International Journal on Semantic Web and Information Systems
Volume 19 • Issue 1

A Scalable Sharding Protocol Based on Cross-Shard Dynamic Transaction Confirmation for Alliance Chain in Intelligent Systems

Nigang Sun, School of Microelectronics and Control Engineering, Changzhou University, China

Junlong Li, School of Computer Science and Artificial Intelligence, Changzhou University, China*

Yining Liu, School of Computer and Information Security, Guilin University of Electronic Technology, China


https://orcid.org/0000-0002-6487-7595

Varsha Arya, Department of Business Administration, Asia University, Taiwan, & Center for Interdisciplinary Research, University of Petroleum and Energy Studies (UPES), Dehradun, India, & Lebanese American University, Beirut, Lebanon, & Chandigarh University, Chandigarh, India

ABSTRACT

Applying sharding protocol to address scalability challenges in alliance chain is popular. However, inevitable cross-shard transactions significantly hamper performance even at low ratios, negating scalability benefits when they dominate as shard scale grows. This article proposes a new sharding protocol suitable for alliance chain that reduces cross-shard transaction impact, improving system performance. It adopts a directed acyclic graph ledger, enabling parallel transaction processing, and employs dynamic transaction confirmation consensus for simplicity. The protocol's sharding process and node score mechanism can deter malicious behavior. Experiments show that compared with mainstream sharding protocols, the protocol performs better when affected by cross-shard transactions. Moreover, its throughput has shown improvement compared to high-performance protocols without cross-shard transactions. This solution suits systems requiring high throughput and reliability, maintaining a stable performance advantage even as cross-shard transactions increase to the usual maximum ratio.

KEYWORDS

Alliance Chain, Blockchain, Blockchain Scaling, Consensus Mechanism, Cross-Shard Transaction, Scalability, Sharding Protocol

INTRODUCTION

Alliance chain is a type of blockchain that offers decentralization and node management capabilities, garnering significant attention for its widespread adoption in various industries, such as Internet of things (IoT), smart city, big data finance, and healthcare (Y. Li et al., 2021). However, with the

DOI: 10.4018/IJSWIS.333063 *Corresponding Author

emergence of scenarios involving increased nodes and heightened computational demands, such as in the case of data-intensive IoT (Lv et al., 2022; Memos et al., 2018; Plageras et al., 2018; Raj & Pani, 2022) and complex intelligent systems (Afify et al., 2022; Fatemidokht et al., 2021; D. Li et al., 2019; Sharma et al., 2022), alliance chains face notable scalability challenges, such as communication congestion and reduced throughput (Dinh et al., 2017). This phenomenon primarily arises from the requirement for all transactions in the blockchain structure to utilize nodes' computing and storage resources (Du et al., 2021; C. Li et al., 2021), leading to substantial wastage of time and space (Qi et al., 2020). Sharding protocol improves scalability by partitioning different responsibilities and resources to different sets of nodes (Yu et al., 2020). In 2016, Elastico (Luu et al., 2016) pioneered by integrating the sharding protocol with blockchain, thereby proposing an innovation that prevents each participating node from incurring redundant communication and computation overhead. Sharding protocol has gradually become the prevailing on-chain solution in the subsequent development, and many studies (Al-Bassam et al., 2017; Hellings & Sadoghi, 2023; Hong et al., 2021; Huang et al., 2022; Kokoris-Kogias et al., 2018; Zamani et al., 2018) have proved its effectiveness and reflected its advantages. In a sharding protocol, nodes participating in the consensus process are divided into different shards, each responsible for generating and maintaining a specific portion of the alliance chain. As a result, the entire system comprises multiple shards, essentially representing parallel chains within a single network. Within each shard, most communication predominantly occurs among the nodes residing there. Transactions are distributed to shards based on specific rules. A vital category of transactions, known as cross-shard transactions, necessitates the involvement of multiple shards (Wang & Raviv, 2021). While cross-shard transactions do not encompass all transactions, sharding protocols degenerate the blockchain into multiple independent systems (Das et al., 2020) without this support. However, processing cross-shard transactions requires participation in complex communication and coordination protocols between shards (Hong et al., 2021). Unfortunately, many existing solutions for handling cross-shard transactions rely on protocols that require splitting such transactions into several sub-transactions, which complicates the process by calling the consensus algorithm multiple times. Even in a typical setup where the workload comprises a low percentage of cross-shard transactions, the performance experiences a substantial decline due to the processing flow (Amiri et al., 2019). The significance of addressing cross-shard transactions becomes evident as their proportion increases with the number of shards. For example, cross-shard transactions can account for 99.98% of the total transaction volume in a scenario with 16 shards (Zamani et al., 2018). Consequently, the way these transactions are handled results in significant processing overhead, negating the scalability enhancements provided by the sharding protocol (Deepa et al., 2022). Addressing these challenges is essential to unlock the full potential of sharding protocols, enhancing their efficiency, scalability, and applicability across various domains.

Researchers have improved the sharding protocol to reduce consensus calls and message complexity between shards, thereby providing valuable assistance in achieving efficient cross-shard transaction processing (Liu et al., 2023). In Attested HyperLedger (AHL), Dang et al. (2019) reduced the number of nodes needed in each shard by enhancing the consensus protocol and randomly assigning nodes to shards. The system proposed an additional set of nodes acting as coordinators, employing classic two-phase commit (2PC) and two-phase locking (2PL) protocols to handle cross-shard transactions. The application is limited by insufficient scalability and an unbalanced workload (Asgaonkar, 2022). Amiri et al. (2019) proposed Sharper, a sharding protocol for permissioned blockchain. Each transaction is treated as a block, in this protocol, and two flattened consensus protocols are employed to enable parallel processing of transactions across different shards. Sharper still faces challenges in the complexity of consensus algorithms and issues related to data access efficiency (Hashim et al., 2022). ZyconChain (Sohrabi & Tari, 2020) is a scalable and versatile sharding blockchain. It utilizes various consensus algorithms to create blocks, with each algorithm possessing unique characteristics that render it suitable for specific types of blocks

(Khor et al., 2023). The main disadvantages that hinder the deployment of this protocol are its cross-shard protocol adopting the complex and difficult-to-implement view change subprotocol and the honest client assumption (Sohrabi et al., 2022). The K-prototype clustering byzantine fault tolerance algorithm (KBFT) (Wu et al., 2023) adopts a consensus mechanism that combines Boneh-Lynn-Shacham multisignature and byzantine fault tolerance (BFT) algorithm, enabling swift transaction confirmation within shards. This scheme eliminates the design of state sharding, and transactions are managed by all shards, making it crucial to solving its node storage issue in practical scenarios (Tan et al., 2023). IGD-ScoreChain (Mehraein & Nourmohammadi, 2023) is a lightweight and scalable sharding protocol tailored for blockchain-based IoT. It employs intelligent routing algorithms to delegate transaction processing to cloud nodes, efficiently alleviating the heavy computing load associated with cross-shard transactions in the fog layer. The storage of blockchain data in the cloud and its reliance on cloud layer nodes for cross-shard transactions introduce centralization risks, and network failures can cause system operations to be affected (Baranwal et al., 2023). In traditional cross-shard transaction processing methods, the widely adopted 2PC protocol often necessitates node asset locking during cross-shard transaction processing to maintain transaction consistency. This locking results in nodes holding resources for extended periods during transaction processing, leading to increased transaction latency and decreased system throughput. Furthermore, achieving cross-shard transactions relies on BFT consensus algorithms, which involve multiple rounds of voting and intricate message exchanges, consequently elevating communication overhead. Despite advancements in sharding protocol research to optimize alliance chain performance, inefficiencies persist in cross-shard transaction processing. As cross-shard transactions become more frequent (Hong et al., 2021), such as more shards due to increased devices (Guebli & Belkhir, 2021) and the uneven distribution of resources (Tiwari & Garg, 2022), the bandwidth and time costs required for communication rise sharply. This challenge poses a considerable obstacle to implementing sharding protocols for high-performance alliance chains, as it mitigates the scalability enhancements protocols offer, rendering them less apparent.

This paper proposes a scalable sharding protocol for the alliance chain. The protocol accomplishes the parallel process of transactions without overlapping shards by structuring the alliance chain ledger as a directed acyclic graph (DAG), where each shard solely handles transactions within the path it maintains, thus avoiding the considerable communication and conflict resolution overhead associated with cross-shard transactions. The protocol employs the dynamic transaction confirmation consensus mechanism suitable for alliance chains, which achieves transaction consensus independently and in parallel within each shard, significantly enhancing transaction processing efficiency and reducing the conflict of cross-shard messages. The sharding process first clusters and then classifies to form each shard, thereby preventing nodes from colluding with malicious behavior. The protocol calculates a behavior score as one of the node attributes based on the node's activity history in transaction processing. This mechanism is to ensure the consistent operation of the alliance chain. In general, the protocol markedly alleviates the impact of cross-shard transactions on alliance chain performance while enhancing transaction processing efficiency.

Simulation experiments show that, unlike mainstream sharding protocols, this protocol does not lead to a sharp drop in performance as the proportion of cross-shard transactions increases to the typical settings. It has higher throughput, and the latency of 10% cross-shard transactions is 56% and 36% lower than AHL and Sharper. Moreover, the protocol's throughput regarding no cross-shard transaction has shown improvements, with a 9% increase compared to KBFT and a 6% increase compared to IGD-ScoreChain. The protocol the authors propose in this paper is a highly effective solution for enhancing the performance of alliance chains in new computing scenarios, such as intelligent systems with numerous devices that require high scalability and involve a large number of cross-shard transactions.

RELATED CONCEPTS

This section provides a comprehensive explanation for the key concepts and applications of alliance chains and sharding protocols, along with a discussion of their advantages and disadvantages in various scenarios.

Alliance Chain

Alliance chain is a permissioned blockchain network involving authorized entities to manage and maintain data and transactions collaboratively (X. Li et al., 2020). Unlike the public chain, the participants of the alliance chain establish a cooperative relationship, jointly make decisions, and manage the operation of the chain.

The primary purpose of the alliance chain is to create a credible collaboration platform within a specific industry or organization. It can be used for various purposes, such as payment and settlement between financial institutions, supply chain management, sharing of medical records, and data exchange between government agencies (Yang et al., 2008). By providing decentralized, transparent, traceable, and secure transaction records, alliance chains enhance participant trust and enable efficient data sharing and automation of business processes (Stuart et al., 2007).

In the alliance chain, participants can join the network through authorized identity verification and jointly maintain the operation and security of the blockchain. The consensus algorithm of the alliance chain may vary depending on the design and purpose of the network (De Angelis et al., 2018). Usually, a consensus mechanism with higher performance is selected, such as proof of authority (Al Asad et al., 2020), BFT (Gao et al., 2021) or proof of stake (Ge et al., 2022). These consensus mechanisms usually achieve high throughput and low latency, and are suitable for scenarios with fewer participants in the alliance chain (Y. Chen et al., 2022).

Although the security of the alliance chain is relatively high due to the involvement of authorized entities, there is still a certain degree of centralization risk, compared with a fully decentralized public chain (Zhu et al., 2019). In addition, the governance and consensus mechanisms of the alliance chain face scalability challenges when the number of nodes and transactions rises, which leads to slower and more complicated transaction confirmation processes. Despite these limitations, the alliance chain is a valuable blockchain solution in specific cooperation scenarios. It provides trusted data sharing and efficient management of business processes, making it suitable for industries and organizations that prioritize security, collaboration, and data integrity (F. Wang et al., 2021).

Sharding Protocol

Network sharding, transaction sharding, and state sharding are state-of-the-art mechanisms for implementing blockchain sharding protocols in the modern world (Zhou et al., 2020). Network sharding divides the entire blockchain network into multiple shards so that different shards can process some transactions in the entire blockchain at the same time. Transaction sharding distributes transactions to different shards and allows them to execute concurrently. State sharding separates the entire ledger and saves it in shards, reducing the network node storage burden.

Sharding protocols can be decomposed into the following phases: Shard configuration, intra-shard consensus, cross-shard consensus, and reconfiguration (G. Wang et al., 2019). The shard configuration phase determines which shards a node belongs to and which transactions each shard will handle (Dang et al., 2019; Kokoris-Kogias et al., 2018; Luu et al., 2016). After completing the previous step, nodes in the same shard pass messages according to the internal consensus protocol to reach a consensus on transactions within the shard. The cross-shard consensus protocol uses transaction-related shards as the basic unit for processing cross-shard transactions. Since the design of shards is relatively independent, implementing cross-shard transactions involves coordination and communication between multiple shards (Kokoris-Kogias et al., 2018; Zamani et al., 2018). The reconfiguration step shuts down nodes and swaps to other shards after a period to maintain each shard's integrity and avoid attacks from

slowly adapting adversaries (Luu et al., 2016; Zamani et al., 2018). Figure 1 shows the layout of a blockchain based on the sharding protocol. It includes network sharding, transaction sharding, and state sharding, also called complete sharding. Although experimental setups or approaches to verify different techniques may vary, throughput and latency are standard metrics for evaluating protocol performance (G. Wang et al., 2019).

A SCALABLE SHARDING PROTOCOL FOR ALLIANCE CHAIN

This section provides a detailed description of the components and mechanisms of the sharding protocol, and their design features receive full consideration to address the challenges. The protocol constructs the alliance chain ledger as a DAG, realizing that each shard only processes transactions within its path. It uses a dynamic transaction confirmation algorithm to establish consensus within a single shard and cross shards, thereby directly improving the efficiency of transaction verification in the alliance chain. Furthermore, the protocol adopts a sharding configuration combining clustering and subsequent classification, complementing a node behavior score mechanism. This fusion is harnessed to bolster the validity of nodes tasked with upholding the consensus process. Figure 2 shows the architecture of the complete protocol and reflects the connections between different components. Among them, 1 and 2 are the alliance chain network configuration, 3 is the ledger implementation, 4 is the dynamic confirmation threshold setting, 5, 6, and 7 are transaction consensus, and 8 is the operation of the score mechanism.

Figure 1. The layout of a blockchain based on the sharding protocol (Note: A node network has only one unique master node. Each shard has a leader node and multiple consensus nodes. Intra-shard consensus and cross-shard consensus are distinct methods for block generation. Each chain of shard consists of blocks containing only transactions related to the shard.)

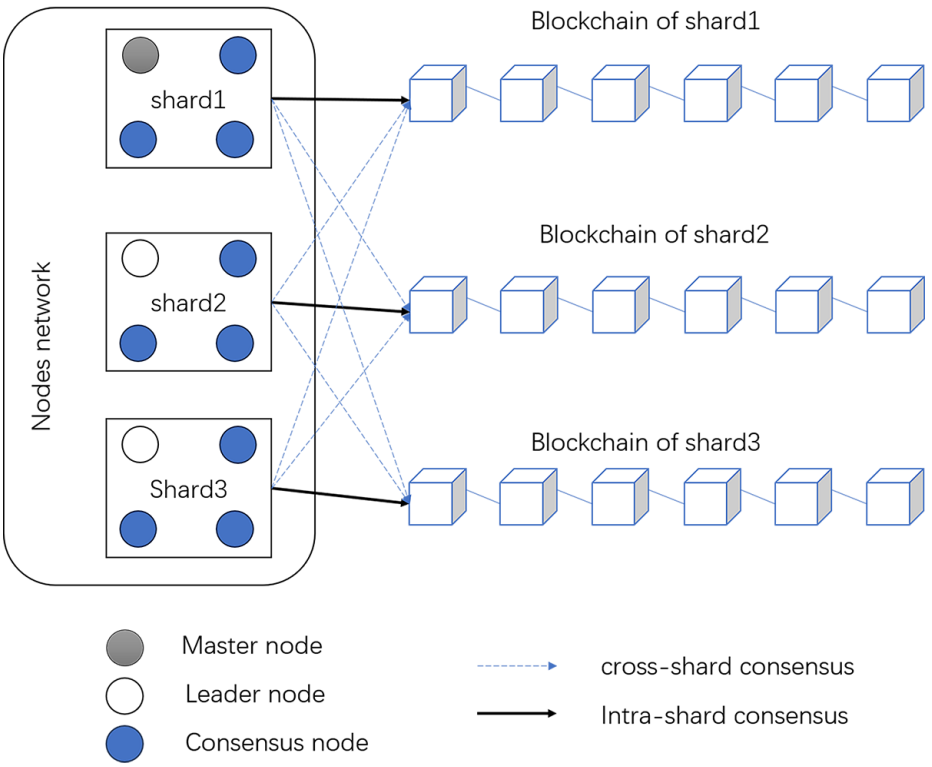
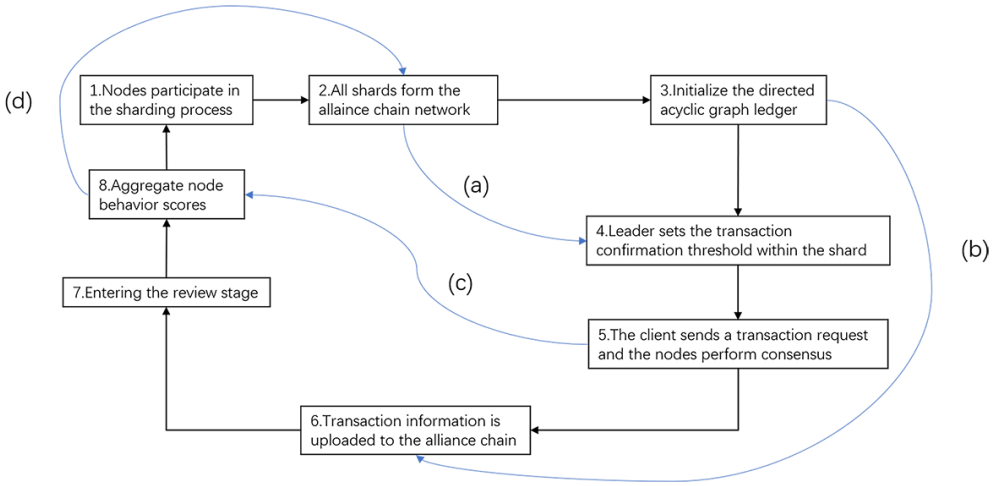


Figure 2. Protocol structure and operation (Note: (a) The leader node has the function of sending and adjusting transaction confirmation threshold (TCT), which impacts consensus; (b) the ledger type determines the block storage method; (c) the consensus process and results determine the score; (d): leader and master nodes in the network are determined by scores.)



Network Infrastructure

The protocol design incorporates three types of nodes: Consensus, leader, and master nodes. Each committee consists of all consensus nodes and a leader node in a single shard, and a unique consensus committee consists of all leader nodes and the only master node. Consensus nodes validate transactions and submit final consensus results to the leader node within their committees. The leader node packages the consensus-completed transactions into blocks and uploads them to the alliance chain. The consensus nodes in the shard can review the transactions during this period. The master node accepts the challenge message from the consensus node and processes the transaction it reviews. After the status of all nodes is synchronized, each node will generate a score according to the transaction history. When an epoch ends, all the scores of each node will be integrated into a sum as its attribute.

A more significant disparity in node types within a shard typically indicates node characteristics and performance variations across various dimensions, including identity, processing capacity, historical performance, and geographical location. Within a shard, greater diversity in node types ensures the system's resilience, enabling it to continue functioning normally, even if some nodes are attacked or engage in malicious behavior. Furthermore, this diversity enhances the system's availability under varying workloads and environmental conditions. As a result, the authors devised a sharding process that involves an initial clustering phase and a subsequent classification phase. The protocol uses the K-prototype clustering algorithm to assign nodes to clusters according to their numeric and categorical attributes, and the number of clusters is the same as the number of nodes in the shard. Numeric attributes have numerical or continuous characteristics and can include the identity and score of a node. Category attributes have discrete characteristics, typically represented as categories or labels, which can include information such as the organization to which a node belongs, Internet protocol address, and geographic location.

Let the node dataset be $X = \left\{X_1, X_2, X_3, \ldots, X_n\right\}$, where n is the number of node objects in dataset X, and each node data in the node dataset has m attributes (i.e., $X_i = \left\{X_{i1}, X_{i2}, X_{i3}, \ldots, X_{ip}, X_{i(p+1)}, X_{i(p+2)}, \ldots, X_{im}\right\}$, where there are p numeric attributes in front and $m - p$ categorical attributes in the back). Specify a positive integer g as the number of clusters to be divided according to actual application requirements, and the clustering algorithm will randomly

select *g* nodes as the initial prototype (central point). In the clustering algorithm, the dissimilarity of mixed attributes is divided into numeric and categorical attributes to be evaluated separately and then added. The dissimilarity of numeric attributes is calculated by Euclidean distance. The dissimilarity of categorical attributes is calculated by Hamming distance, and Equation 1 shows the details:

$$\delta\left(X_{ij},Y_j\right)=\begin{cases}0, & X_{ij}=Y_j\\ 1, & X_{ij}\neq Y_j\end{cases} \tag{1}$$

where 0 indicates the same attribute value, 1 indicates different attribute values, and Y_j is the attribute j of the cluster Y prototype. Equation 2 shows the distance (dissimilarity) between the data and the cluster:

$$d\left(X_i,Y\right)=\sum_{j=1}^{p}\left(X_{ij}-Y_j\right)^2+u\sum_{j=p+1}^{m}\delta\left(X_{ij},Y_j\right) \tag{2}$$

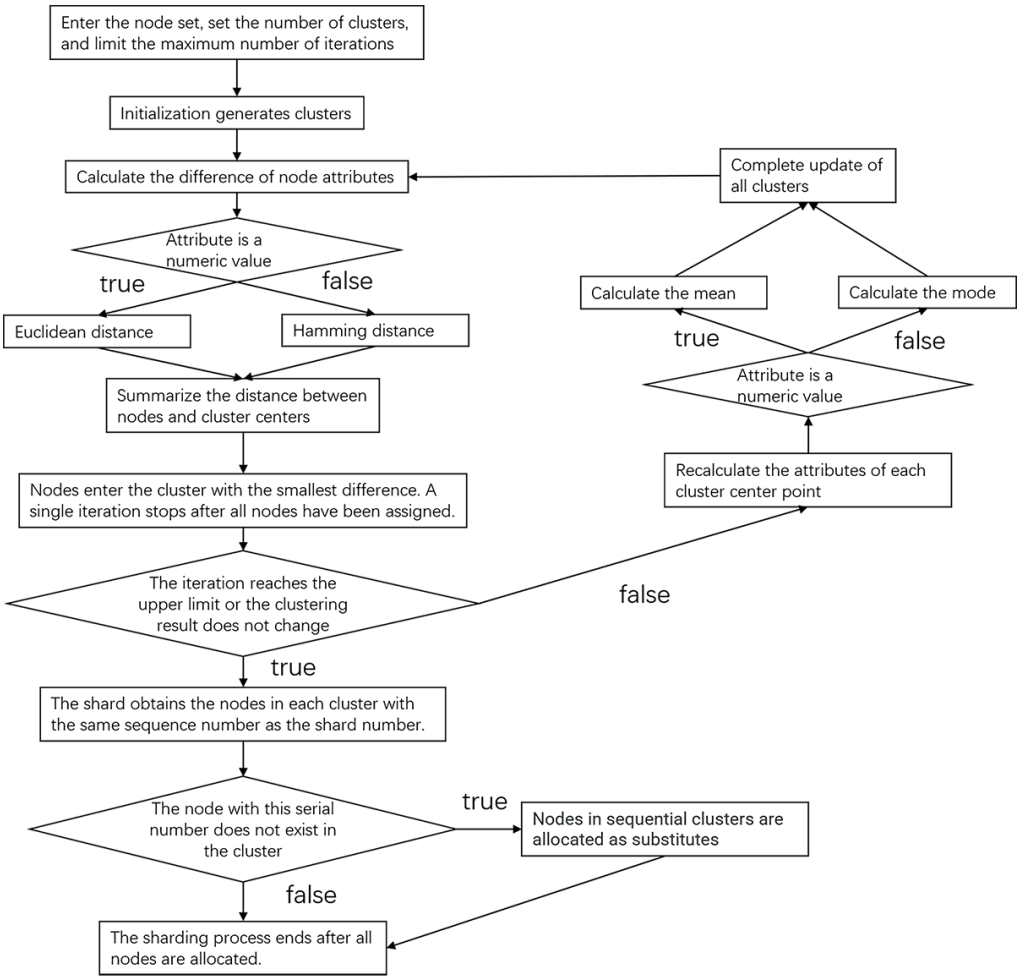
where u is the weight factor of the categorical attribute that can be set.

Each node is assigned to the nearest cluster based on the dissimilarity between the nodes. After each node allocation, the cluster's prototype (center point) needs to be updated. For numeric attributes, the numeric prototype of a cluster will be calculated as the mean of the numeric attribute of all nodes in the cluster. For category attributes, the category prototype of a cluster will be calculated as the mode (i.e., the value with the highest frequency of occurrence) of the category attributes of all nodes in the cluster. The above node allocation and prototype updating steps are performed repeatedly until no node changes its cluster. Once the cluster allocation is complete and stable, the algorithm allocates the nodes in each cluster to shards based on their order. For example, the first shard includes the first node from each cluster, and the second shard includes the second node from each cluster. Figure 3 shows the complete algorithm flow of the two-stage sharding process. Figure 4 (a) shows the clustering results, and Figure 4 (b) shows the classification after clustering. Each shard contains nodes of different clusters as much as possible to maintain the balance of computing functions, security, and network quality (Mehraein & Nourmohammadi, 2023). The node with the highest score in each committee becomes the current Leader node, and the Leader node with the highest score in the consensus committee becomes the current Master node. Node state transitions are depicted in Figure 5.

Blockchain Ledger

The blockchain ledger is a data structure that allows only appending operations to record and store transaction information in a hash chain. DAG ledgers differ significantly from traditional blockchain ledgers. In a traditional blockchain, transactions are packaged into blocks chronologically, with each block linked to the previous block, forming a linear chain. The DAG ledger uses a more flexible structure, connecting transactions with directed edges to form a graph. This structure means that transactions can have multiple parallel paths and are no longer subject to a single chain structure. The initialization process of the DAG ledger includes steps such as the creation of the genesis block, the generation of an initial transaction, transaction verification, and the establishment of directed edges. The initial transaction is used to create the initial state of the ledger, and then transactions are gradually added to the ledger, building a DAG structure. The protocol follows this methodology: Every block consists of an individual transaction, and each data shard is duplicated across all nodes within the committee. Consequently, to guarantee data consistency, it is essential to establish a complete order among transactions (intra- and cross-shard) that access the identical data shard. The

Figure 3. Sharding process (Note: The clustering process's number of clusters and iterations are configured based on specific requirements. The number of clusters is determined by dividing the number of nodes by the number of shards. A higher number of iterations ensures a more comprehensive clustering outcome. Nevertheless, increased iterations also lead to longer processing times, necessitating a trade-off between computational efficiency and result accuracy.)



total order of transactions in this blockchain ledger is achieved by chaining transactions together, where each block contains either a sequence number or a cryptographic hash of the previous block. As cross-shard transactions involve multiple committees, the ledger takes on the structure of a DAG.

Figure 6 illustrates the ledger constructed in this protocol model, depicted as (a), and comprising four committees, namely, p_1, p_2, p_3, and p_4. In Figure 6, the genesis block α, both intra-shard and cross-shard transactions are depicted. t_{10} and t_{13} pertain to intra-shard transactions within committee p_1. Each cross-shard transaction is labeled $t_{e1,\ldots,ek}$, where k is the number of committees involved, and e_i denotes the order of the transactions in the transactions of the i committee. $t_{11,21}$ and $t_{12,22,32,42}$ are two cross-shard transactions, where $t_{11,21}$ accesses p_1 and p_2, while $t_{12,22,32,42}$ accesses all four shards.

Figure 6 demonstrates the establishment of order among transactions, encompassing both intra-shard and cross-shard transactions with a specific shard, such as t_{10}, $t_{11,21}$, $t_{12,22,32,42}$ and t_{13} are interlinked within a chain. Intra-shard transactions originating from distinct committees can be simultaneously incorporated into the blockchain ledger, such as t_{10}, t_{20}, t_{30}, and t_{40} are processed in parallel by disparate committees. Similarly, in scenarios where two cross-shard transactions engage

Figure 4. Clustering and classification (Note: In the process, 32 nodes are clustered and then classified into a shard configuration of four nodes per shard.)

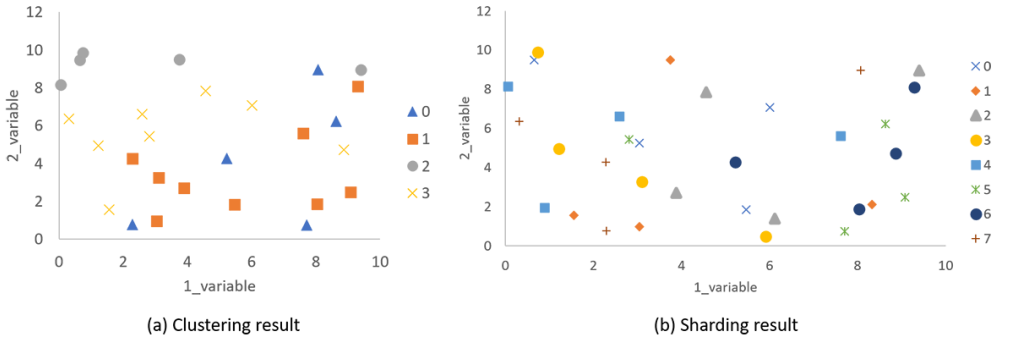
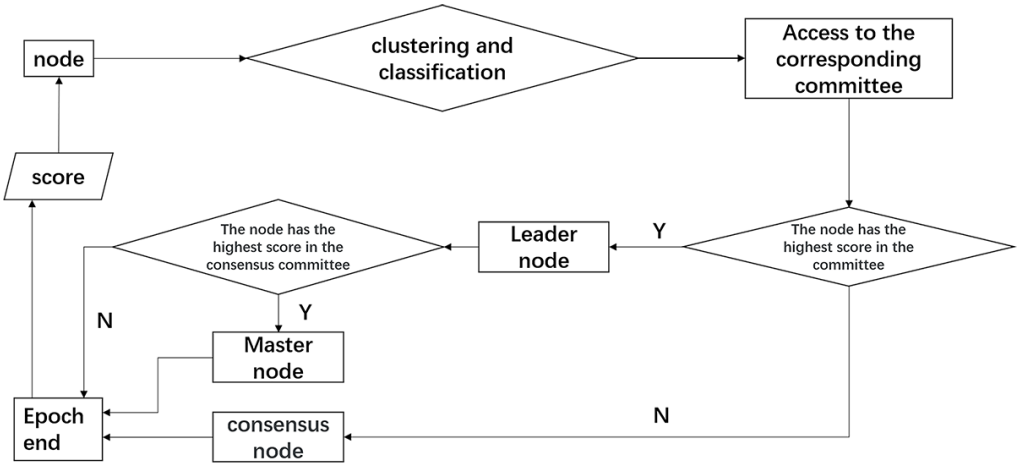


Figure 5. State transition of node (Note: Nodes determine their shard membership through shard configuration. The node enters the shard and participates in the election process of the subsequent leader node. If a node becomes a leader node, it will also participate in electing a master node. The score becomes an attribute of the node after an epoch ends.)



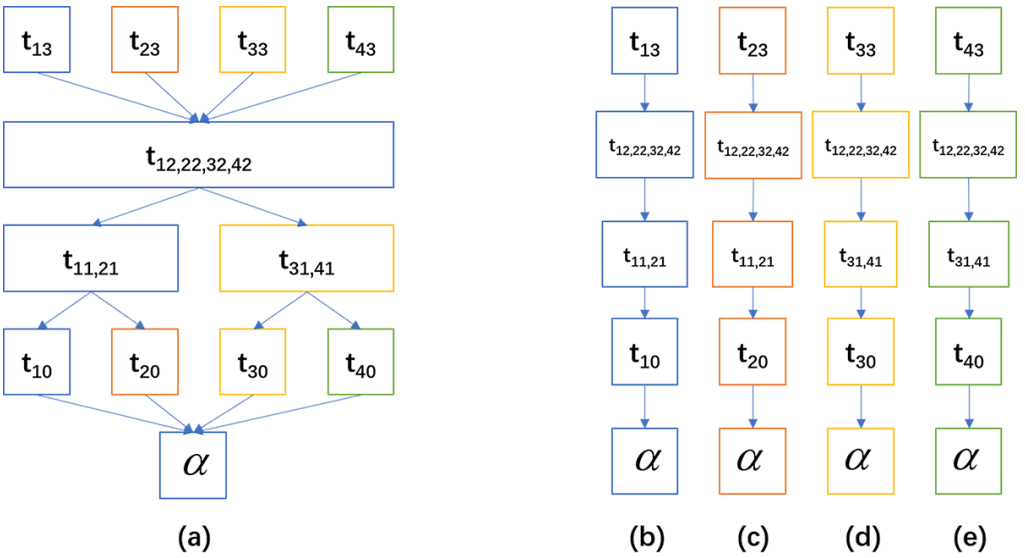
disjoint subsets of shards, they can be added to the ledger in parallel, such as the concurrent addition of transactions $t_{11,21}$ and $t_{31,41}$.

In this protocol, no committee maintains the entire blockchain ledger. Each committee manages only one path of transactions associated with its shard. The blockchain ledger emerges as a fusion of these paths. As Figure 6 (b)—(e) shows, each of the committees p_1, p_2, p_3, and p_4 manifests a distinct ledger path encompassing transactions associated with its respective shard.

Transaction Consensus

Before the formal consensus process begins, the leader node communicates the TCT to the consensus nodes within the shard. Subsequently, the client broadcasts a transaction request to the shards associated with the transaction. This transaction request contains detailed information about the transaction, including transaction content, sender, receiver, and transaction amount. After receiving a transaction request, the node verifies whether the transaction has a valid digital signature to confirm that the sender of the transaction is correct. The node checks whether the sender's account has sufficient balance to perform the correct transaction. If it involves the execution of a smart contract, the node

Figure 6. DAG ledger (Note: (a) represents DAG ledger consisting of four shards; (b), (c), (d), and (e) represent graph paths from four different shards.)

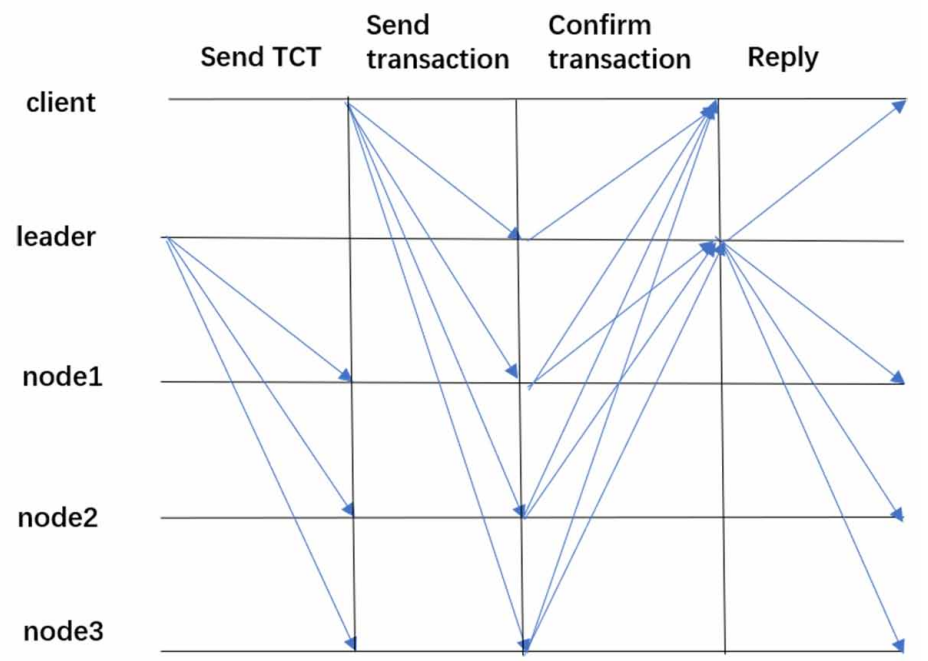


will verify whether the execution results of the contract comply with the contract rules. The consensus node adds the transactions with correct verification results to the transaction pool and sends a confirmation message to the leader node. At the same time, the consensus node replies to the client. This reply activity is one of the score calculation steps and does not affect the consensus. Once the number of confirmed consensus nodes reaches TCT, the leader node packages the transaction into a new block if it solely pertains to that shard. The block is broadcasted to all consensus nodes in the shard, realizing the synchronization of the state of the alliance chain. Figure 7 shows the intra-shard consensus mechanism.

The leader node follows a distinct action if a transaction involves multiple shards. It dispatches a shard confirmation message to the leader nodes of the corresponding shards and awaits their acknowledgments. In addition to detailed information about the transaction, the verification of these confirmation messages also requires the complete intra-shard verification process of the sending shard and the signature of the participating nodes to ensure that the transaction is processed correctly between each shard. After receiving confirmation details from all relevant shards, the confirmed leader node packages the transaction into the new block. This block is then disseminated among other nodes within the shard, promoting the synchronization of the alliance chain's state. Figure 8 illustrates the cross-shard consensus.

The consensus node receives the block transmitted by the leader node and compares the block's contents with the authenticated transactions in the local transaction pool. When the consensus node encounters a transaction submitted by the leader node that has not yet been verified, it triggers a review process by sending a challenge message (containing transaction details to be reviewed) to the master node. The master node sends the received challenge message to the remaining $k - 1$ shards to initiate a validation process, excluding the particular shard that initiated the challenge. After verification, if the results from over $2k/3$ shards indicate a potential error in the transaction, the master node merges the challenge message and the corresponding review result. This merged information is then encapsulated within a block and distributed to all participating shard nodes engaged in the transaction.

Figure 7. Intra-shard consensus (Note: The flow chart of intra-shard consensus mechanism in the case of the shard with four nodes. There are four consensus stages: Send TCT, send transaction, confirm transaction, and reply.)



Node Score Mechanism

During each consensus period, all participating nodes undergo a behavioral evaluation, with the primary metric being the time it takes a node to process a transaction. Based on this metric, each node receives a score, which plays a crucial role in the comprehensive evaluation of node behavior within an epoch. The score also serves as node attributes for the next epoch's sharding process and node election. Once a new epoch begins and the node's new state is determined, the score from the previous epoch is reset to enable a fresh evaluation of the node's behavior within the new epoch.

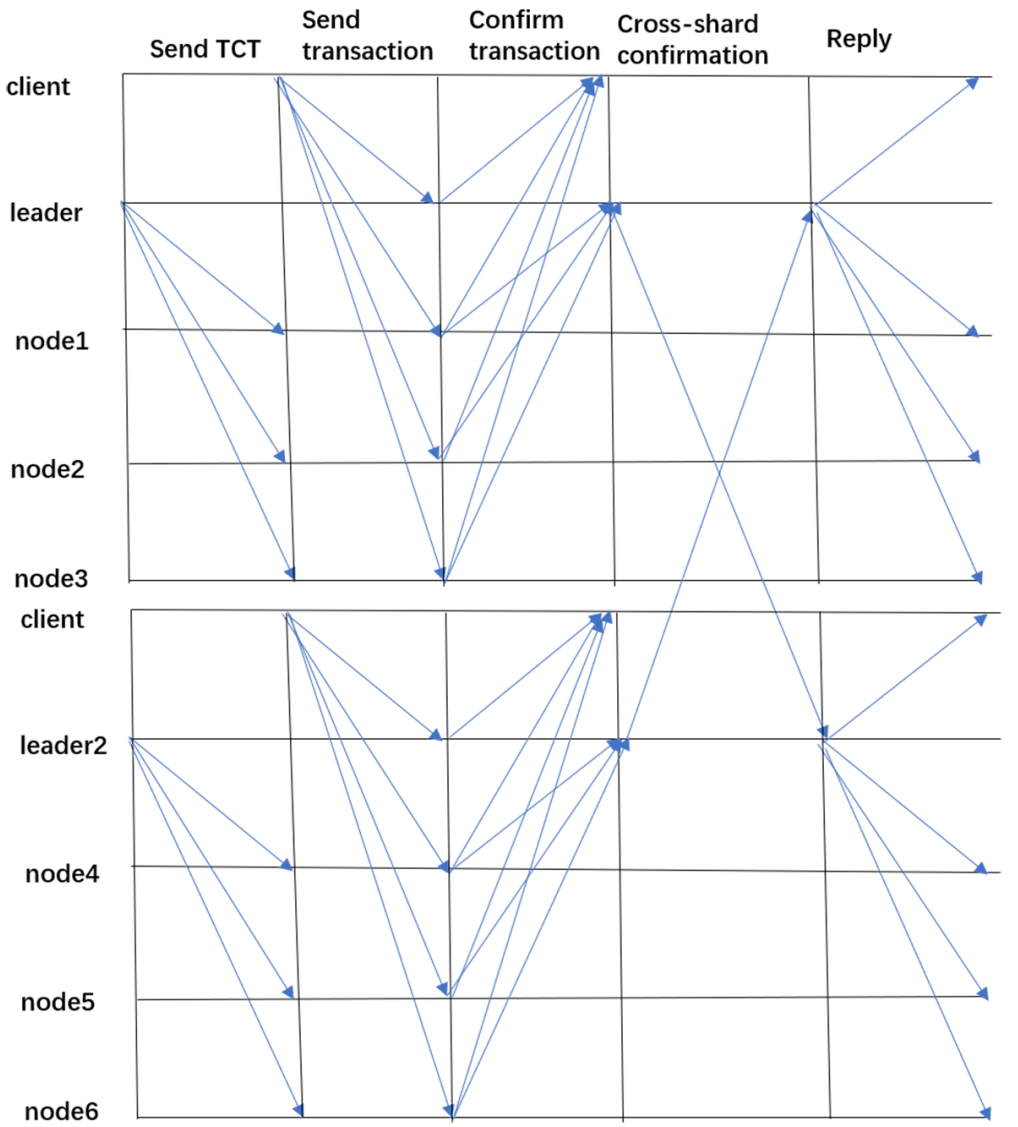
Let $t^i_{j,Tx} = t^i_{j(confirm)} - t^i_{(send)}$, where $t^i_{(send)}$ is the time when the client sends a request to a member of shard *i*, and $t^i_{j(confirm)}$ is the time when a node *j* in a shard *i* confirms a transaction *Tx*. $\overline{t^i_{Tx}}$ is the average time for all nodes in a shard to confirm *Tx*, which is defined as Equation 3:

$$\overline{t^i_{Tx}} = \frac{\sum_{j=1}^{\frac{n}{k}} t^i_{j,Tx}}{\frac{n}{k}} \tag{3}$$

where *n/k* is the number of nodes in each shard.

Let $d = t^i_{j(receive)} - t^i_{j(confirm)}$ be the delay from confirmation to delivery of the transaction to the client, where $t^i_{j(receive)}$ is the time when node of the response received by the client. Let d_0 be the preset delay standard, and *MAX(D)* be the maximum acceptable delay for delivery transaction confirmation. As Equation 4 illustrates, β_d is the ratio by which the score needs to be reduced due to unexpected

Figure 8. Cross-shard consensus (Note: The flow chart of cross-shard consensus mechanism in the case of two shards, where both clients are the same. There are five consensus stages: Send TCT, send transaction, confirm transaction, cross-shard confirmation, and reply.)



events in the network (e.g., a sudden increase in traffic) delaying the transaction delivery to the client, provided that the node confirms the transaction correctly:

$$\beta_d = \frac{d}{MAX\left(D\right)} \tag{4}$$

As Equation 5 shows, β is a coefficient representing the correctness of the transaction verification result and the activity of consensus:

$$\beta = \begin{cases} 0, & not\ confirmation\ or\ inconsistently\ confirmation \\ 1, & correct\ confirmation\ with\ d \le d_0 \\ 1-\beta_d, & correct\ confirmation\ with\ d_0 < d \le MAX(D) \\ 0, & d > MAX(D) \end{cases} \tag{5}$$

After each transaction is completed, the system scores the nodes according to the processing time and confirmation results. $\left(s_j^i\right)_{Tx}$ is the score of node *j* in shard *i* after confirming transaction *Tx*, defined as Equation 6:

$$\left(s_j^i\right)_{Tx} = \frac{\overline{t_{Tx}^i}}{t_{j,Tx}^i} \tag{6}$$

Finally, $\left(s_j^i\right)_{final}$ is the total score of node *j* in shard *i* after confirmed *T* transaction, defined as Equation 7:

$$\left(s_j^i\right)_{final} = \sum_{Tx=1}^{T} \left(s_j^i\right)_{Tx} \tag{7}$$

Figure 9 illustrates the complete process of the protocol monitoring and evaluating node behavior through the score mechanism.

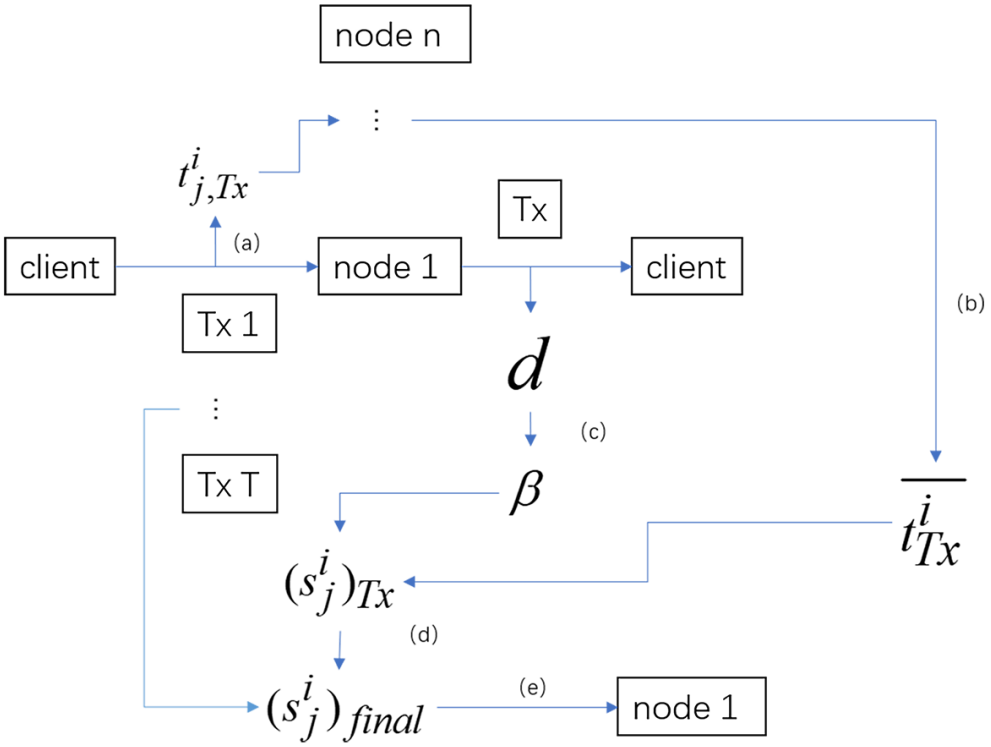
The score mechanism serves as a means to restrict node behavior during transaction consensus. Also, it incentivizes well-performing nodes by offering them increased opportunities to assume roles such as leader or master nodes during elections. Despite the effectiveness of this score mechanism in guiding node behavior, it cannot eliminate the possibility of malicious activities. Therefore, a corresponding penalty mechanism is also designed to increase the cost of nodes performing malicious behaviors. If a node makes an error during the verification process or fails the challenge during the review process, its total score within the current epoch will be reduced by 50%. If a node engages in the malicious behavior mentioned above again, the system will delete the node from the network. The identification of malicious behavior and whether a deleted node may reenter the network will depend on the specific usage scenario.

Adjustment of Transaction Confirmation Threshold

TCT determines how many consensus nodes are needed to confirm a transaction, thus directly affecting the security and credibility of the transaction. When the TCT increases, the number of nodes required for consensus also increases accordingly, which means more nodes will participate in confirming transactions, thereby improving the system's overall security. This mechanism can effectively deal with potential network attacks and threats, ensuring that transactions are considered valid only after enough nodes reach consensus.

However, when the network is unstable, an excessively high TCT will make it difficult for nodes to send confirmation messages, leading to network congestion and ultimately threatening normal consensus progress. This design allows users to adjust TCT according to conditions to balance performance, security, and stability. Especially when facing network fluctuations or attacks, dynamic TCT can ensure system availability.

Figure 9. Score calculation process (Note: (a) Calculate the time interval from the client sending the transaction request to confirm the transaction to determine the node's confirmation time. (b) Calculate the average value from the confirmation times of all nodes participating in the consensus. (c) Measure the node consensus activity using the time interval from the node confirming the transaction to delivering the transaction to the client and the correctness of the verification. (d) After each transaction is processed, a node's score is calculated to evaluate the node's performance in confirming a specific transaction. (e) The total score of a node is calculated after processing all transactions in the epoch. The total score is a numerical attribute for the node's sharding and election in the next epoch.)



The protocol stipulates that TCT can only be adjusted once by the Leader node in the shard during its term, and the adjustment range is limited. This limitation increases the cost for potential attackers because attackers need to control multiple consecutive Leader nodes to affect the TCT of the system in order to harm the system. This mechanism further enhances the alliance chain's security and availability, protecting the integrity of transactions and data.

CORRECTNESS ARGUMENT

In this section, discussions focus on the overall improvements to each component in the above design. These discussions encompass consensus duration, communication complexity, computational cost, component functionality, and their connections.

Performance Analysis

There are n/k nodes in the shard, the time for a node to process a message is fixed at t_1, and the message delivery time is fixed at t_2. T is the time required to complete the cross-shard confirmation process involving two shards in this protocol. According to the dynamic transaction confirmation consensus process, there are four stages of message transmission. Nodes process a transaction message, TCT

confirmation messages, cross-shard confirmation messages, and the block. Equation 8 shows the calculation of the time complexity for achieving consensus:

$$T = 4t_2 + \left(TCT + 3\right)t_1 \tag{8}$$

T' is the time required to complete cross-shard transaction flows for byzantine nodes in SharPer, a permissioned chain sharding protocol employed for handling cross-shard transactions within a network comprising byzantine nodes, which has demonstrated good outcomes in performance and security aspects. The consensus process has a total of four stages of message transmission. Nodes in a shard need to process a request message, a prepare message, *4n/(3k)* accept messages, and *4n/(3k) + 2* commit messages. Equation 9 shows the calculation of the time complexity of achieving consensus:

$$T' = 4t_2 + \left(\frac{8n}{3k} + 4\right)t_1 \tag{9}$$

Equation 10 originates from Equations 8 and 9:

$$T' - T = \left(\frac{8n}{3k} + 1 - TCT\right)t_1 \tag{10}$$

In the cross-shard transaction consensus phase, the dynamic transaction confirmation consensus outperforms Sharper in terms of efficiency. The specific value of the improvement is jointly determined by t_1, *n/k*, and *TCT*.

The values of TCT are set to TCT_1 and TCT_2, and the difference in message confirmation completion time is equal to the reduced transaction confirmation latency $\triangle T$, as Equation 11 shows:

$$\triangle T = \left|TCT_1 - TCT_2\right|t_1 \tag{11}$$

The transaction latency will also change if the TCT in the dynamic transaction confirmation consensus changes. $t_1$is usually measured in milliseconds, so TCT has little impact on system throughput.

The proposed protocol adopts a dynamic transaction confirmation consensus mechanism and a DAG ledger to address the challenges of cross-shard transaction processing. The intra-shard consensus complexity of the dynamic transaction confirmation consensus mechanism is lower than the mainstream BFT consensus protocol. The processing of cross-shard transactions only relies on the Leader node and does not require an additional node set, which helps reduce communication overhead. The alliance chain uses the DAG ledger structure to achieve more transaction parallelism, allowing independent blocks to be added to the ledger simultaneously. The synergy of these mechanisms enables the protocol to efficiently handle high-frequency and large-scale cross-shard transactions while the system remains secure and available.

Stability and Safety

In this protocol, the client segment sends transaction information to each node in the shard (a total of *n/k* messages), and each node will send a confirmation message. The Leader node sends a cross-shard confirmation message and broadcasts block information (a total of *n/k* — 1 messages) in the

shard. Equation 12 shows the number of messages *S* required to complete transaction confirmation in two shards:

$$S = \frac{6n}{k} - 2 \tag{12}$$

According to the Sharper process, the client sends a request message to the Leader node, which broadcasts a prepare message to each consensus node (total of *2n/k* — 1 messages). Each node broadcasts an accept message (total of $[2n/k — 1]^2$ messages) and a commit message (total of $[2n/k — 1]^2$ messages). Equation 13 shows the number of messages required to achieve consensus in a shard with the same configuration:

$$S' = 2\left(\frac{2n}{k} - 1\right)^2 + \frac{2n}{k} \tag{13}$$

The overhead of the proposed work is primarily calculated based on the algorithm's communication complexity. During the consensus process, its communication complexity is *O(n)*, which exhibits clear advantages compared to the BFT consensus algorithm, with a complexity of $O(n^2)$. The *O(n)* complexity signifies that the computational overhead of the consensus process scales linearly with the number of nodes, enabling more efficient execution within large-scale networks. Furthermore, the *O(n)* complexity approach is more cost-effective in computing resource consumption, thereby reducing hardware expenditures. The consensus process operates swifter, significantly reducing transaction confirmation delays, a crucial feature for application scenarios that demand rapid transaction verification. In addition to consensus, the complexity associated with the sharding process and score calculation must be considered. In the sharding stage, the K-prototype clustering algorithm is employed. During the initialization phase, this algorithm entails the random selection of initial cluster centers, and its complexity depends on the number of clusters and node attributes. In the iterative phase, the algorithm updates cluster centers, and the number of clusters and node attributes similarly influences its complexity. The primary iterative complexity of the algorithm is determined by factors such as the number of iterations, the number of samples, the number of clusters, and the number of attributes. For the case where only the number of nodes is a variable, the complexity is *O(n)*. In the process of node score calculation, each transaction only requires a single node calculation without the need for an iterative process, resulting in a complexity of *O(n)*.

In identical computer and network conditions, this protocol's consensus is less susceptible to the negative impact of node scaling and rising proportions of cross-shard transactions compared to the transaction flow involving byzantine nodes. The protocol boasts enhanced stability and can curtail network resource consumption and storage capacity. When the system undergoes multiple verifications due to mishandled transactions or erroneous transaction messages by the verifying nodes, the performance of the alliance chain will not return to its usual state until the malevolent nodes are purged. Both malicious nodes and wrong transactions can result in TCT exerting an influence on stability.

The existing protocols for selecting a master node have certain characteristics. One approach involves selecting a different node as a Master node in each epoch or round according to the rotation rules, which introduces latency and additional communication overhead and allows an attacker to control the rotation order (Amiri et al., 2019; Zamani et al., 2018). Another approach is staking-based selections, which incentivize nodes to follow the rules and maintain normal behavior (Fitzi et al., 2020). Over time, this mechanism may lead to centralization within the system. Performance-based selection can improve system efficiency, but can cause some nodes to become the master node,

continually reducing the utilization of others (Kokoris-Kogias et al., 2018). The protocol design utilizes the behavior score of nodes to select. This approach has several advantages, such as efficiently allocating resources, improving overall network throughput, and maintaining a low likelihood of consensus errors.

Regarding node assignment, methods based on node properties (e.g., liveness and performance) have centralization risks (Luu et al., 2016; Zamani et al., 2018), while free-choice assignment methods are less resistant to malicious behavior (Chen & Wang, 2019). In the protocol design, node allocation first clusters the nodes with high similarity and then assigns the nodes of each cluster to different shards. Clustering is achieved by calculating the attribute (e.g., the node's identity, score, the organization to which the node belongs, internet protocol address, and geographical location) dissimilarity between nodes. While not eliminating the possibility of manipulating individual parameters, this guarantees a robust and manipulation-resistant identity generation process.

Alliance chains often require permission to enter, which helps control access to the network, but there are still some security risks. Attackers may deliberately delay the confirmation of transactions, thereby affecting the performance of the entire alliance chain system. The node score mechanism can solve this attack by evaluating node performance, motivating nodes to process transactions quickly, and punishing malicious nodes. A group of dishonest nodes may collude in the same shard to corrupt the validation of transactions involving that shard. The shard configuration process reduces the likelihood of such collusion attacks by ensuring each shard contains nodes with different properties. Attackers may try to occupy the position of the leader or master node and thereby control the consensus process. The node election process ensures that high score nodes hold these key positions, so the risk of pivotal nodes being controlled is low. Attackers may conduct network attacks to interrupt the communication of the alliance chain and interfere with the consensus process. The dynamic consensus confirmation threshold mechanism relaxes the network quality requirements for system operation and increases the cost of attacks.

The various components of this protocol interact to ensure that the system is available while maintaining high performance. The impact of transaction information and ledger structure in the ledger design on the formula is that they can change the consensus algorithm's computational complexity, storage requirements, verification efficiency, and privacy protection. The node score mechanism significantly improves the reliability of the system. Rewarding nodes with good performance and punishing malicious nodes reduces the failure rate and malicious behavior of nodes and enhances the entire system's stability. Regarding scalability, the node score mechanism ensures that high-quality nodes are allocated to different shards, improving each shard's performance and security and increasing the system's robustness. At the same time, it limits the participation of dishonest nodes, prevents repeated verification, and further enhances the system's scalability. The sharding process divides a node into multiple shards, each responsible for processing a specific range of transactions. This process helps improve the system's scalability as different shards can process transactions in parallel, thus increasing the overall system's throughput. The sharding process also needs to ensure the consistency and security of cross-shard transactions, so leader nodes need to be elected to coordinate consensus between different shards. Node selection ensures that nodes with high scores and good performance become leader nodes, which can effectively manage the consensus process and ensure system reliability.

Consensus nodes are responsible for verifying transactions and sending challenge messages. Both wrong verification and challenge failure will reduce the score of the consensus node. The leader node is responsible for adjusting the TCT, counting the number of confirmation messages, and generating blocks. If the leader node sends different TCTvalues to each consensus node or does not send to some consensus nodes, the consensus node will replace the leader node due to inconsistent status. If the consensus node finds that the TCT and voting information in the block are wrong, it will consider the leader node malicious and send a challenge message to the master node. The access mechanism of the alliance chain and the upper limit of adjustment of TCT make it very expensive for the leader

node to adjust TCT malicious. The master node is responsible for processing challenge messages. If the master node fails to process the challenge message in time, the node that sent the challenge proposes to all nodes to replace the master node. After the master node is replaced, the new master node will process unprocessed challenge messages. In summary, the protocol is safe and practical.

Experimental Design

Performance testing takes place in a simulated system to verify the impact of the protocols analyzed above on system scalability. The experiment involves comparing the performance of the sharding protocol with the mainstream sharding protocol used in the alliance chain with different cross-sharding protocol ratios. Additionally, it tests how the system performance is affected by varying TCTs and block sizes. The simulated alliance chain is designed to have a consistent system architecture and network model, and any discrepancies are minimized to remain within the protocol scope.

Experiment and Configuration

Experiments simulate clients and nodes using a C++ simulation system that utilizes multithreading. In this testing scenario, clients are responsible for transmitting transactions, while each shard comprises multiple consensus nodes alongside a leader node. The system is categorized into distinct modules: the transaction module and the consensus module. The system's performance and scalability assessment revolve around throughput and transaction latency metrics. Table 1 shows the detailed configuration.

Experimental Testing

This experiment measures the performance of an alliance chain using three different types of sharding protocols, but with a fixed number of shards and nodes. By gradually adjusting the proportion of cross-shard transactions from 1% to 100%, it is possible to observe changes in transaction confirmation latency and throughput to measure the impact of cross-shard transactions on the efficiency of different protocols. Latency is the maximum interval between the client sending a transaction and the completion of processing within an epoch, and its unit is seconds. Throughput is the average number of transactions a system processes per second. Figure 10 shows that the transaction processing latencies of the three protocols are similar when the proportion of cross-shard transactions is low. When the cross-shard transaction ratio is 10%, a typical setting for partitioned databases, the protocol's latency is 36% lower than Sharper and 56% lower than AHL. This latency advantage continues to persist as the volume of cross-shard transactions expands, underscoring the superior scalability of this protocol.

Figure 11 presents a comparative assessment of transaction confirmation throughput as the proportion of cross-shard transactions increases. The throughput of this protocol has been significantly ahead of Sharper and AHL, when the cross-shard transaction ratio is 10%. In the impact of the subsequent expansion of the proportion of cross-shard transactions, although the performance difference has changed, the protocol still maintains its throughput advantage over other protocols. For example, after the gap with AHL narrows, the throughput increases by at least 76%.

Table 1. Software and hardware environment configuration

Software and hardware environment	Configure
CPU	2.40 GHz Intel Core i5-9300H
RAM	16GB 2667 MHz DDR4
System	Windows 11

Note: CPU (central processing unit) is the primary processing unit of a computer. RAM (random access memory) is a computer memory that temporarily stores running programs and data. System refers to the computer's operating system.

Figure 10. Comparison of latency for different cross-shard transaction ratios (Note: System latency averages comparison for cross-shard transaction proportions ranging from 1%, 5%, …, 100%. Other factors (shard counts = 4, number of nodes in shard = 4, TCT = 3, and transaction counts = 400) are the same except for the sharding configuration.)

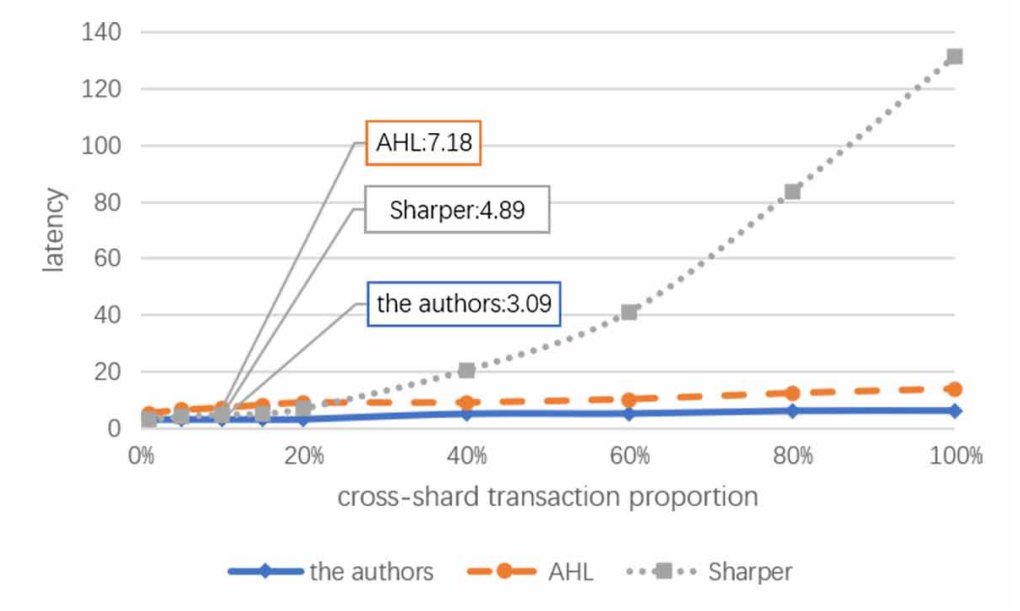
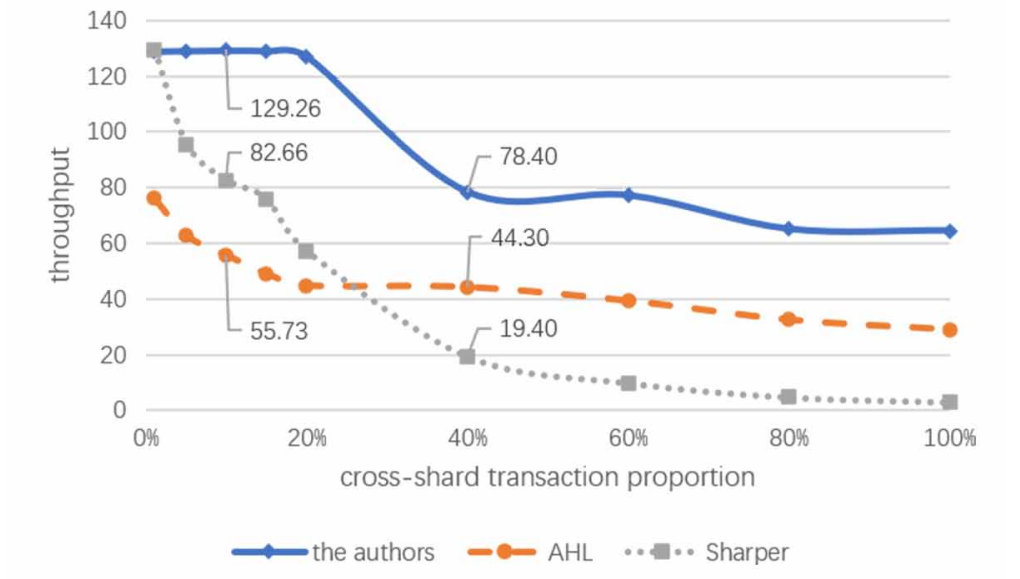
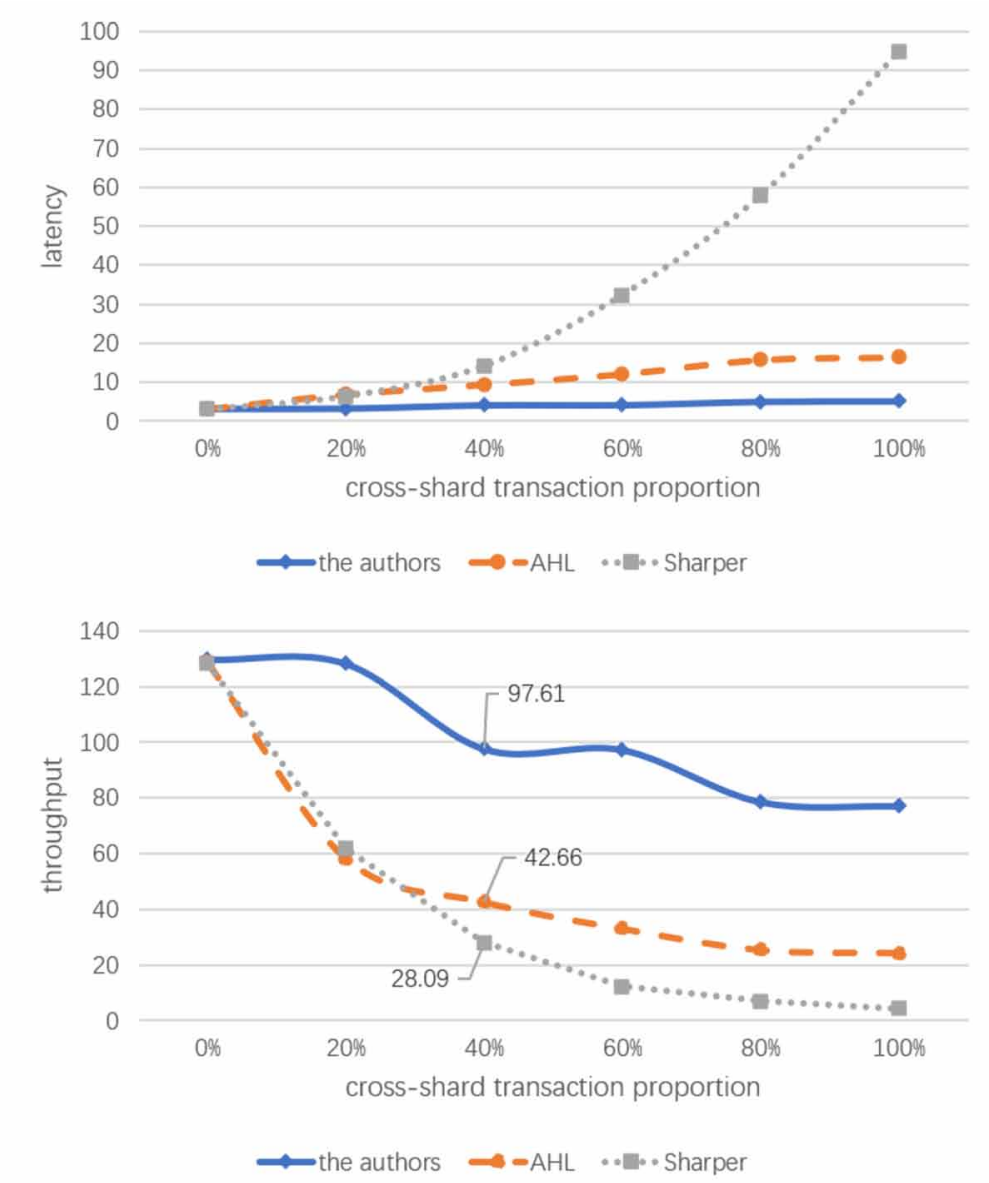


Figure 11. Comparison of throughput for different cross-shard transaction ratios (Note: System throughput averages comparison for cross-shard transaction proportions ranging from 1%, 5%, …, 100%. Other factors (shard counts = 4, number of nodes in shard = 4, TCT = 3, and transaction counts = 400) are the same except for the sharding configuration.)



In addition, adjust the sharding configuration to test the expanded alliance chain. Figure 12 shows the performance comparison between different sharding protocols after the number of nodes increases. Compared to other protocols, the advantages of this protocol become more evident as the node scale increases.

Figure 12. Comparison of protocol performance after adding nodes (Note: System throughput and latency average comparison for cross-shard transaction proportions ranging from 0%, 20%, …, 100%. Other factors (shard counts = 5, number of nodes in shard = 4, TCT = 3, and transaction counts = 400) are the same except for the sharding configuration.)



The above experimental results are derived from the average of multiple measurements, which is sufficient to demonstrate that this protocol can steadily improve performance. The dynamic transaction confirmation consensus mechanism rationally utilizes system resources under the increasing proportion of cross-shard transactions, including transmitting transactions between various shards and ensuring conflict handling and consistency.

This experiment measures the performance of an alliance chain using three different types of sharding protocols, but with a fixed number of shards. It is necessary to adjust the number of nodes in

each shard from four to 12, observe the changes in latency and throughput of the alliance chain within an epoch, and compare how the increased number of nodes affects the efficiency of different protocols while keeping the units of latency and throughput. As Figure 13 illustrates, in the scenario where there are no cross-shard transactions, the latency of this protocol is significantly reduced compared with high-performance sharding protocols. It has a minimum advantage of 10% over KBFT on 24 nodes and 6% over IGD-ScoreChain on 32 nodes. This latency advantage remains firmly maintained as the number of nodes increases.

Figure 14 shows a test of system throughput as the number of nodes increases. Although the numerical gap keeps changing, the protocol maintains a stable lead over the other two in throughput. For example, its throughput is 9% higher than KBFT in the case of 24 nodes and 6% higher than IGD-ScoreChain in the case of 32 nodes, and these are only the minimal improvements shown.

As the experimental results showed, the data confirm the performance advantages of this protocol. This protocol has varying performance advantages in scenarios with different nodes, and the overall gap tends to expand as the number of nodes increases. This situation results from the designed consensus mechanism, which optimizes the transaction confirmation process, ensuring that the required communication scale remains low even when dealing with a large number of nodes. The data presented above represents averages from multiple tests.

TCT plays a decisive role in the consensus process of transactions, and the authors explained its importance in the analysis process. Therefore, designing experiments to study how to set up dynamic TCT is necessary. This experiment uses two alliance chains with different sharding configurations (i.e., four shards, each shard containing 8 or 9 nodes) to study the impact of TCT on system performance. Gradually increase TCT in each shard and observe the changes in the alliance chain latency and throughput within an epoch. Figure 15 shows a summary of the results.

Experimental results show that in the absence of erroneous transactions, dynamic TCT does not significantly impact system throughput or latency. Therefore, in the early stages of system operation, it is recommended to configure the configuration within the range of 1/3 to 2/3 of the total number

Figure 13. Comparison of latency without cross-shard transaction (Note: System latency averages comparison for node counts of per shard ranging from 4, 5, …, 12. Other factors (TCT = 3 and transaction counts = 800) are the same except for the sharding configuration.)

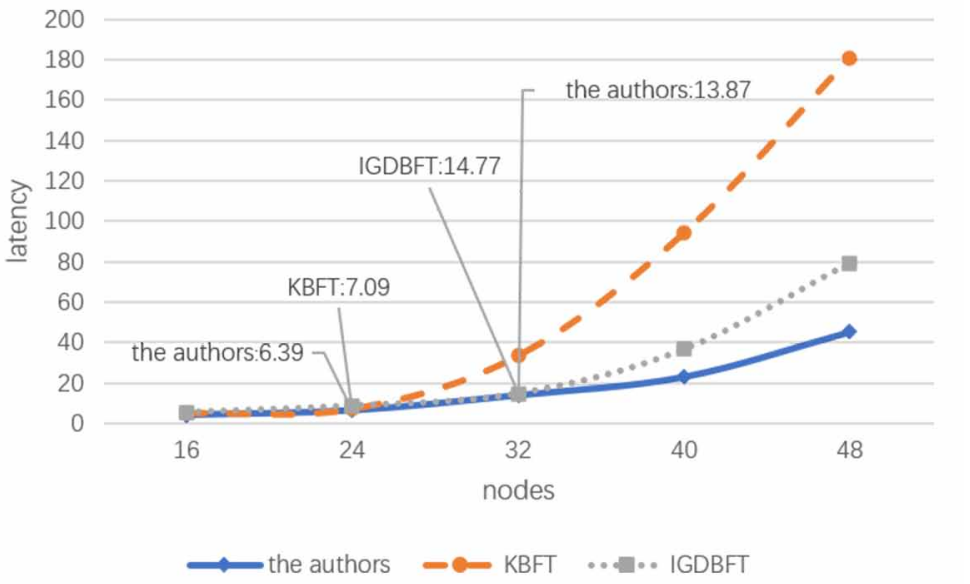
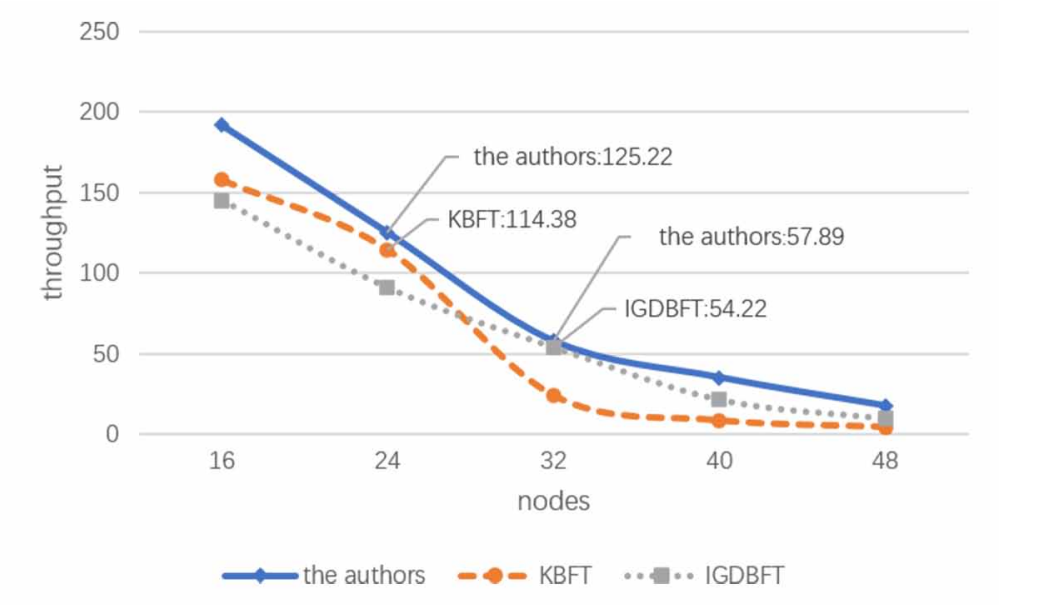


Figure 14. Comparison of throughput without cross-shard transaction (Note: System throughput averages comparison for node counts per shard ranging from 4, 5, ..., 12. Other factors (TCT = 3 and transaction counts = 800) are the same except for the sharding configuration.)



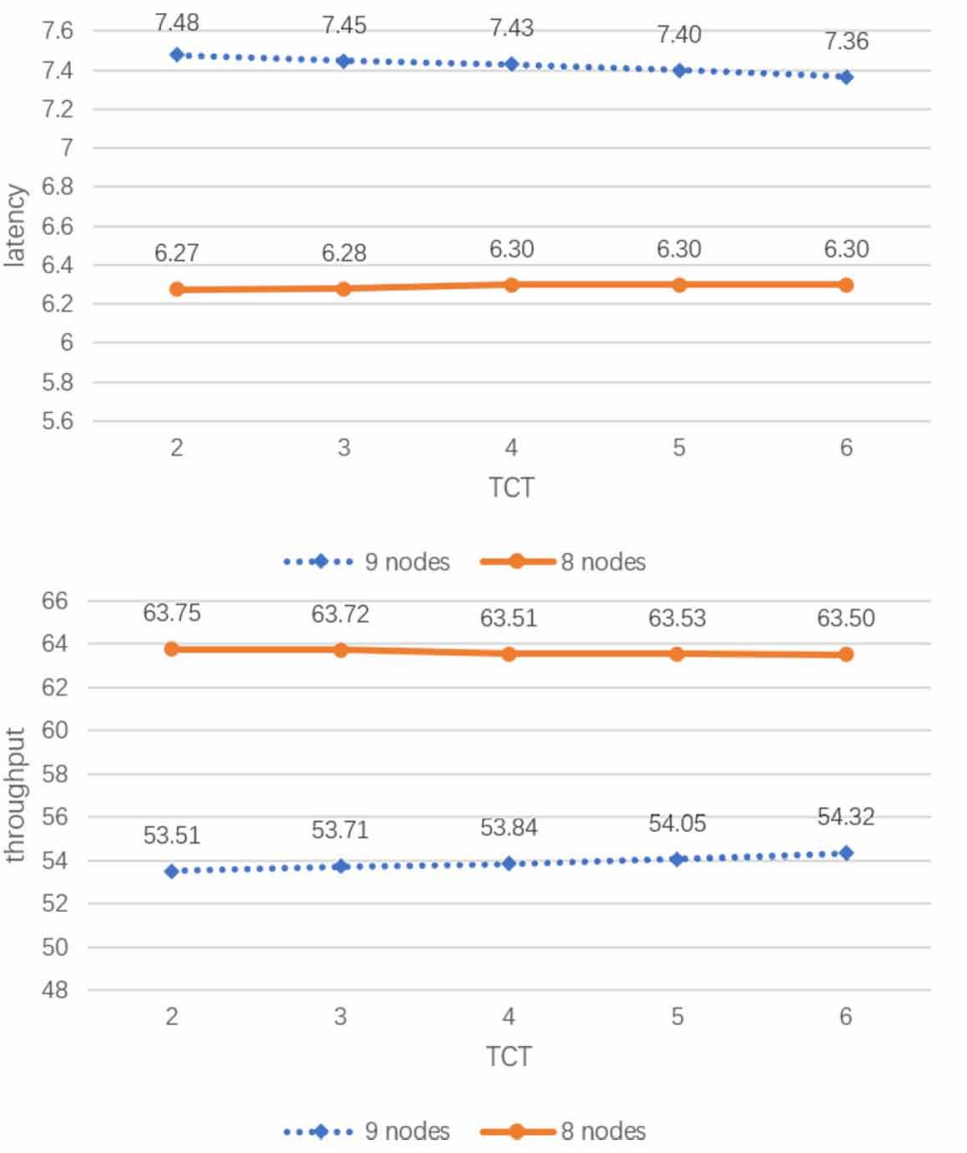
of shard nodes within the range of conventional alliance chains to prevent malicious behavior of byzantine nodes. Consistent with the analysis, the experimental results show that this parameter adjustment ensures the system's stability.

The number of transactions in an epoch will affect the throughput and latency of the entire system because it involves the frequency of node allocation. This experiment studies the impact of total transaction volume over a period on the performance of this protocol. While the shard configuration, cross-shard transaction ratio, and TCT remain unchanged, only the total transaction count is adjusted. The results combine performance data from multiple measurement systems (Figure 16). The units of delay and throughput are the same as in the above experiments, and transaction size is the number of transactions in an epoch.

Increasing the total transaction size enhances throughput, albeit at the expense of extended block confirmation times. Hence, the practical realization of a comprehensive transaction necessitates an evaluation of performance requirements within distinct application contexts. Striking the optimal equilibrium between throughput and transaction confirmation latency is essential to satisfy user expectations and demands. Achieving this balance entails meticulously assessing the blockchain protocol's design and configuration, ensuring alignment with the demands of the specific use case.

While simulations may not perfectly replicate the intricacies of real-world distributed settings, they provide a controlled avenue for researchers to analyze and evaluate the protocol's behavior. These experiments serve the purpose of comprehending the holistic procedure and gauging the scalability enhancements introduced by this protocol compared to existing methodologies. It is essential to acknowledge that disparities might arise when transitioning this protocol into an actual distributed environment. These differences should not hinder the deployment of protocol processes in a distributed setup or undermine the potential for enhanced scalability and the advantageous position over other protocols.

Figure 15. Comparison of system performance for different TCT (Note: System latency and throughput averages comparison for different TCT values from 2, 3, . . ., 6. Experiments are conducted in two shard configurations, where the number of shards is 4, but the number of nodes within the shards is 8 and 9 (the total number of nodes is 32 and 36). Other factors (the typical settings include cross-shard transaction proportion = 10% [Amiri et al., 2019], transaction counts = 400) are the same except for the sharding configuration.)

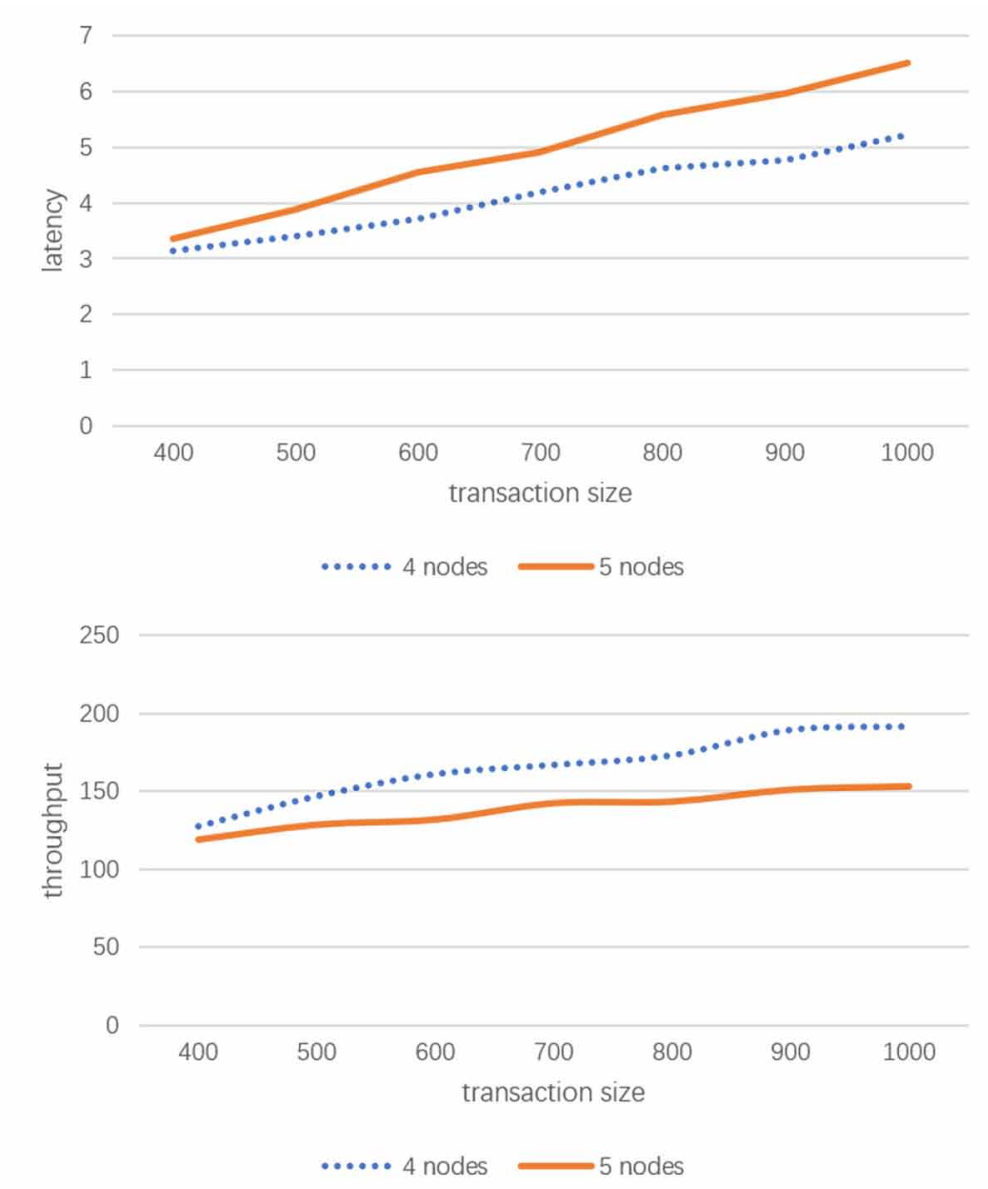


RELATED WORK

In this section, the authors conduct a review of relevant previous studies, methods, and protocols, with an emphasis on highlighting the unique advantages of this protocol in terms of performance through these comparisons.

Many classic cross-sharding schemes have been proposed, but they all have some limitations. Omniledger (Kokoris-Kogias et al., 2018) is designed based on Elastico (Luu et al., 2016) and adopts

Figure 16. Comparison of system performance for different transaction sizes (Note: Performance averages for different total transaction sizes from 400, 500, ..., 1000. Experiments are conducted in two shard configurations, where the number of shards is 4, but the number of nodes within the shards is 4 and 5. Other factors (the typical settings include cross-shard transaction proportion = 10% [Amiri et al., 2019] and TCT = 2).)



a client-driven cross-shard transaction processing method. However, a malicious leader node may provide false proof of acceptance, causing inconsistencies between shards. Furthermore, no detailed construction method for rejection proofs has been provided. The collection and delivery of evidence increases the burden on clients. RapidChain (Zamani et al., 2018) splits cross-shard transactions into multiple single-input single-output transactions and commits them sequentially. However, this increases the total number of transactions, thereby increasing the processing and storage burden on the entire network. Additionally, no detailed instructions are provided on how to generate a specific

shard transaction. Chainspace (Al-Bassam et al., 2017) combines 2PC with BFT, but transactions can only be processed individually, resulting in many BFT calls. Monoxide (Wang & Wang, 2019) proposed a relay mechanism in an account/balance-based system. Each cross-shard transaction is divided into subtransactions, including internal and relay transactions. Each internal transaction corresponds to an associated shard. A relay transaction is required between every two consecutive internal transactions. AHL (Dang et al., 2019) adopts trusted hardware to limit the malicious behavior of nodes. However, this system has several disadvantages. First, running a fault-tolerant protocol among 80 nodes results in high latency. Second, the protocol requires an additional set of nodes to form the reference committee, resulting in high communication overhead between nodes and the reference committee. Finally, since a single reference committee handles cross-shard transactions, the protocol cannot parallel cross-shard transactions with non-overlapping clusters. SharPer (Amiri et al., 2019) is a permissioned blockchain system with a ledger designed as a DAG, which is not maintained by any node. Therefore, SharPer can process cross-shard transactions in nonoverlapping clusters in parallel, but consensus in this process requires cross-shard communication from each participating node. Pyramid (Hong et al., 2021) proposes a new cross-shard block structure and provides a hierarchical shard consensus to expand the number of relevant shards in cross-shard blocks. Its work relies on frequent BFT consensus calls from bridge shards that handle transactions across multiple shards.

Although the above sharding protocols can guarantee certain atomicity and consistency of cross-shard transactions, they require cumbersome consensus to handle cross-shard transactions. In systems, particularly those employing complete sharding protocols, each cross-shard transaction undergoes segmentation into numerous subtransactions. All the associated subtransactions must be verified and processed to submit a cross-shard transaction successfully. This severely degrades sharding performance in terms of throughput and acknowledgment latency. Compared with these existing studies, the proposed protocol avoids the large communication and conflict resolution overhead associated with cross-shard transactions through a DAG ledger and a dynamic confirmation consensus mechanism. In addition, the protocol can utilize the sharding process and behavior score mechanism to quickly process transactions while preventing nodes from colluding in malicious behavior. Multiple benefits and a broad spectrum of application domains become apparent when implementing the protocol within a typical intelligent system. Its elevated throughput rate renders it apt for data-intensive operations, such as extensive data analysis and machine learning training, thereby enhancing system efficiency. Its low confirmation latency promises expedited user response times for interactive applications like virtual assistants and intelligent search engines, ultimately augmenting the user experience. In the realm of IoT, it lends support for high throughput and swift transaction processing within expansive systems, guaranteeing the punctual handling of sensor data and device control.

CONCLUSION

This paper proposes a scalable sharding protocol for alliance chain systems with dynamic confirmation cross-shard transactions. The protocol employs a DAG to structure the alliance chain ledger and utilizes dynamic transaction confirmation consensus as the consensus process. This approach enables independent and parallel transaction consensus within each shard, significantly improving intra-shard and cross-shard consensus efficiency. The protocol incorporates a sharding process involving clustering before classification and a node behavior score mechanism to deter attackers from concentrating their controlled nodes into specific shards. This measure reduces system delays and mitigates repeated message propagation caused by potential attacks.

Experimental results demonstrate that the proposed sharding protocol outperforms other methods when dealing with cross-shard transactions, substantially enhancing the scalability of the alliance chain. It is crucial to adjust the dynamic confirmation threshold and block capacity to ensure users access the alliance chain frequently and utilize many fault-free nodes within a smooth network

environment, especially those intelligent systems that usually require multiple smart devices and sensors to work together. The protocol features high throughput and low latency, allowing related devices to transmit data and respond to commands more quickly, thereby improving the performance and reliability of intelligent systems, such as smart city and industrial IoT. Moreover, the protocol remains applicable even in intelligent systems with poorly performing node facilities and scenarios where computing models and environments change frequently. The protocol has been confirmed by analysis and experiments as an improvement to the alliance chain, which improves efficiency and reduces costs of related equipment, thereby improving the performance and reliability of intelligent systems. This research is expected to provide strong support for intelligent systems to realize data integration in information systems through data collection, intelligent analysis and reasoning, real-time response, and control.

The protocol employs a clustering algorithm during the sharding phase, which consumes computing power to establish node identities rather than processing transactions. Further work will design corresponding resharding and node election mechanisms to improve the utilization of equipment computing power. Additionally, diverse case usage and performance evaluations will be conducted to improve other protocol components. These efforts are intended to enhance the protocol's utility and enable researchers to broadly explore its application potential in more new intelligent systems that require efficient data exchange and collaborative work, including but not limited to the IoTs with a large number of smart devices, sensors, and infrastructure. Notably, these computing environments continue to evolve, making further improvements in scalability critical for future research, which involves combining the sharding protocol with other conflict-free scaling solutions and designing a lightweight consensus mechanism as a component of the sharding protocol.

AUTHOR NOTE

This research was supported by Postgraduate Research & Practice Innovation Program of Jiangsu Province [KYCX22_3059].

REFERENCES

Afify, M., Loey, M., & Elsawy, A. (2022). A robust intelligent system for detecting tomato crop diseases using deep learning. *International Journal of Software Science and Computational Intelligence*, *14*(1), 1–21. doi:10.4018/IJSSCI.304439

Al Asad, N., Elahi, M. T., Al Hasan, A., & Yousuf, M. A. (2020). Permission-based blockchain with proof of authority for secured healthcare data sharing. In *the 2020 2nd International Conference on Advanced Information and Communication Technology (ICAICT)* (pp. 35-40). Dhaka, Bangladesh. doi:10.1109/ICAICT51780.2020.9333488

Al-Bassam, M., Sonnino, A., Bano, S., Hrycyszyn, D., & Danezis, G. (2017). Chainspace: A sharded smart contracts platform. *CoRR, abs/1708.03778*, 1—16. 10.1109/ICAICT51780.2020.9333488

Amiri, M. J., Agrawal, D., & El Abbadi, A. (2019). SharPer: Sharding permissioned blockchains over network clusters. In *Proceedings of the 2021 International Conference on Management of Data,* China. doi:10.1145/3448016.3452807

Asgaonkar, A. (2022). Scaling blockchains and the case for ethereum. In D. A. Tran (Ed.), *Handbook on blockchain* (pp. 197–213). Springer. doi:10.1007/978-3-031-07535-3_6

Baranwal, G., Kumar, D., & Vidyarthi, D. P. (2023). Blockchain based resource allocation in cloud and distributed edge computing: A survey. *Computer Communications*, *209*, 469–498. doi:10.1016/j.comcom.2023.07.023

Chen, H., & Wang, Y. (2019). Sschain: A full sharding protocol for public blockchain without data migration overhead. *Pervasive and Mobile Computing*, *59*, 101055. doi:10.1016/j.pmcj.2019.101055

Chen, Y., Li, M., Zhu, X., Fang, K., Ren, Q., Guo, T., Chen, X., Li, C., Zou, Z., & Deng, Y. (2022). An improved algorithm for practical byzantine fault tolerance to large-scale consortium chain. *Information Processing & Management*, *59*(2), 102884. doi:10.1016/j.ipm.2022.102884

Dang, H., Dinh, T. T. A., Loghin, D., Chang, E. C., Lin, Q., & Ooi, B. C. (2019). Towards scaling blockchain systems via sharding. In *Proceedings of the 2019 International Conference on Management of Data,* New York, NY, USA. doi:10.1145/3299869.3319889

Das, S., Krishnan, V., & Ren, L. (2020). Efficient cross-shard transaction execution in sharded blockchains. *CoRR, abs/2007.14521*, 1—16. https://doi.org/10.48550/arXiv.2007.14521

De Angelis, S., Aniello, L., Baldoni, R., Lombardi, F., Margheri, A., & Sassone, V. (2018). *PBFT vs. proof-of-authority: Applying the CAP theorem to permissioned blockchain.* In the 2nd Italian Conference on Cyber Security, ITASEC 2018 tenutosi a Milan, Italy. https://iris.uniroma1.it/handle/11573/1337256

Deepa, N., Pham, Q.-V., Nguyen, D. C., Bhattacharya, S., Prabadevi, B., Gadekallu, T. R., Maddikunta, P. K. R., Fang, F., & Pathirana, P. N. (2022). A survey on blockchain for big data: Approaches, opportunities, and future directions. *Future Generation Computer Systems*, *131*, 209–226. doi:10.1016/j.future.2022.01.017

Dinh, T. T. A., Wang, J., Chen, G., Liu, R., Ooi, B. C., & Tan, K.-L. (2017). Blockbench: A framework for analyzing private blockchains. In *Proceedings of the 2017 ACM International Conference on Management of Data* (pp. 1085-1100). ACM. doi:10.1145/3035918.3064033

Du, Z., Pang, X., & Qian, H. (2021). PartitionChain: A scalable and reliable data storage strategy for permissioned blockchain. *IEEE Transactions on Knowledge and Data Engineering*, *35*(4), 4124–4136. doi:10.1109/TKDE.2021.3136556

Fatemidokht, H., Rafsanjani, M. K., Gupta, B. B., & Hsu, C.-H. (2021). Efficient and secure routing protocol based on artificial intelligence algorithms with UAV-assisted for vehicular ad hoc networks in intelligent transportation systems. *IEEE Transactions on Intelligent Transportation Systems*, *22*(7), 4757–4769. doi:10.1109/TITS.2020.3041746

Fitzi, M., Gazi, P., Kiayias, A., & Russell, A. (2020). Proof-of-stake blockchain protocols with near-optimal throughput. *Cryptology ePrint Archive.* https://eprint.iacr.org/2020/037

Gao, W., Mu, W., Huang, S., Wang, M., & Li, X. (2021). Improved byzantine fault-tolerant algorithm based on alliance chain. *Wireless Communications and Mobile Computing*, *2021*, 1–10. doi:10.1155/2021/8455180

Ge, L., Wang, J., & Zhang, G. (2022). Survey of consensus algorithms for proof of stake in blockchain. *Security and Communication Networks*, *2022*, 1–13. doi:10.1155/2022/2812526

Guebli, W., & Belkhir, A. (2021). Inconsistency detection-based LOD in smart homes. [IJSWIS]. *International Journal on Semantic Web and Information Systems*, *17*(4), 56–75. doi:10.4018/IJSWIS.2021100104

Hashim, F., Shuaib, K., & Zaki, N. (2022). Sharding for scalable blockchain networks. *SN Computer Science*, *4*(1), 2. doi:10.1007/s42979-022-01435-z

Hellings, J., & Sadoghi, M. (2023). Byshard: Sharding in a byzantine environment. *The VLDB Journal*, 1—25. 10.1007/s00778-023-00794-0

Hong, Z., Guo, S., Li, P., & Chen, W. (2021). Pyramid: A layered sharding blockchain system. In *the IEEE INFOCOM 2021-IEEE Conference on Computer Communications* (pp. 1-10). BC, Canada. doi:10.1109/INFOCOM42981.2021.9488747

Huang, H., Peng, X., Zhan, J., Zhang, S., Lin, Y., Zheng, Z., & Guo, S. (2022). Brokerchain: A cross-shard blockchain protocol for account/balance-based state sharding. In *the IEEE INFOCOM 2022-IEEE Conference on Computer Communications* (pp. 1968-1977). London, United Kingdom. doi:10.1109/INFOCOM48880.2022.9796859

Khor, J. H., Sidorov, M., & Zulqarnain, S. A. B. (2023). Scalable lightweight protocol for interoperable public blockchain-based supply chain ownership management. *Sensors (Basel)*, *23*(7), 3433. doi:10.3390/s23073433 PMID:37050490

Kokoris-Kogias, E., Jovanovic, P., Gasser, L., Gailly, N., Syta, E., & Ford, B. (2018). Omniledger: A secure, scale-out, decentralized ledger via sharding. In the 2018 IEEE symposium on security and privacy (SP) (pp. 583-598). San Francisco, CA, USA. doi:10.1109/SP.2018.000-5

Li, C., Zhang, J., Yang, X., & Youlong, L. (2021). Lightweight blockchain consensus mechanism and storage optimization for resource-constrained IoT devices. *Information Processing & Management*, *58*(4), 102602. doi:10.1016/j.ipm.2021.102602

Li, D., Deng, L., Gupta, B. B., Wang, H., & Choi, C. (2019). A novel CNN based security guaranteed image watermarking generation scenario for smart city applications. *Information Sciences*, *479*, 432–447. doi:10.1016/j.ins.2018.02.060

Li, X., Lv, F., Xiang, F., Sun, Z., & Sun, Z. (2020). Research on key technologies of logistics information traceability model based on consortium chain. *IEEE Access : Practical Innovations, Open Solutions*, *8*, 69754–69762. doi:10.1109/ACCESS.2020.2986220

Li, Y., Qiao, L., & Lv, Z. (2021). An optimized byzantine fault tolerance algorithm for consortium blockchain. *Peer-to-Peer Networking and Applications*, *14*(5), 2826–2839. doi:10.1007/s12083-021-01103-8

Liu, Y., Xing, X., Cheng, H., Li, D., Guan, Z., Liu, J., & Wu, Q. (2023). A flexible sharding blockchain protocol based on cross-shard byzantine fault tolerance. *IEEE Transactions on Information Forensics and Security*, *18*, 2276–2291. doi:10.1109/TIFS.2023.3266628

Luu, L., Narayanan, V., Zheng, C., Baweja, K., Gilbert, S., & Saxena, P. (2016). A secure sharding protocol for open blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (pp. 17-30). ACM. doi:10.1145/2976749.2978389

Lv, L., Wu, Z., Zhang, L., Gupta, B. B., & Tian, Z. (2022). An edge-AI based forecasting approach for improving smart microgrid efficiency. *IEEE Transactions on Industrial Informatics*, *18*(11), 7946–7954. doi:10.1109/TII.2022.3163137

Mehraein, E., & Nourmohammadi, R. (2023). IGD-ScoreChain: A novel lightweight-scalable blockchain based on nodes sharding for the Internet of things. *Cryptology ePrint Archive*. https://eprint.iacr.org/2023/576

Memos, V. A., Psannis, K. E., Ishibashi, Y., Kim, B.-G., & Gupta, B. B. (2018). An efficient algorithm for media-based surveillance system (EAMSuS) in IoT smart city framework. *Future Generation Computer Systems*, *83*, 619–628. doi:10.1016/j.future.2017.04.039

Plageras, A. P., Psannis, K. E., Stergiou, C., Wang, H., & Gupta, B. B. (2018). Efficient IoT-based sensor BIG Data collection–processing and analysis in smart buildings. *Future Generation Computer Systems*, *82*, 349–357. doi:10.1016/j.future.2017.09.082

Qi, X., Zhang, Z., Jin, C., & Zhou, A. (2020). BFT-Store: Storage partition for permissioned blockchain via erasure coding. In *the 2020 IEEE 36th International Conference on Data Engineering (ICDE)* (pp. 1926-1929). IEEE. doi:10.1109/ICDE48307.2020.00205

Raj, M. G., & Pani, S. K. (2022). Chaotic whale crow optimization algorithm for secure routing in the IoT environment. *International Journal on Semantic Web and Information Systems*, *18*(1), 1–25. doi:10.4018/IJSWIS.300824

Sharma, P., Raj, B., & Gill, S. S. (2022). Spintronics based non-volatile MRAM for intelligent systems: Memory for intelligent systems design. *International Journal on Semantic Web and Information Systems*, *18*(1), 1–16. doi:10.4018/IJSWIS.310056

Sohrabi, N., & Tari, Z. (2020). ZyConChain: A scalable blockchain for general applications. *IEEE Access : Practical Innovations, Open Solutions*, *8*, 158893–158910. doi:10.1109/ACCESS.2020.3020319

Sohrabi, N., Tari, Z., Voron, G., Gramoli, V., & Fu, Q. (2022). SAZyzz: Scaling AZyzzyva to meet blockchain requirements. *IEEE Transactions on Services Computing*, *16*(3), 2139–2152. doi:10.1109/TSC.2022.3214976

Stuart, T. E., Ozdemir, S. Z., & Ding, W. W. (2007). Vertical alliance networks: The case of university–biotechnology–pharmaceutical alliance chains. *Research Policy*, *36*(4), 477–498. doi:10.1016/j.respol.2007.02.016

Tan, B., Chen, Y., Zhuang, G., Zhou, Y., & Dong, Z. (2023). A novel dynamic practical byzantine fault tolerance protocol based on node grouping. Authorea, *2023*(1), 1—3. 10.22541/au.168922622.23659736/v1

Tiwari, A., & Garg, R. (2022). Adaptive ontology-based IoT resource provisioning in computing systems. *International Journal on Semantic Web and Information Systems*, *18*(1), 1–18. doi:10.4018/IJSWIS.306260

Wang, C., & Raviv, N. (2021). Low latency cross-shard transactions in coded blockchain. In *the 2021 IEEE International Symposium on Information Theory (ISIT)* (pp. 2678-2683). IEEE. doi:10.1109/ISIT45174.2021.9518047

Wang, F., Ji, Y., Liu, M., Li, Y., Li, X., Zhang, X., & Shi, X. (2021). An optimization strategy for PBFT consensus mechanism based on consortium blockchain. In *Proceedings of the 3rd ACM International Symposium on Blockchain and Secure Critical Infrastructure* (pp. 71-76). ACM. doi:10.1145/3457337.3457843

Wang, G., Shi, Z. J., Nixon, M., & Han, S. (2019). Sok: Sharding on blockchain. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies* (pp. 41-61). ACM. doi:10.1145/3318041.3355457

Wang, J., & Wang, H. (2019). Monoxide: Scale out blockchains with asynchronous consensus zones. In *Proceedings of the 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)* (pp. 95—112). USENIX Association. https://www.usenix.org/conference/nsdi19/presentation/wang-jiaping

Wu, X., Jiang, W., Song, M., Jia, Z., & Qin, J. (2023). An efficient sharding consensus algorithm for consortium chains. *Scientific Reports*, *13*(1), 20. doi:10.1038/s41598-022-27228-1 PMID:36593262

Yang, J., Wang, J., Wong, C. W., & Lai, K.-H. (2008). Relational stability and alliance performance in supply chain. *Omega*, *36*(4), 600–608. doi:10.1016/j.omega.2007.01.008

Yu, G., Wang, X., Yu, K., Ni, W., Zhang, J. A., & Liu, R. P. (2020). Survey: Sharding in blockchains. *IEEE Access : Practical Innovations, Open Solutions*, *8*, 14155–14181. doi:10.1109/ACCESS.2020.2965147

Zamani, M., Movahedi, M., & Raykova, M. (2018). Rapidchain: Scaling blockchain via full sharding. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (pp. 931-948). ACM. doi:10.1145/3243734.3243853

Zhou, Q., Huang, H., Zheng, Z., & Bian, J. (2020). Solutions to scalability of blockchain: A survey. *IEEE Access : Practical Innovations, Open Solutions*, *8*, 16440–16455. doi:10.1109/ACCESS.2020.2967218

Zhu, L., Yu, H., Zhan, S., Qiu, W., & Li, Q. (2019). Research on high-performance consortium blockchain technology. *Journal of Software*, *30*(6), 1577–1593. doi:10.13328/j.cnki.jos.00573

Nigang Sun is an associate professor and deputy dean in School of Microelectronics and Control Engineering, Changzhou University, China. His research interests are in the areas of spread spectrum communication, stream cipher, consensus mechanism, and privacy protection in blockchain.

Junlong Li is a postgraduate at Changzhou University. His major research field is blockchain consensus mechanism and sharding protocol.

Yining Liu received his B.S. degree in applied mathematics from Information Engineering University, Zhengzhou, China, in 1995, his M.S. degree in computer software and theory from the Huazhong University of Science and Technology, Wuhan, China, in 2003, and his Ph.D. degree in mathematics from Hubei University, Wuhan, in 2007. He is currently a professor with school of Computer and Information Security, Guilin University of Electronic Technology, Guilin, China. His research interests include data privacy, security and privacy in VANETs, image security, data mining, and machine learning.

Varsha Arya did Master's degree from Rajasthan University, India in 2015 and has been working as a researcher for the last 7 years. She published more than 25 papers in top journals and conferences. Her research interests include business administration, technology management, Cyber physical systems, cloud computing, healthcare and networking. Currently, she is doing research at Asia University, Taiwan.