unwavering support, guidance, and inspiration throughout my Ph.D. studies. Their enthusiasm, profound knowledge, and patience have been pivotal in my research journey, helping me navigate complex challenges and fostering my growth as a researcher. Special thanks are also due to my external supervisor, Dr. Guangsheng Yu at CSIRO, whose insights and guidance were invaluable during both the research and writing phases of this thesis. His expertise and constructive feedback have greatly enriched my work.

I am sincerely grateful to Dr. Ying Wang, Ming Zhang, and Dr. Caijun Sun for their encouraging words and perceptive comments, which have significantly contributed to my research. Their support and critical insights have been instrumental in refining my ideas and enhancing the quality of my research. I would also like to acknowledge all the colleagues at the GBDTC, UTS, for their support, camaraderie, and the stimulating academic environment they provided throughout this journey. Their collaboration and friendship have made this experience both rewarding and enjoyable.

My heartfelt appreciation goes to my family. To my parents, Tao Zhang and Meisong Wang, thank you for your endless love and belief in me, which have been the bedrock of my perseverance. To my wife, Ying Wang, your unwavering support, understanding, and encouragement have been a constant source of strength. Your patience and sacrifices have made this journey possible, and for that, I am forever grateful. This accomplishment would not have been possible without your presence and support.

Finally, I would like to take a moment to acknowledge my own efforts and perseverance throughout this journey. Pursuing a Ph.D. is a challenging endeavor that requires constant self-improvement and resilience in the face of setbacks. I am grateful to myself for not giving

up during difficult times and for maintaining my passion for knowledge and commitment to my goals. It is this determination and hard work that have allowed me to reach this point and complete this significant research. This journey has not only equipped me with valuable knowledge and skills but also taught me the importance of perseverance and self-growth.

Zixu Zhang

January 2, 2025

Sydney, Australia

Contents

Chapter 1

Introduction

Blockchain technology has emerged as a pivotal innovation, transforming various sectors by offering decentralized [1], secure [2, 3], and transparent solutions [4, 5]. Its significance is particularly evident in industries such as finance [6–8], healthcare [9, 10], and supply chain management [11, 12], where it enables streamlined processes, enhanced data security, and improved traceability. By providing a tamper-resistant ledger, blockchain offers a robust framework for transactions and data management, reducing the need for intermediaries and thus enhancing efficiency. Moreover, blockchain serves as the backbone of Web 3.0, the next generation of the internet that envisions a decentralized ecosystem empowering users with ownership of their data and enabling new paradigms in digital interaction [13]. However, despite its growing adoption and the promise it holds, blockchain technology is not without its intrinsic challenges.

One of the primary challenges lies in its scalability [14–16]. As blockchain networks grow and transaction volumes surge, the system's ability to maintain high throughput and low latency becomes increasingly strained. Traditional blockchain architectures, such as Bitcoin [17, 18] and Ethereum [19–21], suffer from limited transaction processing capabilities due to their reliance on a global consensus mechanism. This limitation not only leads to increased transaction times and fees but also restricts the technology's potential for large-scale adoption. Furthermore, various types of nodes—such as accounts, validators, and servers—play crucial roles in the network's operations, each contributing to the overall security and functionality of the blockchain. Meanwhile, security [2, 22] remains a critical concern. As blockchains become more popular, they become attractive targets for malicious actors. Ensuring the integrity and security of transactions, especially in a decentralized and open network, is paramount [2, 3].

These challenges—scalability and security—pose significant obstacles to the broader adoption and efficiency of blockchain systems.

To address these pressing issues, research on sharded blockchain architectures has gained prominence, offering potential solutions to enhance both scalability and security [23]. Sharded blockchain [24–27] partitions the network into smaller, more manageable segments called shards. Each shard processes a subset of transactions in parallel, thus increasing the system's overall throughput and reducing latency. This parallel processing mechanism effectively addresses the key bottlenecks of traditional blockchain systems, making sharding a promising approach to scalability. However, sharding introduces new complexities and security risks, particularly in terms of how nodes are allocated to shards. The security of a sharded blockchain is dependent on ensuring that no shard is overrun by malicious nodes, which could compromise the integrity of the entire network.

The effectiveness of sharding in blockchain systems is inherently tied to the strategies used for node allocation. Proper allocation of nodes is crucial for maintaining the security and enhancing the performance of sharded blockchains [28]. Poor allocation strategies can result in significant vulnerabilities, such as an adversary seizing control of a single shard, thereby compromising the security of the entire network. To fully realize the benefits of sharding, it is important to develop robust allocation strategies that consider transaction behaviors between accounts, the heterogeneity of nodes, network dynamics, and potential adversarial actions, ensuring a distribution that bolsters security without sacrificing operational efficiency.

This thesis addresses these intricate challenges, exploring how strategic allocation of nodes can optimize blockchain performance and facilitate its broader application across diverse domains. The research focuses on three critical areas: transaction-based account allocation, trust-based node allocation, and node allocation in overlapping shards. By fortifying the foundational architecture of sharded blockchains, the study aims to mitigate a range of security threats while enhancing scalability. Through a comprehensive analysis of how different allocation methods impact network security and performance, this thesis provides valuable insights to guide the future development of blockchain systems, ultimately contributing to the creation of more secure, scalable, and efficient blockchain solutions ready for widespread adoption across various industries.

1.1 Overview of Blockchain Technology

Blockchain technology is a distributed ledger system that offers a secure and immutable way to record transactions across multiple devices, ensuring that no single point of control or failure can manipulate the data. This revolutionary technology emerged with the publication of the Bitcoin white paper by Satoshi Nakamoto in 2008 [17] and has since evolved into various forms and applications beyond cryptocurrencies, including supply chain management [29], healthcare [30], and decentralized finance (DeFi) [31].

1.1.1 Blockchain Structure

A blockchain consists of a chain of blocks that securely store data in a decentralized and tamper-resistant manner. Each block in the blockchain contains a list of transactions, a timestamp, a reference to the previous block, and a cryptographic hash. As shown in Fig 1.1, this structure ensures that once data is recorded, it becomes immutable and tamper-evident.

Block: A block is the fundamental unit of the blockchain that stores a batch of transactions. Each block contains two components, block header and block body.

Header: The block header includes metadata such as the timestamp, the cryptographic hash of the current block's data, the hash of the previous block, and a nonce used in the consensus process (e.g., Proof of Work).

Transactions: The core part of the block contains a list of transactions. Each transaction represents a transfer of value or execution of a smart contract, and includes details such as sender, receiver, amount, and digital signatures to verify authenticity.

Merkle Root: The block also contains a Merkle root, a single hash representing the combined hash of all transactions in the block. This data structure allows for efficient and secure verification of individual transactions within the block.

Linking Blocks: Each block is linked to the previous one through its hash value, forming a continuous chain back to the first block, known as the genesis block. This chain of hashes ensures the integrity of the entire blockchain; if any block is altered, the hash values of all subsequent blocks will change, immediately signaling that tampering has occurred.

Hashing and Cryptography: The use of cryptographic hashing in the blockchain ensures

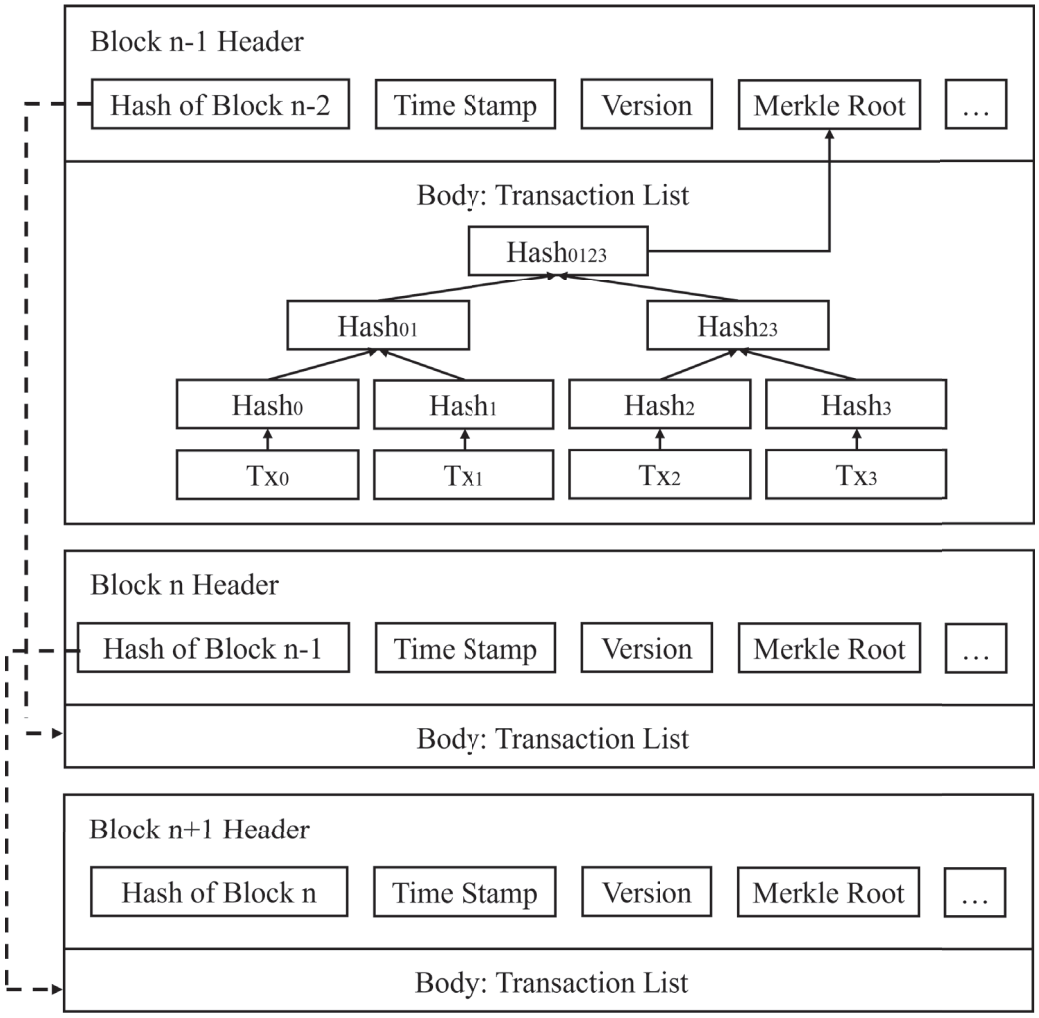


Figure 1.1: Block Structure

data integrity and security. Hash functions take input data and generate a fixed-size output (hash) that uniquely represents the data. In the blockchain structure, each block's hash is included in the next block, creating an immutable sequence. Any change to the data in a block would alter its hash, thereby breaking the chain and revealing the tampering.

Distributed Ledger: The blockchain structure is maintained across a distributed network of nodes. Each node has a copy of the blockchain and participates in the consensus process to agree on the state of the ledger. This decentralized nature ensures that no single point of control or failure can compromise the data.

This structure of linked blocks, cryptographic hashing, and distributed ledger technology forms the foundation of blockchain's security, immutability, and trustless nature, making it a powerful tool for various applications beyond just cryptocurrencies.

1.1.2 Types of Blockchain:

Blockchains can be categorized into three main types: Public Blockchains, Private Blockchains, and Consortium Blockchains.

- **Public Blockchain:** Public blockchains are open and decentralized networks where anyone can participate as a node, validate transactions, and access the ledger. These blockchain are permissionless, meaning that they do not require any authorization for participation. Examples include Bitcoin [17] and Ethereum [19]. These blockchains provide high levels of security and transparency but often face challenges related to scalability and transaction speed due to their consensus mechanisms, such as Proof of Work (PoW) [32] or Proof of Stake (PoS) [33].
- **Private Blockchain:** Private blockchains are permissioned networks controlled by a single organization. Private blockchain restrict access to authorized participants only, allowing for faster transaction processing and enhanced privacy. Examples of private blockchain platforms include Multichain [34] and Hyperledger Fabric [35]. While private blockchains offer enhanced performance and control, they trade off some of the decentralization and trustless nature that characterize public blockchains.
- **Consortium Blockchain:** Consortium blockchains, also known as federated blockchains, represent a hybrid approach where a group of organizations jointly manage the blockchain network. These blockchains are permissioned networks managed by a group of organizations rather than a single entity. The model of Consortium blockchains strikes a balance between decentralization and centralized control, making it suitable for scenarios where multiple stakeholders need to collaborate while maintaining a degree of privacy and security. A notable example of a consortium blockchain framework is R3 Corda [36], which is specifically designed for financial institutions to enable secure, efficient, and scalable transactions.

1.1.3 Multi-layered conceptual model in Blockchain network

The blockchain ecosystem can be understood through a six-layer conceptual model. This model outlines the various components and functions that work together to form a complete blockchain network. The six layers are the Application Layer, Contract Layer, Incentive Layer, Consensus Layer, Network Layer, and Data Layer.

Application Layer: This is the topmost layer where end-users interact with the blockchain through various applications and services. It encompasses a wide range of use cases, including cryptocurrencies, decentralized applications (dApps), supply chain management [37–40]. These applications provide tangible value to end-users by allowing them to perform transactions, interact with smart contracts, and leverage blockchain-based solutions for different purposes.

Contract Layer: The contract layer supports the execution of smart contracts and decentralized applications [41, 42]. It defines the rules, logic, and conditions for automated transactions and agreements on the blockchain. This layer enables complex functionalities, such as token issuance, decentralized exchanges, and autonomous organizations, by executing programmable contracts that automatically enforce terms without the need for intermediaries.

Incentive Layer: Primarily present in public blockchains, the incentive layer deals with the economic aspects of the blockchain ecosystem [43–45]. It includes rewards, transaction fees, and other economic incentives designed to encourage honest participation in the network. By providing financial motivation, this layer ensures that participants act in ways that uphold the network's integrity and security.

Consensus Layer: The consensus layer is responsible for how nodes in the network agree on the current state of the blockchain. It encompasses various consensus mechanisms like PoW [46,47], PoS [48–50], Practical Byzantine Fault Tolerance (PBFT) [51–53], and Delegated Proof of Stake (DPoS) [54]. This layer plays a critical role in maintaining the blockchain's security, performance, and scalability by determining how transactions are validated and blocks are added to the chain.

Network Layer: The network layer manages the communication and data propagation among nodes in the blockchain network. It ensures that all nodes maintain a consistent view of the blockchain by using networking protocols (e.g., TCP/IP, gossip protocols) and the necessary software and hardware infrastructure [37,55]. This layer includes peer-to-peer networking, relay networks, and data propagation algorithms, facilitating the secure and efficient dissemination of blocks and transactions across the network.

Data Layer: This foundational layer contains the raw data of the blockchain, including transactions, blocks, and smart contracts [56–58]. It also involves cryptographic components such as hashing algorithms, digital signatures, and Merkle trees [59], which ensure data immutability and security. The data layer provides the basic building blocks for creating and verifying

blockchain data, ensuring the integrity and consistency of the entire system.

Together, these six layers offer a comprehensive view of the blockchain ecosystem, detailing how various components interact to enable the secure, decentralized, and automated execution of transactions. By categorizing the blockchain network into these distinct layers, this model aids in understanding the underlying architecture and functionality of blockchain technology.

1.1.4 Blockchain Consensus

Consensus mechanisms are the backbone of blockchain technology, ensuring that all participants in the network agree on the current state of the ledger. In sharded blockchains, achieving consensus becomes even more complex due to the partitioning of the network into multiple shards, each functioning as a semi-independent blockchain.

Traditional consensus mechanisms like PoW [46, 47] and PoS [48–50] struggle to scale efficiently in sharded environments because they require global coordination across all nodes. To address this, sharded blockchain systems employ specialized intra-shard and cross-shard consensus protocols.

Consensus mechanisms in sharded blockchains must address the unique challenge of achieving agreement not only within shards but also across the entire network. In traditional, non-sharded blockchains, consensus is reached by having all nodes validate and agree on each transaction. However, in a sharded system, each shard independently processes transactions, which raises the challenge of ensuring that cross-shard transactions are consistently and atomically recorded across all affected shards.

1.2 Sharding Concept

Sharding in blockchain adapts traditional database partitioning techniques to a decentralized environment, distributing the transaction load across multiple shards [24]. Each shard operates independently, processing its own set of transactions and maintaining a unique segment of the overall state, this division not only alleviates the burden on individual nodes but also enhances the network's capacity to handle larger volumes of transactions concurrently. By reducing the number of transactions processed by each node, sharding can significantly improve network latency, leading to faster transaction validation times, making it ideal for complex applications

such as financial services and large-scale IoT [60–62] deployments.

1.2.1 Sharded Blockchain

Sharding is a technique designed to improve the scalability of blockchain networks by dividing the network into smaller, manageable segments called shards. Each shard processes a subset of the network's transactions, which can significantly increase the overall throughput. However, as shown in Tab. 1.1, this approach introduces several challenges, particularly in terms of security, cross-shard communication, and consensus.

Types of Sharding:

- **Network Sharding:** This involves partitioning the network's nodes into different shards. For instance, Elastico [63] and Omniledger [64] use this approach to reduce the computational load on individual nodes and increases the network's overall throughput.
- **Transaction Sharding:** In this approach, transactions are distributed across different shards. Monoxide [65] uses asynchronous consensus zones where each shard processes a unique set of transactions, thereby improving parallelism and reducing latency.
- **State Sharding:** State sharding divides the blockchain's state (e.g., account balances) among different shards. This approach significantly reduces storage requirements for each node but complicates cross-shard communication. Ethereum 2.0 [66] employs state sharding to enhance its scalability while maintaining security through a hybrid PoS consensus mechanism.

Sharded blockchain architectures provide a comprehensive solution to the scalability and security challenges faced by traditional blockchain systems. By partitioning the network into smaller, more manageable segments known as shards, these architectures facilitate parallel transaction processing, significantly increasing overall network capacity and efficiency. This section discusses the technological foundations of sharded blockchains including sharding technology, scalability, and security aspects, as well as Distributed Randomness Generation (DRG) protocol is, leader election, and consensus mechanisms.

- **Intra-shard Consensus:** Each shard operates its own consensus mechanism to validate transactions within the shard. For example, Elastico uses Practical Byzantine Fault Tolerance (PBFT) [67] within each shard to ensure that all nodes agree on the transactions

Table 1.1: Comparison of Sharding Protocols

Protocol	Sharding Type	Consensus Mechanism	Cross-shard Communication	Scalability	Security Features
Elastico [63]	Network Sharding	PBFT	Directory Committee	High	33% Byzantine Fault Tolerance
Omniledger [64]	Network Sharding	PBFT + Atomix	Atomix Protocol	High	DRG, VRF-based leader election
RapidChain [25]	Network + Transaction Sharding	PBFT	Cuckoo Rule	High	Byzantine Resilient DRG
Zilliqa [68]	Network Sharding	PBFT	Collective Signing (CoSi)	High	Scilla for Smart Contracts
Monoxide [65]	Network + Transaction Sharding	PoW + Eventual Atomicity	Chu-ko-nu Mining	High	PoW-based Sharding
Ethereum 2.0 [70]	State Sharding	PoS (Casper FFG)	Beacon Chain	High	Multi-stage RNG
Pyramid [69]	Layered Sharding	Layered Consensus	Overlapping Shards	High	Hierarchical DRG

included in a block. Similarly, Zilliqa [68] employs a two-layer consensus mechanism where a smaller committee first reaches consensus on the block and then broadcasts it to the entire shard for validation.

- **Cross-shard Consensus:** Transactions that span multiple shards require coordination across those shards. Omniledger introduces Atomix [64], a cross-shard atomic commit protocol that ensures transactions involving multiple shards are either fully committed or fully aborted, maintaining the atomicity of transactions. RapidChain [25], on the other hand, uses a variant of the Cuckoo rule to facilitate efficient cross-shard communication, reducing the overhead and latency typically associated with such operations.

- **Hybrid Approaches:** Some systems combine different consensus mechanisms to enhance security and scalability. For instance, Pyramid [69] integrates a hierarchical consensus mechanism that allows overlapping shards to share and validate information, thus reducing the complexity and latency of cross-shard transactions. This layered approach allows Pyramid to maintain high throughput while ensuring strong consistency across the network.

Challenge of Sharded Blockchain

- **Challenges of large number of Cross-Shard Transactions:** Scalability remains a major challenge for blockchain adoption, as traditional blockchains require every node to process all transactions and store a full copy of the ledger. This results in high latency and increased costs as the network grows. Sharding offers a solution by dividing the blockchain into smaller shards that can process transactions simultaneously, improving throughput and resource efficiency. However, as the number of shards increases, cross-shard transactions—transactions involving data from multiple shards—become more frequent and pose significant challenges. These transactions require coordination between shards, leading

to higher communication overhead, delays, and increased complexity in maintaining consistency across the blockchain. This limits the scalability benefits of sharding and complicates transaction validation and finalization, making it difficult to achieve low latency and high throughput in large-scale networks.

- **Challenges of Comprised Security:** While sharding significantly enhances the scalability of blockchain technology, it also introduces complex security challenges. The primary concern is ensuring the integrity and consistency of data across multiple shards. Transactions that span multiple shards need to be handled with strict coordination to prevent double-spending and ensure atomicity. Additionally, the smaller node count in individual shards could potentially lower the barrier for certain types of attacks, such as the 51% attack, where an attacker gains control of a majority of the nodes in a shard. Developing robust cryptographic protocols and consensus mechanisms that can secure intra-shard communication and validate cross-shard transactions is crucial for maintaining the security and immutability of the ledger.

- **Challenges of High Latency:** Latency is another critical challenge in sharded blockchains, particularly for cross-shard transactions [71, 72]. Since transactions that involve multiple shards require coordination across those shards, the process can become time-consuming. This inter-shard communication can introduce significant delays as the transaction must be validated and confirmed by multiple shards, each operating independently. The increased complexity in managing cross-shard dependencies and the need for synchronization among different shards can lead to higher latency, impacting the overall performance and user experience of the blockchain system. Addressing this issue requires developing efficient protocols for cross-shard communication and transaction validation that minimize delays while ensuring data consistency and security.

To overcome these challenges, cross-shard communication protocols are implemented. These protocols are designed to ensure the consistency and atomicity of transactions that span multiple shards. One example is the Omniledger protocol, which integrates intra-shard consensus with a global consensus layer. This global consensus ensures that the state of the entire blockchain remains synchronized, even as individual shards process transactions independently [25]. The use of atomic commit protocols, such as the two-phase commit or the Byzantine atomic commit, further guarantees that cross-shard transactions are either fully committed or fully aborted,

thereby preserving the consistency of the blockchain.

Moreover, sharded blockchain systems often employ novel consensus mechanisms such as RapidChain [25], which optimizes cross-shard transaction processing by dynamically adjusting shard sizes and utilizing fast intra-shard consensus algorithms. These innovations help reduce the latency and overhead associated with cross-shard communications, thereby enhancing the overall performance and scalability of the blockchain system.

In summary, the combination of robust DRG protocols, secure leader election mechanisms, and advanced cross-shard consensus protocols forms the foundation of a resilient and scalable sharded blockchain architecture. These components work in tandem to ensure that the system can handle a high volume of transactions while maintaining security, consistency, and decentralization across all shards.

1.3 Research Objective

Objective 1: Addressing the Challenges of Cross-Shard Transactions

Research Question 1: How can the efficiency of cross-shard transactions be improved to ensure scalability without compromising the security and atomicity of transactions?

Research Target: This research aims to develop and evaluate a novel cross-shard transaction protocol that optimizes transaction throughput while maintaining strong security guarantees. The proposed protocol will be designed to reduce the complexity of cross-shard communication, ensuring that transactions spanning multiple shards are processed efficiently and atomically. This will be explored in detail in Chapter III.

Objective 2: Enhancing Security in Sharded Blockchain Networks

Research Question 2: What security mechanisms can be implemented to prevent and mitigate attacks in sharded blockchains, particularly in environments with smaller node counts per shard?

Research Target: The research focuses on designing robust security framework that enhance the security of individual shards while maintaining overall network integrity. The target is to minimize the risk of attacks, such as the 51% attack, by implementing advanced artificial intelligent techniques and trust table. This objective will be thoroughly investigated in Chapter

IV.

Objective 3: Reducing Latency in Sharded Blockchain Systems

Research Question 3: How can latency be minimized in sharded blockchain systems, especially in the context of cross-shard transaction validation and communication?

Research Target: The research aims to develop and test innovative protocols that reduce the latency associated with cross-shard communications. The goal is to create a low-latency environment for blockchain operations by optimizing shard synchronization and transaction validation processes. This will be achieved by exploring efficient consensus mechanisms and cross-shard communication strategies, as detailed in Chapter V.

1.4 Organization of the Thesis

The thesis is based on the research results of 3 conference papers and 1 journal papers. The thesis is organized as follows:

- *Chapter 1*: This chapter sets the stage for the thesis by providing an overview of blockchain technology and its significance in modern digital transactions. It introduces the concept of blockchain sharding as a pivotal enhancement aimed at solving scalability and security issues inherent in traditional blockchain systems.

- *Chapter 2*: This chapter delves into the foundational theories and existing research surrounding blockchain sharding, emphasizing the role of community detection algorithms and their application to enhancing blockchain scalability and security. It explores trust-based and DRL-based sharding mechanisms, highlighting their potential to improve efficiency and robustness.

- *Chapter 3*: This chapter introduces a novel sharding scheme based on community detection algorithm aimed at optimizing the scalability and cost-efficiency of blockchain networks. It details the system model and the specific community detection mechanism used, proposing a budget-friendly sharding framework that minimizes transaction fees and processing overhead. The effectiveness of this scheme is demonstrated through rigorous experimental testing, with results suggesting significant improvements in transaction throughput and reduced latency.

- *Chapter 4*: In this chapter, the focus shifts to the TbDd (Trust-based and Deep Reinforcement Learning-based) Sharding Framework, which integrates trust mechanisms and DRL to dynamically manage shard allocation and optimize blockchain performance. The chapter outlines the theoretical adversary model, implementation details, and the practical application of DRL in managing shard behavior. Experimental results are presented to validate the framework's efficiency in handling adversarial scenarios and improving blockchain scalability and security.

- *Chapter 5*: This chapter introduces a novel framework specifically designed to address the complexities of cross-shard smart contract interactions in blockchain networks. The proposed framework leverages overlapping shards and an optimized xPBFT mechanism to transform cross-shard smart contract calls into more manageable intra-shard operations. The chapter provides a detailed analysis of the overlapping shard architecture, explaining how it integrates shard leaders to form a cohesive and efficient transaction processing unit. It also delves into the mechanics of xPBFT, tailored to enhance the speed and security of consensus in overlapping shards. Through comprehensive experimental evaluations, the chapter demonstrates how this framework significantly reduces transaction latency and enhances security, thereby offering a robust solution to the challenges of cross-shard smart contract execution.

- *Chapter 6*: In the concluding chapter, the research findings are synthesized, highlighting the significant contributions made to the field of blockchain sharding. This chapter reflects on the theoretical advancements and practical implications of the study, emphasizing the innovative solutions introduced for managing cross-shard smart contract interactions. The framework for overlapping shards and the xPBFT consensus mechanism are reviewed, summarizing their impact on improving efficiency and security in blockchain networks. The chapter also identifies limitations encountered during the research and proposes directions for future investigation. It suggests exploring the scalability of the proposed frameworks in larger and more complex networks, integrating emerging technologies, and developing adaptive strategies to further enhance the capabilities of sharding in blockchain systems. Recommendations are provided to guide subsequent research efforts aimed at building on the foundations established by this study and addressing the evolving challenges in blockchain technology.

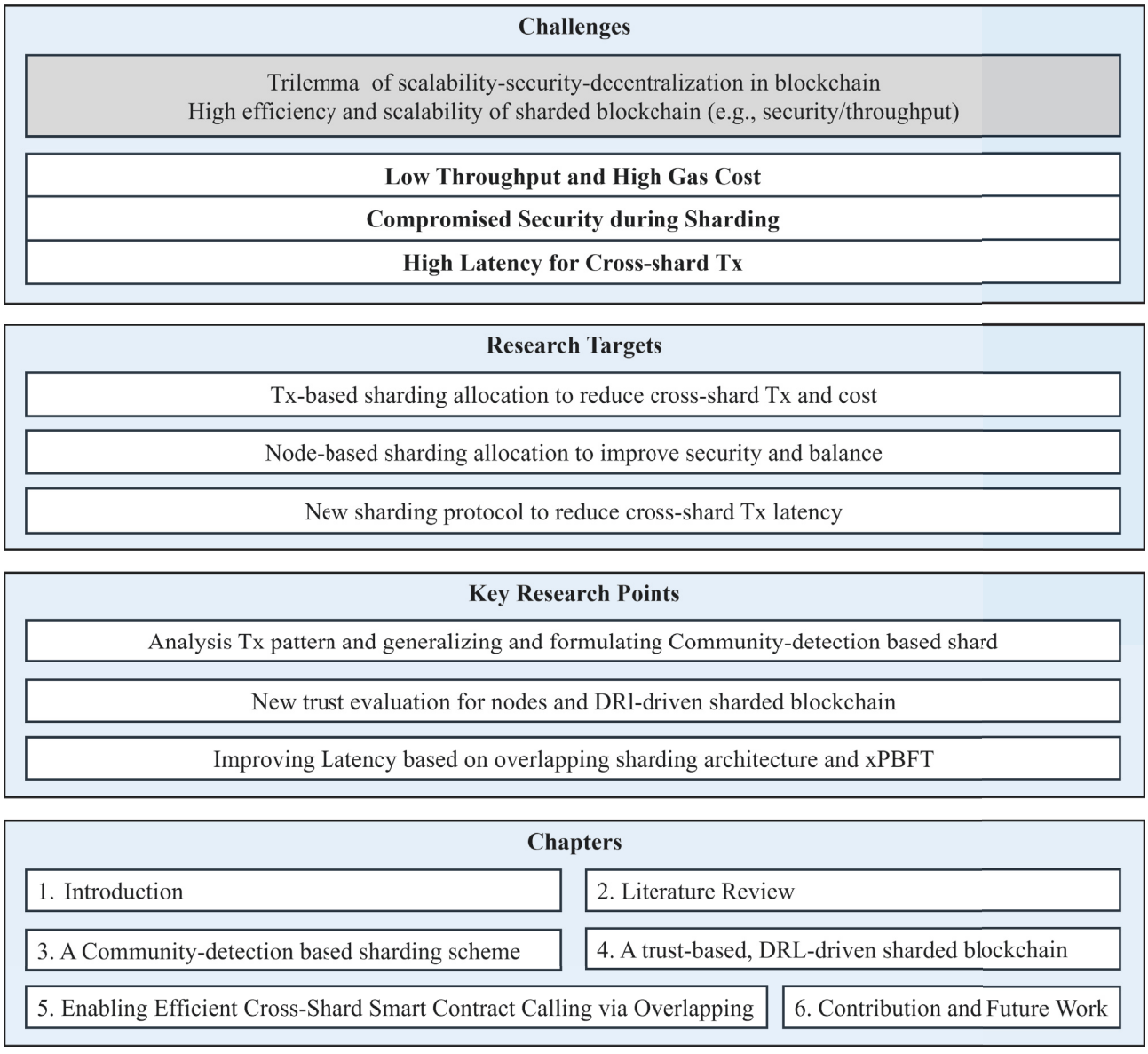


Figure 1.2: Thesis organization

Chapter 2

Literature Review

In this chapter, existing scaling solutions for blockchain from various layers are discussed in Section 2.1. Section 2.2 provides an overview of efficient sharding schemes. Security and balanced sharding approaches are presented in Section 2.3, followed by a summary of latency issues in blockchain sharding in Section 2.4. Finally, a conclusion is provided at the end of this chapter.

2.1 Blockchain Scalability Solutions

Blockchain scalability remains a key challenge, and various solutions have been proposed across different layers of the blockchain technology stack. These can be categorized into Layer 0 [73], Layer 1 (on-chain) [74–76], and Layer 2 (off-chain) approaches [27, 67].

2.1.1 An Overview of Layered Blockchain Scalability Solutions

Layer 0 Solutions

Layer 0 solutions focus on improving the integration between blockchain technology and traditional networking protocols, optimizing the dissemination of transactions and blocks across the network. These solutions enhance the underlying network infrastructure, enabling more efficient communication and data transmission. Polkadot [77] is a prominent example of *Layer 0* optimization, providing a foundational architecture called the Relay Chain, which allows various blockchains to interoperate while maintaining security and consensus. Similarly, jae *et al.* propose Cosmos [78] to creates an interoperable ecosystem of blockchains by enabling them

to communicate and exchange data through its Inter-Blockchain Communication (IBC) protocol. These projects exemplify the role of *Layer 0* in establishing a scalable and interconnected blockchain environment.

In addition to these architectures, several protocols aim to further optimize block and transaction propagation. For example, solutions like Compact Blocks [79], which related to block compression during block propagation, are also considered *Layer 0* optimizations as they enhance the efficiency of data dissemination across the network. Naumenko *et al.* [80] propose Erlay to reduces bandwidth consumption for Bitcoin's transaction relay. Chawla *el al.* propose Velocity [81] to improve block propagation by utilizing Fountain codes, a type of erasure code, to minimize the amount of data that needs to be propagated, further increasing the efficiency of the network. Rohrer *et al.* [82] introduce Kadcast, which is a novel peer-to-peer protocol for efficient block propagation using the structured overlay topology of Kademlia [83]. Kadcast improves broadcast efficiency by using adjustable overhead and includes forward error correction (FEC) to boost reliability. Klarman *et al.* [84] propose a Blockchain Distribution network, called bloXroute, designed to enhance cryptocurrency scalability. It introduces a neutral transport layer that optimizes the propagation of blocks and transactions, allowing for larger blocks and shorter intervals without changing the existing consensus mechanisms.

Layer 0 solutions enhance blockchain scalability, but their integration can be complex and costly, potentially leading to compatibility issues. Moreover, these optimizations may introduce new security risks, such as an increased susceptibility to network attacks like eclipse or DoS attacks, as faster communication often relies on fewer nodes. Cross-chain protocols like Polkadot [77] and Cosmos [78] add complexity in maintaining security across multiple chains. Additionally, *Layer 0* solutions primarily focus on network performance and do not directly address consensus mechanisms or transaction processing speeds, limiting their ability to fully resolve blockchain scalability challenges.

layer 1 Solutions

Layer 1 solutions have been widely studied, yet there remains room for further exploration to fully address blockchain scalability challenges. These solutions aim to improve scalability by directly modifying the blockchain's structure, consensus mechanisms, or transaction processing methods. One of the most promising approaches is sharding [26, 85], which divides the blockchain into smaller, independent units called shards. Each shard processes its own subset of

transactions in parallel, allowing the system to scale as more nodes are added. Sharding maintains decentralization while significantly boosting throughput. Sharding in Layer 1 solutions can be broadly classified into two main approaches: vertical scaling and horizontal scaling.

- Vertical scaling attempts to increase blockchain throughput by enhancing the capacity of individual nodes. For example, Bitcoin has experimented with increasing the number of transactions allowed per block or reducing the block generation time to improve throughput. However, these methods come with significant trade-offs, as they require more storage, computation, and bandwidth from each node [86, 87]. Such resource demands may strain smaller nodes, leading to potential centralization risks. Beyond these changes, Ethereum introduced the Greedy Heaviest Observed Subtree (GHOST) protocol [88], which organizes blocks in a tree structure rather than a simple linear chain. This modification increases throughput by allowing more flexible block validation. The GHOST protocol was later extended into a Directed Acyclic Graph (DAG) structure [89,90], where transactions are organized in a graph instead of a linear sequence.

- Horizontal Scaling, as exemplified by sharding, distributes the workload across multiple nodes. By dividing the blockchain into shards, each processing its own transactions in parallel, horizontal scaling allows the system to grow efficiently while maintaining decentralization.

Other notable Layer 1 solutions include:

- Bigger Blocks: Increasing block size allows more transactions to be processed per block, but this approach can lead to longer block propagation times and risks centralizing control in nodes with more resources [91].

- Directed Acyclic Graphs (DAGs): In DAG-based systems like IOTA [92] and Spectre [93], transactions are organized in a graph structure, enabling parallel processing of multiple transactions. While this design improves scalability, it introduces challenges in maintaining security and transaction verification. In the DAG model, each transaction references previous transactions, allowing them to be processed in parallel, which significantly improves node resource utilization and overall network throughput [94,95]. However, vertical scaling has inherent limitations, as it relies on enhancing individual node capacity, which becomes impractical when hardware and bandwidth improvements fail to keep pace with the exponential growth of transactions. Additionally, higher resource demands may ex-

clude resource-constrained nodes, undermining decentralization and increasing the risk of centralization.

Among these, sharding stands out as one of the most promising solutions for blockchain scalability, offering significant potential to increase throughput while maintaining decentralization and security.

Layer 2 (off-chain)

Layer 2 solutions, such as Payment Channels and Sidechains, move transaction processing off-chain to reduce the load on the main blockchain. These solutions handle high-frequency transactions off-chain and periodically update the blockchain, improving scalability without altering the base blockchain [96].

- Payment Channels: Solutions like the Lightning Network and Raiden Network enable transactions to occur off-chain, with only the final transaction states being recorded on the blockchain. This reduces the number of on-chain transactions and enhances scalability for micropayments [97, 98].

- Side Chains: Side chains like Plasma and Bitcoin Rootstock enable the off-chain execution of smart contracts and transactions, which are periodically anchored to the main chain. This method allows for increased transaction throughput without sacrificing the security of the main blockchain [99].

Blockchain scalability remains a persistent challenge, with solutions being developed at multiple layers, including the consensus, network, and application layers. At the consensus layer, protocols such as PoS and DPoS have been proposed to reduce the resource intensity of PoW systems while enhancing throughput. BFT-based consensus algorithms, like Tendermint [100], also aim to improve transaction finality and scalability by enabling faster agreement among nodes.

At the network layer, Layer 2 solutions such as payment channels (e.g., Lightning Network [97]) and state channels [101] have emerged to alleviate congestion on the main blockchain by handling transactions off-chain and only submitting final states to the blockchain. Rollups, particularly optimistic and zk-rollups [102], have also gained traction, enabling off-chain computation with periodic state updates to the blockchain, drastically reducing on-chain transaction volume.

At the application layer, dApps [103] rely on scalable infrastructure for seamless operation. Vertical scaling approaches, including sharding techniques, are widely considered the future for blockchain scalability. These approaches aim to divide the blockchain into smaller, more manageable shards that can process transactions independently, as discussed in detail in Section 2.2.

2.1.2 Sharding: A Promising layer 1 Scaling Solution

Sharding has emerged as a promising Layer 1 scaling solution, offering a practical approach to overcoming blockchain scalability challenges. By partitioning the blockchain into smaller, independent units known as shards, sharding enables the parallel processing of transactions, significantly increasing throughput while preserving decentralization. This horizontal scaling approach distributes computational and storage workloads across multiple nodes, ensuring the network can grow without relying on centralized control or excessively powerful individual nodes. Pioneering sharding protocols such as Omniledger [64], Zilliqa [104], and Ethereum 2.0 [66] have introduced advanced consensus mechanisms and randomness generation techniques to enhance security and efficiency. These techniques ensure fair node assignment and mitigate the risks of collusion or shard takeovers. However, sharding also presents notable challenges, particularly in managing cross-shard communication and ensuring the security of individual shards. Transactions spanning multiple shards require efficient coordination mechanisms to maintain consistency and prevent bottlenecks. Moreover, the presence of malicious nodes within a shard can compromise its integrity, posing a significant security threat.

2.2 An Efficiency Sharding Scheme

Sharding is increasingly recognized as a key strategy for boosting the scalability of blockchain systems. A primary concern in sharding is ensuring the security of individual shards while also improving the overall system throughput. The process of allocating nodes to shards has evolved significantly, with DRG protocols playing a critical role in ensuring the randomness of node assignments. For example, in Elastico [63], nodes are assigned to shards based on the last few bits of a PoW puzzle solution, which is influenced by DRG-derived epoch randomness. Similarly, Omniledger [64] uses a decentralized ledger system where validators are randomly assigned to shards, with shard leaders using the RandHound DRG protocol to produce randomness.

RapidChain [25] also adopts unique random allocation methods, utilizing the Feldman Verifiable Secret Sharing (VSS) DRG to generate unbiased random outputs and the Commensal Cuckoo rule for node assignment.

While DRG protocols are widely used, other methods also exist for node allocation. Monoxide [65] allocates nodes to shards based on address prefixes, whereas Chainspace [105] employs a more democratic approach, allowing nodes to move between shards based on votes from other nodes, facilitated by the Manage Shards smart contract. In Fabric [35], sharding is achieved by deploying different shards on different channels, with a trusted entity handling cross-shard transactions.

Beyond security, another critical aspect of sharding is the reduction of cross-shard transactions, which directly impacts throughput. Classical graph partitioning algorithms like Kernighan-Lin [106] have been proposed as effective methods for optimizing the sharding process by minimizing interconnections between communities, thus reducing cross-shard transactions. Advanced techniques such as community detection, as proposed by Zhang *et al.* [107], strategically cluster nodes with high transactional interactions into the same shard, further minimizing the overhead of cross-shard transactions.

Despite these advancements, existing blockchain sharding schemes—whether based on randomness, voting, or channels—have not fully addressed the challenge of improving scalability by reducing cross-shard transactions. While promising, graph partitioning and community detection algorithms remain underdeveloped for blockchain sharding, especially for blockchains that require high scalability. A more integrated approach that balances security, throughput, and user experience is necessary for the next generation of scalable blockchain systems.

2.3 Achieving Secure and Balanced Blockchain Sharding Solutions

Trust-based sharding divides the blockchain network into shards based on node trust, with the aim of enhancing security and performance by grouping trusted nodes together while evenly distributing dishonest nodes. Notably, research by Yun *et al.* [108] introduced a Trust-Based Shard Distribution (TBSD) scheme that aims to distribute nodes across shards based on trust metrics to minimize collusion risks and ensure the security of transactions. However, while

this approach addresses security concerns, it often overlooks the impact on system throughput and load balancing, which can lead to inefficiencies. Huang *et al.* [109] introduced RepChain, a reputation-based system that uses a double-chain architecture for high throughput and security, though this increases system complexity and resource requirements. Zhang *et al.* [28] proposed a sharding model that considers trust, latency, and node count differences to reduce the risk of blockchain failure, though its effectiveness depends on obtaining accurate information, which can be challenging in dynamic environments.

In addition to these sharding techniques, the application of Deep Reinforcement Learning (DRL) in blockchain sharding has garnered attention for its potential to optimize sharding processes in real-time. Liu *et al.* [110] were early pioneers in applying DRL to blockchain frameworks for Industrial IoT (IIoT), though their work did not address dishonest attacks. Similarly, Liu *et al.* [111] utilized DRL with Ethereum for IIoT data security but overlooked throughput scalability. Qiu *et al.* explored DRL for refined block production and bandwidth allocation in service-oriented blockchain solutions [112,113], but these lacked centralized security measures. Yun *et al.* [114] introduced a DQN-optimized framework for sharded blockchains, which, while advanced, did not address intricate attack strategies in DRL-based sharding. Yang *et al.* [115] incorporated K-means clustering with DRL for optimization, but their approach was computationally intensive.

Most existing DRL-based sharding methods lack effective analysis of dishonest attacks, particularly in IoT contexts. The proposed framework, TbDd, addresses this gap by leveraging DRL to optimize sharding in real-time, providing a robust defense against strategic collusion and ensuring balanced node distribution. TbDd enhances the security and scalability of blockchain systems within IoT scenarios, improving throughput and efficiency and adapting to the evolving demands of real-world IoT environments.

2.4 Low latency Blockchain Sharding

Sharding is a crucial technique for enhancing blockchain scalability by dividing the network into smaller, independently functioning segments known as shards. While this approach significantly improves throughput, it also introduces complexities, particularly in managing cross-shard transactions involving smart contracts, which demand substantial coordination across shards. To address these challenges, the PYRAMID model introduced by Hong *et al.* [116]

offers a layered sharding approach where shards can overlap, allowing nodes to participate in multiple shards simultaneously. This overlapping design enables nodes to directly validate and execute cross-shard transactions without decomposing them into sub-transactions, thereby streamlining the process. However, despite these improvements in throughput, the security of cross-shard transactions remains a critical concern.

Liu *et al.* [117] tackled these security issues by proposing a secure cross-shard view-change protocol, designed to safeguard against malicious shard leaders and ensure transaction integrity across the blockchain. Complementing this, Sonnino *et al.* [118] focused on enhancing consensus mechanisms for sharded blockchains, utilizing Byzantine Fault Tolerance (BFT) algorithms to ensure consistency and fault tolerance across shards, thereby securing the validation and commitment of transactions. Zamani *et al.* [25] further explored sharding protocols that optimize resource usage while maintaining scalability as the number of nodes and transactions increases. Additionally, Yang *et al.* [119] expanded on the concept of overlapping shards, allowing nodes to belong to multiple shards. This innovation converts cross-shard transactions into intra-shard transactions, reducing latency and improving security. They also integrated machine learning techniques to dynamically optimize shard configurations based on real-time network conditions.

Despite these advancements, many existing solutions [25, 64, 116, 118, 119] continue to face challenges, such as high latency in cross-shard smart contract transactions and difficulties in handling their complex dynamics. To overcome these obstacles, this paper presents a novel framework for enabling efficient cross-shard smart contract calling. By leveraging overlapping shards and the xPBFT consensus mechanism, this framework treats cross-shard transactions as intra-shard operations, significantly reducing latency and enhancing security. This approach offers a robust solution for the scalability challenges inherent in blockchain architectures, particularly in the context of smart contract execution.

2.5 Conclusion

In conclusion, this chapter has examined the critical challenges and advancements in blockchain sharding, focusing on community detection algorithm, trust table, DRL optimization, and the efficient handling of cross-shard smart contract transactions. While current sharding techniques offer significant improvements in scalability and security, they often struggle with high latency and the complexity of cross-shard interactions. The proposed framework, which leverages

overlapping shards and the xPBFT consensus mechanism, offers a robust solution by converting cross-shard transactions into intra-shard operations. This approach not only reduces latency but also enhances security, providing a scalable and efficient architecture for blockchain systems, particularly in the context of smart contract execution. Future research should explore further integration of machine learning techniques for dynamic shard optimization and the potential application of these methods to broader blockchain scenarios, ensuring the system's adaptability to evolving network conditions.

Chapter 3

Community Detection-based Sharding Scheme

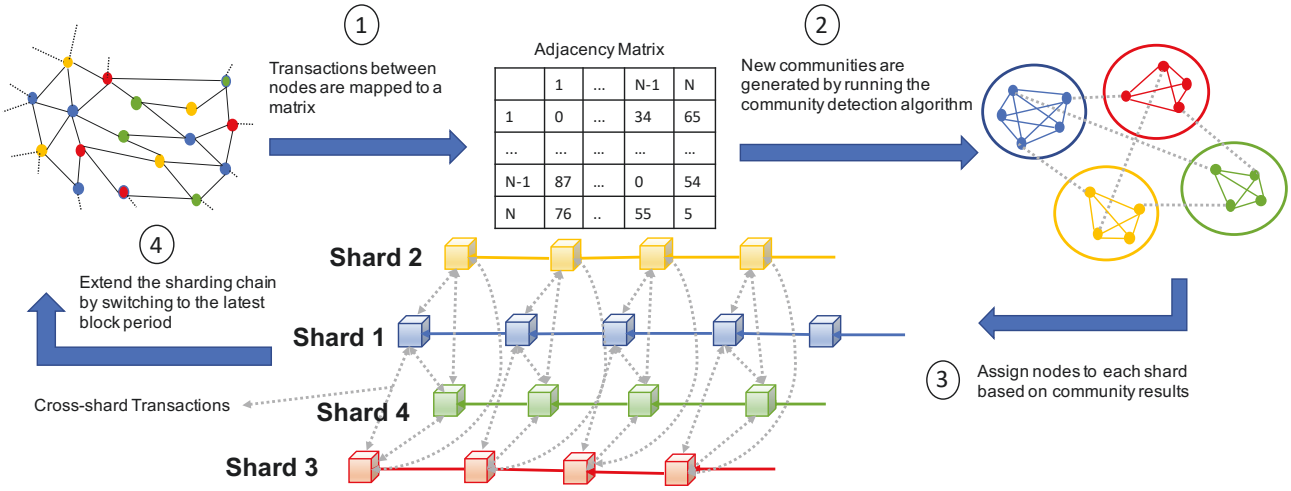
3.1 Introduction

Sharding has been considered a promising approach to improving blockchain scalability. However, multiple shards result in a large number of cross-shard transactions, which require a long confirmation time across shards and thus restrain the scalability of sharded blockchains. In this chapter, we convert the blockchain sharding challenge into a graph partitioning problem on undirected and weighted transaction graphs that capture transaction frequency between blockchain addresses. We propose a new sharding scheme using the community detection algorithm, where blockchain nodes in the same community frequently trade with each other. The detected communities are used as shards for node allocation. Futhermore, considering about the promise of blockchain networks is occasionally overshadowed by challenges related to scalability and the escalation of gas fees, we propose a budget-friendly approach to sharding by leveraging community detection algorithms. By clustering tightly-knit transactional communities, our method encourages more intra-shard transactions, thus minimizing the higher fees associated with cross-shard transactions.

3.2 The Proposed Method

3.2.1 The Proposed Community Detection-based Sharding

We propose a novel blockchain sharding scheme using the community detection algorithm. An adjacency matrix representing transactions (TXs) between node pairs is the input parameter for the system model we presented. Our model uses the Louvain algorithm in community detection [120] to obtain communities that respond to the sharding result. All nodes are assumed to be trustworthy in the model. Under our assumption, cross-shard transactions are reduced to a lower frequency based on our model. Therefore, our community detection-based sharding model is more suitable for permissioned chains since security issue is out of scope. Notations used in this paper are collected in Tab. 3.1.



	1	...	N-1	N
1	0	...	34	65
...	...	...	...	...
N-1	87	...	0	54
N	76	..	55	5

Figure 3.1: The flow diagram of the proposed Re-sharding system comprises the following four stages. ① Graph generation: generate an adjacency matrix according to the number of transactions between nodes. ② Community detection: run the community detection algorithm to identify node communities so that nodes frequently exchange with nodes in the same community and transact less with nodes in different communities. ③ Community-based sharding: allocate nodes to shards according to detected communities, where nodes in the same community are in the same shard, and implement the sharding result. ④ Chain extension: extend the chains in parallel.

The blockchain sharding network is reviewed as an undirected weighted graph $G = (V, E)$. The set of vertices V represents node addresses, and the set of edge E represents the transaction number between node pairs. This G is incorporated into the community detection algorithm

using the adjacency matrix format.

Table 3.1: notation and definition in the blockchain sharding system.

Notation	Description
$\mathcal{N}$	Set of nodes
$\mathcal{C}$	Set of community
$\mathcal{S}$	Set of shard
$\mathcal{B}_p$	An epoch of blocks
$\mathcal{E}_t$	t-th block epoch
r	The cut weight ratio
$\mathcal{A}$	The adjacency matrix of the transaction graph
s	The number of shards
ϕ_i	The number of Intra-shard transactions
ϕ_c	The number of Cross-shard transactions
ρ	Threshold for the cross-shard transaction ratio
TX	The transaction dataset
TX_n	Number of transactions
R	The transaction fee
R_{in}	The total of Intra-shard transaction fee
R_{cr}	The total of Cross-shard transaction fee
R_{total}	The total transaction fee
η	The system expenditure
$\mathcal{B}$	The adjacency matrix of the transaction fee
γ	The adjacency matrix of the system expenditure
φ_{cr}	The ratio of Cross-shard transactions

Fig. 3.1 illustrates the sharding system we use to reduce R_c. This system performs four stages: Graph generation, Community detection, Community-based sharding, and Chain extension. The following Eq. 3.1 calculates the ratio of cross-shard transactions.

$$\varphi_c = \frac{\phi_c}{\phi_i + \phi_c}. \tag{3.1}$$

STAGE 1: Graph generation (Alg. 1 lines 1-7). The model generates an adjacency matrix $\mathcal{A}$ based on the number of transactions between nodes. As shown in Alg. 1, the adjacency matrix

Algorithm 1: Community-based Sharding

▷ ***Sharding($\mathcal{B}_p$, s)***

Input:

$\mathcal{B}_p$: An epoch of blocks;

s: The number of shards.

Output:

$\mathcal{S}$: Shards with nodes;

r: The cut weight ratio.

$\mathcal{A} = \mathbf{0}$

$\mathcal{N} \leftarrow$**GetAllNodes**($\mathcal{B}_p$)

`// Sort nodes according to the number of transactions.`

$\mathcal{N}_s \leftarrow$**Sort**($\mathcal{N}$, TX)

for $i \leq |\mathcal{N}_s|$ **do**

  for $j \leq |\mathcal{N}_s|$ **do**

    $a_{i,j} \leftarrow$**CountTX**($\mathcal{B}_p$, N_i, N_j)

$\mathcal{C} \leftarrow$ **CommunityDetection**($\mathcal{A}$, s)

`// Calculate the cut weight ratio, which is the sum of weights of the edges crossing the communities divided by the sum of weights of all edges.`

$r \leftarrow$ **CalculateCutRatio**($\mathcal{A}$, $\mathcal{C}$)

$\mathcal{S} \leftarrow$ **MapNodesToCommunities**($\mathcal{N}_s$, $\mathcal{C}$)

return $\mathcal{S}$, r

▷ ***ShardsExtension()***

Input:

ρ: The threshold of cross-shard TX ratio;

s: The number of shards.

while *True* **do**

  ShardsBlockMining()

  if *Shard heights reach an epoch* **then**

    $\mathcal{B}_p \leftarrow$**Blocks in the latest period**

    $\phi_i \leftarrow$**CountIntraShardTX**($\mathcal{B}_p$)

    $\phi_c \leftarrow$**CountCrossShardTX**($\mathcal{B}_p$)

    if $\frac{\phi_c}{\phi_i+\phi_c} > \rho$ **then**

      $\mathcal{S},\ r =$ **Sharding**($\mathcal{B}_p$, s)

      `// Reallocate nodes to new shards.`

      ReAllocateNodes($\mathcal{S}$)

      $\rho \leftarrow \max(\rho, r)$

is initialized with zero. The algorithmic flow starts with obtaining the dataset of the node $\mathcal{N}$ that sends and receives transactions from the selected block epoch. Then, node dataset $\mathcal{N}$ are sorted based on the number of transactions to a sorted node dataset $\mathcal{N}_s$. An adjacency matrix $\mathcal{A}$ is generated after traversing all node sets. The size of the $\mathcal{A}$ depends on the node number. The i-th row and the j-th column in matrix $a_{i,j}$ represents the number of transactions between nodes N_i and N_j. Matrix diagonal elements represent the number of transactions between nodes and themselves.

STAGE 2: Community detection (Alg. 1 lines 8-9). The community detection algorithm generates a community set $\mathcal{C}$ using an adjacency matrix $\mathcal{A}$ and a fixed number of shards s as input parameters. An algorithm for detecting communities facilitates more frequent exchanges between nodes within a community and less exchange between nodes from different communities. To measure the quality of the sharding result, we define a cut weight ratio r. Cut weight ratio is calculated by dividing the sum of edge weights crossing communities by the sum of all edge weights.

STAGE 3: Community-based sharding (Alg. 1 lines 10-11). Based on the optimized outputs of the community set $\mathcal{C}$ from previous stages, sorted nodes are assigned to the new communities and the filled shard set $\mathcal{S}$ is returned. Filled shards set $\mathcal{S}$ with sorted nodes and a cut weight ratio r are returned at the end of each epoch.

STAGE 4: Chain extension (Alg. 1 lines 12-21). As the new shards return, each chain mines blocks and extend in parallel until it reaches the end of an epoch. According to returned sharding results, the number of intra-chain transactions ϕ_i and the number of cross-chain transactions ϕ_c are separately counted. According to Eq. 3.1, the statistical results of both transactions are used to calculate a ratio of cross-shard transactions φ_c. We set a threshold ρ for cross-shard transactions in our system. The threshold determines whether or not a node needs to be reassigned during a block epoch $\mathcal{E}_t$. A node is not re-allocated if φ_c is less than the threshold. Otherwise, nodes are re-allocated to other shards based on the community detection algorithm. Additionally, the value of ρ is based on the largest φ_c.

3.2.2 Proposed Budget-friendly Sharding Framework

A budget-friendly sharding framework based on community detection algorithms is proposed in this paper. Our model uses the Louvain algorithm [120] in community detection to dis-

cern communities that align with optimal sharding results. The Louvain algorithm effectively discovers small and large communities within networks by maximizing a modularity score, thereby determining the natural divisions of nodes based on their interactions. By leveraging the strengths of this algorithm, the system embarks on a multi-stage procedure, articulated as stages 1 through 4, as shown in Fig. 3.2. These stages encompass the generation of adjacency matrices, expenditure calculation, community formulation, and a trigger mechanism for node resharding.

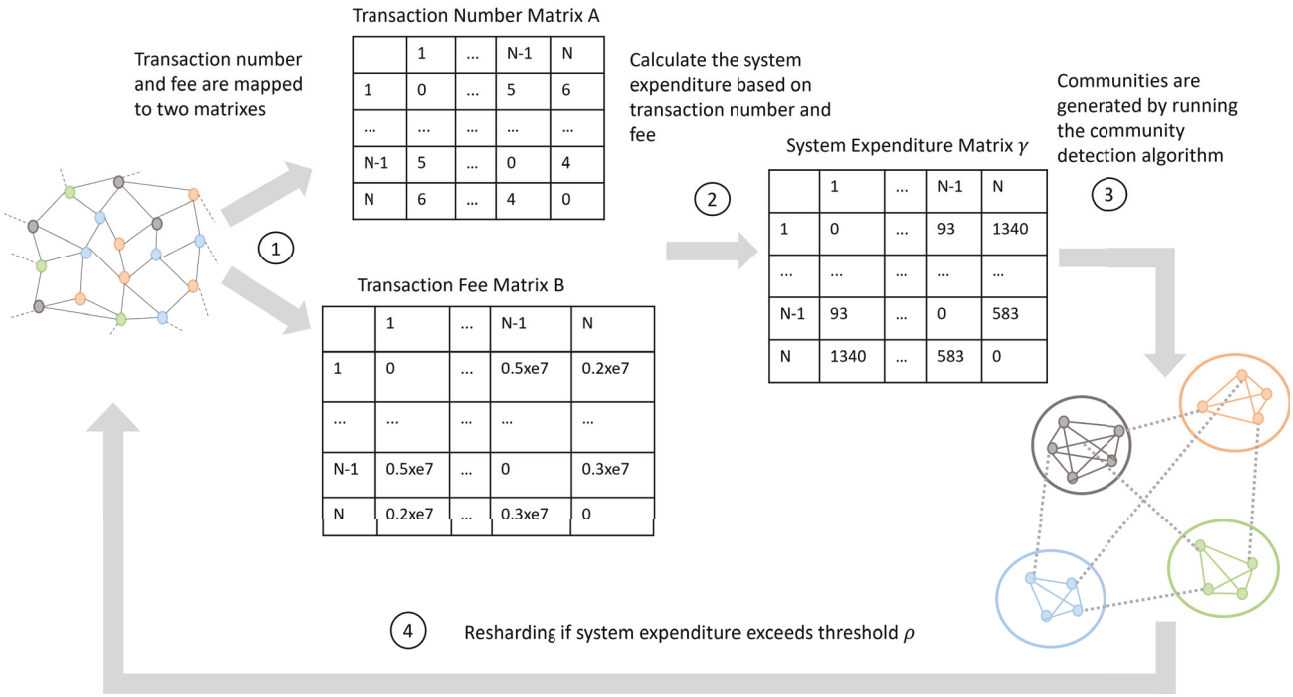


Figure 3.2: The proposed Re-sharding system flow diagram comprises the following four stages. ① Transaction number and Transaction fee matrix generation: generate two adjacency matrixes according to the number of transactions and Transaction fee. ② System expenditure matrix generation: calculate the system expenditure matrix based on Eq. 3.2. ③ Community detection: Implement the community detection algorithm to establish shards. This arrangement promotes frequent interactions between nodes within the same community, reducing the Transaction fee expenditure by the system. ④ Resharding: If the system's expenditure breaches the predefined threshold, a resharding procedure is triggered.

The system model is primarily anchored on adjacency matrixes constructed from the transaction counts between nodes and the corresponding transaction fee expenditures. A new system expenditure matrix is crafted by merging these matrices, serving as a pivotal input. Our design aims to minimize cross-shard transactions, reducing the system's cumulative gas expenditure.

All pertinent notations utilized throughout this paper are cataloged and can be referenced in Tab. 3.1.

We conceptualize the blockchain sharding network as an undirected weighted graph, denoted as $G = (V, E)$. In this formulation, the graph G is subsequently processed through community detection using its adjacency matrix portrayal. The vertex set V corresponds to node addresses, whereas the edge set E encapsulates the aggregate transaction numbers between node pairs and their associated gas fee expenditures. The gas fee expenditure R is calculated as:

$$R = gas \times gas\_price \tag{3.2}$$

where the gas is the unit of measure that quantifies the computational effort required to execute operations, and the gas_price is the amount of Ether (ETH) a user is willing to spend on every unit of gas [121].

Thus, the system expenditure η is calculated as follows:

$$\eta = \alpha TX_n + \beta R \tag{3.3}$$

where TX_n is the number of transaction, R is the gas fee expenditure, α and β are the weight of the TX_n and R.

The total transaction expenditure R_{total} is calculate as:

$$R_{total} = R_{in} + \mu R_{cr} \tag{3.4}$$

where R_{in} and R_{cr} represent the intra-shard transaction fee and cross-shard transaction separately, and μ is the increased transaction fee weight for cross-shard transactions.

STAGE 1: Matrix generation (Alg. 2 lines 1-7). In this framework, two adjacency matrices are formulated: $\mathcal{A}$, signifying the transaction counts between nodes, and $\mathcal{B}$, denoting the transaction fee based on Eq. 3.2. As outlined in Alg. 2, these matrices start with zero values. The procedure commences by extracting the transaction dataset, which enumerates transactions between senders and receivers. Then, the $\mathcal{N}$ dataset is organized into the node dataset $\mathcal{N}_s$, sorted by transaction count. The dimensions of $\mathcal{A}$ and $\mathcal{B}$ are contingent upon the node count. In matrix $\mathcal{A}$, the element $a_{i,j}$ indicates the transactions between nodes N_i and N_j. Similarly, in matrix $\mathcal{B}$, $b_{i,j}$ conveys the transaction fee from node N_i to N_j. Diagonal elements in both matrices account for self-transactions within nodes, capturing transaction counts and corresponding transaction fees.

Algorithm 2: Gas expenditure-based sharding

1 ▷ ***sharding()***

Input:

TX: The transactions set;

s: The number of shards;

R: The transaction fee;

Output:

$\mathcal{S}$: Shards with nodes;

3 $\mathcal{A} = \mathbf{0}$, $\mathcal{B} = \mathbf{0}$, $\gamma = \mathbf{0}$

4 $\mathcal{N} \leftarrow$**GetAllNodes**(TX)

5 $\mathcal{N}_s \leftarrow$**Sort**$(\mathcal{N}, TX_n)$ `// Sort nodes according to the transaction number.`

7 **for** $i \leq |\mathcal{N}_s|$ **do**

8 **for** $j \leq |\mathcal{N}_s|$ **do**

9 $a_{i,j} \leftarrow$**CountTXNumber**(TX_n, N_i, N_j)

10 $b_{i,j} \leftarrow$**CountTXFee**(R, N_i, N_j)

11 $\gamma \leftarrow$ **CalculateSystemExpenditure**$(\mathcal{A}, \mathcal{B})$

12 `//Calculate the system expenditure based on Eq. 3.3.`

14 $\mathcal{C} \leftarrow$ **CommunityDetection**(γ, s)

15 $r \leftarrow$ **CalculateCutRatio**$(\gamma, \mathcal{C})$

16 `//Calculate the cut weight ratio, which is defined as the collective weight of edges that traverse communities divided by the overall weight of every edge in the system.`

18 $\mathcal{S} \leftarrow$ **MapNodesToCommunities**$(\mathcal{N}_s, \mathcal{C})$

19 **return** $\mathcal{S}, r$

20 ▷ ***Resharding()***

Input:

ρ: The threshold of system expenditure;

s: The number of shards.

22 **while** *True* **do**

23 **TransactionUpdate**(TX)

24 $R_{in} \leftarrow$**CountIntraShardTXFee**(TX)

25 $R_{cr} \leftarrow$**CountCrossShardTXFee**(TX)

26 $R_{total} \leftarrow$**CountTotalTXFee**(R_{in}, R_{cr})

27 `//Calculate the overall transaction fee based on Eq. 3.4.`

29 **if** $R_{total} > \rho$ **then**

30 $\mathcal{S}$, r=**Sharding**(TX, s)

31 `// Reallocated nodes to new shards.`

33 **ReAllocateNodes**$(\mathcal{S})$

34 $\rho \leftarrow \max(\rho, r)$

STAGE 2: System expenditure calculation(Alg. 2 lines 8). According to Eq. 3.3, the system expenditure η is calculated based on the transaction number TX_n and transaction fee R, forming a new matrix γ.

STAGE 3: Community generation (Alg. 2 lines 9-12). The optimized outputs of the

community set $\mathcal{C}$ from previous stages assign sorted nodes to new communities. Consequently, this process generates the filled shard set $\mathcal{S}$ and returns it with the accompanying cut weight ratio denoted as r.

STAGE 4: Resharding triggering(Alg. 2 lines 13-21). As the new transaction dataset is updated, the intra-shard transaction fee R_{in} and the cross-shard transaction fee R_{cr} are separately counted. To mimic a situation where cross-shard transactions fees incur greater value than intra-shard ones, our model amplifies the weight attributed to the transaction fee expense of cross-shard transactions when calculating the cumulative transaction fee. The threshold ρ dictates the reassignment necessity of a node. If R_{total} falls below this threshold, the node remains in its current position. Conversely, if it exceeds, nodes are relocated to different shards following the community detection algorithm. Moreover, ρ is determined by the maximum value of R_{total}.

Following the completion of stage 4, the nodes undergo a reassignment phase, marking the initiation of the resharding process. This adjustment is inherently responsive, recalibrating to accommodate newly emerging transaction datasets. By iterating through these stages repeatedly, the system is poised to adapt continually, offering flexibility and efficiency.

3.3 Experimental Results

3.3.1 Experiment Settings for the Proposed Community Detection-based Sharding

An experimental framework is implemented in a local environment (MacBook Pro with 2.5 GHz Quad-Core Intel Core i7 and 16GB memory) to evaluate a proposed blockchain sharding scheme. We create a virtual machine using Conda on Visual Studio Code and implement our framework in Python 3.10.44. Ethereum block data ranging from 13.7 million to 15.04 million are downloaded and sorted from the Ethereum public endpoint [122]. There are 1.34 million blocks in the captured block range. Experiments are conducted on consecutive 100,000 blocks randomly selected from captured block ranges. Our randomly selected block period is between 14 million and 14.9 million. We divide the selected block period into ten equal parts, each containing 10,000 blocks.

Data Overview

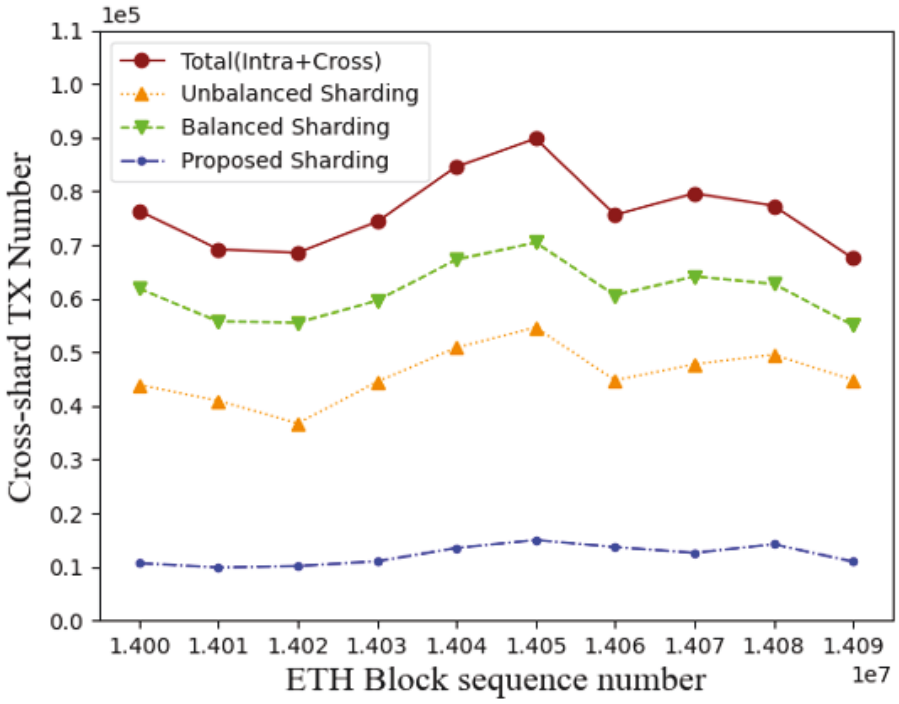
We have 762,203 node addresses and 3,735,641 transactions during our first test block epoch. The community detection algorithm uses the top 90 addresses [1] sending the most transaction numbers out of 762,203 addresses. We define K as the number of addresses with the most transactions in subsequent tests. After sorting the first 90 nodes, we see that the address with the most transactions belongs to OpenSea [2]. In our first tested 10,000 blocks, OpenSea generated 154,879 transactions. Besides OpenSea, big companies like Coinbase [3] and Uniswap [4] also have large numbers of transactions on Ethereum. According to the adjacency matrix, 31 nodes do not have any transactions with any of the first 90 nodes. The row sum of all the 31 sender addresses in the matrix is zero. Thus we remove these 31 addresses, and the top 90 most transacted addresses form a 59×59 adjacency matrix.

Sharding Analysis

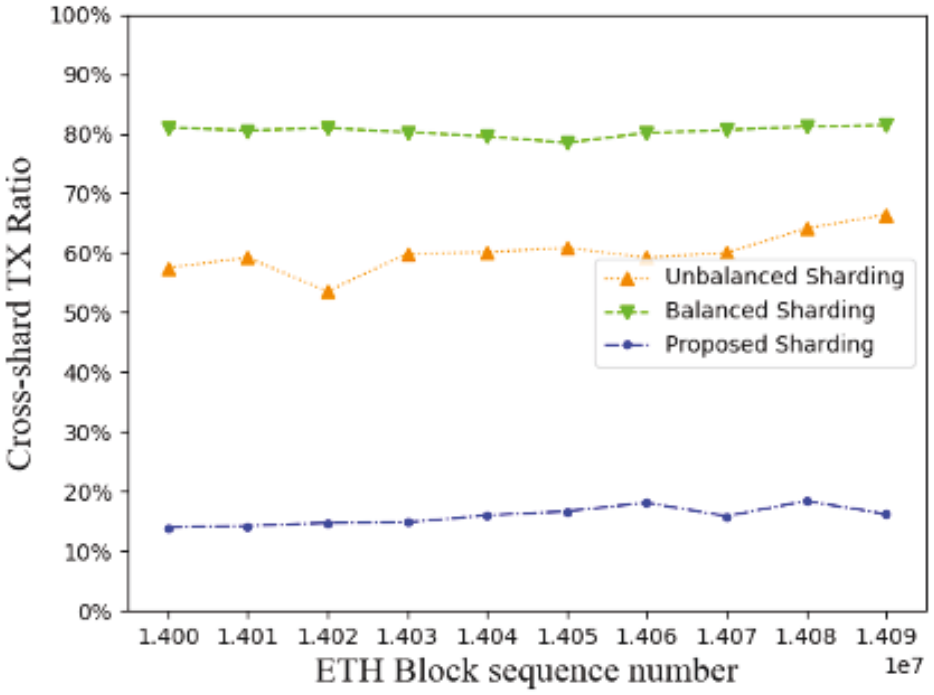
The comparison between a community detection-based sharding method and a random sharding method is shown in Fig. 3.3. In Fig. 3.3a, the vertical axis represents the number of cross-shard transactions, while the horizontal axis represents the Ethereum block sequence number. Fig. 3.3b differs from Fig. 3.3a in that the vertical axis represents the ratio of cross-shard transactions, while the horizontal axis remains the same. During the first block epoch, we divide the 59 addresses into four shards, which generate 76,369 transactions. As a result of adopting the community detection algorithm, 65,703 transactions are intra-shard transactions, and 10,666 are cross-shard transactions.

To compare with the community detection-based sharding scheme, we simulate a process of randomly assigning addresses, similar to the process of nodes randomly assigned to varying shards in Omniledger [64]. Nodes can be randomly allocated into shards of the same size. Alternatively, nodes can be randomly divided into shards of different sizes. Therefore, we develop two methods for randomly allocating nodes to shards in the experiments. Keeping each shard with the same number of nodes is balanced sharding. Sharding with more or fewer nodes in each shard is unbalanced.

[1] Due to the limited computational capacity, the local computer can only handle up to 90 addresses.
[2] 0x7be8076f4ea4a4ad08075c2508e481d6c946d12b, https://opensea.io/
[3] 0x503828976d22510aad0201ac7ec88293211d23da, https://www.coinbase.com/
[4] 0xd3d2e2692501a5c9ca623199d38826e513033a17, https://uniswap.org/



(a) Unbalanced random allocation method



(b) Balanced random allocation method

Figure 3.3: Testing Ethereum address in sharding framework under unbalanced random allocation method, balanced random allocation method, and community detection-based allocation method while fixing tested addresses number $K = 90$ and shard number $s = 4$.

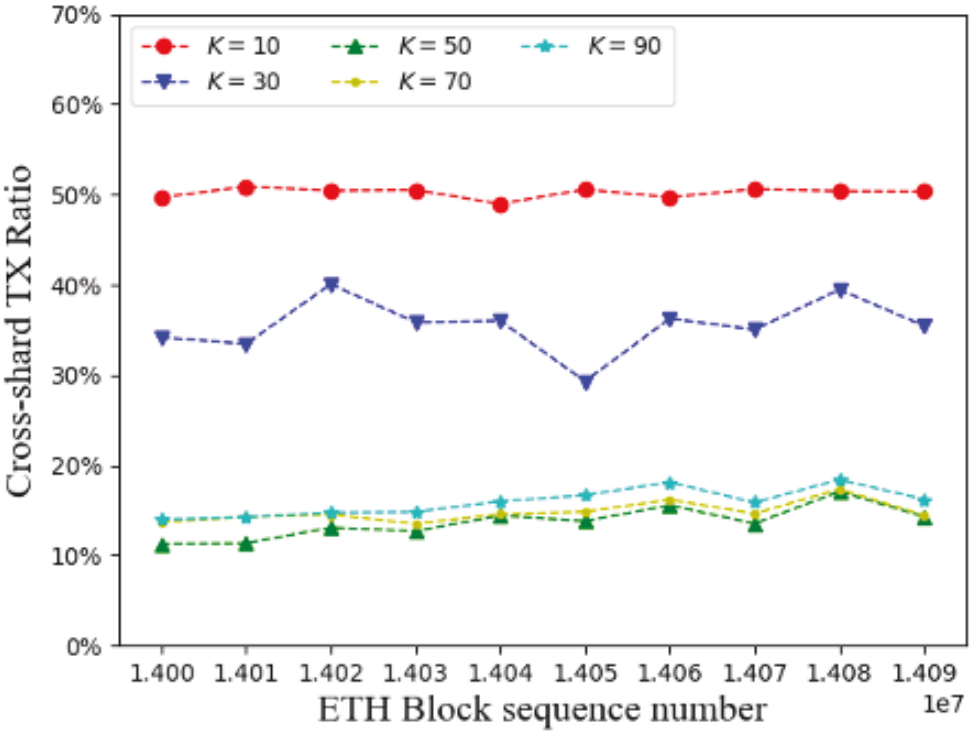


Figure 3.4: Cross-shard TX ratio, the top 10 to 90 most traded addresses. Varying K within$\{10, 30, 50, 70, 90\}$, and fixing $s = 4$.

As shown in Fig. 3.3a, the proposed sharding scheme keeps the number of cross-shard transactions around 10,000. Additionally, there is a transaction peak of around 14.05 million blocks. In the peak block, the random balanced sharding method generates 70,455 cross-shard transactions, while the unbalanced sharding method generates 54,629 cross-shard transactions. Community detection-based sharding generates only 14,972 cross-shard transactions, approximately one-fifth of the number generated by random sharding. With our proposed sharding technique, the peaks of shard lines appear relatively flat because it is based on a community detection algorithm. Fig. 3.3b shows that random balanced and unbalanced sharding leads to a high ratio of cross-shard transactions, i.e., around 80% and 60% respectively. Compared with random sharding with balanced shards, the ratio of cross-shard transactions is reduced from 80% to 20% by implementing the community-detected sharding rather than random sharding.

We explore our proposed scheme by changing the numbers of addresses or numbers of shards. According to Fig. 3.4, we demonstrate changes in the ratio of cross-shard transactions when the number of addresses varies. Tab. 3.2 presents the number of deleted addresses and the number of remaining addresses for each K value.

Table 3.2: Tested ETH Top K Addresses and Remaining Addresses

Tested ETH Address Number	K=**10**	K=**30**	K=**50**	K=**70**	K=**90**
Deleted Address Number	3	9	14	21	31
Remaining Address Number	7	21	36	49	59

We evaluate our proposed community detection-based sharding approach by varying ETH most traded top K addresses from the range $\{10, 30, 50, 70, 90\}$ and fixing shard number $s = 4$. After traversing ten epochs, experiments on the ratio of ϕ_c in Fig. 3.4 reveal an interesting pattern of change. We see that a decrease in the ϕ_c occurs with an increase in K from 10 to 50, and an increase in the ϕ_c occurs with an increase in K from 50 to 90. The ratio of cross-shards is lowest when $K = 50$.

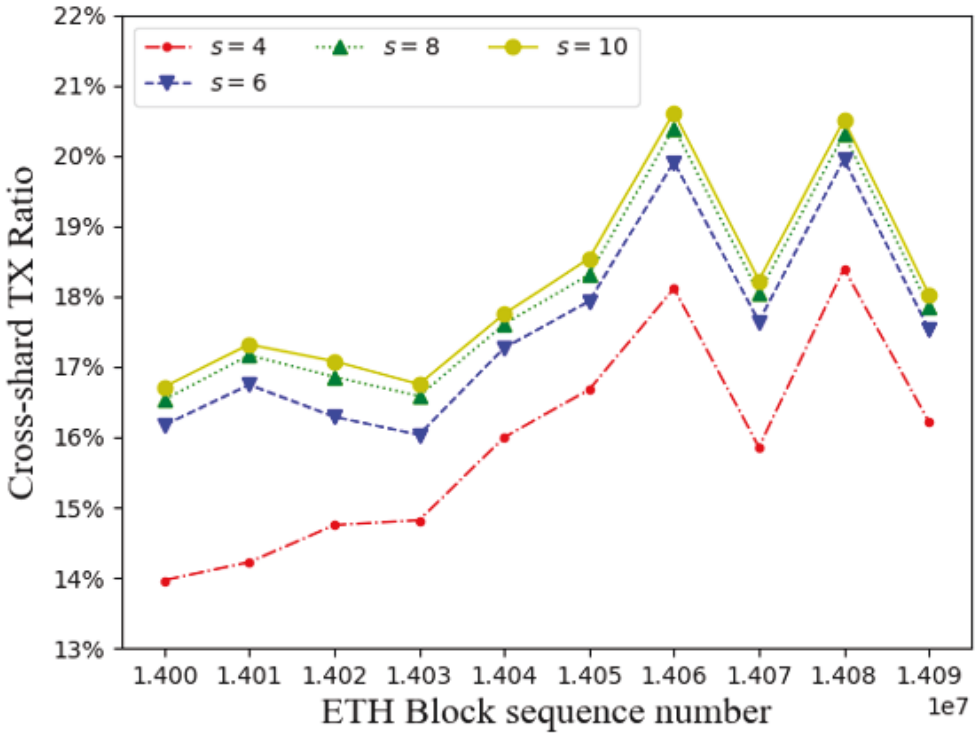


Figure 3.5: Cross-shard TX ratio, while varying shards number within $\{4, 6, 8, 10\}$, and fixing $K = 90$.

In addition to varying the top K, we test sharding performance by changing the shard number s from the range $\{4, 6, 8, 10\}$ and fixed $K = 90$. The observation in Fig. 3.5 indicates that an increase in s leads to an increase in ϕ_c. Although the s increases with ϕ_c, the s and the ϕ_c

do not follow a linear relationship. A significant change in the ratio of ϕ_c occurs when the s is increased from 4 to 6, followed by a continually smaller change as the s increases. Sharding results show that fewer shards result in a lower ratio of cross-shard transactions. Also, cross-shard transactions will reach upper bounds as more shards are added.

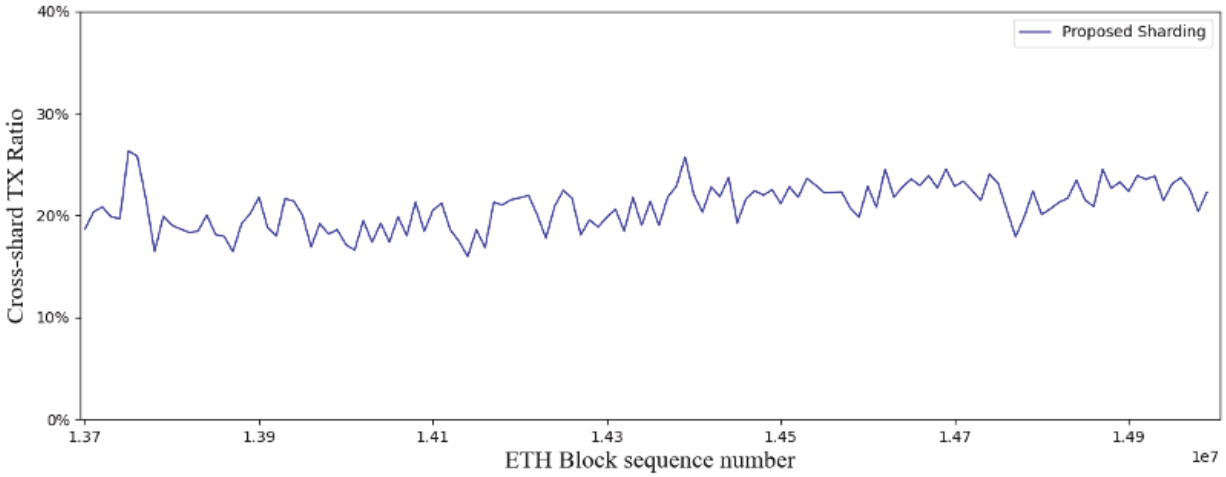


Figure 3.6: Variation of cross-shard TX ratios in 1.3 million blocks.

Fig. 3.6 shows the test results of our proposed sharding method from the block range of 13.7 million to 15.04 million. We display the overall ratio of cross-shard transactions during the tested period. Community detection-based sharding reduces transactions between shards to 20% and stays stable.

3.3.2 Experiments Settings for the Proposed Budget-friendly Sharding Framework

To assess our proposed blockchain sharding framework, we set up an experimental framework on a local system (MacBook Pro, equipped with a 2.5 GHz Quad-Core Intel Core i7 and 16GB RAM). We establish a virtual machine and test the sharding framework in 3.10.44.

Data Overview

The latest 100,000 Ethereum transaction data from the first week of September 2023 was downloaded from the website called Kaggle [123]. The data contains 48,864 sender addresses and 24,860 receiver addresses. According to the adjacency matrix, 33 nodes do not have any transactions with any of the first 50 nodes. The row sum of all the 33 sender addresses in the

matrix is zero. Thus we remove these 33 addresses, and the top 50 most transacted addresses form adjacency. ted addresses form a 59×59 adjacency matrix.

Comparison with random-based sharding

Table 3.3: Tested ETH Top K Addresses and Remaining Addresses

Tested ETH Address Number	K=**10**	K=**20**	K=**30**	K=**40**	K=**50**
Deleted Address Number	7	10	17	24	33
Remaining Address Number	3	10	13	16	17

Tab. 3.3 presents the number of deleted addresses and the number of remaining addresses for each K value. We evaluate our proposed community detection-based sharding approach by varying ETH most traded top K addresses from the range $\{10, 20, 30, 40, 50\}$ and fixing shard number $s = 2$.

To compare with the community detection-based sharding mechanism, we conducted a simulation involving random allocation of addresses. This mirrors the method in Omniledger [64], where nodes are randomly assigned to various shards. There are two ways in which nodes can be allocated to shards in our simulation. The first method ensures each shard has an equal number of nodes, termed as balanced random sharding. The second allows for shards to have varying numbers of nodes, either more or less, and is referred to as unbalanced random sharding.

Table 3.4: Comparison of Total ETH TX Fee, Community Detection VS Random Allocation

Total TX Fee	K=**10**	K=**20**	K=**30**	K=**40**	K=**50**
Community	0.22e10	0.95e10	1.20e10	1.23e10	1.82e10
Balanced Random	0.82e10	2.56e10	2.87e10	2.77e10	2.78e10
Unbalanced Random	0.67e10	2.43e10	2.16e10	2.48e10	3.77e10

Fig. 3.7 and Tab. 3.4 shows the effectiveness of the community detection algorithm proposed in this article in reducing transaction fee. The proposed sharding scheme in this paper consistently

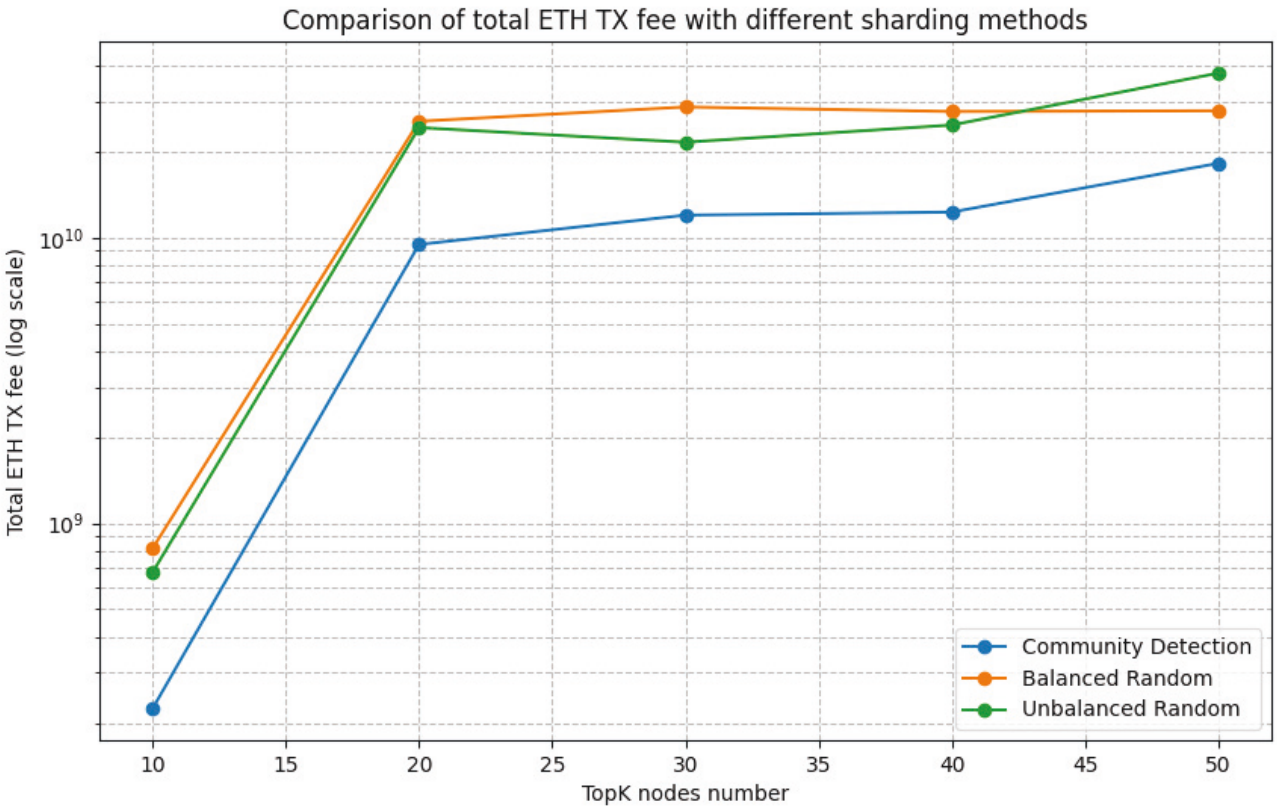


Figure 3.7: Comparison of total ETH TX fee, Community detection VS Random Allocation, $K = \{10, 20, 30, 40, 50\}$, and fixing $s = 2$.

results in lower transaction fees as the system node count rises, compared to the other two random-based sharding techniques. Compared to the balanced random sharding scheme, our method decreases transaction fees by 54.06%. Compared to the unbalanced random sharding scheme, the reduction in transaction fees is 52.91%. The experiment result indicates that the community detection-based sharding method can effectively reduce the system's unnecessary costs on invalid or zero-value transactions.

Fig. 3.8 and Tab. 3.5 shows the effectiveness of the proposed algorithm in reducing the proportion of cross-shard transactions. As shown in the figure, the proportion of cross-shard transactions of the sharding scheme proposed in this article is lower than the other two random sharding methods. Compared with the balanced and unbalanced random sharding schemes, our method reduces the proportion of cross-shard transactions by 45.28% and 46.13%, respectively. Experimental results show that this sharding method can effectively improve system efficiency.

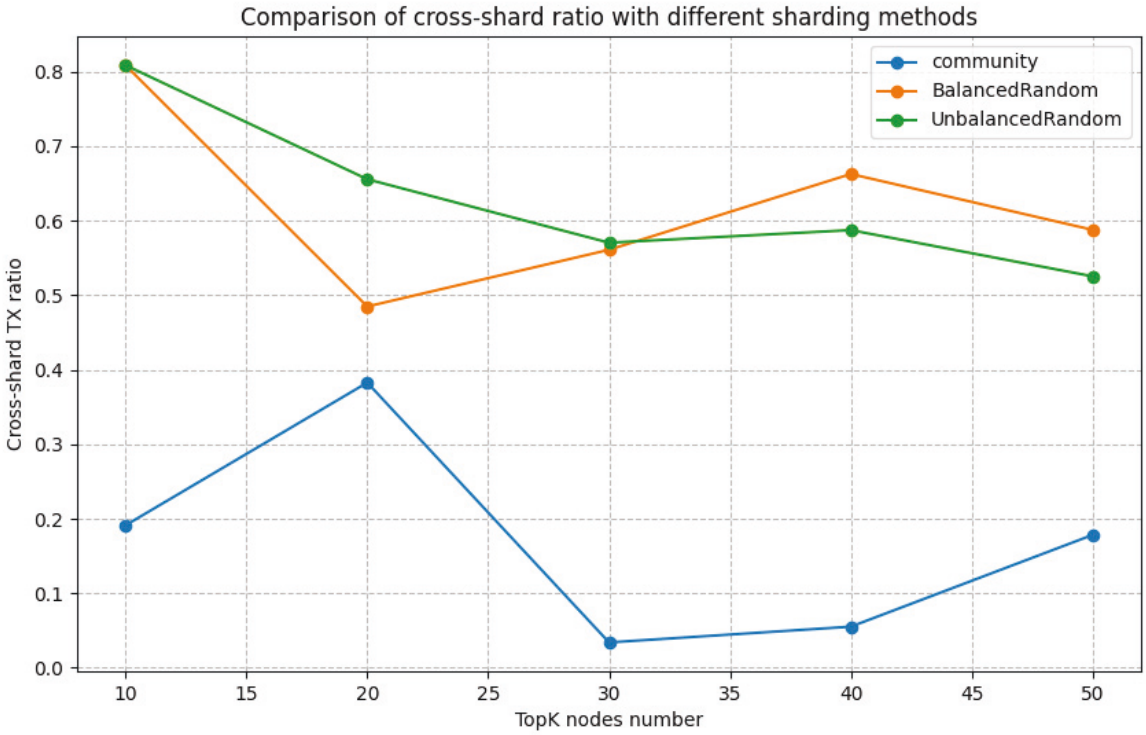


Figure 3.8: Comparison of cross-shard TX ratio, Community detection VS Random Allocation, $K = \{10, 20, 30, 40, 50\}$, and fixing $s = 2$.

Table 3.5: Comparison of Cross-Shard TX Ratio, Community Detection VS Random Allocation

Cross-shard TX ratio	K=**10**	K=**20**	K=**30**	K=**40**	K=**50**
Community	19.12%	38.28%	3.40%	5.52%	17.84%
Balanced Random	80.88%	48.49%	56.22%	66.28%	58.78%
Unbalanced Random	80.88%	65.57%	57.06%	58.75%	52.54%

Comparison the Proposed Framework Performance with Different Shard Number

To analyze the performance of the proposed budget-friendly framework under different shard number and different TopK, we conducted comparative experiments on transaction fee and cross-shard ratio for $K = \{20, 30, 40, 50\}$, and $s = \{2, 3, 4, 5, 6\}$.

Fig. 3.9 and Fig. 3.10 respectively represent the impact of an increasing number of nodes on ETH transaction fees and the cross-shard ratio using the proposed algorithm. These effects are

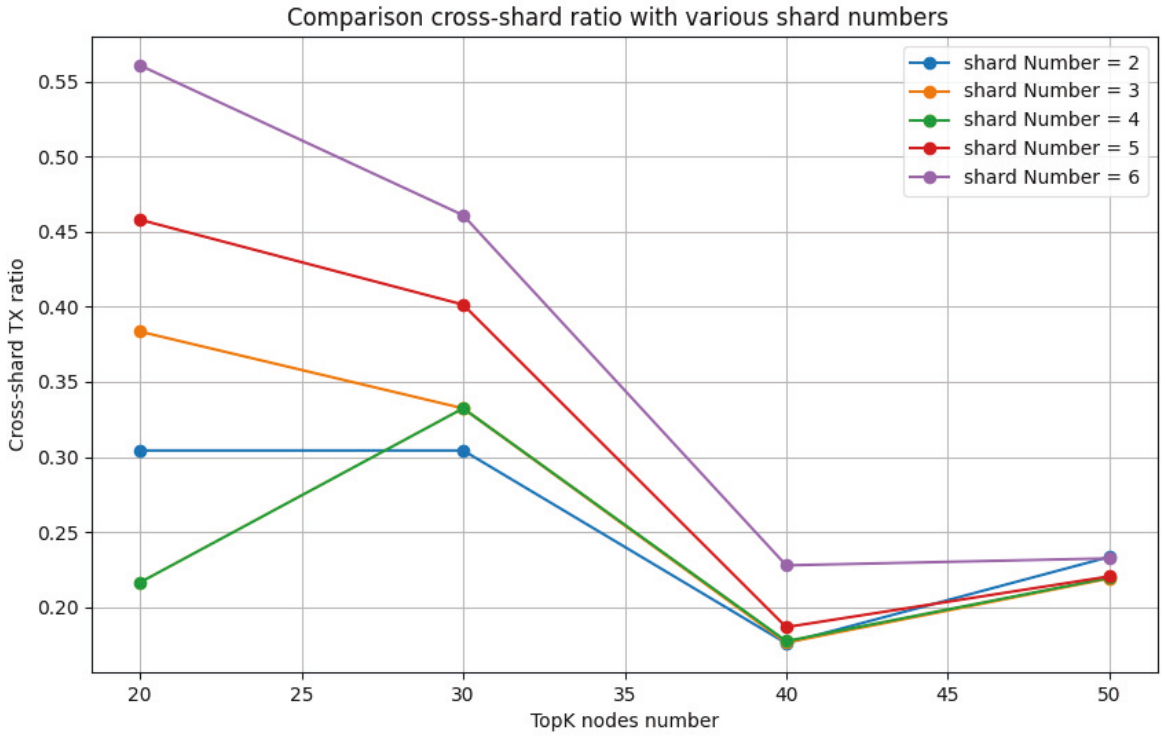


Figure 3.9: Comparison of Cross shard TX Ratio under various shard numbers, $K = \{20, 30, 40, 50\}$, and $s = \{2, 3, 4, 5, 6\}$.

observed under various numbers of shard configurations. These two figures indicate a direct correlation: a higher proportion of cross-shard transactions results in increased transaction fees, detrimentally affecting the user experience.

Table 3.6: Comparison of Cross Shard TX Ratio under Various Shard Numbers

Cross Shard TX Ratio	K**=20**	K**=30**	K**=40**	K**=50**
Shard number = 2	30.43%	30.43%	17.59%	23.34%
Shard number = 3	38.35%	33.23%	17.63%	21.9%
Shard number = 4	21.64%	33.23%	17.73%	21.95%
Shard number = 5	45.79%	40.14%	18.66%	22.04%
Shard number = 6	56.07%	46.1%	22.78%	23.24%

Fig. 3.9 and Tab. 3.6 reveal that the proportion of cross-shard transactions is notably varied

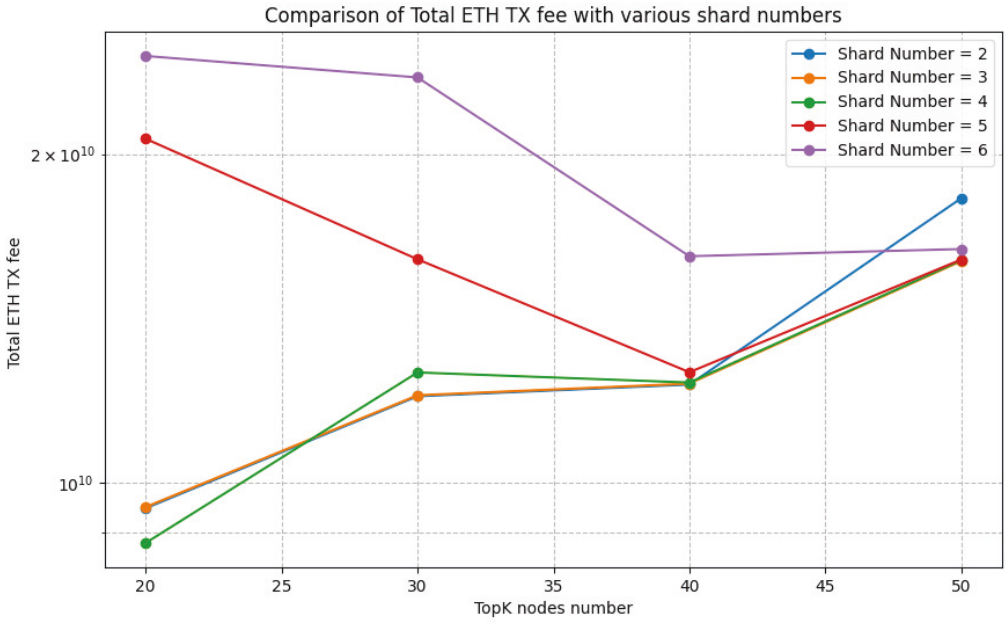


Figure 3.10: Comparison of total TX fee expenditure under various shard numbers, $K = \{20, 30, 40, 50\}$, and $s = \{2, 3, 4, 5, 6\}$.

with many shards but limited nodes. However, as the number of shards and nodes increases, the change in the proportion of cross-shard transactions stabilizes, resulting in minimal differences.

Table 3.7: Comparison of Total TX Fee under Various Shard Numbers

Total TX Fee	K**=20**	K**=30**	K**=40**	K**=50**
Shard number = 2	0.95e10	1.20e10	1.23e10	1.82e10
Shard number = 3	0.95e10	1.20e10	1.23e10	0.16e10
Shard number = 4	0.88e10	1.26e10	1.23e10	1.60e10
Shard number = 5	2.07e10	1.60e10	1.26e10	1.60e10
Shard number = 6	2.50e10	2.35e10	1.61e10	1.64e10

In Fig. 3.10 and Tab. 3.7, it's evident that with the escalation in the number of shards from 2 to 6, there's a corresponding 35.5% rise in the average transaction fee. When there is a high number of shards coupled with fewer nodes, the system displays a marked difference in its transaction fee outlays. This can be attributed mainly to the fact that a surge in the cross-

shard ratio inherently escalates the transaction fees. Yet, as both shard and node numbers increase, the variation in transaction fees becomes less pronounced, suggesting a trend toward fee stabilization.

3.4 Conclusion

In this chapter, we presented a community detection-based sharding framework aimed at reducing cross-shard transactions and optimizing transaction processing efficiency in blockchain systems. First, we converted blockchain data into an undirected and weighted graph and applied a community detection algorithm to identify communities. By assigning nodes to different communities, we achieved sharding of nodes, which effectively reduced the frequency of cross-shard transactions while maintaining system stability. Evaluation on a real-world Ethereum dataset demonstrated that this approach successfully reduced the ratio of cross-shard transactions to 20%. Furthermore, we optimized the sharding framework by considering both the number of transactions and transaction fee expenditures, converting ETH transactions into a weighted undirected graph model. Compared with two separate random-based sharding strategies, the community detection-based sharding approach significantly reduced transaction fees, achieving a 50% reduction. The proposed methods not only enhanced the user experience but also alleviated the transaction processing burden for end-users.

Chapter 4

TbDd Sharding Framework

4.1 Introductions

Sharding boosts blockchain scalability by dividing its nodes into parallel shards, yet it is vulnerable to the 1% attacks where dishonest nodes target a shard to corrupt the entire blockchain. Balancing security with scalability is pivotal for such systems. In this chapter, a Trust-based and DRL-driven (TbDd) framework is presented, crafted to counter collusion attack risks and dynamically adjust node allocation, enhancing throughput while maintaining network security. Deep Reinforcement Learning (DRL) adeptly handles dynamic, complex systems and multi-dimensional optimization. With a comprehensive trust evaluation mechanism, TbDd discerns node types and performs targeted resharding against potential threats based on the trust threshold. Moreover, its architecture can leverage edge servers to efficiently manage shard operations in resource-constrained environments.

4.2 System Model: Mining in Permissioned Sharded Blockchain Networks

The architecture of the proposed sharded blockchain designed to support an IoT network is presented in this section, followed by roles involved in the system, providing a workflow overview and elucidating the system assumptions.

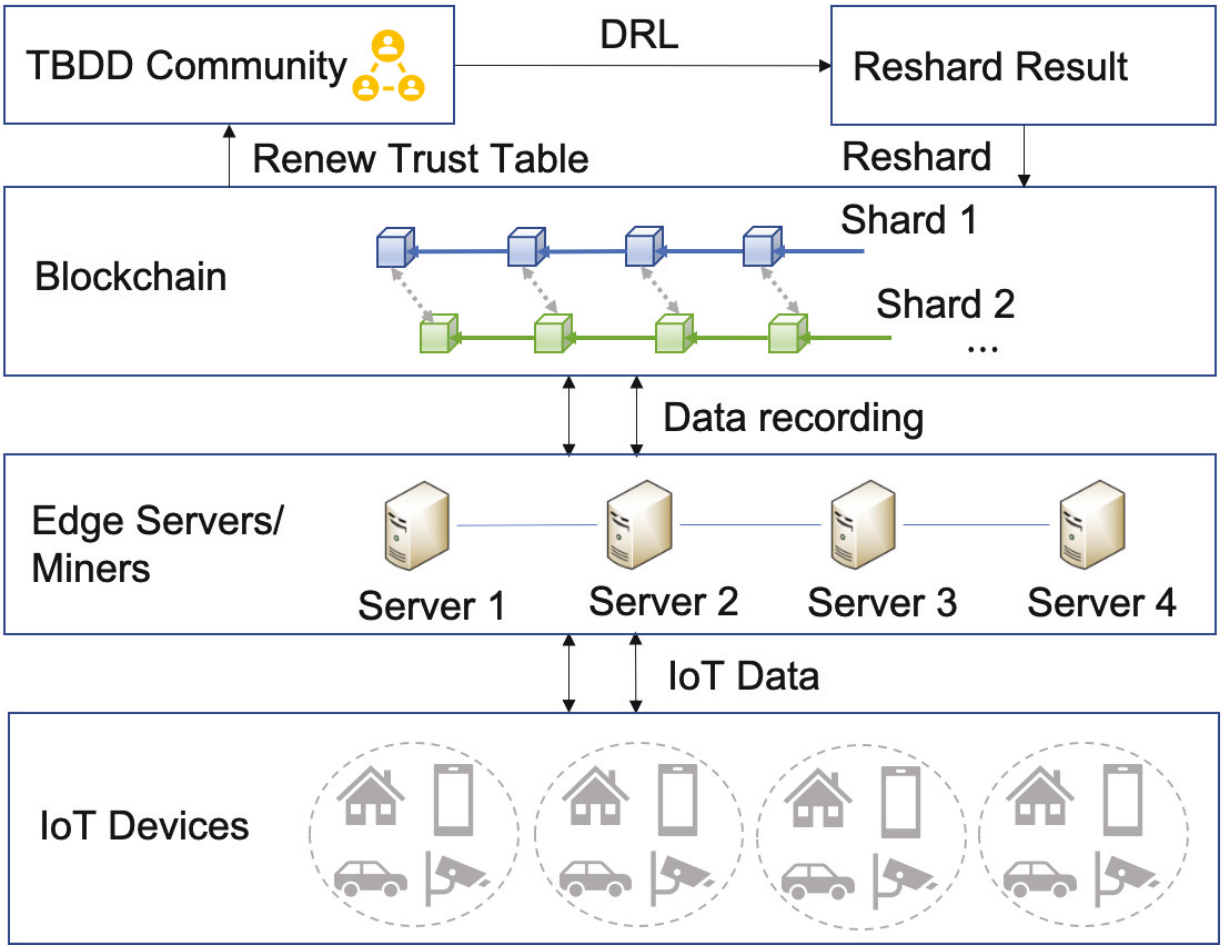


Figure 4.1: System model.

4.2.1 System Overview

System Model. Fig. 4.1 illustrates the proposed sharding framework used in IoT deployments. This framework aims to enhance the scalability and efficiency of blockchain applications within the vast and interconnected realm of IoT devices. Central to the architecture is the sharding mechanism that partitions the broader network into smaller, more manageable shards. Each shard handles a subset of the overall transactions, allowing for simultaneous processing and increasing throughput. Additionally, the design ensures a balanced distribution of nodes across shards to mitigate adaptive collusion attacks.

Architecture. Fig. 4.2 illustrates TbDd's architecture. Within each shard managed by an edge server, any node can propose a block as a leader. Each node maintains a local trust table with trust scores assigned to other network nodes. The introduced TbDd Committee (TC) is composed of members democratically selected by the network's users. Drawing inspiration from Elastico [63], the TC ensures decentralized and reliable oversight. This design prevents

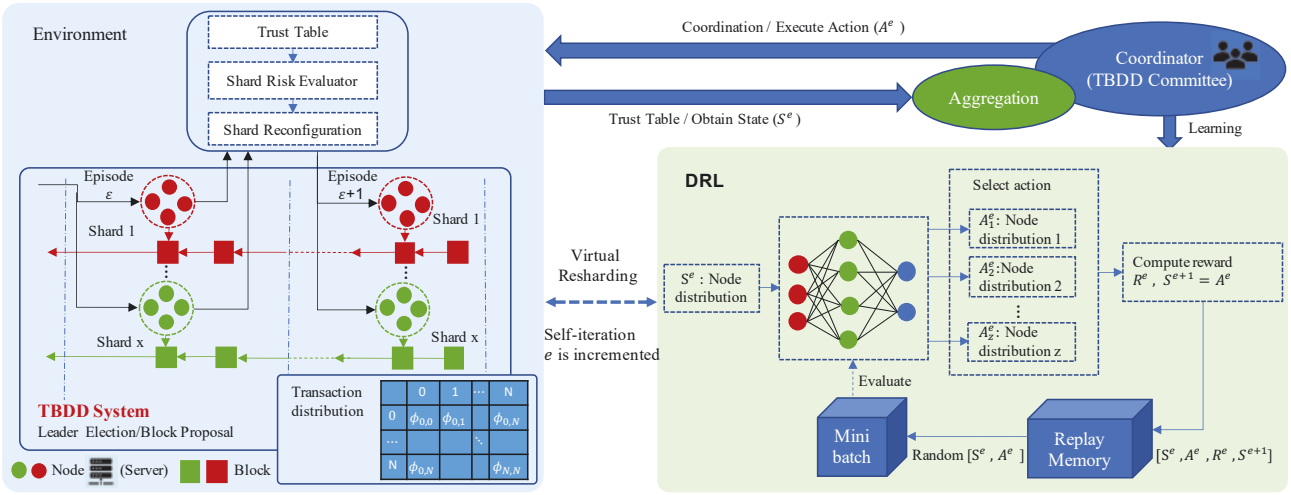


Figure 4.2: This figure illustrates the proposed blockchain sharding system architecture - TBDD. The left section shows the TBDD system with nodes in red and green indicating different shards, where blocks are created and linked red and green arrows within each shard across episodes ε and $\varepsilon + 1$. Black arrows signify the TBDD monitoring and resharding processes. The transaction distribution table represents node transaction volumes, highlighting potential discrepancies between honest and dishonest nodes. On the right, the DRL mechanism evaluates node distributions and selects actions to optimize shard configuration. The coordinator, symbolized by the blue oval, aggregates information and guides the learning process. This cohesive depiction aims to clarify the intricacies of our sharding approach and its operational flow.

single points of failure and minimizes risks from a centralized authority. The TC functions as a decentralized, trusted overseer, managing the node list for the entire network and assigning nodes to different shards. By aggregating nodes' local trust tables, it derives a global trust metric per node, thus ensuring the integrity of trust and node distribution. A key component of TBDD is its resharding mechanism, which routinely reassigns nodes among shards. Such adaptability is crucial, enabling the system to remain scalable and secure, and to handle increasing transaction volumes without sacrificing security.

Roles. In the TBDD framework, participants in the network, referred to as nodes or servers, are categorized into two distinct roles: validators and leaders. Each node in the network has the potential to serve as a validator, participating in the validation of block proposals within a network that is segmented into D shards. The network consists of N nodes, denoted as $\mathbb{N} = \{1, \cdots, N\}$. An episode, represented by ε, is defined as a blockchain period during which

every node has successfully served as a leader at least once. In this role, a node is responsible for proposing blocks to its shard peers for validation. Notably, the leadership role is considered trivial, as the system's focus is not on the consensus algorithm but on the distributed verification process.

4.2.2 Workflow Overview

The proposed TbDd framework follows the workflow in Fig. 4.3.

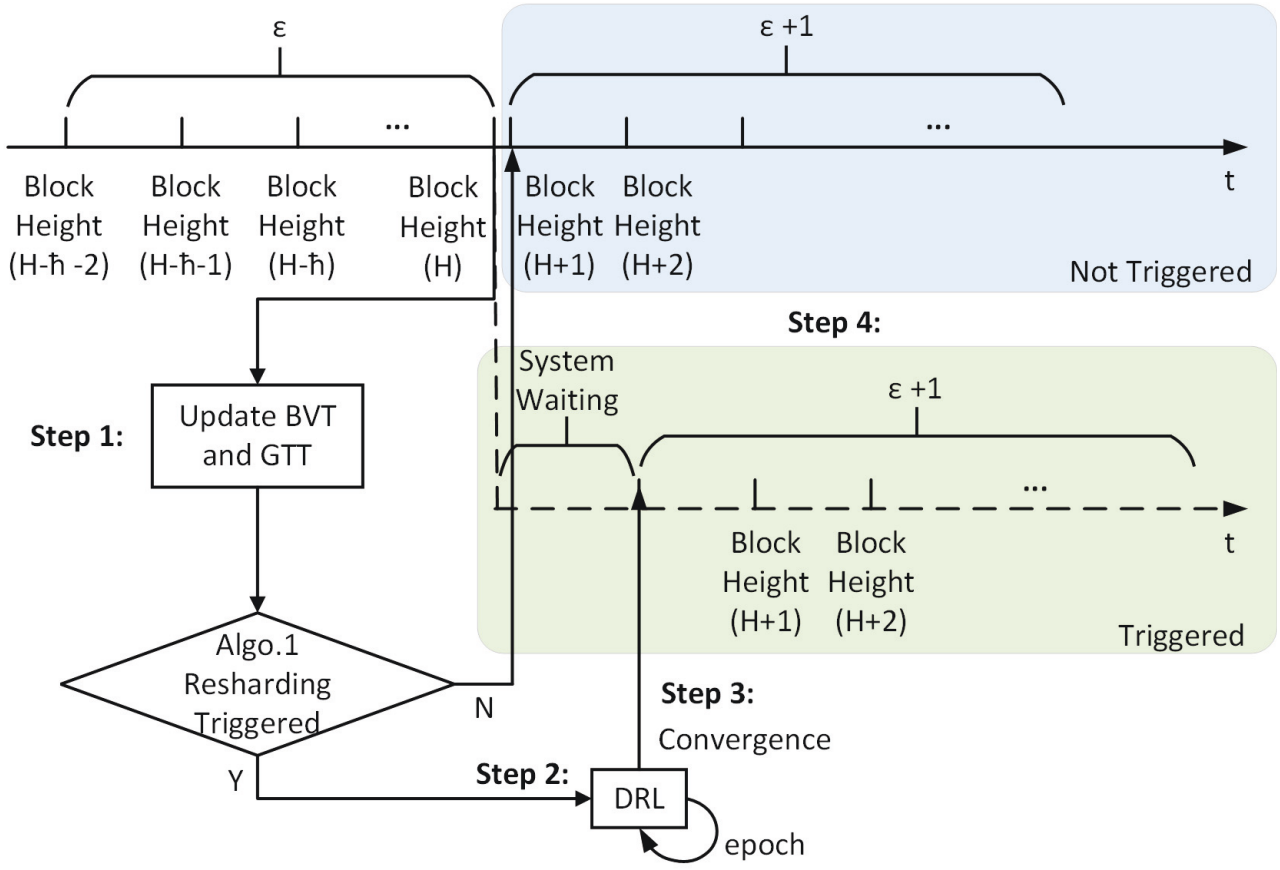


Figure 4.3: The proposed blockchain sharding system flowchart - TbDd, including four steps. ① Trust table updated: update the Block Verification Table (BVT) and Global Trust Table (GTT), checking whether triggering Algo. 3. ② Train the DRL algorithm: use the DRL-based model to virtually resharding through several epochs and output a new node allocation result. ③ Update shard allocation: allocate nodes to the shards according to the DRL-based training result. ④ Monitor network performance: monitor and step into retraining.

Step-1. *Trust table updated.* The first step in the workflow is updating the trust table. The trust table shows the global trust scores for individual users. Then, the system checks if

conditions for triggering Algo. 3 are met.

Step-2. *Train the DRL algorithm.* This step trains the DRL algorithm using the collected network data, which includes running simulations to optimize shard allocation policies based on real-time network conditions. After initiating the DRL training process, the system enters a waiting state and dedicates its computational resources to the training procedure. Note that the TC conducts "virtual resharding" trials to evaluate outcomes and rewards for different actions; however, these trials occur solely within the TC, and the final sharding action is not implemented until it has fully converged.

Step-3. *Update shard allocation.* TC updates shard allocation policies in real time once the DRL model completes training and allocates nodes to shards based on the result.

Step-4. *Monitor network performance.* As shard allocation policies are updated, monitoring network performance is important, which includes tracking network metrics such as transaction distribution and volume, node location, and colluding risk.

After ***Step-4***, the TC moves on to the retraining phase of the DRL model, incorporating the latest information from the trust table obtained in ***Step-1***. Subsequently, the TC updates the shard allocation strategy in ***Step-3*** based on the new insights gained from the retrained DRL algorithm. This cyclical process follows a similar set of steps, beginning from ***Step-1*** and progressing through ***Step-4***.

4.2.3 System assumptions

There are several assumptions in this paper. These assumptions are considered from multiple perspectives, such as attack and system environment.

Attack assumption

The collusion behavior of nodes refers to the situation where multiple nodes work together to manipulate the network and gain some unfair advantage. This dishonest behavior can overload the network, leading to delays and service disruptions. An attack model focusing on collusion attacks in blockchain sharding is designed in this paper, which considers two types of participants: **dishonest nodes** and **honest nodes**.

Dishonest nodes. Dishonest nodes aim to be allocated in the same shard and then inten-

tionally propose invalid results to honest nodes' blocks. This consequence can be achieved by sending many invalid transactions across dishonest nodes, regardless of whether they are cross-shard or intra-shard transactions. Suppose a dishonest node finds no other teammates in the same shard. In that case, it may hide by pretending to be honest and following honest nodes' behavior, avoiding suspicion. Dishonest nodes can also utilize the global trust table's information to enhance their trust by imitating the conduct of honest nodes, selectively endorsing or opposing blocks according to proposers' trust scores.

In (4.1), the trust score $\mathcal{G}_i^{\varepsilon}$ is normalized of the i-th node to a range of $[0, 1]$. Let $\mathcal{G}_{min}^{\varepsilon}$ and $\mathcal{G}_{max}^{\varepsilon}$ be the minimum and maximum possible raw trust scores. The current episode's normalized value $\mathbf{G}_i$ is calculated by the previous episode's global trust score.

$$\mathbf{G}_i = \frac{\mathcal{G}_i^{\varepsilon-1} - \mathcal{G}_{min}^{\varepsilon-1}}{\mathcal{G}_{max}^{\varepsilon-1} - \mathcal{G}_{min}^{\varepsilon-1}}. \tag{4.1}$$

The probabilities related to voting behavior are utilized, in which the probability of a node voting for a block is denoted as P_{vote}, while the probability of voting against a block is represented by $P_{\text{not\_vote}}$. Furthermore, the probabilities of voting behavior between dishonest and honest nodes are distinguished. The probability of dishonest nodes engaging in honest voting and dishonest voting is respectively denoted as $P_{\text{vote}}^{\text{dishonest}}$ and $P_{\text{not\_vote}}^{\text{dishonest}}$. Let F be the probability of a node's vote failing to reach others due to network issues, with $F \in [0, 1]$. Assume A is the proportion of dishonest nodes in the shard, with $A \in [0, 1]$. A strategy threshold τ can be defined, with $\tau \in [0, 1]$, determining when a dishonest node would try to hide by pretending to be honest. If $A < \tau$, the dishonest nodes pretend to be honest and follow honest nodes' behavior; otherwise, they follow the collusion strategy and favor their conspirators. A block verification result, denoted by U, takes a $U = 1$ if it matches the local version and $U = 0$ otherwise. A weighting factor w_G based on the normalized trust score $\mathbf{G}$ and a weighting factor w_U can be defined based on the block verification result in U. The probability distribution for dishonest nodes (4.2) and (4.3) can be defined as follows:

$$P_{\text{vote}}^{\text{dishonest}} = \begin{cases} (1-F) \cdot w_G \cdot \mathbf{G}_i \cdot w_U \cdot U, & \text{if } A < \tau \\ (1-F) \cdot \textbf{CollusionStrategy } (\kappa), & \text{otherwise} \end{cases} \tag{4.2}$$

$$P_{\text{not\_vote}}^{\text{dishonest}} = 1 - P_{\text{vote}}^{\text{dishonest}}, \tag{4.3}$$

where **CollusionStrategy** (κ) signifies a strategy in which attackers consistently vote in favor of their teammates while voting against honest leaders with a probability denoted by $\kappa \in [0, 1]$.

The collusion strategy (4.4) is to give valid results to all dishonest partners, attack honest nodes according to a particular proportion, and give them non-valid.

$$\textbf{CollusionStrategy}\ (\kappa) = \begin{cases} 1, & \text{dishonest nodes} \\ 1-\kappa, & \text{honest nodes} \end{cases} \tag{4.4}$$

Honest nodes. Honest nodes rely on the global trust table, providing an overview of high-risk or low-risk users without explicitly identifying dishonest nodes. During the intra-consensus phase, honest nodes rely on block verification, voting for a block only if it matches their local version. A composite probability distribution considering the trust-based voting distribution and the block verification distribution is proposed in this paper by weighting the probability of voting for a block based on the proposer's trust score and adjusting the weight according to the block verification result. This composite distribution allows honest nodes to make more informed decisions when voting for or against proposed blocks, considering both the proposer's trust score and the consistency of the proposed block with their local version. The combined probability distribution for honest nodes (4.5) and (4.6) can be calculated as follows:

$$P_{\text{vote}}^{\text{honest}} = (1-F) \cdot w_G \cdot \mathbf{G}_i \cdot w_U \cdot U, \tag{4.5}$$

$$P_{\text{not\_vote}}^{\text{honest}} = 1 - P_{\text{vote}}^{\text{honest}}. \tag{4.6}$$

The block verification table can be obtained by simulating honest and dishonest nodes' behavior using these attack models. The effects of collusion attacks on the overall system's security and performance in blockchain sharding systems can be analyzed.

Along with honest and dishonest nodes, the concept of high-risk and low-risk nodes is introduced in the proposed system. It is important to clarify that high-risk and low-risk nodes differ from honest and dishonest nodes. The node evaluation principle classifies nodes as high-risk when their trust scores fall below a specific threshold, while nodes with trust scores above this threshold are categorized as low-risk nodes. These classifications do not necessarily mean high-risk or low-risk nodes are inherently honest or dishonest. Instead, they indicate the level of trustworthiness based on the evaluation criteria. By considering the presence of high-risk and low-risk nodes in the network, The system can better identify and respond to potential collusion attacks, improving the security and integrity of the blockchain sharding framework.

Environment assumption

The system operates within permissioned blockchain systems, restricting network access to specific nodes. Despite this limitation, the system remains decentralized as it is maintained by the TC, a committee fairly elected by all peers within the network. For the training process to be considered trustworthy, it is important that the TC is reliable and has a transparent record when training the DRL model. Furthermore, Assumptions involve at least a certain number of members per shard, and the claim that the count of dishonest nodes in each shard does not surpass a certain number of the overall node count is based on the Byzantine Fault Tolerance (BFT) principle [124]. In case of node failure or a dishonest node attack, the system continues functioning with the remaining nodes.

4.3 TBDD: A Trust-Driven and DRL-based Approach to Optimize Throughput and Security

TbDd is proposed based on the results and analysis in Section 4.2, which is composed of the Block Verification Table (BVT), Local Trust Table (LTT), and Global Trust Table (GTT), Shard Risk evaluator and Shard reconfiguration (see Fig. 4.4 and Tab. 4.1).

4.3.1 Trust Scheme

Block Verification Table (BVT)

The BVT aims to record the validation results for each shard. The verification table for the x-th shard in the ε-th episode is denoted as VT_x^{ε}. The size of VT_x^{ε} is determined by the number of nodes in the shard, with dimensions of $N_x \times N_x$, where N_x represents the number of nodes in the x-th shard. The verification result table can be visualized as a two-dimensional matrix where each cell corresponds to the set of verification results generated by the node that proposed a block. The size of each cell in the matrix corresponds to the number of times the leader produced blocks. During the trust table update episode, assume that there are sufficient votes to guarantee that each node within the shard will be elected as a leader at least once, resulting in block production. In this assumption, the leader can expect to obtain a minimum of one ballot from himself. Furthermore, the consensus process complies with the weak synchrony assumption [51], which indicates a finite upper limit on message delays.

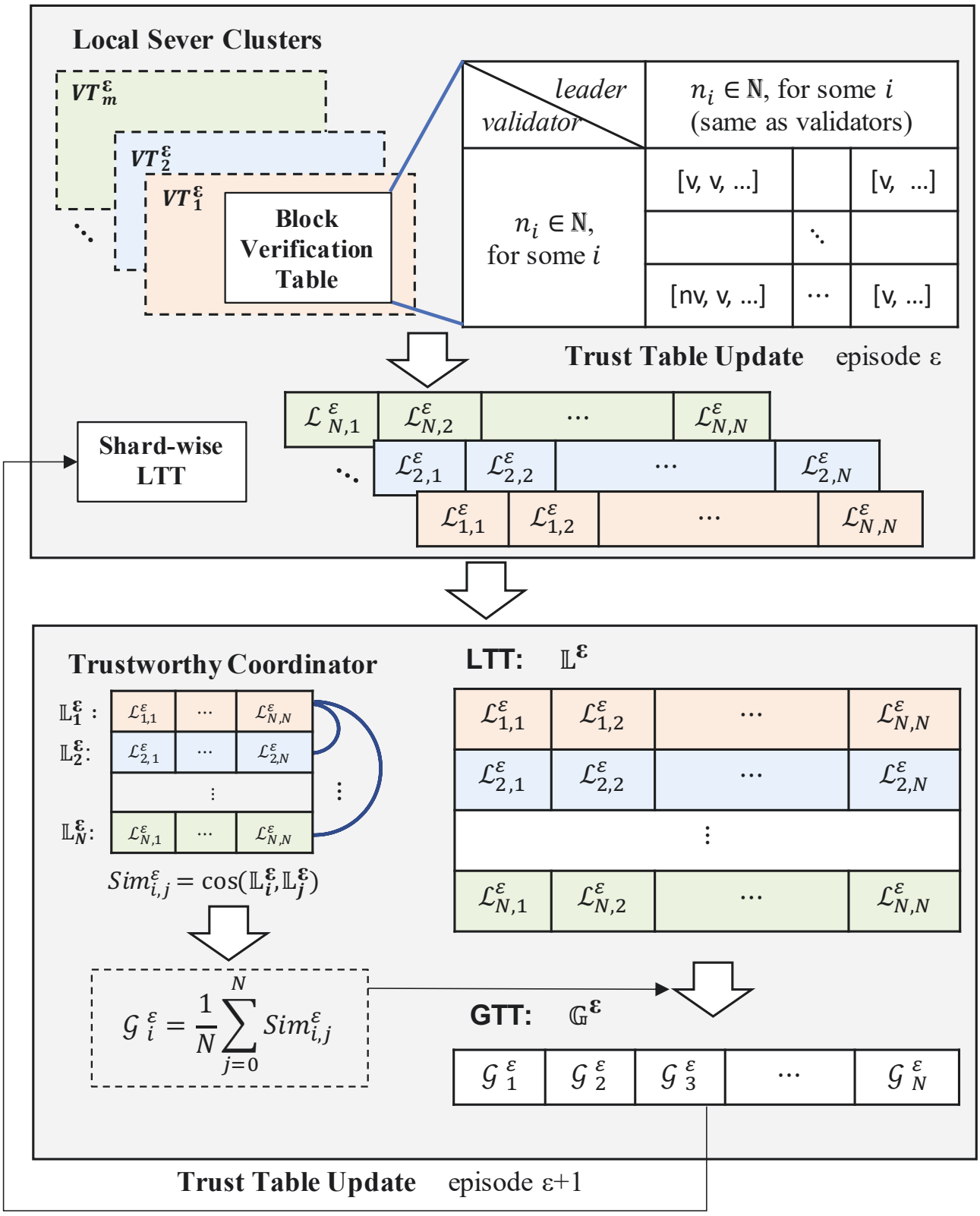


Figure 4.4: The flow diagram of the proposed computing trust score system comprises BVT, LTT, and GTT.

Table 4.1: Notation and Definition Used in The Trust Table

Notation	Description
N	The number of nodes in the network
N_x^{ε}	The number of nodes in the x-th shard in episode ε
$\mathbb{N}$	The set of all nodes in the network
$n_i \in \mathbb{N}$	The i-th node in the network
ι_j^{ε}	The number of times when the j-th node is elected as the leader in episode ε
D	The number of shards in the network
ε	The trust table update in episode ε
e	The e-th epoch during the DRL iteration
$\mathcal{F}_{i,j}^{\varepsilon}$	The indirected feedback of j-th leader from the i-th node in episode ε
$\hat{\mathcal{F}}_{i,j}^{\varepsilon}$	The directed feedback from the j-th node to the i-th leader in episode ε
$\mathcal{L}_{i,j}^{\varepsilon}$	The local trust from the i-th node to the j-th node in episode ε
$\mathbb{L}_i^{\varepsilon}$	The local trust table for the i-th node in episode ε
$\mathbb{L}^{\varepsilon}$	The concatenated local trust tables across all nodes in episode ε
$\mathcal{G}_i^{\varepsilon}$	The global trust for the i-th node in episode ε
$\mathbb{G}^{\varepsilon}$	The global trust table in episode ε
$\mathbf{G}_i$	The normalized trust score of the i-th node
θ_x^{ε}	The average global trust of x-th shard in episode ε
$\bar{\theta}^{\varepsilon}$	The average value of all θ_x^{ε}
h_x	The list of high-risk nodes in the x-th shard
f_{total}	The entire network's fault tolerance threshold for dishonest nodes
f_{intra}	The shard's fault tolerance threshold for dishonest nodes
ϕ_{in}	The number of intra-shard transactions (ISTs)
ϕ_{cr}	The number of cross-shard transactions (CSTs)

Trust Table

In TBDD, two distinct trust tables are utilized: LTT and GTT. These tables are dynamic and regularly refreshed to capture the latest data on each node's performance and behaviors within the network. By constantly monitoring and evaluating the LTT and GTT, the TBDD system can make strategic and informed shard allocation decisions, ensuring a reliable and secure network operation.

The local trust table is denoted as $\mathbb{L}^{\varepsilon}$. Each row of this table is specifically assigned to a node within the shard, and these nodes are represented as $1 \times N$ entries within the table, which is term as the local trust table of the i-th node at the ε epoch, denoted as $\mathbb{L}_i^{\varepsilon}$. When these individual local trust tables are concatenated, it is represented as $\mathbb{L}^{\varepsilon}$, forming a comprehensive $N \times N$ table where $i, j \leq N$. Within this table, each element represents the trust score sent from the i-th node to the j-th node, denoted as $\mathcal{L}_{i,j}^{\varepsilon}$. The computation of each element in the LTT of i-th node is shown in (4.7):

$$\mathcal{L}_{i,j}^{\varepsilon} = \begin{cases} \alpha \mathcal{F}_{i,j}^{\varepsilon} + \beta \hat{\mathcal{F}}_{i,j}^{\varepsilon} + \mu \mathcal{G}_i^{\varepsilon-1}, & \text{if } n_i, n_j \text{ in the same shard} \\ \mathcal{G}_i^{\varepsilon-1}, & \text{if } n_i, n_j \text{ in different shards} \end{cases} \tag{4.7}$$

when calculating the trust score in the same shard, three feedbacks are included: Indirected feedback $\mathcal{F}_{i,j}^{\varepsilon}$, Directed feedback $\hat{\mathcal{F}}_{i,j}^{\varepsilon}$ and Global trust score from the last episode $\mathcal{G}_i^{\varepsilon-1}$. Each component has a distinct proportion represented by α, β, μ. These proportions sum up to 1 (i.e., $\alpha + \beta + \mu = 1$). Only the global trust score from the previous episode is considered for calculating the trust scores of nodes in different shards.

Indirected feedback of each leader. The proportion of verification that the j-th node passes when he is the leader at ε episode is shown in (4.8):

$$V_j^{\varepsilon} = \frac{\sum_{i=1}^{N_x^{\varepsilon}} v_{i,j}^{\varepsilon}}{\iota_j^{\varepsilon} N_x^{\varepsilon}}, \tag{4.8}$$

where ι_j^{ε} represents the times of the j-th node being elected as the leader in the ε-th episode, and $v_{i,j}^{\varepsilon}$ signifies the total number of valid votes v cast by the i-th node for the j-th leader. Then, the indirected feedback $\mathcal{F}_{i,j}^{\varepsilon}$ is calculated as follows (4.9):

$$\mathcal{F}_{i,j}^{\varepsilon} = \gamma V_j^{\varepsilon} + \frac{\gamma^2}{N_x^{\varepsilon} - 2} \sum_{p \in \{i,j\}^{\mathrm{C}}} \delta_{p,j}^{\varepsilon} \left(V_p^{\varepsilon}\right)^{\iota_j^{\varepsilon} - \delta_{p,j}^{\varepsilon} + 1}, \tag{4.9}$$

where the node, denoted as n_p, provides indirect feedback and participates in the voting process for the current leader l_j. The n_p node does not include block proposer l_j and the voter n_i itself.

$\delta^{\varepsilon}_{p,j}$ refers to the ratio of non-empty votes (both valid and non-valid votes) cast from the p-th node for the j-th leader. γ is a discount rate similar to that used in reinforcement learning.

Directed feedback of each leader. The direct feedback of trust score is calculated as shown in (4.10):

$$\hat{\mathcal{F}}^{\varepsilon}_{i,j} = \frac{v^{\varepsilon}_{j,i}}{\iota^{\varepsilon}_{i}}, \tag{4.10}$$

where $v^{\varepsilon}_{j,i}$ denotes the count of valid votes v cast by the j-th node for the i-th node when n_i is leader. ι^{ε}_{i} indicates that the number of times of the i-th node being elected as the leader during the ε-th episode.

Global trust of each leader from the history. The historical trust of each leader $\mathcal{G}^{\varepsilon-1}_{i}$ is inherited from the last episode.

The global trust table is denoted as $\mathbb{G}^{\varepsilon}$. Inspired by federated learning principles, the coordinator enhances credibility by updating the LTT according to the GTT's outcome after every iteration.

Cosine similarity calculation. Comparisons of cosine similarity among LTT rows reflect deviations in node-scoring behavior (4.11):

$$Sim^{\varepsilon}_{i,j} = cos(\mathbb{L}^{\varepsilon}_{i}, \mathbb{L}^{\varepsilon}_{j}), \quad i, j < N. \tag{4.11}$$

The global trust score for each node is determined by computing the mean of the cosine similarity between the node's scoring behavior in the LTT and the entire LTT (4.12):

$$\mathcal{G}^{\varepsilon}_{i} = \frac{1}{N}\sum_{j=1}^{N} Sim^{\varepsilon}_{i,j}, \tag{4.12}$$

$\mathcal{G}^{\varepsilon}$ is a $1 \times N$ vector capturing the global trust, where element $\mathcal{G}^{\varepsilon}_{i}$ is the global trust of the i-th node in the current shard.

4.3.2 Resharding Trigger: The Shard Risk Evaluator

The resharding process is triggered when certain conditions are met, leading to the trigger phase (the green block in Fig. 4.3), as shown in Algo. 3. A global trust threshold ρ_t is defined to differentiate between low-risk and high-risk nodes. Nodes are high-risk and possibly dishonest if their $\mathcal{G}^{\varepsilon}$ are lower than ρ_t. Nodes are *more likely* to be honest if their global trust exceeds ρ_t. The fault tolerance threshold f_{intra} is defined within each shard. If the number of dishonest

Algorithm 3: Shard risk evaluator

Input: x: The x-th shard;

N_x^{ε}: Number of nodes in the x-th shard at ε episode;

$\mathcal{G}_i^{\varepsilon}$: Global trust of the i-th node;

f_{intra}: The faulty tolerance of dishonest nodes in any shard;

φ_{cr}: The CST ratio since last episode;

ρ_t: The threshold of differentiating a low-risk or high-risk node in terms of trust score;

ρ_{cr} : The threshold in terms of CST;

Output:

{*Trigger*, *Not trigger*}

```
ρ_cr := 0
h_x := [] for each shard x
for x ∈ [1, D) do
    for n_i ∈ x and i ∈ [1, N) do
        if G_i^ε < ρ_t then
            h_x ← n_i
    if |h_x| / N_x > f_intra then
        return Trigger
if φ_cr > ρ_cr then
    return Trigger
return Not Trigger
```

nodes exceeds the threshold f_{intra} in the shard, the shard is labeled as corrupted and triggers resharding. Furthermore, resharding is also triggered when the ratio of CST surpasses the setting threshold ρ_{cr}. The ratio of the CST is calculated as follows:

$$\varphi_{cr} = \frac{\phi_{cr}}{\phi_{cr} + \phi_{in}}, \tag{4.13}$$

where ϕ_{cr} represents the CST count, as given by:

$$\phi_{cr} = \frac{1}{2}\left(\sum_{i=1}^{N}\sum_{j=1}^{N}\phi_{i,j} - \sum_{x=1}^{D}\sum_{i=1}^{N_x^{\varepsilon}}\sum_{k=1}^{N_x^{\varepsilon}}\phi_{i,k}\right), \tag{4.14}$$

where $\phi_{i,j}$ represents the transaction count between the i-th and j-th nodes in the network. $\phi_{i,k}$ represents the transaction count between the i-th node and the k-th node in the shard, which equals the sum of the Intra-Shard Transaction (IST). The ϕ_{cr} is obtained by subtracting the IST count from the total transactions count among network nodes.

4.3.3 DRL Framework

Optimizations, Rewards, and DRL

The DRL-based model enhances blockchain sharding systems by dynamically allocating nodes depending on network conditions and automating complex decision-making procedures. The reward function optimizes node allocation while maintaining security constraints through DRL adaptation. In TbDd, the agent aims to maximize earnings by calculating the objective function that is composed of 6 reward components, where $E_a, E_b, \lambda_a, \lambda_a$, and λ_c are all constant.

Shard load balance (ξ). Aiming to distribute the number of nodes across each shard evenly, (4.15) is defined. If there is a significant disparity in the node count per shard, resharding is augmented to counteract potential 1% attacks. [125]

$$\xi = \begin{cases} E_a, & \text{shard load balance} \\ -E_a, & \text{shard load unbalance} \end{cases} \tag{4.15}$$

Corrupted shards portion (ϱ). As shown in (4.16), the agent receives a reward if no shard is occupied. Otherwise, the agent receives a penalty.

$$\varrho = \begin{cases} E_b, & \text{no shard is corrupted} \\ -E_b, & \text{at least one shard is corrupted} \end{cases} \tag{4.16}$$

CST ratio (η). As shown in (4.17), if the CST ratio is less than the specified threshold ρ_{cr}, the agent gets rewarded; otherwise, it receives a penalty.

$$\eta = \lambda_a \left(\rho_{cr} - \varphi_{cr}\right) \lambda_b^{\lambda_c |\varphi_{cr} - \rho_{cr}|}, \tag{4.17}$$

Nodes shifting ratio (ψ). As shown in (4.18), if a node switches the shard, its action is noted as 1; otherwise, it is 0. The overall count of shifted nodes corresponds with the penalty score. The more nodes relocate, the more computational resources are consumed by shard synchronization, resulting in a higher level of punishment.

$$\psi = \frac{1}{N} \sum_{j=1}^{N} \upsilon_j, \quad \upsilon_j = \begin{cases} 1, & \text{nodes moving} \\ 0, & \text{nodes staying} \end{cases} \tag{4.18}$$

Intra-shard's trust variance (Ω_{in}). As shown in (4.19), trust variance represents the trust distribution within each shard. A larger trust variance of a shard indicates a better distinction

between trustworthy and untrustworthy participants. Thus, a larger intra-shard trust variance collected by the agent serves as a reward to indicate a clearer differentiation between honest and dishonest nodes.

$$\Omega_{in} = \frac{1}{D}\sum_{x=1}^{D}\frac{1}{N_x^{\varepsilon}}\sum_{k=1}^{N_x^{\varepsilon}}\left|\mathcal{G}_k^{\varepsilon} - \theta_x^{\varepsilon}\right|^2, \tag{4.19}$$

where $\mathcal{G}_k^{\varepsilon}$ is the global trust value of nodes in the x-th shard and θ_x^{ε} is the average of global trust in the x-th shard.

Cross-shard's trust variance (Ω_{cr}). As shown in (4.20), it represents the deviation of trust value among different shards. A minor cross-shard trust variance indicates a more uniform distribution of dishonest nodes.

$$\Omega_{cr} = \frac{1}{D}\sum_{x=1}^{D}\left|\theta_x^{\varepsilon} - \bar{\theta}^{\varepsilon}\right|^2, \tag{4.20}$$

where $\bar{\theta}^{\varepsilon}$ is the average value of all $\theta_x^{\varepsilon}, x \in D$. Thus, the objective function can be defined as:

$$R = \xi + \varrho + \eta - \psi + \Omega_{in} - \Omega_{cr}, \tag{4.21}$$

$$\begin{aligned} &\text{Let } \mathbf{R} = [\xi, \varrho, \eta, \psi, \Omega_{in}, \Omega_{cr}], \\ &\text{objective: } \max_{\mathbf{R}} \sum^{\varepsilon_{max}} R(\mathbf{R}), \\ &\text{s.t.} \quad |h_x| < N_x f_{intra}, \\ &\qquad N_x \geq N_{min}, \end{aligned} \tag{4.22}$$

where ξ is the balance reward for shard load; ϱ is the reward for the number of corrupted shards; η signifies the reward for low-level CST; ψ indicates the reward for the number of shifted nodes; Ω_{in} stands for the reward of intra-shard trust variance; and Ω_{cr} represents the reward of cross-shard trust variance. $\mathbf{R}$ is defined as the set of all rewards. Restrictions of the objective function R ensure the dishonest node count stays below the shard's fault tolerance. N_{min} denotes the minimum node requirement for each shard. Each shard mandates a minimum of four nodes in the setting, i.e., $N_{min} = 4$.

DRL-based Sharding Optimization Model

The DRL model is used to assist in the blockchain reconfiguration process. The agent of the DRL model is acted by the TC, a committee elected by all peers in the network. The agent obtains the state from the node allocation. Then, the agent gains the reward by virtually

resharding, deciding the optimal allocation strategy and executing the action for the new node allocation.

Agent: The agent is perceived as the TC, which consists of several nodes. These nodes implement the PBFT protocol to achieve consensus, representing the collective action of the TC. The agent executes a learning process and decides shard allocation based on real-time network conditions.

Environment: As illustrated in Fig. 4.2, the environment is viewed as a black box that executes the **Action** of the agents and obtains the **State**. In this paper, the sharding reconfiguration process in the blockchain sharding network is the environment.

The agent is considered to obtain a state $s \in \mathcal{S}$ based on the node allocation at the current episode ε in this paper. s denotes the distribution of all nodes across various shards in the current episode, while m precisely identifies the specific node present within a particular shard.

$$s = [m_{x,1}, \cdots, m_{x,i}, \cdots, m_{x,N}], x \in D, i \in N, \tag{4.23}$$

where $m_{x,i}$ is the x-shard to which the i-th node belongs.

Episode (ε). An episode in the blockchain context is characterized as a period during which each node has successfully assumed the role of a leader at least once. The initiation of a new episode is signaled by the updating of the trust table, indicating that the duration of an episode can be flexibly modified to align with diverse requirements.

Actions ($\mathcal{A}$). The agent executes an action $a \in \mathcal{A}$ based on its decision produced by its local DRL-based learning during the current episode ε. The valid action space for the agent is formatted identically to $\mathcal{S}$, as given by $\mathcal{A} = \mathcal{S}$ with $s^{\varepsilon+1} = a^{\varepsilon}$.

Policy (π). A policy determines what action the agent would take next based on a set of rules. In this case, the process of nodes assigned to different shards is based on the policy that is continually updated and trained. i.e., $\pi(s, a) : \mathcal{S} \to \mathcal{A}$.

Reward function (r). The reward function is inherited from the objective function, $r : R(\xi, \varrho, \phi_{cr}, \psi, \Omega_{in}, \Omega_{cr})$.

DRL is a versatile method that allows agents to make decisions in dynamic environments effectively. Its ability to handle complex data and its proactive defense against malicious attacks make it an ideal solution for managing node and transaction interactions in sharding systems.

The sharding optimization problem, which involves assigning each node to a specific shard, is a known non-convex, NP-hard problem due to the binary nature of decision variables [126]. Such complexity highlights the appropriateness of DRL as an approach to address this challenge compared to traditional methods such as convex optimization, underlining its significance in managing the complexities of sharded blockchain systems.

4.4 Experiment and Evaluation

The experiment assesses the DRL-based sharding approach regarding convergence performance and stability under various environment settings, including the number of shards, the number of nodes, the resource distribution, and other practical settings. Note that the parameters in the following experiments align with real-world IoT scenarios, such as Mobile Edge Computing (MEC), where edge servers on vehicles or drones collect data from end devices such as sensors. These servers initiate resharding to optimize data processing by redistributing workload when moving across regions.

Table 4.2: Hyperparameters

Notation	Description	Value
e_{max}	The number of epochs in DRL	$[30, 100]$
F	The probability of node failing to vote	20%
A	The fraction of dishonest nodes within one shard	$[0, 30]$
τ	The threshold for triggering colluding strategy	10%
γ	The discount factor for calculating OT of each leader	0.9
ρ_t	The threshold of differentiating low-risk nodes and high-risk nodes in terms of trust	0.67
f_{intra}	The threshold of faulty tolerance within one shard	$\lfloor (N_x - 1)/3 \rfloor$
ρ_{cr}	The threshold of triggering resharding in terms of CST ratio	0.4

4.4.1 Experiment Framework

An experimental framework is implemented in an environment with 2.5 Ghz, 20 cores, 2 × Intel(R) Xeon(R) Gold 6248 CPU, NVIDIA Quadro P4000, 768GB memory to evaluate a

proposed blockchain sharding scheme. A virtual machine is created using Python 3.8.10 and Pytorch 1.13.1. In this setting, the discrete DRL algorithms, DQN and PPO, are used and run over $30 - 100$ epochs. The range of total nodes from $0 - 16$ is set in the environment. The transaction distribution model between nodes follows the normal distribution. The trustworthy coordinator is implemented by deploying the smart contract.

In the experimental setup, the blockchain sharding system TbDd is assessed against other sharding techniques such as random-based sharding [63], community-based sharding [107], and trust-based sharding [108]. The dishonest nodes in the range of $0 - 5$ are accounted for, which employ collusion strategies during block verification. These nodes may produce deceptive block verifications and send CSTs across different shards. Initially, these dishonest nodes are scattered randomly across shards. Positive noises are introduced to their transaction counts to model interactions between dishonest nodes in distinct shards. The resulting transaction distribution table is influenced by normally distributed transactions. The experiments' hyperparameters are detailed in Tab 4.2.

As the intra-shard fault tolerance threshold f_{intra} is set to $\lfloor (N_x - 1)/3 \rfloor$, the relationship between the total number of nodes N in the network, the number of shards D, and the total fault tolerance f_{total}, is evaluated as (4.24)

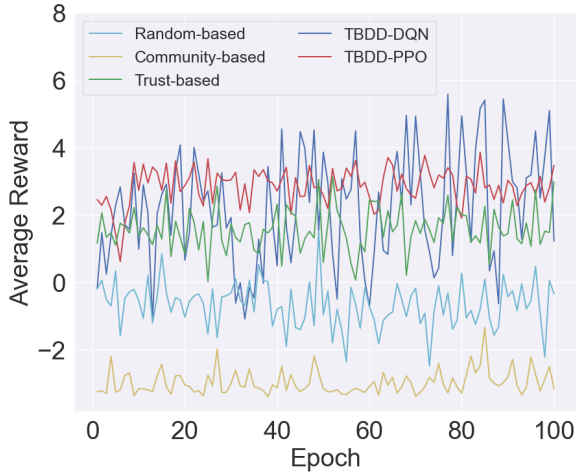
$$f_{total} = \left\lfloor \left(\left\lfloor \frac{N}{D} \right\rfloor - 1 \right) /3 \right\rfloor \times D, \tag{4.24}$$

by which one can realize whether the entire network has failed.

4.4.2 Experiment Results

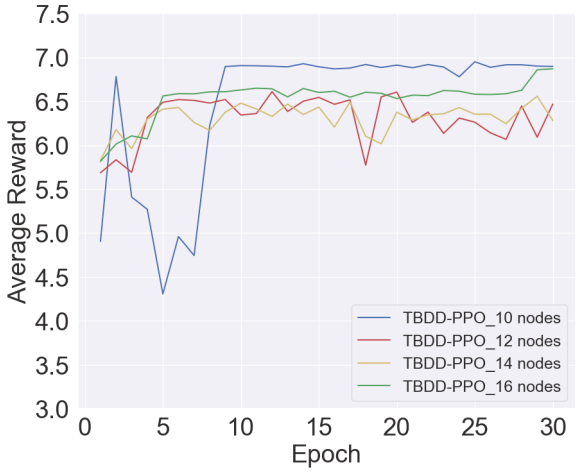
In the experimental evaluations, the performance of the TbDd system is compared with random-based, community-based, and trust-based sharding approaches using metrics such as CST ratio, shard risk variance, corrupted shard number, and convergence speed. Through these evaluations, the effectiveness and robustness are validated by the proposed approach. In the following figures, each figure is labeled in terms of the number of nodes N, shards D, and dishonest nodes h, respectively.

Fig. 4.5(a) illustrates the average rewards achieved in a single epoch with different sharding schemes. The community-based sharding technique recorded the lowest reward. While the scheme effectively reduces cross-shard transactions, it remains susceptible to adaptive collusion



(a) $N = 16, D = 2, h = 4$.

(b) $N = \{10, 12, 14, 16\}$, $D = 2$, $h = 2$.



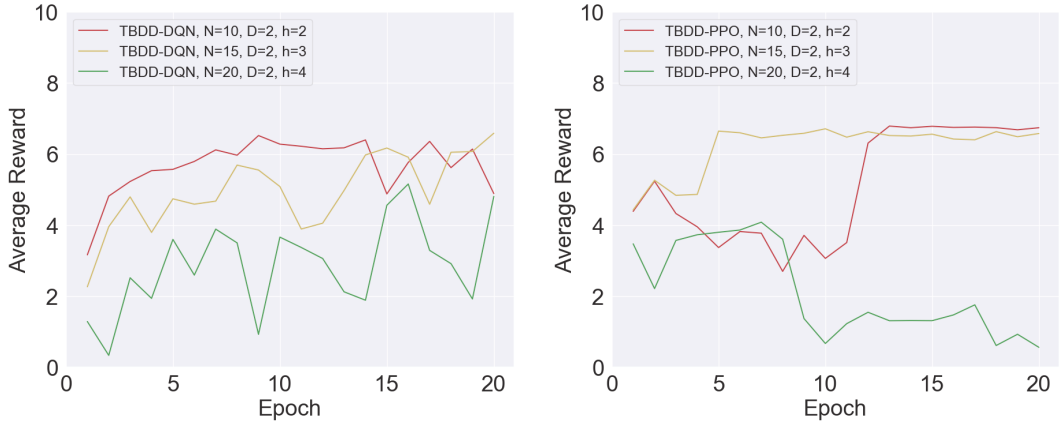
(c) $N = \{10, 12, 14, 16\}$, $D = 2$, $h = 2$.

Figure 4.5: (a) represents the comparison among Random-based, Community-based, Trust-based, TBDD-DQN, and TBDD-PPO across 100 epochs. (b) and (c) represent the reward performance between TBDD-DQN and TBDD-PPO across varying node numbers in the environment settings across 30 epochs, respectively.

attacks due to its tendency to group a large number of dishonest nodes within the same shard, thereby compromising the shard's security. The random-based scheme fares slightly better, with its rewards hovering around zero. The trust-based sharding technique is more proficient at evenly distributing dishonest nodes across different shards, mitigating the risk of a single shard being dominated. However, it leads to a higher number of cross-shard transactions. Consequently, its rewards are marginally less than those of DQN and PPO. Upon reaching the 10^{th} epoch, both TBDD-PPO and TBDD-DQN consistently outperform the other sharding schemes, exhibiting consistently high rewards. This observation indicates that the agent

received the maximum reward associated with the action.

Figs. 4.5(b)–4.5(c) depict the convergence of rewards under various node numbers for TbDd-DQN and TbDd-PPO, respectively. The reward becomes more stable as the number of nodes increases. Across Figs. 4.5(a)–4.5(c), a consistent trend emerges caused by the constrained approach in the policy update process. PPO achieves more stable rewards. The instability of DQN is because it combines the direct value function estimation method and ϵ greedy exploration strategy.



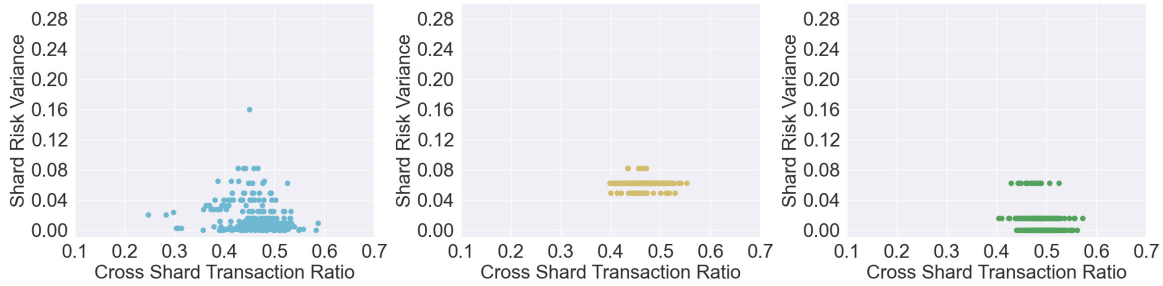
(a) $N = \{10, 15, 20\}$, $D = 2$, $h = \{2, 3, 4\}$, TbDd-DQN

(b) $N = \{10, 15, 20\}$, $D = 2$, $h = \{2, 3, 4\}$, TbDd-PPO

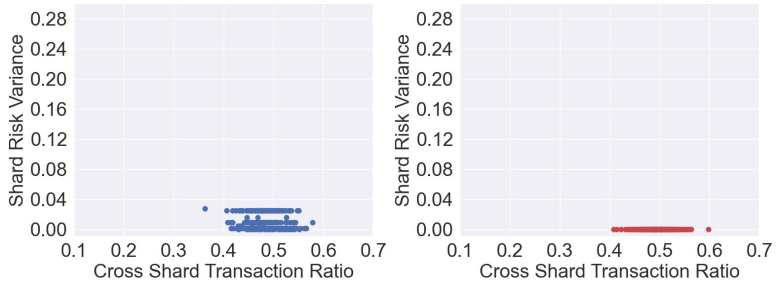
Figure 4.6: Tracing the impact of the number of nodes under the same dishonest node ratio with respect to the reward performance, $N = \{10, 15, 20\}$, $D = 2$, $h = \{2, 3, 4\}$.

Fig. 4.6 shows the impact of varying the network size—10, 15, and 20 blockchain nodes—under a constant dishonest-to-honest node ratio (i.e., 0.2). Although the ratio remains unchanged, the absolute number of dishonest nodes increases alongside N, heightening the likelihood of collusion attacks. Consequently, fewer nodes tend to yield higher average rewards, thereby enhancing security and efficiency for both the TbDd-DQN and TbDd-PPO mechanisms. Moreover, the distribution of dishonest nodes across shards may fluctuate as N changes, causing performance disparities even under the same ratio. Additionally, the results indicate that the PPO method exhibits greater stability than the DQN scheme, partly due to its continuous policy updates and more robust handling of these distributional effects.

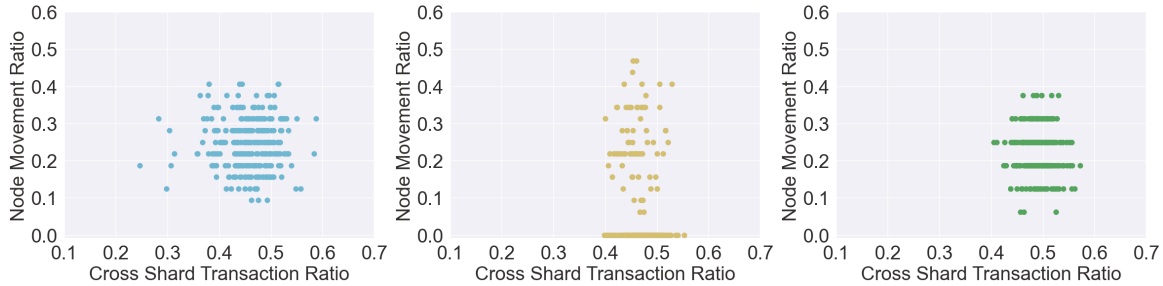
Figs. 4.7(a)–4.7(e) (Shard Risk Variance vs. CST Ratio) illustrate the trade-offs between the risk distribution among shards and the system's ability to handle cross-shard transactions. The ideal positioning in these scatter plots is towards the bottom-left corner, indicating low shard



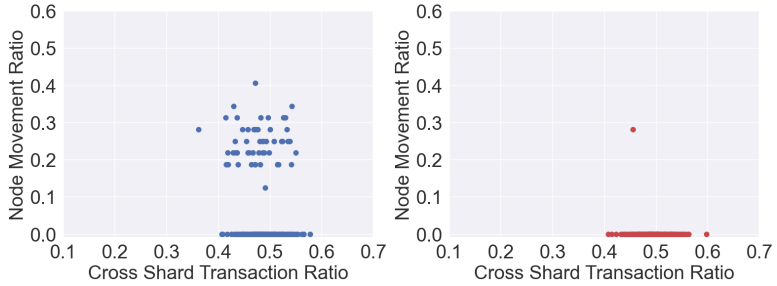
(a) Random-based (b) Community-based (c) Trust-based



(d) TbDd-DQN (e) TbDd-PPO



(f) Random-based (g) Community-based (h) Trust-based



(i) TbDd-DQN (j) TbDd-PPO

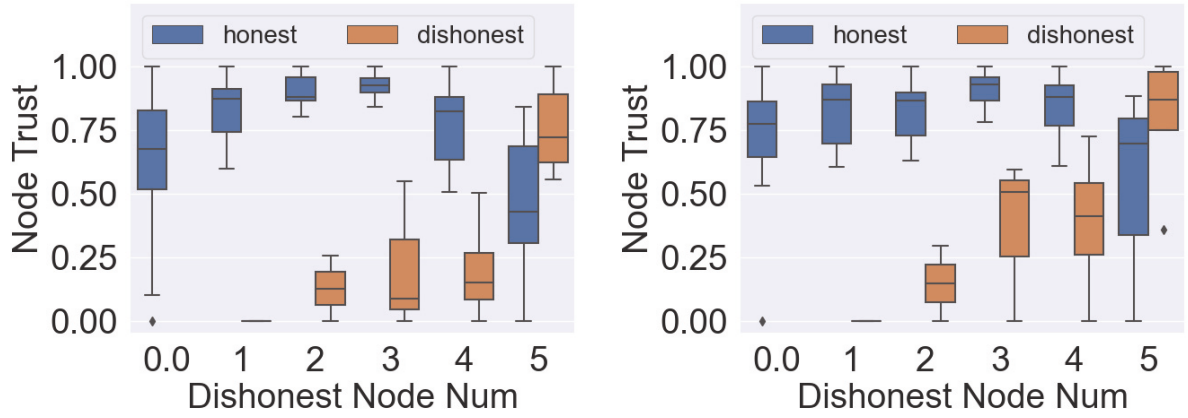
Figure 4.7: Comparison of different sharding schemes between random-based, community-based, trust-based, TbDd-DQN, and TbDd-PPO with $N = 16, D = 2, h = 4$ over 100 epochs. The x-axis represents the CST ratio, and the y-axis represents the cross-shard's trust variance through Figs. (a)-(e), while through (f)-(j), the x-axis represents the CST ratio and the y-axis represents the Node Movement Ratio. Lower values in the Shard Risk Variance and Node Movement Ratio indicate enhanced security and system stability, respectively. Lower CST Ratios indicate higher scalability of the sharding scheme.

risk variance (a secure and balanced distribution of nodes) and low CST ratio (high scalability with minimal cross-shard transactions). The random-based approach shows a scattered distribution, signifying a lack of predictability in achieving security and scalability. Community-based and trust-based strategies display a compromise between scalability and security, with community-based leaning towards scalability, sacrificing some security, and trust-based focusing heavily on security, which may limit scalability. The proposed TbDd-DQN and TbDd-PPO methods demonstrate a more balanced approach, achieving lower risk variance without significantly compromising on the CST ratio, thereby endorsing the efficacy of the TbDd framework.

Figs. 4.7(f)–4.7(j) (Node Movement Ratio vs. CST Ratio) present the scalability and efficiency aspect, where a lower Node Movement Ratio implies reduced overhead and increased stability due to less frequent reassignments of nodes to different shards. The Random-based sharding approach results in an unpredictable pattern, leading to potentially frequent node reassignments that can increase system complexity. The adaptive collusion behavior of dishonest nodes significantly impacts both community-based and trust-based sharding strategies. This malicious behavior intentionally misleads the sharding mechanisms, resulting in increased node movements for both strategies. In contrast, the TbDd-DQN and TbDd-PPO strategies demonstrate a reduction in node movements, with TbDd-PPO showing the greatest stability and efficiency in system operation among the evaluated strategies. Fig. 4.7 highlights the TbDd framework's ability to achieve a balanced and secure blockchain system, capable of addressing the complex trade-offs involved in blockchain system design.

In Fig. 4.8, it is shown that the average trust of dishonest nodes surpasses that of honest nodes as the number of dishonest nodes increases. The uppermost horizontal line in each column represents the maximum trust value among all nodes, while the bottom horizontal line represents the minimum trust value. The horizontal line at the midpoint signifies the median trust value of all nodes. When the total number of nodes is 16, and the total shard number is 2, the overall fault tolerance threshold f_{total} is 4 based on (4.24). Consequently, a shard becomes vulnerable and is corrupted when the number of dishonest nodes $h \geq 5$.

As shown in Figs. 4.9(a)–4.9(l), the blue curves represent the average trust of honest nodes, and the red curves are dishonest nodes. Consistent results from Fig. 4.8 reveal that the system can effectively perform sharding if the proportion of dishonest nodes remains below f_{total} of the total node count. However, if the number of dishonest nodes exceeds the f_{total} threshold, any

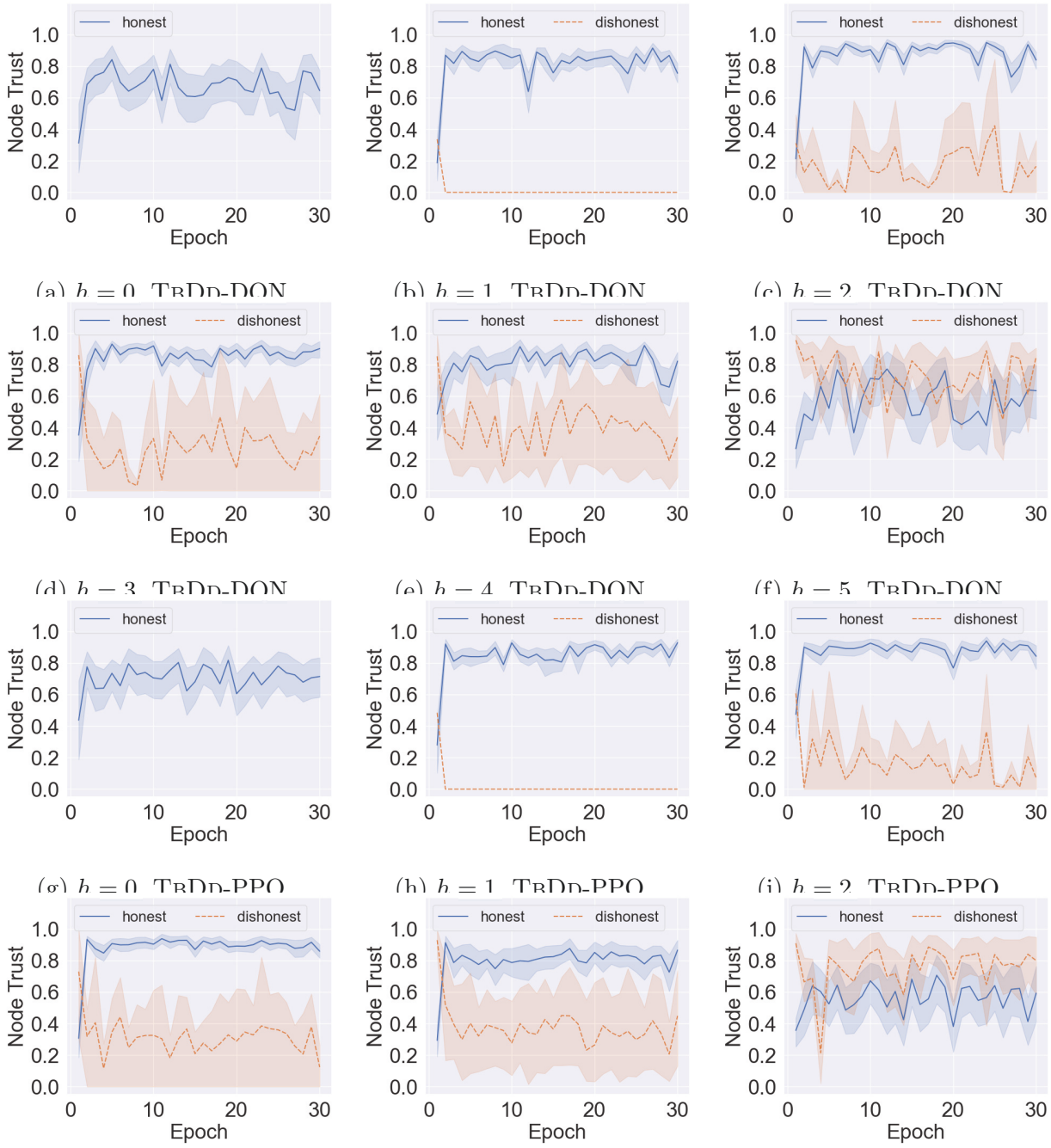


(a) $N = 16, D = 2, h = \{0, 1, 2, 3, 4, 5\}$, TbDd-DQN (b) $N = 16, D = 2, h = \{0, 1, 2, 3, 4, 5\}$, TbDd-PPO

Figure 4.8: Tracing the impact of the number of dishonest nodes for node trust with DRL approach, $N = 16, D = 2, h = \{0, 1, 2, 3, 4, 5\}$. The left subfigure uses the DQN algorithm, and the right subfigure uses the PPO algorithm.

sharding approaches, including the proposed schemes TbDd-DQN and TbDd-PPO, cannot safely execute sharding and are vulnerable to the 1% attack, which is beyond the scope of the investigation. On the other hand, having a lower count of dishonest nodes still enables the implementation of more shards within the system, leading to improved overall performance and scalability.

The normalized throughput metric evaluates throughput across different sharding methods influenced by dishonest nodes. This metric compares the throughput with and without dishonest nodes. A higher ratio suggests dishonest nodes have minimal impact on transaction throughput, while a lower one indicates a significant negative effect. Additionally, the benefits of the proposed system are highlighted by comparing the shard corruption ratio over the last 100 rounds among various sharding approaches. As depicted in Figs. 4.10(a)–4.10(b), random-based sharding exhibits the lowest throughput and compromised security. As dishonest node counts rise, there is a pronounced decline in scalability coupled with an increased number of corrupted shards. The Community-based sharding method has the best scalability and maintains a high throughput. Yet, its oversight on the security front results in a higher rate of shard corruption. In comparison, the trust-based sharding method reduces the chances of shard corruption but lags in throughput, signaling its scalability constraints. Without collusion attacks and within the tolerable limit of dishonest nodes, the proposed TbDd-DQN and TbDd-PPO in this paper achieves approximately a 10% throughput improvement over random-based sharding



(a) $h = 0$, TBDD-DQN (b) $h = 1$, TBDD-DQN (c) $h = 2$, TBDD-DQN

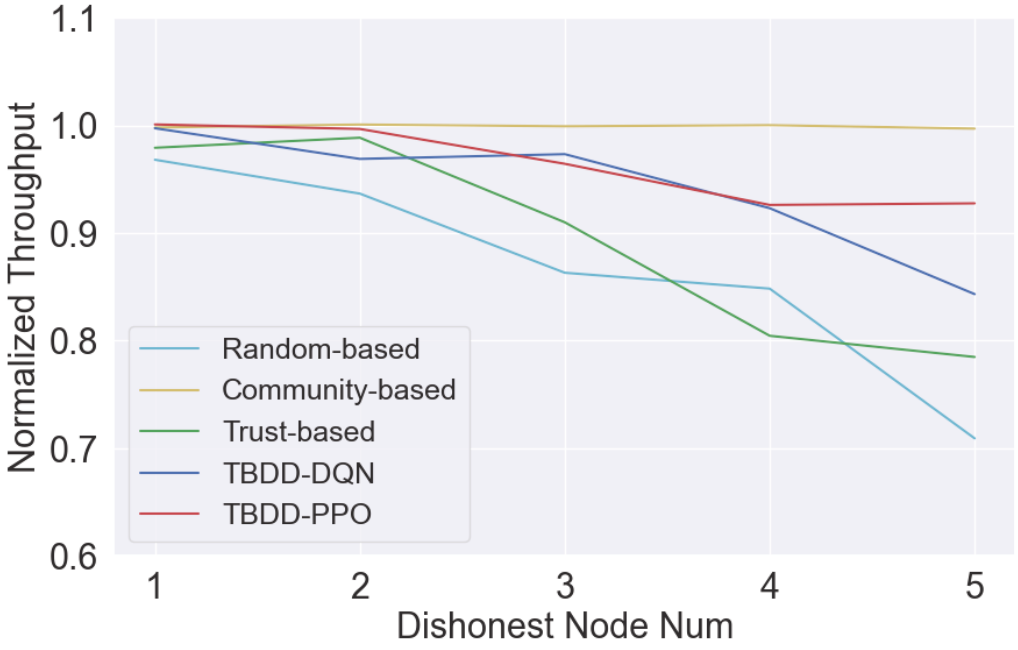
(d) $h = 3$, TBDD-DQN (e) $h = 4$, TBDD-DQN (f) $h = 5$, TBDD-DQN

(g) $h = 0$, TBDD-PPO (h) $h = 1$, TBDD-PPO (i) $h = 2$, TBDD-PPO

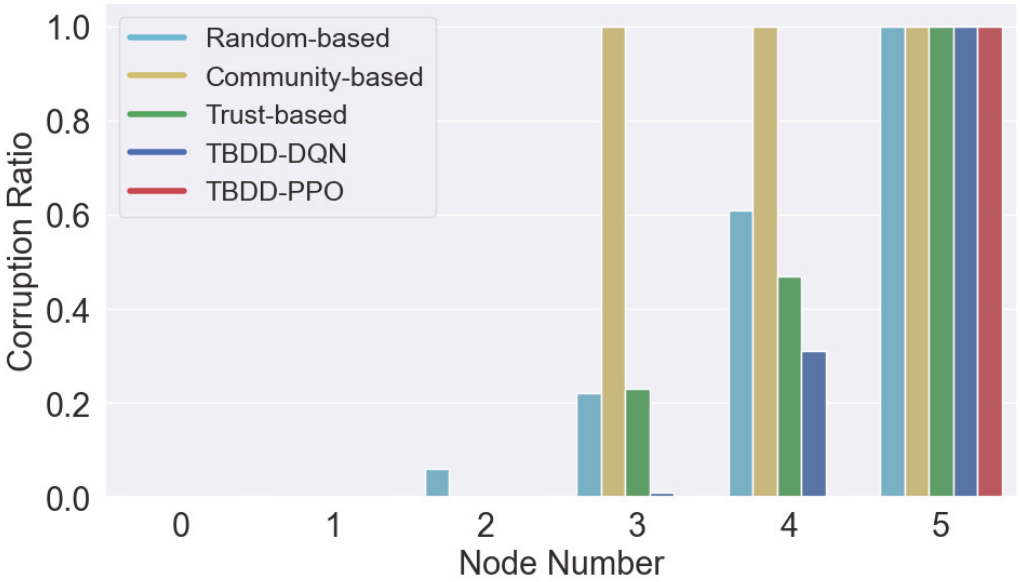
(j) $h = 3$, TBDD-PPO (k) $h = 4$, TBDD-PPO (l) $h = 5$, TBDD-PPO

Figure 4.9: Tracking global trusts $\mathcal{G}$ between honest and dishonest nodes as the number of dishonest nodes escalates within the TBDD-DQN and TBDD-PPO schemes with $N = 16$, $D = 2$, $h = \{0, 1, 2, 3, 4, 5\}$.

method and a 13% increase compared to the trust-based method.



(a) $N = 16, D = 2, h = \{0, 1, 2, 3, 4, 5\}$



(b) $N = 16, D = 2, h = \{0, 1, 2, 3, 4, 5\}$

Figure 4.10: (a) The relationship between the number of dishonest nodes, system throughput, and corrupted shards is interconnected. (b) Corrupted shard ratio with different sharding techniques across 100 epochs.

4.4.3 Discussion

Latency. The evaluation of network latency and overall system latency is presented in Table 4.3. Distributed blockchain nodes employ the Practical Byzantine Fault Tolerance (PBFT) consensus [51] algorithm for block generation. The latency across distributed nodes follows a

unified normal distribution with mean latencies of 0.1, 0.5, and 1 seconds and a standard deviation of 0.1 seconds. The DRL algorithms in this paper, i.e., DQN and PPO, are executed on a high-performance computer featuring a 2.5GHz CPU and 768GB memory. Throughout the experiment, a sharding cycle of one day is considered [25,64], during which the DRL algorithms are executed daily, and block consensus proceeds following the results of DRL-based sharding to calculate the per-block latency resulting from the DRL algorithms.

Regarding the impact of network latency, Table 4.3 illustrates that the block time is around three times the network latency under the normal distribution network latency model and PBFT consensus protocol. A slight decrease in consensus latency is observed when the number of dishonest nodes remains constant while the total number of nodes increases. Conversely, an increase in the number of dishonest nodes is linked to a notable rise in consensus latency, indicating that a greater presence of dishonest nodes negatively impacts the time required to achieve consensus.

In terms of the impact of varying numbers of nodes on system latency, the experimental results in Table 4.3 indicate a small variation in block time with changes in the total number of nodes. Despite the exponential growth in DRL latency as the total number of nodes increases, the latency per block remains manageable. Table 4.3 further demonstrates that the PPO latency is lower than the DQN latency for smaller node quantities. However, the PPO latency increases at a faster rate compared to the latency of DQN. This observation highlights a trade-off between reward performance and system latency, as the PPO method typically yields superior rewards compared to the DQN method.

Other types of latency include block verification, sharding strategy optimization, and resharding phases. Tolerating delay during block verification is a necessary trade-off to prevent double-spending issues [17]. However, an extended offline DRL learning phase for the sharding strategy is not favorable. Mitigation can be achieved by aligning online sharding strategy learning with the execution of resharding. Latency concerns during resharding, due to extensive node synchronization, can be alleviated through state channels [101] that expedite off-chain transactions, thus reducing blockchain load and hastening resharding. Integrating state channels into TbDd enables certain IoT transactions to be executed off-chain during proposed block verification, which lessens the node verification load and enhances overall system responsiveness.

Edge-driven Protocol. The architecture of TbDd uniquely operates on edge servers, not

Nodes	Network Latency (s)	Block (s)	DQN (s)	DQN per block (s)	PPO (s)	PPO per block (s)
$N = 10, h = 2$	0.1	0.458	2092	0.011	1369	0.007
	0.5	1.727		0.043		0.028
	1	3.223		0.080		0.052
$N = 12, h = 2$	0.1	0.334	2320	0.009	1787	0.007
	0.5	1.602		0.044		0.034
	1	3.094		0.085		0.065
$N = 14, h = 2$	0.1	0.374	2412	0.011	2194	0.010
	0.5	1.633		0.047		0.043
	1	3.139		0.090		0.082
$N = 15, h = 3$	0.1	0.375	2478	0.011	2646	0.012
	0.5	1.649		0.049		0.052
	1	3.150		0.093		0.100
$N = 16, h = 3$	0.1	0.446	4021	0.022	4388	0.024
	0.5	1.730		0.084		0.093
	1	3.227		0.157		0.173
$N = 18, h = 3$	0.1	0.361	4475	0.020	10031	0.047
	0.5	1.636		0.089		0.215
	1	3.140		0.172		0.412
$N = 20, h = 4$	0.1	0.418	11592	0.065	49466	0.560
	0.5	1.703		0.264		2.281
	1	3.203		0.496		4.289

Table 4.3: Latency (in seconds) with different numbers of nodes, dishonest nodes, and network latency.

directly on IoT sensor end nodes. This strategic placement ensures that the system can manage extensive computations associated with blockchain operations without overwhelming individual IoT devices. As a result, the scaling exhibited in the experiments is consistent and realistic, representing a practical implementation in real-world IoT networks. This edge-driven approach aligns with the broader move towards edge computing in IoT, capitalizing on its benefits to improve scalability and responsiveness.

Decentralized Coordination. Our design leverages a decentralized TC, acting as a trustworthy third-party coordinator, eliminating the vulnerabilities associated with a single centralized coordinator. This approach significantly mitigates the risks of a single point of failure in the system. The intra-consensus security within the decentralized TC ensures robust and reliable decision-making processes. Such a structure has been mirrored in existing studies [63, 64], affirming its practicality and security. By embedding this design, TbDd further ensures system

robustness, sustaining its promises of trustworthiness and integrity in diverse IoT settings.

4.5 Conclusion

In this chapter, we introduce TbDd, a novel trust and DRL-based sharding framework, which represents a significant advancement in blockchain technology for IoT environments. TbDd surpasses existing random, community, and trust-based methods, demonstrating a 10% throughput advantage over random-based sharding and a 13% improvement compared to trust-based methods, along with achieving the lowest rate of corrupted shards. This enhancement in security and scalability makes it well-suited for real-world IoT applications such as smart cities and industrial IoT. By integrating trust mechanisms with DRL, TbDd effectively addresses major IoT issues such as scalability and security, elevating sharding technology and offering a robust, efficient solution for diverse IoT contexts. Its adaptability across various node sizes and minimal system latency showcase its potential to set new standards in the field.

Chapter 5

Enabling Efficient Cross-Shard Smart Contract Calling via Overlapping

5.1 Introduction

The scalability and efficiency of blockchain networks have been significantly enhanced through the adoption of sharding, which partitions the network into smaller, manageable segments called shards. However, managing cross-shard transactions, particularly those involving smart contracts, remains a complex and latency-prone process due to the need for extensive coordination between shards. This chapter introduces an innovative framework designed to address these challenges by leveraging overlapping shards and an optimized consensus mechanism. This approach transforms cross-shard transactions into more efficient intra-shard operations, thereby reducing latency and improving transaction security.

5.2 Cross-Shard Smart Contract Framework: Overlapping Shards and xPBFT Consensus

In this chapter, we detail the architecture of overlapping shards designed to address the high latency and security challenges inherent in cross-shard transactions involving smart contracts. Several leaders are randomly selected from each shard to form an overlapping shard, ensuring that these leaders have access to comprehensive transaction data from all shards. Within this novel framework, the xPBFT consensus mechanism operates through an overlapping shard

structure.

5.2.1 Sharding with Overlapping shard

Each shard maintains its ledger in the proposed architecture, but the leader within the overlapping shard can access the full transaction data across the network. This setup is crucial for managing both intra-shard and cross-shard transactions effectively. The role of overlapping shard leaders is not only to facilitate cross-shard transactions but also to ensure that intra-shard transactions adhere to the global state of the blockchain, thereby preserving consistency and integrity throughout the system. Tab. 5.1 presents the notations used throughout the paper.

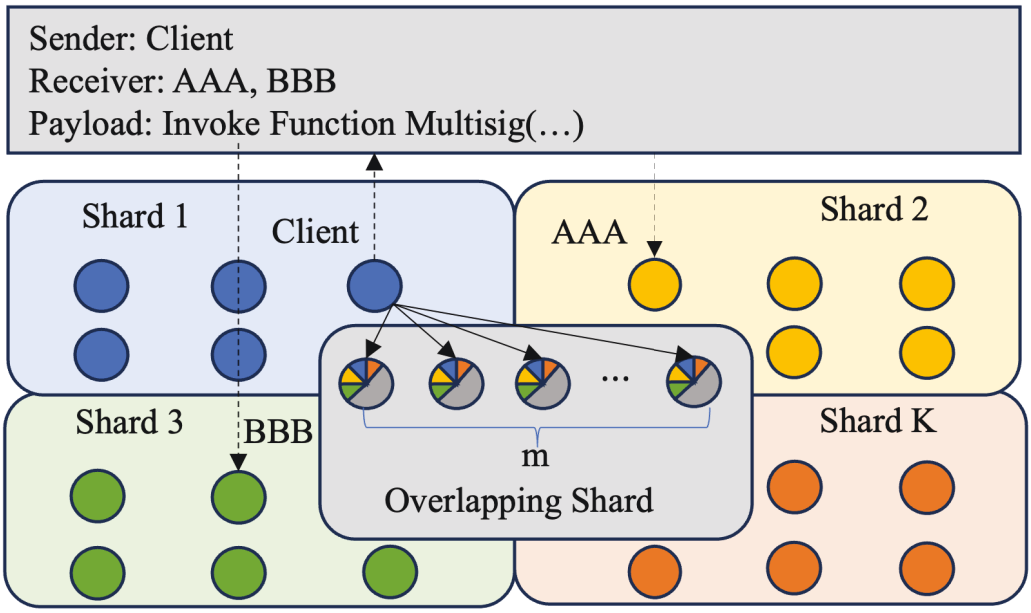


Figure 5.1: A client proposes a transaction and broadcasts it to the overlapping shard containing several leaders from each shard.

Fig. 5.1 illustrates the process of proposing and broadcasting a transaction to the overlapping shard. It highlights the central role of leaders from each shard in the initial transaction handling. Each shard operates independently and includes multiple nodes, designated as n_k. From these, a subset, m_k, are chosen as leaders. These leaders collectively form the overlapping shard, consisting of m total nodes, thus representing the combined leadership from all participating shards.

Table 5.1: Notations

Notation	Description
n	Total number of nodes
n_k	Total number of nodes in the k-th shard
K	Total number of shards
m	Total number of leader nodes in overlapping shard
m_k	Total number of leader nodes in the k-th shard
$\mathbb{M}$	Set of nodes in overlapping shard
T^{ε}	Total latency in the ε consensus round
T^{ε}_{pkg}	Block packing latency in the ε consensus round
T^{ε}_{cp}	Consensus latency in the ε consensus round
	including phases: $T^{\varepsilon}_{cp,pre-prepare}$, $T^{\varepsilon}_{cp,prepare}$, $T^{\varepsilon}_{cp,submit}$, $T^{\varepsilon}_{cp,result}$, $T^{\varepsilon}_{cp,verify}$
$B^{\varepsilon}_{pre-prepare}$	Block data size in the pre-prepare stage at the ε
$B^{\varepsilon}_{prepare}$	Block data size in the prepare stage at the ε
B^{ε}_{submit}	Block data size in the commitment stage at the ε
B^{ε}_{result}	Block data size in the reply stage at the ε
S	Transmission rate
c_k	Total CPU cycle requirement in the k-th shard
f_i	Computational Capability (CPU cycles per second) for i-th node
F	Total number of signatures
V	Total number of Message Authentication Codes
D	Total storage requirement
$\mathcal{B}_k$	Block storage requirement in the k-th shard

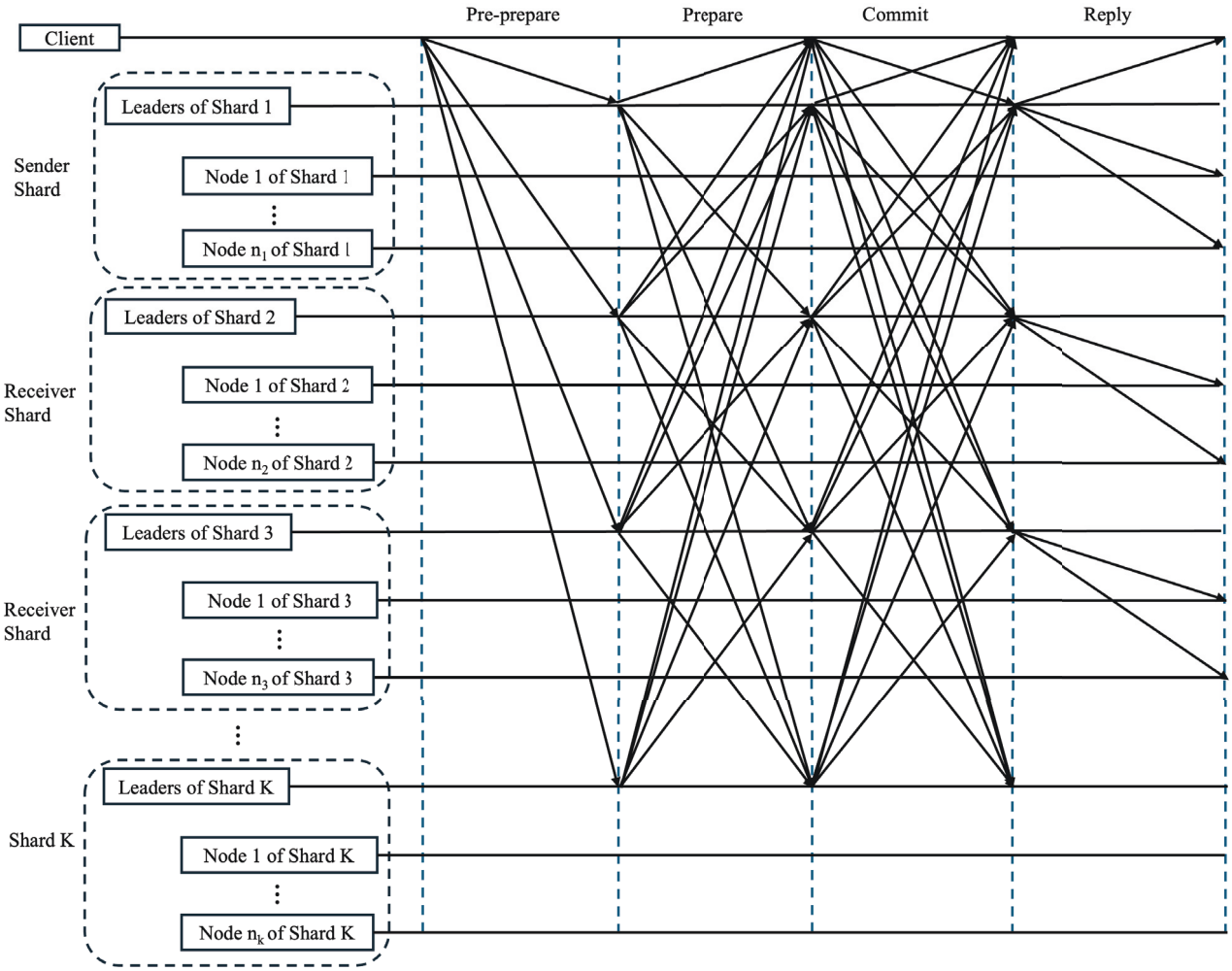


Figure 5.2: The proposed PBFT workflow based on the overlapping sharding framework. The security is ensured by keeping malicious nodes in the overlapping shard below the threshold α, defined as one-third of all leader nodes ($\alpha = \frac{1}{3}(m_1 + m_2 + \cdots + m_k)$). Only the sender and receiver shards involved in the transaction require replies. The client can be any node initiating a transaction from any shard. Leaders from Shard 1 to Shard K form the overlapping shard, coordinating to reach a consensus on cross-shard smart contract transactions.

5.2.2 The Proposed xPBFT Consensus

As depicted in Fig. 5.2, the workflow of the proposed xPBFT consensus within the overlapping sharding framework is designed to optimize both the speed and security of transaction processing. Utilizing an overlapping shard that exists for all shard leaders, the system can rapidly reach consensus on intra-shard and cross-shard transactions, significantly reducing the communication overhead typically associated with such operations.

The xPBFT consensus process, as outlined in Alg. 4, starts when a client submits a transac-

Algorithm 4: PBFT Consensus for Overlapping Shards

Input: $REQ(C, op, t, Sig)$: The transaction request from client C with operation op, timestamp t, and signature Sig;

Output: Confirmation of transaction commitment;

$v \leftarrow$ **GetCurrentViewNumber(*REQ*)** ;

$\phi \leftarrow$ **GetUniqueSequenceNumber(*REQ*)** ;

$d \leftarrow$ **ComputeDigest**(REQ) ;

BroadcastToAllLeaders$(PP(v, \phi, d, REQ))$;

// Broadcast pre-prepare message to all leaders in the overlapping shard.

foreach $leader \in \mathbb{M}$ **do**

    $P \leftarrow$ **ReceivePrepareMessage**$(PP(v, \phi, d, REQ))$;

    // Each leader receives prepare messages from others.

    if ***ValidatePrePrepare***$(\mathcal{P}(v, \phi, d, i).)$ **then**

        AddToPreparedList$(\mathcal{P}(v, \phi, d, i).\ )$;

        // Validate and add to prepared list if valid.

if ***ConsensusReached***$(\mathcal{P}(v, \phi, d, i).\ )$ **then**

    $\mathcal{C} \leftarrow$ **CommitTransaction**$(\mathcal{P}(v, \phi, d, i))$;

    BroadcastToAllNodesInShards$(\mathcal{C}(v, \phi, d, i, s))$;

    //Commit the transaction and notify all nodes. ;

    return "Transaction Committed Successfully" ;

else

    return "Transaction Failed to Reach Consensus" ;

tion request to the overlapping shard's leaders during the **Pre-prepare Phase**. This request, denoted as $REQ(C, op, t, \text{Sig})$, consists of the client submitting an operation op, client identifier C, timestamped with t, and secured by a digital signature Sig. During this phase, each transaction is encapsulated with a view number v, sequence number ϕ, and a digest d of the request, represented as $\mathcal{PP}(v, \phi, d, REQ)$. This initial phase guarantees uniform receipt and verification of transaction data by all leaders in the overlapping shard, mitigating the risk of double spending attacks and promoting a consistent approach to transaction validation.

After the Pre-prepare Phase, the transaction enters the **Prepare Phase**, where the detailed transaction data is further verified and prepared for final consensus. The Prepare message includes the view number v, sequence number ϕ, a digest d of the request and identification of the node that sent this prepare message i, represented as $\mathcal{P}(v, \phi, d, i)$. During the Prepare Phase, each leader within the overlapping shard independently validates the transaction, facilitating a consensus-driven verification process.

Upon achieving consensus, the transaction advances to the **Commitment Phase**, where it is formally committed to the blockchain. This phase marks the definitive agreement among the leaders, ensuring the transaction's integrity and finality. The commitment is denoted as

$\mathcal{C}(v, \phi, d, i, s)$, representing a complete and verified transaction.

Following the commitment, the **Reply Phase** initiates. During this phase, leaders from the overlapping shard communicate the transaction results to all nodes within the initiating client's and recipient's shard. This ensures that each shard involved is synchronized with the final outcome, maintaining a consistent and updated state across the entire network.

5.2.3 Security Analysis

During the investigation of the security dynamics within overlapping shards, a probabilistic approach is utilized to assess the distribution and behavior of nodes across multiple shards. The hypergeometric distributions [127, 128] are critical in evaluating the security of a sharded blockchain by modeling the randomness and potential risks associated with the distribution of malicious nodes across the overlapping shard.

Single Shard Case.

$$P(X_1 = i) = \frac{C_{h_1}^{i} C_{n_1-h_1}^{m_1-i}}{C_{n_1}^{m_1}}, \tag{5.1}$$

where n_1 is the total number of nodes in the shard-1. m_1 is the number of leader nodes from the shard-1 selected to join the overlapping shard. h_1 is the number of malicious nodes in the shard-1. i represents the number of malicious nodes from shard-1 that end up in the overlapping shard.

General Case for K Shards.

$$\begin{aligned} P\left(\sum_{k=1}^{K} X_k \leq \alpha\right) = \sum_{i_1=0}^{\alpha} \Bigg\{ P(X_1 = i_1) \times \Bigg[\sum_{i_2=0}^{\alpha - i_1} P(X_2 = i_2) \times \\ \cdots \times \sum_{i_{K-1}=0}^{\alpha - \left(\sum_{j=1}^{K-2} i_j\right)} \Bigg[P(X_{K-1} = i_{K-1}) \times \\ P(X_K \leq \alpha - \left(\sum_{j=1}^{K-1} i_j\right)) \Bigg] \Bigg] \Bigg\}, \end{aligned} \tag{5.2}$$

where $P(X_1 = i_1)$ represents the probability that i_1 node joins the overlapping shard from shard-1. X_k is defined as the random variable representing the number of malicious nodes assigned to the k-th shard. The formula shown aims to compute the security probability P that the aggregate number of malicious nodes $\sum_{k=1}^{K}$ across all shards K remains below a specific threshold α. This threshold α is defined as one-third of the total number of nodes joining the

overlapping shard from all shards involved, expressed as $\alpha = \frac{1}{3}\lfloor(m_1 + m_2 + \cdots + m_k)\rfloor$. The significance of α is to establish a bound at which the presence of malicious leaders within the overlapping shard would not pose a security risk. The term $P(X_K \leq \alpha - \left(\sum_{j=1}^{K-1} i_j\right)$ represents the cumulative probability that the number of malicious nodes in the K-th shard does not exceed $\alpha - \left(\sum_{j=1}^{K-1} i_j\right)$. This term ensures that the sum of malicious nodes across all K shards stays within the threshold α.

The cumulative probability function for X_K is defined as:

$$P\left(X_K \leq y\right) = \sum_{i=0}^{y} P\left(X_K = i\right), \tag{5.3}$$

where $y = \alpha - \left(\sum_{j=1}^{K-1} i_j\right)$ indicates the maximum allowable number of malicious nodes in shard-K that, when combined with the malicious nodes from other shards, does not exceed α. Each probability $P\left(X_K = i\right)$ within the sum is computed based on the hypergeometric distribution, which accounts for the likelihood of having exactly i malicious nodes in shard-K given the total nodes in that shard and the overall number of malicious nodes.

5.2.4 Complexity Analysis

Communication Complexity. Reducing communication complexity is pivotal in scaling distributed blockchain networks in public blockchain environments that utilize PBFT mechanisms. Traditionally, PBFT implementations suffer from a communication complexity of $O(n^2)$, where n is the total number of nodes in the network [53]. This inefficiency arises because each node must broadcast messages to every other node to achieve consensus, leading to significant operational challenges as the network expands.

The introduction of sharding, especially with an overlapping shard design where only select leaders from each shard participate in the consensus process, significantly reduces the communication complexity to $O(m^2)$, where m represents the number of nodes in the overlapping shard. This reduction from $O(n^2)$ to $O(m^2)$ enhances scalability and allows the network to support more nodes and handle a higher volume of transactions without a proportional increase in resource demand or latency.

Consensus Latency Costs. The latency T^{ε} for a consensus round ε is a critical metric in evaluating the efficiency of the proposed PBFT consensus mechanism for overlapping shards.

The equation for total latency is expressed as:

$$T^{\varepsilon} = \min(T^{\varepsilon}_{\text{pkg}} + T^{\varepsilon}_{\text{cp}}), \tag{5.4}$$

where $T^{\varepsilon}_{\text{pkg}}$ denotes the block packing latency, and $T^{\varepsilon}_{\text{cp}}$ represents the overall consensus latency, encompassing all critical phases of the consensus protocol. Each phase's latency depends on the maximum transmission time of data blocks and the node transmission rate, which is affected by the number of participating leaders, potentially leading to congestion.

In each shard, reaching consensus involves two key operations [129]: 1) the message transmission between nodes, including signature verification, and 2) the validation of the message authentication code (MAC) by the nodes. The total latency involved in a cross-shard transaction process, considering multiple phases of a consensus protocol, is detailed as follows:

$$\begin{aligned}
T^{\varepsilon}_{\text{cp}} &= T^{\varepsilon}_{\text{cp,pre-prepare}} + T^{\varepsilon}_{\text{cp,prepare}} + T^{\varepsilon}_{\text{cp,submit}} \\
&\quad + T^{\varepsilon}_{\text{cp,result}} + T^{\varepsilon}_{\text{cp,verify}} \\
&= \max\left\{ \left(\frac{B^{\varepsilon}_{\text{pre-prepare}}}{S(m)} \right) \middle| i, j \in \mathbb{M}, i \neq j \right\} \\
&\quad + \max\left\{ \left(\frac{B^{\varepsilon}_{\text{prepare}}}{S(m)} \right) \middle| i, j \in \mathbb{M}, i \neq j \right\} \\
&\quad + \max\left\{ \left(\frac{B^{\varepsilon}_{\text{submit}}}{S(m)} \right) \middle| i, j \in \mathbb{M}, i \neq j \right\} + \frac{\sum_{k=1}^{K} c_k}{f_i} \\
&\quad + \max\left\{ \left(\frac{B^{\varepsilon}_{\text{result}}}{S(m)} \right) \middle| i \in \mathbb{M}, j \in \mathbb{P} \right\} \\
&\quad + \max\left\{ \left(T^{\varepsilon}_{\text{cp,verify},i}, T^{\varepsilon}_{\text{cp,verify},j} \right) \middle| i, j \in \mathbb{N}, i \neq j \right\},
\end{aligned} \tag{5.5}$$

where $T^{\varepsilon}_{\text{cp}}$ represents the total latency in a cross-shard transaction, comprising stages like *pre-prepare, prepare, commitment, reply, and verification.* Each stage's latency depends on the maximum transmission time of data blocks ($B^{\varepsilon}_{\text{pre-prepare}}$, $B^{\varepsilon}_{\text{prepare}}$, etc.) and the transmission rate $S(m)$, which decreases as more leaders (m) participate, indicating potential congestion. $\mathbb{M}$ denotes a set of leaders in overlapping shards, critical in coordination across shards. $\mathbb{P}$ denotes all node sets in the sender shards and receiver shards involved in processing the transaction. c_k represents the CPU cycles the k-th shard requires to process a part of a cross-shard smart contract transaction. f_i defines the computational capability of the unit CPU cycles per second for the i-th node. $T^{\varepsilon}_{\text{cp,verify},i}, T^{\varepsilon}_{\text{cp,verify},j}$ represents the maximum verification delay encountered by any pair of nodes in the network, encompassing the time taken for computational tasks such as signature and MAC verification.

The verification latency for each node involved in the consensus process within a shard is quantified using the following formulas:

$$T^{\varepsilon}_{\text{cp,verify},i} = \frac{\alpha F + \beta[F(1+V) + 4(n_k - 1)]}{f_i}, \tag{5.6}$$

$$T^{\varepsilon}_{\text{cp,verify},j} = \frac{\alpha F + \beta[FV + 4(n_k - 1)]}{f_j}, \tag{5.7}$$

where F represents the total number of signatures that need to be verified. V represents the number of MACs that need to be verified for each request. The term $1 + V$ in node i's equation includes verifying each transaction's signature along with an additional signature for the block itself. n_k indicates the number of nodes within the k-th shard. Coefficients α and β measure the CPU cycles needed to verify one signature and one MAC, respectively, quantifying the computational effort required for cryptographic tasks. f_i and f_j denote the computational capability of nodes i and j, measured in CPU cycles per second, which is pivotal in determining the speed at which these nodes can handle cryptographic verification.

Storage Costs. The storage costs in a blockchain system, which includes both overlapping and non-overlapping shards, can be modeled using the following equation:

$$D = \sum_{k=1}^{K} m_k \mathcal{B}_k + \sum_{k=1}^{K} (n_k - m_k)\mathcal{B}_k, \tag{5.8}$$

where D represents the total storage overhead for the entire system, accounting for both overlapping and non-overlapping shards. K is the total number of shards in the system. Each shard has its specific storage requirements and node configurations. n_k denotes the number of non-leader nodes in the k-th shard. m_k indicates the number of leader nodes in the k-th shard. These leaders typically handle additional storage responsibilities due to managing data across multiple shards. $\mathcal{B}_k$ is the block storage requirement in the k-th shard, representing the data capacity needed to store in the chain. The first term of the equation, $\sum_{k=1}^{K} m_k \mathcal{B}_k$, calculates the total storage requirements for leader nodes in all shards. Leader nodes are crucial as they might store information from their shard and others, particularly in systems with overlapping shards. The second term, $\sum_{k=1}^{K} (n_k - m_k)\mathcal{B}_k$, represents the storage required for all non-leader nodes across each shard, which typically store only data relevant to their specific shard.

5.3 Experimental Results

To validate the performance of our proposed framework with overlapping shards and xPBFT consensus mechanism, experimental tests are conducted on a MacBook Pro equipped with a 2.5 GHz Quad-Core Intel Core i7 processor and 16GB of RAM.

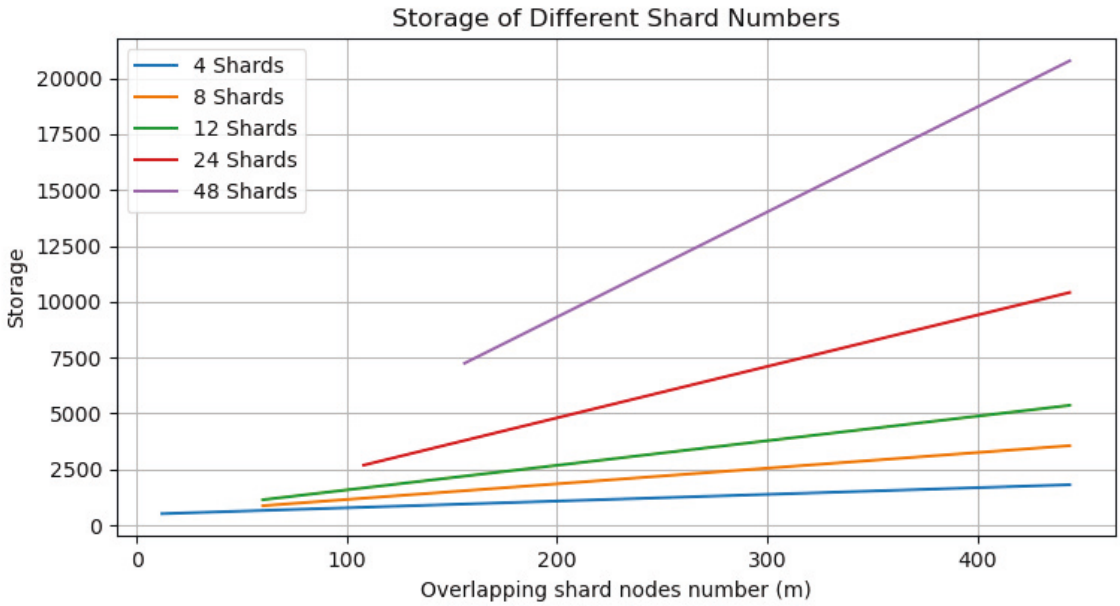


Figure 5.3: Storage escalation with shard number and leader number, $n_k = 100$, $m_k \geq 3$.

Fig. 5.3 shows a linear rise in storage requirements as the number of nodes within overlapping shards and the total number of shards increases. Systems with more shards, such as 48 shards, experience a sharper rise in storage needs compared to systems with fewer shards, such as 4 or 8. This indicates that higher shard counts naturally require more storage due to increased data replication to ensure transaction integrity and system robustness across a diverse network.

Fig. 5.4 illustrates the relationship between the delay and the ratio of leaders to total nodes (m/n) across different shard configurations. The graph shows that with an increase in the number of shards, from 2 to 6, the delay decreases, indicating that fewer leaders relative to the total number of nodes can efficiently handle consensus processes and minimize communication overhead. However, as the m/n ratio nears 1, the delay significantly increases across all shard configurations. This suggests that a higher proportion of leaders leads to improved communication and coordination overhead, slowing down the system. This pattern indicates an optimal m/n ratio where the delay is minimized, and this optimal point shifts slightly with changes in the number of shards.

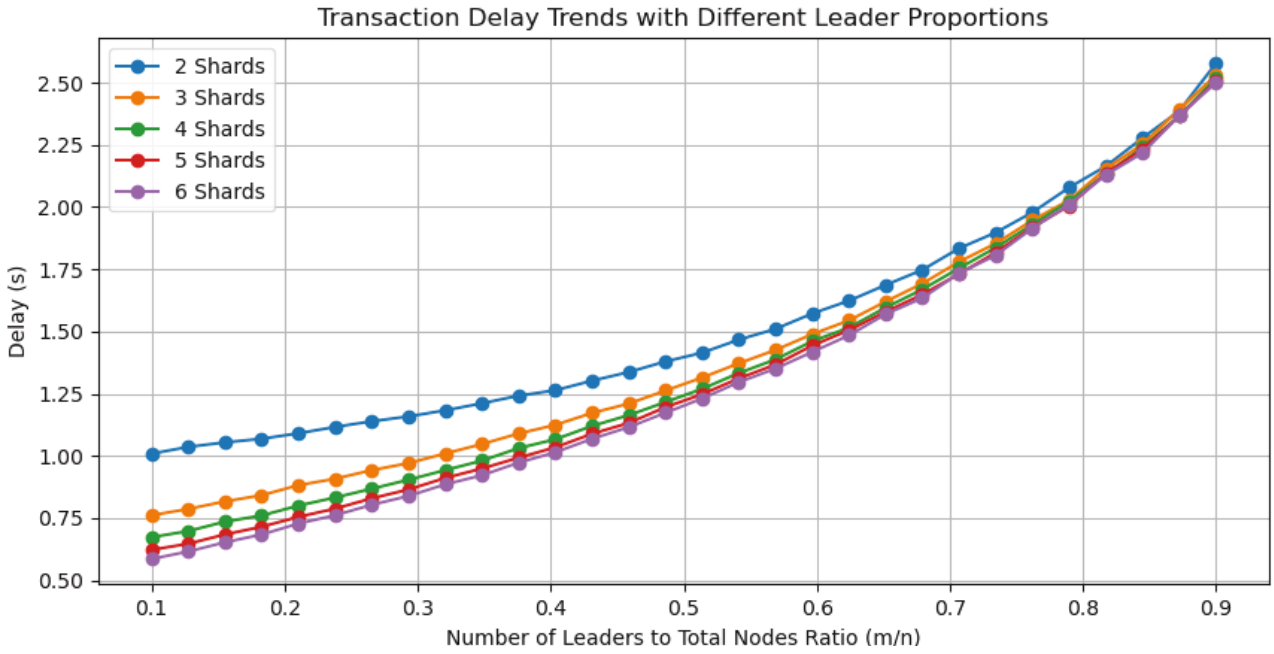


Figure 5.4: Transaction delay variations with different leader proportions, $n = 200$, $k = \{2, 3, 4, 5, 6\}$.

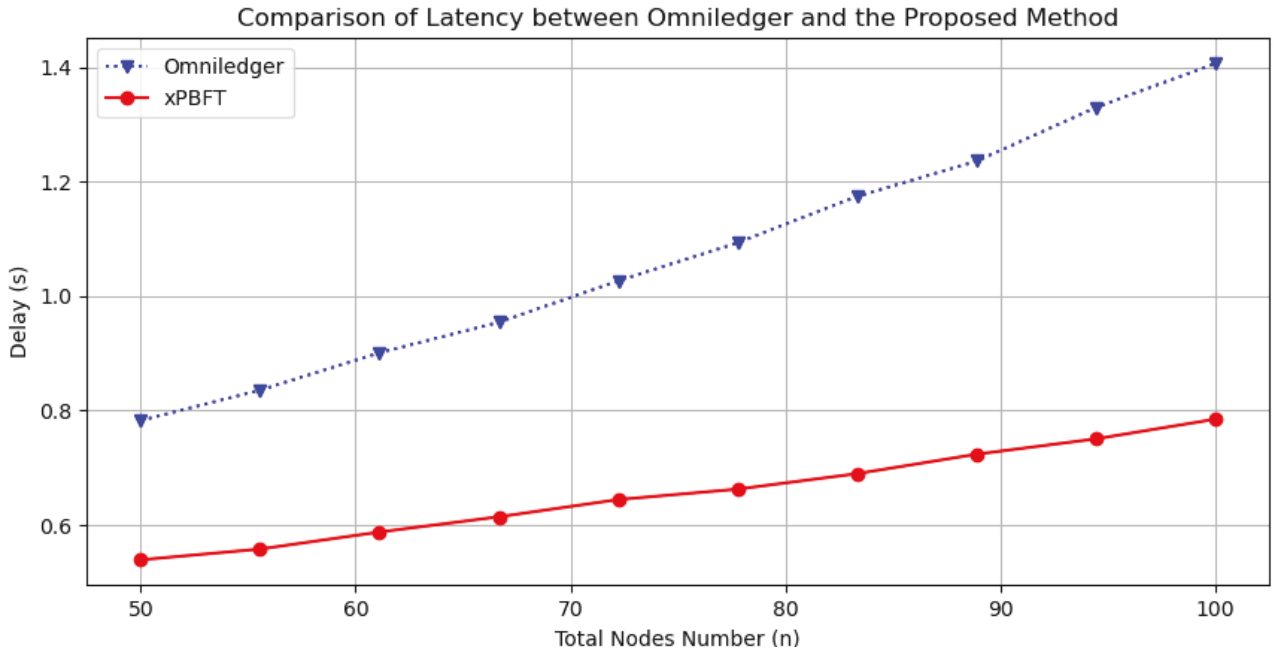


Figure 5.5: Comparison of transaction delays between traditional PBFT consensus (Omniledger) and the proposed xPBFT consensus, $n = [50, 100]$.

Fig. 5.5 illustrates the latency differences between the proposed xPBFT consensus and the traditional PBFT used in Omniledger [64], emphasizing the improvements across varying node counts. The xPBFT demonstrates a consistent 40% reduction in latency compared to Omniledger, underscoring its enhanced efficiency in transaction handling. This efficiency is achieved

through xPBFT's optimized design, which reduces communication overhead and accelerates transaction processing.

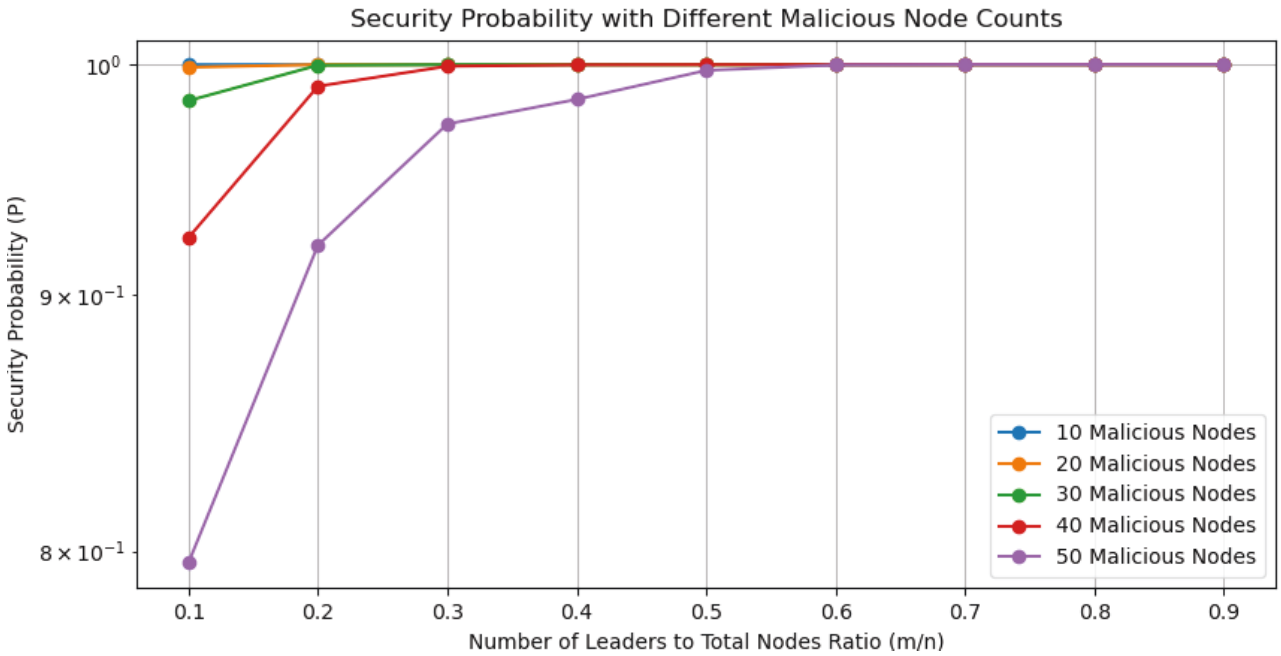


Figure 5.6: Security enhancements with varying leader counts and malicious scenarios, $n = 200$, $k = 2$, $h = \{10, 20, 30, 40, 50\}$.

Fig. 5.6 focuses on the variation in security probability with different counts of malicious nodes. As the number of leader nodes increases relative to the total nodes, systems with a low count of malicious nodes (10 and 20) consistently maintain high-security levels, exceeding the 99% threshold deemed secure. However, in systems burdened with a higher count of malicious nodes (30 to 50), enhancing the proportion of leader nodes significantly boosts security probabilities, illustrating the effectiveness of leader density in countering higher security threats. This effect underscores the importance of strategic leader placement in sharded architectures to safeguard against escalating malicious activities.

Fig. 5.7 evaluates the security probability across different shard counts and shows that the security probability of blockchain systems using overlapping shards stabilizes at high levels as the ratio of leader nodes to total nodes (m/n) increases. This stabilization is especially notable in configurations with fewer shards. Systems achieve nearly perfect security probabilities exceeding 99%, demonstrating robust defenses even with minimal leader representation.

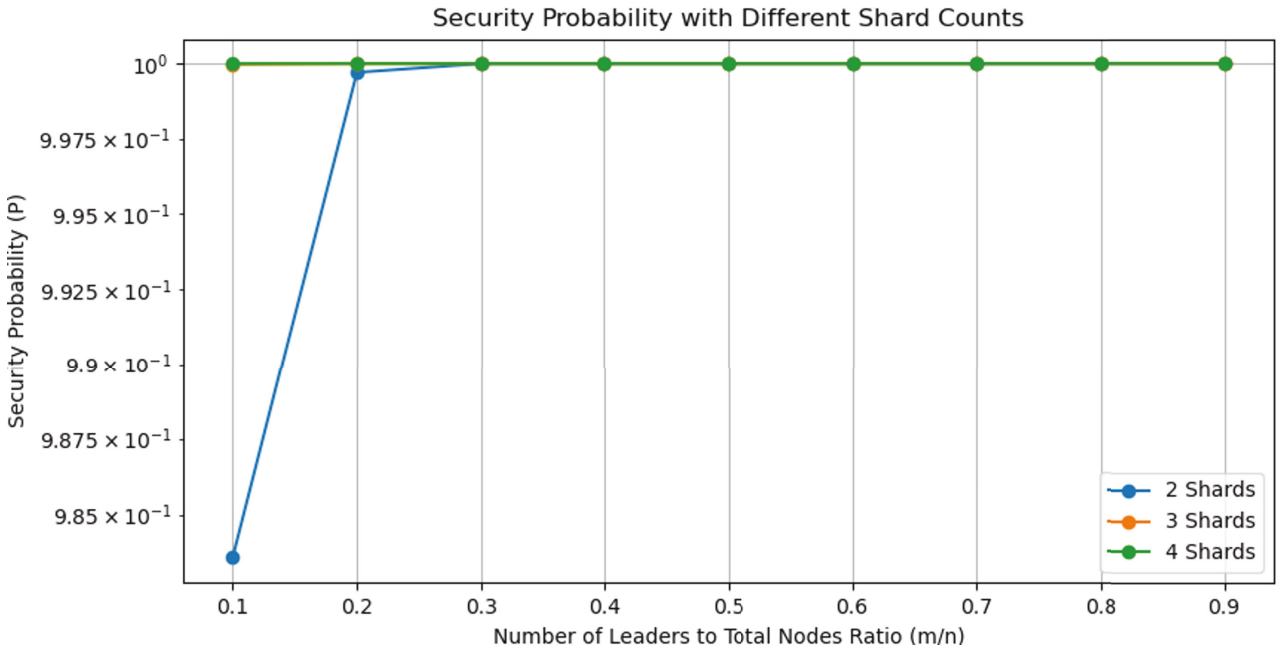


Figure 5.7: Security enhancements with varying leader counts and malicious scenarios, $n_k = 100$, $k = \{2, 3, 4\}$, $h = 30$.

5.4 Conclusion

This chapter proposed a novel framework designed for cross-shard smart con- tract transactions function calling, integrating the overlapping shards with an optimized PBFT consensus mechanism, referred to as xPBFT. By converting cross-shard transactions into intra-shard transactions, this framework streamlines the handling of smart contracts across shards. Experimental results show that the proposed framework reduces latency by approximately 40% compared to traditional blockchain framework, significantly enhancing transaction processing speeds. Moreover, the proposed framework maintained security, providing robust defenses against potential adversarial threats in scenarios involving multiple recipients.

Chapter 6

Conclusions and Future Work

6.1 Summary of Outcomes

This thiesis addresses the critical challenges outlined in the introduction: reducing the number of cross-shard transactions, enhancing security against 51% attacks, and improving system efficiency while minimizing latency. By leveraging a community-detection algorithm, a trust-based, DRL-based framework, and an overlapping shard structure, this thesis provides a comprehensive solution for optimizing sharded blockchain systems.

The first challenge, related to the high number of cross-shard transactions, is optimized through the adoption of a community-detection algorithm. This algorithm effectively clusters accounts that frequently interact, ensuring that they are allocated within the same shard. By reducing the number of cross-shard transactions, this approach minimizes the communication overhead and improves overall system throughput, contributing to enhanced scalability without compromising performance.

The second challenge, mitigating the risk of a strategical collusion attack, is tackled through a trust-based, DRL-based framework. This framework allocates nodes to shards based on their trustworthiness, which is determined by node behavior and interaction history. By prioritizing trusted nodes in shard assignment, this method significantly reduces the probability of an attacker gaining control of the majority of nodes within any single shard, thus bolstering the security of the system.

Lastly, the issue of high latency in cross-shard transactions is optimised through the imple-

mentation of an overlapping shard structure. By allowing nodes to participate across multiple shards, the framework enables more efficient handling of cross-shard dependencies, reducing the need for extensive inter-shard communication. This structure, combined with the optimized xPBFT consensus mechanism, demonstrated a significant reduction in transaction latency—up to 40%, while maintaining security and data consistency.

Together, these three approaches—community detection for reducing cross-shard transactions, a trust-based DRL framework for enhancing security, and an overlapping shard structure for reducing latency—work to address the intertwined challenges of scalability, security, and performance in blockchain systems. The resulting framework provides a robust foundation for advancing the capabilities of sharded blockchains, enabling more secure, scalable, and efficient solutions for real-world applications.

6.2 Recommendations & Future Work

In this thesis, the proposed framework addresses several critical challenges in blockchain sharding, there are areas that warrant further investigation. Future research should focus on the following:

Scalability in Larger Networks: The framework's effectiveness should be evaluated in larger and more complex blockchain networks. This would involve testing its performance in real-world scenarios with varying network conditions and transaction loads.

Integration with Emerging Technologies: Exploring the integration of the proposed framework with emerging technologies such as quantum computing, advanced cryptographic techniques, and machine learning algorithms could further enhance its capabilities and resilience.

Adaptive Strategies: Developing adaptive strategies for dynamic shard reconfiguration and leader selection could improve the framework's efficiency in response to changing network conditions and adversarial behaviors. This would involve designing algorithms that can autonomously adjust the shard structure and consensus parameters to optimize performance.

Cost-Effectiveness: Investigating methods to reduce the computational and storage costs associated with the overlapping shard framework is crucial. This includes optimizing the balance between security, performance, and resource utilization.

Cross-Domain Applications: Applying the proposed framework to various blockchain applications beyond financial transactions, such as supply chain management, healthcare, and IoT, could validate its versatility and practical utility.

Chapter 7

Publication List

[1] **Z. Zhang**, X. Wang, G. Yu, et al. "A community detection-based blockchain sharding scheme," International Conference on Blockchain (Blockchain–ICBC, 2022), Honolulu, 10–14.

[2] **Z. Zhang**, Y. Wang*, G. Yu, et al. "A Community-based Strategy for Blockchain Sharding: Enabling More Budget-friendly Transactions," 2023 IEEE International Conference on Blockchain (Blockchain). IEEE, 2023: 370-376.

[3] **Z. Zhang**, G. Yu, C. Sun*, et al. "TbDd: A new trust-based, DRL-driven framework for blockchain sharding in IoT," Computer Networks (CN), 244, 110343(2024).

[4] **Z. Zhang**, H. Yin, Y. Wang, et al. "Enabling Efficient Cross-Shard Smart Contract Calling via Overlapping," ProvSec 2024. (Accepted).

[5] Z. Sun, J. Feng, L. Yin, **Z. Zhang**, et al. "Fed-DFE: A Decentralized Function Encryption-Based Privacy-Preserving Scheme for Federated Learning," Computer Materials and Continua(CMC), 71, 1(2022).

[6] Y. Wang, F. Qi, **Z. Zhang**, et al. "Super-Resolution Reconstruction Algorithm for Terahertz Imaging below the Diffraction Limited," Chinese Physics B(CPB), 32(3), 1-5(2022).

[7] M. Zhang, Z. Sun , H. Li, B. Niu, F. Li, **Z. Zhang**, et al, "Go-Sharing: A Blockchain-based Privacy-Preserving Framework for Cross-Social Network Photo Sharing," IEEE Transactions on Dependable and Secure (TDSC) 20(5), 3572-3587.

Bibliography

[1] Natarajan Deepa, Quoc-Viet Pham, Dinh C Nguyen, Sweta Bhattacharya, B Prabadevi, Thippa Reddy Gadekallu, Praveen Kumar Reddy Maddikunta, Fang Fang, and Pubudu N Pathirana. A survey on blockchain for big data: Approaches, opportunities, and future directions. *Future Generation Computer Systems*, 131:209–226, 2022.

[2] Huaqun Guo and Xingjie Yu. A survey on blockchain technology and its security. *Blockchain: research and applications*, 3(2):100067, 2022.

[3] Muhammad Nasir Mumtaz Bhutta, Amir A Khwaja, Adnan Nadeem, Hafiz Farooq Ahmad, Muhammad Khurram Khan, Moataz A Hanif, Houbing Song, Majed Alshamari, and Yue Cao. A survey on blockchain technology: Evolution, architecture and security. *Ieee Access*, 9:61048–61073, 2021.

[4] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Xiangping Chen, and Huaimin Wang. Blockchain challenges and opportunities: A survey. *International journal of web and grid services*, 14(4):352–375, 2018.

[5] Gousia Habib, Sparsh Sharma, Sara Ibrahim, Imtiaz Ahmad, Shaima Qureshi, and Malik Ishfaq. Blockchain technology: benefits, challenges, applications, and integration of blockchain technology with cloud computing. *Future Internet*, 14(11):341, 2022.

[6] Philip Treleaven, Richard Gendal Brown, and Danny Yang. Blockchain technology in finance. *Computer*, 50(9):14–17, 2017.

[7] Jaspreet Kaur, Satish Kumar, Balkrishna E Narkhede, Marina Dabić, Ajay Pal Singh Rathore, and Rohit Joshi. Barriers to blockchain adoption for supply chain finance: the case of indian smes. *Electronic Commerce Research*, 24(1):303–340, 2024.

[8] Jayanth Rama Varma. Blockchain in finance. *Vikalpa*, 44(1):1–11, 2019.

[9] Thippa Reddy Gadekallu, MK Manoj, Neeraj Kumar, Saqib Hakak, Sweta Bhattacharya, et al. Blockchain-based attack detection on machine learning algorithms for iot-based e-health applications. *IEEE Internet of Things Magazine*, 4(3):30–33, 2021.

[10] Marko Hölbl, Marko Kompara, Aida Kamišalić, and Lili Nemec Zlatolas. A systematic review of the use of blockchain in healthcare. *Symmetry*, 10(10):470, 2018.

[11] Houtian Wang, Taotao Wang, Long Shi, Naijin Liu, and Shengli Zhang. A blockchain-empowered framework for decentralized trust management in internet of battlefield things. *Computer Networks*, page 110048, 2023.

[12] Xu Wang, Ping Yu, Guangsheng Yu, Xuan Zha, Wei Ni, Ren Ping Liu, and Y. Jay Guo. A high-performance hybrid blockchain system for traceable iot applications. In Joseph K. Liu and Xinyi Huang, editors, *Network and System Security*, pages 721–728, Cham, 2019. Springer International Publishing.

[13] Weikang Liu, Bin Cao, and Mugen Peng. Web3 technologies: Challenges and opportunities. *IEEE Network*, 2023.

[14] Gabriel Antonio F Rebello, Gustavo F Camilo, Lucas Airam C de Souza, Maria Potop-Butucaru, Marcelo Dias de Amorim, Miguel Elias M Campista, and Luís Henrique MK Costa. A survey on blockchain scalability: From hardware to layer-two protocols. *IEEE Communications Surveys & Tutorials*, 2024.

[15] Iqra Sadia Rao, ML Mat Kiah, M Muzaffar Hameed, and Zain Anwer Memon. Scalability of blockchain: a comprehensive review and future research direction. *Cluster Computing*, pages 1–24, 2024.

[16] Dodo Khan, Low Tang Jung, and Manzoor Ahmed Hashmani. Systematic literature review of challenges in blockchain scalability. *Applied Sciences*, 11(20):9372, 2021.

[17] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Satoshi Nakamoto*, 2008.

[18] Harald Vranken. Sustainability of bitcoin and blockchains. *Current opinion in environmental sustainability*, 28:1–9, 2017.

[19] Vitalik Buterin et al. Ethereum white paper. *GitHub repository*, 1:22–23, 2013.

[20] Elnaz Rabieinejad, Abbas Yazdinejad, Reza M Parizi, and Ali Dehghantanha. Generative adversarial networks for cyber threat hunting in ethereum blockchain. *Distributed Ledger Technologies: Research and Practice*, 2(2):1–19, 2023.

[21] Emre Yavuz, Ali Kaan Koç, Umut Can Çabuk, and Gökhan Dalkılıç. Towards secure e-voting using ethereum blockchain. In *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*, pages 1–7. IEEE, 2018.

[22] Rui Zhang, Rui Xue, and Ling Liu. Security and privacy on blockchain. *ACM Computing Surveys (CSUR)*, 52(3):1–34, 2019.

[23] Anamika Chauhan, Om Prakash Malviya, Madhav Verma, and Tejinder Singh Mor. Blockchain and scalability. In *2018 IEEE international conference on software quality, reliability and security companion (QRS-C)*, pages 122–128. IEEE, 2018.

[24] James C Corbett, Jeffrey Dean, Michael Epstein, Andrew Fikes, Christopher Frost, Jeffrey John Furman, Sanjay Ghemawat, Andrey Gubarev, Christopher Heiser, Peter Hochschild, et al. Spanner: Google's globally distributed database. *ACM Transactions on Computer Systems (TOCS)*, 31(3):1–22, 2013.

[25] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. Rapidchain: Scaling blockchain via full sharding. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 931–948, 2018.

[26] Gang Wang, Zhijie Jerry Shi, Mark Nixon, and Song Han. Sok: Sharding on blockchain. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pages 41–61, 2019.

[27] Zicong Hong, Song Guo, Enyuan Zhou, Wuhui Chen, Huawei Huang, and Albert Zomaya. Gridb: Scaling blockchain database via sharding and off-chain cross-shard mechanism. *arXiv preprint arXiv:2407.03750*, 2024.

[28] PeiYun Zhang, WeiFeng Guo, ZiJie Liu, MengChu Zhou, Bo Huang, and Khaled Sedraoui. Optimized blockchain sharding model based on node trust and allocation. *IEEE Transactions on Network and Service Management*, 20(3):2804–2816, 2023.

[29] Xu Wang, Guangsheng Yu, Ren Ping Liu, Jian Zhang, Qiang Wu, Steven W Su, Ying He, Zongjian Zhang, Litao Yu, Taoping Liu, et al. Blockchain-enabled fish provenance

and quality tracking system. *IEEE Internet of Things Journal*, 9(11):8130–8142, 2021.

[30] Cornelius C Agbo, Qusay H Mahmoud, and J Mikael Eklund. Blockchain technology in healthcare: a systematic review. In *Healthcare*, volume 7, page 56. MDPI, 2019.

[31] Fabian Schär. Decentralized finance: On blockchain-and smart contract-based financial markets. *FRB of St. Louis Review*, 2021.

[32] Arthur Gervais, Ghassan O Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 3–16, 2016.

[33] Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August*, 19(1), 2012.

[34] Gideon Greenspan et al. Multichain private blockchain-white paper. *URl: http://www. multichain. com/download/MultiChain-White-Paper. pdf*, 85, 2015.

[35] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference*, pages 1–15, 2018.

[36] Richard Gendal Brown. The corda platform: An introduction. *Retrieved*, 27:2018, 2018.

[37] Sheping Zhai, Yuanyuan Yang, Jing Li, Cheng Qiu, and Jiangming Zhao. Research on the application of cryptography on the blockchain. In *Journal of Physics: Conference Series*, volume 1168, page 032077. IOP Publishing, 2019.

[38] Shi-Qin Zeng, Ru Huo, Tao Huang, Jiang Liu, Shuo Wang, and Wei Feng. Survey of blockchain: principle, progress and application. *Journal on Communications*, 41(1):134–151, 2020.

[39] Alex Norta. Designing a smart-contract application layer for transacting decentralized autonomous organizations. In *Advances in Computing and Data Sciences: First International Conference, ICACDS 2016, Ghaziabad, India, November 11-12, 2016, Revised Selected Papers 1*, pages 595–604. Springer, 2017.

[40] Bin Cao, Zixin Wang, Long Zhang, Daquan Feng, Mugen Peng, Lei Zhang, and Zhu Han. Blockchain systems, technologies, and applications: A methodology perspective. *IEEE Communications Surveys & Tutorials*, 25(1):353–385, 2022.

[41] Yang Xinyi, Zhang Yi, and Yulin He. Technical characteristics and model of blockchain. In *2018 10th international Conference on communication Software and networks (ICCSN)*, pages 562–566. IEEE, 2018.

[42] Changjing Wang, Huiwen Jiang, Jingshan Zeng, YU Min, Qing Huang, and Zhengkang Zuo. A review of blockchain layered architecture and technology application research. *Wuhan University Journal of Natural Sciences*, 26(5):14, 2021.

[43] Rong Han, Zheng Yan, Xueqin Liang, and Laurence T Yang. How can incentive mechanisms and blockchain benefit with each other? a survey. *ACM Computing Surveys*, 55(7):1–38, 2022.

[44] Maher Alharby and Aad Van Moorsel. Blocksim: a simulation framework for blockchain systems. *ACM SIGMETRICS Performance Evaluation Review*, 46(3):135–138, 2019.

[45] Yong Yuan and Fei-Yue Wang. Towards blockchain-based intelligent transportation systems. In *2016 IEEE 19th international conference on intelligent transportation systems (ITSC)*, pages 2663–2668. IEEE, 2016.

[46] Cristian Lepore, Michela Ceria, Andrea Visconti, Udai Pratap Rao, Kaushal Arvindbhai Shah, and Luca Zanolini. A survey on blockchain consensus with a performance comparison of pow, pos and pure pos. *Mathematics*, 8(10):1782, 2020.

[47] Noureddine Lasla, Lina Al-Sahan, Mohamed Abdallah, and Mohamed Younis. Green-pow: An energy-efficient blockchain proof-of-work consensus algorithm. *Computer Networks*, 214:109118, 2022.

[48] Yaqin Wu, Pengxin Song, and Fuxin Wang. Hybrid consensus algorithm optimization: A mathematical method based on pos and pbft and its application in blockchain. *Mathematical Problems in Engineering*, 2020(1):7270624, 2020.

[49] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual international cryptology conference*, pages 357–388. Springer, 2017.

[50] Fahad Saleh. Blockchain without waste: Proof-of-stake. *The Review of financial studies*, 34(3):1156–1190, 2021.

[51] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OsDI*, volume 99, pages 173–186, 1999.

[52] Stefano De Angelis, Leonardo Aniello, Roberto Baldoni, Federico Lombardi, Andrea Margheri, Vladimiro Sassone, et al. Pbft vs proof-of-authority: Applying the cap theorem to permissioned blockchain. In *CEUR workshop proceedings*, volume 2058. CEUR-WS, 2018.

[53] Wenyu Li, Chenglin Feng, Lei Zhang, Hao Xu, Bin Cao, and Muhammad Ali Imran. A scalable multi-layer pbft consensus for blockchain. *IEEE Transactions on Parallel and Distributed Systems*, 32(5):1146–1160, 2020.

[54] Narayan DG, Naveen Arali, and R Tejas. Dposeb: Delegated proof of stake with exponential backoff consensus algorithm for ethereum blockchain. *Computer Science Journal of Moldova*, 32(2), 2024.

[55] Till Neudecker and Hannes Hartenstein. Network layer aspects of permissionless blockchains. *IEEE Communications Surveys & Tutorials*, 21(1):838–857, 2018.

[56] Volkan Dedeoglu, Raja Jurdak, Guntur D Putra, Ali Dorri, and Salil S Kanhere. A trust architecture for blockchain in iot. In *Proceedings of the 16th EAI international conference on mobile and ubiquitous systems: computing, networking and services*, pages 190–199, 2019.

[57] Tao Zhang, Bingyu Li, Yan Zhu, Tianxu Han, and Qianhong Wu. Covert channels in blockchain and blockchain based covert communication: Overview, state-of-the-art, and future directions. *Computer Communications*, 205:136–146, 2023.

[58] Hussam Saeed Musa, Moez Krichen, Adem Alpaslan Altun, and Meryem Ammi. Survey on blockchain-based data storage security for android mobile applications. *Sensors*, 23(21):8749, 2023.

[59] Mingchao Yu, Saeid Sahraei, Songze Li, Salman Avestimehr, Sreeram Kannan, and Pramod Viswanath. Coded merkle tree: Solving data availability attacks in blockchains.

In *International Conference on Financial Cryptography and Data Security*, pages 114–134. Springer, 2020.

[60] Ana Reyna, Cristian Martín, Jaime Chen, Enrique Soler, and Manuel Díaz. On blockchain and its integration with iot. challenges and opportunities. *Future generation computer systems*, 88:173–190, 2018.

[61] Ali Dorri, Salil S Kanhere, and Raja Jurdak. Towards an optimized blockchain for iot. In *Proceedings of the second international conference on Internet-of-Things design and implementation*, pages 173–178, 2017.

[62] Ruoting Xiong, Wei Ren, Xiaohan Hao, Jie He, and Kim-Kwang Raymond Choo. Bdim: A blockchain-based decentralized identity management scheme for large scale internet of things. *IEEE Internet of Things Journal*, 2023.

[63] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. A secure sharding protocol for open blockchains. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 17–30, 2016.

[64] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ewa Syta, and Bryan Ford. Omniledger: A secure, scale-out, decentralized ledger via sharding. In *2018 IEEE symposium on security and privacy (SP)*, pages 583–598. IEEE, 2018.

[65] Jiaping Wang and Hao Wang. Monoxide: Scale out blockchains with asynchronous consensus zones. In *NSDI*, volume 2019, pages 95–112, 2019.

[66] Web3.university. Ethereum Sharding. online, accessed, 2022. https://www.web3.university/article/ethereum-sharding-an-introduction-to-blockchain-sharding.

[67] Abdelatif Hafid, Abdelhakim Senhaji Hafid, and Mustapha Samih. Scaling blockchains: A comprehensive survey. *IEEE access*, 8:125244–125262, 2020.

[68] Zilliqa Team et al. The zilliqa technical whitepaper. *Retrieved September*, 16:2019, 2017.

[69] Zicong Hong, Song Guo, Peng Li, and Wuhui Chen. Pyramid: A layered sharding blockchain system. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, pages 1–10. IEEE, 2021.

[70] V. Buterin. Ethereum sharding faq. Online, 2024. Accessed: Sep. 15, 2024.

[71] Ittai Abraham, Dahlia Malkhi, Kartik Nayak, Ling Ren, and Alexander Spiegelman. Solidus: An incentive-compatible cryptocurrency based on permissionless byzantine consensus. *CoRR, abs/1612.02916*, 2016.

[72] 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 *25th usenix security symposium (usenix security 16)*, pages 279–296, 2016.

[73] Qiheng Zhou, Huawei Huang, Zibin Zheng, and Jing Bian. Solutions to Scalability of blockchain: A survey. *IEEE Access*, 8:16440–16455, 2020.

[74] Wessel Reijers, Iris Wuisman, Morshed Mannan, Primavera De Filippi, Christopher Wray, Vienna Rae-Looi, Angela Cubillos Vélez, and Liav Orgad. Now the code runs itself: On-chain and off-chain governance of blockchain technologies. *Topoi*, 40:821–831, 2021.

[75] Thomas Hepp, Matthew Sharinghousen, Philip Ehret, Alexander Schoenhals, and Bela Gipp. On-chain vs. off-chain storage for supply-and blockchain integration. *it-Information Technology*, 60(5-6):283–291, 2018.

[76] Soohyeong Kim, Yongseok Kwon, and Sunghyun Cho. A survey of scalability solutions on blockchain. In *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 1204–1207. IEEE, 2018.

[77] Gavin Wood. Polkadot: Vision for a heterogeneous multi-chain framework. *White paper*, 21(2327):4662, 2016.

[78] Kwon Jae and E Buchman. Cosmos: a network of distributed ledgers. *URL https://github. com/cosmos/cosmos/blob/7814a6cfe53873eaec1c478971d290c08ce4db7a/WHITEPAPER. md*, 2020.

[79] Bip152. Online. Accessed: Sep. 15, 2024.

[80] Gleb Naumenko, Gregory Maxwell, Pieter Wuille, Alexandra Fedorova, and Ivan Beschastnikh. Bandwidth-efficient transaction relay for bitcoin. *arXiv preprint arXiv:1905.10518*, 2019.

[81] Nakul Chawla, Hans Walter Behrens, Darren Tapp, Dragan Boscovic, and K Selçuk Candan. Velocity: Scalability improvements in block propagation through rateless era-

sure coding. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 447–454. IEEE, 2019.

[82] Elias Rohrer and Florian Tschorsch. Kadcast: A structured approach to broadcast in blockchain networks. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pages 199–213, 2019.

[83] Petar Maymounkov and David Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In *International workshop on peer-to-peer systems*, pages 53–65. Springer, 2002.

[84] Uri Klarman, Soumya Basu, Aleksandar Kuzmanovic, and Emin Gün Sirer. bloxroute: A scalable trustless blockchain distribution network whitepaper. *IEEE Internet of Things Journal*, 2018.

[85] Guangsheng Yu, Xu Wang, Kan Yu, Wei Ni, J Andrew Zhang, and Ren Ping Liu. Survey: Sharding in blockchains. *IEEE Access*, 8:14155–14181, 2020.

[86] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, et al. On scaling decentralized blockchains: (a position paper). In *Financial Cryptography and Data Security: FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers 20*, pages 106–125. Springer, 2016.

[87] Eric Lombrozo, Johnson Lau, and Pieter Wuille. Segregated witness (consensus layer). *Bitcoin Core Develop. Team, Tech. Rep. BIP*, 141, 2015.

[88] Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. In *Financial Cryptography and Data Security: 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers 19*, pages 507–527. Springer, 2015.

[89] Tong Zhou, Xiaofeng Li, and He Zhao. Dlattice: A permission-less blockchain based on dpos-ba-dag consensus for data tokenization. *IEEE Access*, 7:39273–39287, 2019.

[90] Laizhong Cui, Shu Yang, Ziteng Chen, Yi Pan, Mingwei Xu, and Ke Xu. An efficient and compacted dag-based blockchain protocol for industrial internet of things. *IEEE Transactions on Industrial Informatics*, 16(6):4134–4145, 2019.

[91] Jeff Garzik. Block size increase to 2mb. *Bitcoin improvement proposal*, 102, 2015.

[92] IOTA Foundation. What is iota? `https://www.iota.org/get-started/what-is-iota`, n.d. Accessed: Sep. 15, 2024.

[93] Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *arXiv preprint arXiv:1710.09437*, 2017.

[94] Yixin Li, Bin Cao, Mugen Peng, Long Zhang, Lei Zhang, Daquan Feng, and Jihong Yu. Direct acyclic graph-based ledger for internet of things: Performance and security analysis. *IEEE/ACM Transactions on Networking*, 28(4):1643–1656, 2020.

[95] Weikang Liu, Bin Cao, Mugen Peng, and Bo Li. Distributed and parallel blockchain: Towards a multi-chain system with enhanced security. *IEEE Transactions on Dependable and Secure Computing*, 2024.

[96] Joseph Poon and Vitalik Buterin. Plasma: Scalable autonomous smart contracts. *White paper*, pages 1–47, 2017.

[97] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: scalable off-chain instant payments (2016), 2016.

[98] R. Network. Fast, cheap, scalable token transfers for ethereum. Online, 2018. Accessed: Sep. 15, 2024.

[99] S. D. Lerner. *RSK*, 2020. Accessed: Sep. 15, 2024.

[100] Ethan Buchman. *Tendermint: Byzantine fault tolerance in the age of blockchains*. PhD thesis, University of Guelph, 2016.

[101] Andrew Miller, Iddo Bentov, Surya Bakshi, Ranjit Kumaresan, and Patrick McCorry. Sprites and state channels: Payment networks that go faster than lightning. In *International conference on financial cryptography and data security*, pages 508–526. Springer, 2019.

[102] Ethhub. ZK-Rollups. Online, 2021. Accessed: Sep. 15, 2024.

[103] Andreas M Antonopoulos and Gavin Wood. *Mastering ethereum: building smart contracts and dapps*. O'reilly Media, 2018.

[104] A Secure. The zilliqa project: A secure, scalable blockchain platform. 2018.

[105] Mustafa Al-Bassam, Alberto Sonnino, Shehar Bano, Dave Hrycyszyn, and George Danezis. Chainspace: A sharded smart contracts platform. *arXiv preprint arXiv:1708.03778*, 2017.

[106] Brian W Kernighan and Shen Lin. An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal*, 49(2):291–307, 1970.

[107] Zixu Zhang, Xu Wang, Guangsheng Yu, et al. A community detection-based blockchain sharding scheme. In *Blockchain–ICBC 2022: 5th International Conference, Held as part of the Services Conference Federation, SCF 2022, Honolulu, HI, USA, December 10–14, 2022, Proceedings*, pages 78–91. Springer, 2022.

[108] Jusik Yun, Yunyeong Goh, and Jong-Moon Chung. Trust-based shard distribution scheme for fault-tolerant shard blockchain networks. *IEEE Access*, 7:135164–135175, 2019.

[109] Chenyu Huang, Zeyu Wang, Huangxun Chen, Qiwei Hu, Qian Zhang, Wei Wang, and Xia Guan. Repchain: A reputation-based secure, fast, and high incentive blockchain system via sharding. *IEEE Internet of Things Journal*, 8(6):4291–4304, 2020.

[110] Mengting Liu, F Richard Yu, Yinglei Teng, Victor CM Leung, and Mei Song. Performance optimization for blockchain-enabled industrial internet of things (iiot) systems: A deep reinforcement learning approach. *IEEE Transactions on Industrial Informatics*, 15(6):3559–3570, 2019.

[111] Chi Harold Liu, Qiuxia Lin, and Shilin Wen. Blockchain-enabled data collection and sharing for industrial iot with deep reinforcement learning. *IEEE Transactions on Industrial Informatics*, 15(6):3516–3526, 2018.

[112] Chao Qiu, Haipeng Yao, F Richard Yu, Chunxiao Jiang, and Song Guo. A service-oriented permissioned blockchain for the internet of things. *IEEE Transactions on Services Computing*, 13(2):203–215, 2019.

[113] Chao Qiu, F Richard Yu, Haipeng Yao, Chunxiao Jiang, Fangmin Xu, and Chenglin Zhao. Blockchain-based software-defined industrial internet of things: A dueling deep q-learning approach. *IEEE Internet of Things Journal*, 6(3):4627–4639, 2018.

[114] Jusik Yun, Yunyeong Goh, and Jong-Moon Chung. Dqn-based optimization framework for secure sharded blockchain systems. *IEEE Internet of Things Journal*, 8(2):708–722,

2020.

[115] Zhaoxin Yang, Ruizhe Yang, F Richard Yu, Meng Li, Yanhua Zhang, and Yinglei Teng. Sharded blockchain for collaborative computing in the internet of things: Combined of dynamic clustering and deep reinforcement learning approach. *IEEE Internet of Things Journal*, 9(17):16494–16509, 2022.

[116] Zicong Hong, Song Guo, and Peng Li. Scaling blockchain via layered sharding. *IEEE Journal on Selected Areas in Communications*, 40(12):3575–3588, 2022.

[117] Yizhong Liu, Jianwei Liu, Yiming Hei, Yu Xia, and Qianhong Wu. A secure cross-shard view-change protocol for sharding blockchains. In *Information Security and Privacy: 26th Australasian Conference, ACISP 2021, Virtual Event, December 1–3, 2021, Proceedings 26*, pages 372–390. Springer, 2021.

[118] Alberto Sonnino, Shehar Bano, Mustafa Al-Bassam, and George Danezis. Replay attacks and defenses against cross-shard consensus in sharded distributed ledgers. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 294–308. IEEE, 2020.

[119] Xinjie Yang, Tianyi Xu, Fengbiao Zan, Tao Ye, Zhaofang Mao, and Tie Qiu. An overlapping self-organizing sharding scheme based on drl for large-scale iiot blockchain. *IEEE Internet of Things Journal*, 2023.

[120] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[121] etherscan. etherscan. online, accessed, 2020. https://etherscan.io/address/0x7Be8076f4EA4A4AD08075C2508e481d6C946D12b.

[122] Ankr.com. Build on Ethereum With Instant RPC Endpoint. online, accessed, 2022. Available at: https://www.ankr.com/protocol/public/eth/.

[123] Kaggle.com. Ethereum Blockchain. online, accessed, 2023. Available at: https://www.kaggle.com/datasets/bigquery/ethereum-blockchain.

[124] Allen Clement, Edmund Wong, Lorenzo Alvisi, Mike Dahlin, Mirco Marchetti, et al. Making byzantine fault tolerant systems tolerate byzantine faults. In *Proceedings of the*

6th USENIX symposium on Networked systems design and implementation. The USENIX Association, 2009.

[125] Runchao Han, Jiangshan Yu, and Ren Zhang. Analysing and improving shard allocation protocols for sharded blockchains. In *Proceedings of the 4th ACM Conference on Advances in Financial Technologies*, pages 198–216, 2022.

[126] Mengya Li and Yang Qin. Scaling the blockchain-based access control framework for iot via sharding. In *ICC 2021 - IEEE International Conference on Communications*, pages 1–6, 2021.

[127] Kamalani Aiyar, Malka N Halgamuge, and Azeem Mohammad. Probability distribution model to analyze the trade-off between scalability and security of sharding-based blockchain networks. In *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–6. IEEE, 2021.

[128] Abdelatif Hafid, Abdelhakim Senhaji Hafid, and Mustapha Samih. A novel methodology-based joint hypergeometric distribution to analyze the security of sharded blockchains. *IEEE Access*, 8:179389–179399, 2020.

[129] Zhihua Cui, Zhaoyu Xue, Yanan Ma, Xingjuan Cai, and Jinjun Chen. A many-objective optimized sharding scheme for blockchain performance improvement in end-edge enabled internet of things. *IEEE Internet of Things Journal*, 2023.