DRDST: Low-Latency DAG Consensus Through Robust Dynamic Sharding and Tree-Broadcasting for IoV

Runhua Chen, Haoxiang Luo, *Graduate Student Member, IEEE*, Gang Sun, *Senior Member, IEEE*, Hongfang Yu, *Senior Member, IEEE*, Dusit Niyato, *Fellow, IEEE*, and Schahram Dustdar, *Fellow, IEEE*

***Abstract*—The Internet of Vehicles (IoV) is emerging as a pivotal technology for enhancing traffic management and safety. Its rapid development demands solutions for enhanced communication efficiency and reduced latency. However, traditional centralized networks struggle to meet these demands, prompting the exploration of decentralized solutions such as blockchain. Addressing blockchain's scalability challenges posed by the growing number of nodes and transactions calls for innovative solutions, among which sharding stands out as a pivotal approach to significantly enhance blockchain throughput. However, existing schemes still face challenges related to *a*) the impact of vehicle mobility on blockchain consensus, especially for cross-shard transaction; and *b*) the strict requirements of low latency consensus in a highly dynamic network. In this paper, we propose a DAG (Directed Acyclic Graph) consensus leveraging Robust Dynamic Sharding and Tree-broadcasting (DRDST) to address these challenges. Specifically, we first develop a standard for evaluating the network stability of nodes, combined with the nodes' trust values, to propose a novel robust sharding model that is solved through the design of the Genetic Sharding Algorithm (GSA). Then, we optimize the broadcast latency of the whole sharded network by improving the tree-broadcasting to minimize the maximum broadcast latency within each shard. On this basis, we also design a DAG consensus scheme based on an improved hashgraph protocol, which can efficiently handle cross-shard transactions. Finally, the simulation proves the proposed scheme is superior to the comparison schemes in latency, throughput, consensus success rate, and node traffic load.**

Received 24 April 2025; revised 2 July 2025; accepted 13 August 2025. Date of publication 18 August 2025; date of current version 3 December 2025. This work was supported by the Innovation Fund of Glasgow College, University of Electronic Science and Technology of China, in part by the Singapore Ministry of Education (MOE) Tier 1 under Grant RG87/22 and Grant RG24/24, in part by the NTU Centre for Computational Technologies in Finance (NTU-CCTF), in part by the RIE2025 Industry Alignment Fund-Industry Collaboration Projects (IAF-ICP) under Grant I2301E0026, and in part by the A∗STAR. Recommended for acceptance by L. Kong. *(Runhua Chen and Haoxiang Luo contributed equally to this work.) (Corresponding author: Gang Sun.)*

Runhua Chen is with the Glasgow College, University of Electronic Science and Technology of China, Chengdu 611731, China, and also with the James Watt School of Engineering, University of Glasgow, G12 8QQ Glasgow, U.K. (e-mail: 2839936c@student.gla.ac.uk).

Haoxiang Luo, Gang Sun, and Hongfang Yu are with the School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: lhx991115@163.com; gangsun@uestc.edu.cn; yuhf@uestc.edu.cn).

Dusit Niyato is with the College of Computing and Data Science, Nanyang Technological University, Singapore 639798 (e-mail: dniyato@ntu.edu.sg).

Schahram Dustdar is with the Distributed Systems Group, TU Wien, 1040 Vienna, Austria, and also with the ICREA, 08002 Barcelona, Spain (e-mail: dustdar@dsg.tuwien.ac.at).

Digital Object Identifier 10.1109/TMC.2025.3599385

***Index Terms*—Internet of Vehicles, blockchain, sharding, tree-broadcasting, DAG consensus.**

I. Introduction

THE advent of the Internet of Vehicles (IoV) marks a significant milestone in the evolution of intelligent transportation systems. As a subset of the Internet of Things (IoT), IoV facilitates the exchange of data between vehicles and with infrastructure, enhancing traffic flow, safety, and user experience [1], [2]. To address the security needs of IoV, blockchain offers a promising avenue by enabling decentralized data storage and cryptographic assurance of data integrity [3], [4]. In particular, the consensus provides an autonomous decision-making paradigm for IoV that does not rely on a trusted third party [5]. It has the potential to enhance the trust among network participants, thereby facilitating efficient information exchange among vehicles in the IoV. However, the high-speed mobility and large-scale interaction of vehicle communication put forward extremely strict requirements on the rate of information exchange between vehicles. Traditional blockchains have shortcomings in delay and scalability due to complex consensus, broadcast processes.

To surmount these obstacles, sharding has emerged as a feasible solution. By partitioning the blockchain network into smaller, manageable segments, or "shards," it becomes possible to process transactions and achieve consensus in parallel, thereby enhancing throughput and reducing latency [5]. Initially, sharding schemes were predominantly based on random node allocation, such as Elastico [6], OmniLedger [7], and RapidChain [8]. However, random sharding schemes can compromise the security of blockchain systems, as a disproportionate number of Byzantine nodes may be aggregated within one shard, causing 51% of attacks. Therefore, the subsequent focus shifted towards incorporating trust mechanisms, such as Trust-Based Shard Distribution (TBSD) [9], and the methods proposed in [10] and [11]. This method can balance the concentration of malicious nodes within each shard. Furthermore, PolyShard [12] introduces a protocol for decentralized coded storage and computation in blockchains, leveraging polynomial coding to achieve linear scaling in both efficiency and security. DynaShard [13] integrates adaptive shard management, hybrid consensus, and state synchronization to optimize blockchain scalability under dynamic workloads and cross-shard transactions.

Additionally, the IoV involves a substantial amount of data exchange between vehicles and infrastructure, implying that the sharding system must achieve high throughput to efficiently manage traffic information. Hence, researchers have focused on combining sharding with DAG architecture in recent years. The sharding-hashgraph framework can provide a high-performance solution for blockchain [14], even with a low scalable Practical Byzantine Fault Tolerance (PBFT) consensus, which can greatly improve transaction throughput and scalability [15].

A. Research Motivation

Although the introduction of DAG can lead to a blockchain network with higher throughput, achieving rapid ordering of transactions remains the most significant challenge [16], [17], [18], especially under high-concurrency conditions. The necessity of transaction ordering in blockchain systems is crucial for supporting smart contracts, which rely on a consistent and irreversible sequence of events to execute contract terms autonomously and securely [17], [18]. This implies that in IoV, cross-shard communication resulting from the mobility of vehicles and transaction processing in such dynamic environments require more effective mechanisms to ensure data consistency and order.

Fortunately, tree-broadcasting [19], [20], [21], [22], as a structured protocol, establishes strict parent-child communi-cation relationships, reducing unnecessary message replication and transmission. This approach is particularly beneficial given that while the DAG structure facilitates parallel transaction processing, the introduction of sharding exacerbates the complexity of rapid transaction ordering and confirmation within the DAG. To harness the synergies between sharding and DAG, we ingeniously leverage tree-broadcasting as an important bridge between them, ensuring the quality of throughput with efficient communication.

Additionally, to construct a stable tree-broadcasting structure among nodes, robust sharding must be performed first, as sharding determines with whom to communicate, while the tree structure determines the mode. Prior works often address the benefits of sharding and tree-broadcasting in isolation [15], [22]. However, we identify a gap in recognizing the interdependent relationship between these two mechanisms. We reveal a critical insight: the two techniques are not merely related, but sequentially dependent, with sharding serving as the foundation for the subsequent tree-broadcasting structure. However, the unpredictable behavior of nodes implies that sharding should be dynamic, capable of adjusting the nodes divided into each shard in response to changes in the network environment, thereby ensuring the robustness of the sharding process. This is all in line with our key motivation: to meet the low-latency demands of IoV while achieving high throughput and a high consensus success rate.

B. Our Contributions

To address the shortcomings of existing work, we propose DRDST, a novel low-latency DAG consensus scheme tailored for IoV. The contributions of this paper are as follows:

- We design a network stability scoring criterion based on node continuous online time, computing power, and failure probability. This standard enables construction of robust and dynamically adaptive shard models.
- In order to solve a complex multi-objective optimization problem generated by the shard model, we design a new Genetic Sharding Algorithm (GSA), determine the most reasonable sharding scheme, and ensure a secure and more efficient consensus process.
- To reduce latency, we introduce the S-MLBT (S-Minimum Latency Broadcast Tree, where S denotes a set of q shards), denoting our tree-based broadcasting. This strategy ensures that information propagates efficiently within each shard, significantly reducing the maximum broadcast latency among shards.
- Shifting from a gossip-based to a tree-based communication pattern, we substantially enhance the performance of the traditional hashgraph protocol. Through the application of DAG consensus mechanism to sharding scenario, our approach efficiently handles cross-shard transactions, thereby directly addressing the challenges posed by vehicle mobility, ensuring integration and consistency of the blockchain in highly dynamic vehicular environments.

C. Organization of the Paper

The rest of the paper is arranged as follows. Section II reviews the related work. Section III delves into the system model, providing a comprehensive framework of our IoV network and DRDST architecture. In Section IV, we provide the details of the robust sharding model. In Section V, we introduce the S-MLBT, explaining our tree broadcasting protocol. Our DAG consensus scheme is covered in Section VI. Sections VII and VIII present the performance simulation of DRDST and comparison results with other schemes, respectively. Finally, Section IX concludes this paper.

II. Related Work

A. Sharding in IoV

The integration of sharding into the IoV landscape represents a significant stride towards addressing the efficiency and scalability challenges inherent in decentralized vehicular networks. Several pioneering studies have endeavored to harness the potential of sharding in IoV.

Singh et al. [10] introduced a decentralized trust management scheme for the IoV that leverages blockchain sharding to reduce the main chain's workload and increase transaction throughput. However, the scheme does not account for scenarios where vehicles frequently switch between different RSUs, posing challenges to its blockchain consensus. Zhang et al. [15] presented a dynamic network sharding approach to enhance the efficiency of the PBFT consensus algorithm in the IoV by minimizing the number of consensus nodes. This method aims to achieve a more focused and efficient consensus. However, the relatively stable nodes selected by the scheme may still lose connection due to sudden lane changes, acceleration, or deceleration of vehicles,

thereby leading to consensus failure. Chen et al. [23] introduced VT-chain, a vehicular trust blockchain framework that employs a multi-shard system to tackle scalability and performance issues within vehicular networks. It incorporates a hierarchical Byzantine Fault Tolerance (HierBFT) consensus protocol to maintain efficient consensus across shards. Despite the integration of Trusted Execution Environments (TEEs), its sharding configuration also ignores vehicles' frequent handover between RSUs. Consequently, these existing sharding schemes are deemed not fully applicable in highly dynamic vehicular environments due to their inability to effectively manage cross-shard transactions and their potential to lead to unstable connections that may result in consensus failures.

Based on these understandings, our work proposes an innovative RSU-level sharding model that aims to overcome the limitations of previous approaches. Dynamically adjust the nodes allocated to each shard as the network environment changes, and optimize the consensus mechanism that supports cross-shard transactions and reduces latency. Notably, vehicles serve only as client-side devices and not consensus nodes in our design, therefore they will not cause any shard topology changes. This distinction allows for simpler maintenance of a stable sharding architecture while only needing to accommodate impacts of vehicles' high mobility on cross-shard transactions.

B. Broadcasting

In broadcasting protocols, the existing relevant work has inspired further enhancements in the efficiency of broadcasting within IoV. Demers et al. [24] introduced gossip, a robust algorithm for replicating database updates across all replicas, but it has high message overhead and propagation delay issues and doesn't handle dynamic network topologies or efficient broadcasting well. Rohrer et al. [19] introduced the Kadcast protocol, which improves the traditional gossip-based broadcast mechanism by employing a structured tree approach derived from the Kademlia protocol [25], later enhanced with congestion control in Kadcast-NG [20]. However, Kadcast doesn't account for network heterogeneity and node dynamics. Zhu et al.'s MLBT [21] aimed to reduce latency and enhance stability in blockchains by leveraging network heterogeneity, but it lacks scalability and adaptability for highly dynamic networks. Zheng et al.'s DHBN [22] incorporates MLBT to address the challenges of network dynamics and heterogeneity in blockchain networks by maintaining multiple MLBTs and balancing relay tasks among participants.

In the realm of IoV, multi-hop broadcast is vital for timely data dissemination. Selecting the next hop in multi-hop routing is crucial for performance and application reliability. However, high mobility, dynamic network topologies, and complex channel conditions pose challenges to optimal next-hop selection. Palazzi et al. [26] proposed a fast multi-hop broadcast algorithm that may struggle with GPS limitations in areas with poor coverage or obstructions. Tian et al. [27] introduced a protocol based on geographic coordinates for Vehicular Ad Hoc Network (VANET), potentially ineffective in complex traffic scenarios. Wang et al. [28] developed a crowdsensing model with a cluster-optimized framework and a delay-sensitive routing algorithm for IoV event propagation, using stochastic theory and intersection data for relay selection, but it could be vulnerable to misinformation, affecting vehicle and traffic management.

The above studies provide insights that the characteristics of vehicles, such as their high mobility, and reliance on multi-hop wireless broadcasting, further complicate the task of information sharing and traffic management. Therefore, by selecting RSUs as consensus nodes instead of vehicles, we effectively minimize the adverse impacts of these vehicle characteristics. This allows us to focus on harnessing tree-broadcasting protocols' efficiency at the RSU level. Additionally, traditional MLBT-based works focuses on optimizing within a single blockchain network and does not address the optimization of cross-shard communication, while our trees enable cross-tree communication. This innovation is pivotal in facilitating cross-shard transactions and lays the foundation for constructing a DAG ledger.

C. DAG Consensus

In pursuing enhancing the throughput of blockchain and scalability, numerous studies have delved into integrating DAG with blockchain systems [29]. The GHOST protocol [30] deviates from the longest-chain rule by selecting the hea-viest subtree in blockchain forks, enhancing block validation but risking centralization. Conflux [16] uses a tree graph with parent and reference edges, complicating system implementation. IOTA's Tangle [31] replaces blocks with a DAG, eliminating mining and fees, but faces scalability and security issues, particularly Sybil attacks. SPECTRE [17] focuses on honest transaction sequencing through a voting algorithm within a DAG, but its complexity and computational demands are high. PHANTOM [18] builds on SPECTRE, creating a scalable consensus protocol that establishes a total order across a DAG of blocks (blockDAG), distinguishing between compliant and intransigent nodes.

In response to the burgeoning development of DAG technology, scholars have been proactively integrating DAG structures into the IoV systems to enhance the reliability and efficiency of data sharing. Lu et al. [32] introduce a hybrid blockchain architecture known as PermiDAG, which combines a permissioned blockchain with a local DAG to enhance data security and reliability in the IoV. Cui et al. [33] presented a Blockchain-Based Containerized Edge Computing Platform (CUTE) designed for the IoV. The platform schedules DAG-based computation tasks efficiently and integrates blockchain technology to enhance network security and reduce computation latency in IoV applications. Fu et al. [34] proposed a DAG blockchain-based framework that incorporates a two-stage Stackelberg game for optimizing bandwidth allocation and pricing strategies in the IoV.

However, the common limitation in existing DAG-based blockchain systems is the challenge of achieving rapid ordering of the transactions, particularly under high-concurrency conditions prevalent in IoV. Besides, cross-shard events are not efficiently handled. Our proposed scheme addresses these issues by integrating robust dynamic sharding with tree-broadcasting

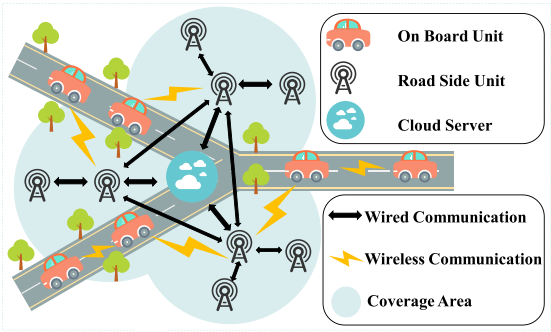

Fig. 1. The IoV model.

to optimize transaction processing and order, thereby enhancing the efficiency and security of consensus within the IoV network.

III. System Model

A. Network Model

At the vanguard of vehicular communication systems, IoV integrates advanced sensors, powerful processors, and resilient communication protocols, promising significant improvements in traffic safety, efficiency, and connectivity [35]. As shown in Fig. 1, our network model is designed with a focus on three critical components: the On-Board Unit (OBU), the Roadside Unit (RSU), and the Cloud Server (CS), each playing a distinct role in the ecosystem.

OBU: The OBUs are the mobile traffic data collectors with a dynamic topology that enables data aggregation from diverse sources. Despite their limited computational and storage capabilities and the instability of network connections due to mobility, they actively send the collected traffic data to each other and to the RSUs through wireless communication. Since they frequently switch between different RSUs, cross-shard transactions can be created.

RSU: The RSUs serve as the network's backbone, executing consensus algorithms due to their stationary nature, providing reliable connections, greater bandwidth, storage, and computational power compared to OBUs. The RSUs receive data from OBUs, utilize a tree-based communication pattern for consensus, and record outcomes on local ledgers before syncing with the Cloud Server.

CS: The CS is a trusted role managed by the government to offer various services to the network. It possesses substantial computational and storage capabilities. This strategy is similar to [36] and [37], where centralized entities are employed to provide essential services. CS aggregates consensus outcomes from RSUs across shards, maintains the global DAG ledger, and is in charge of the execution of the RSU sharding algorithm. The centralized ledger storage provides a comprehensive transaction record and facilitates the swift integration of new, trustworthy RSUs by enabling them to download the latest ledger and engage in consensus activities immediately. Nevertheless, it is important to acknowledge that this centralized approach may introduce security issues. To address the challenges, the robust key management mechanisms can effectively mitigate the risks associated with centralization and ensure the overall security of our proposed scheme [38], [39]. However, a detailed exploration of these mechanisms is beyond the scope of this paper.

Similar to [10] and [23], our strategy implements sharding at the RSU layer rather than at the vehicle layer. This approach leverages the fact that RSUs provide more stable connectivity and have superior computational capabilities than OBUs. However, both [10] and [23] neglect vehicles' frequent handover between RSUs. In our design, by sharding RSUs, vehicles function only as client-side devices rather than consensus nodes, which does not lead to shard topology changes. Besides, with this sharding strategy, the primary challenge posed by vehicles' frequent switch between RSUs could now be focused on efficiently handling a growing number of cross-shard transactions. To eventually serve for consensus, our approach takes full account of nodes' potential behavior and conducts a more comprehensive evaluation of each RSU before implementing sharding.

B. DRDST Architecture

Our proposed DRDST scheme has three major components, the robust sharding model, the S-MLBT broadcasting, and the DAG consensus. Fig. 2 depicts the overall architecture diagram of DRDST. The workflow starts from the initialization of RSUs distributed in different areas. Each RSU is annotated with its trust value and network stability score, which are critical metrics for the sharding algorithm. Upon the cloud server's execution of the algorithm named GSA, which is a more adopted method rather than random [6], [7], [8], RSUs are dynamically assigned to different robust shards. Within each shard, RSUs, guided by the establishment of S-MLBT, engage in a consensus process using an improved hashgraph algorithm. This process results in the formation of local blockchains by each RSU, which are then integrated into a global DAG ledger stored on the cloud server. The DAG structure ensures a coherent and secure ledger across the network, providing a comprehensive view of all transactions and enhancing the overall reliability of the IoV. The design details will be provided from Sections IV and V. The key notations used in our design are succinctly defined in Table I.

IV. Robust Sharding Model

A. Node Trust and Stability

Network dynamics refers to the frequent changes in the state of nodes within a network, such as the alterations in network topology caused by the movement of vehicles [40]. Conversely, network stability denotes the ability of a network to maintain consistency and reliability in the face of dynamic changes [1], [41].

In our sharding model, malicious nodes are defined as those who intentionally deviate from the protocol specifications to disrupt the consensus process. Their adversarial behavior is categorized under Byzantine faults, which encompass double-spending attacks and the intentional propagation of invalid transactions or false information, as well as other forms of malicious activities that undermine the integrity and reliability of the network. Given the unpredictable nature of network attacks, it is essential to

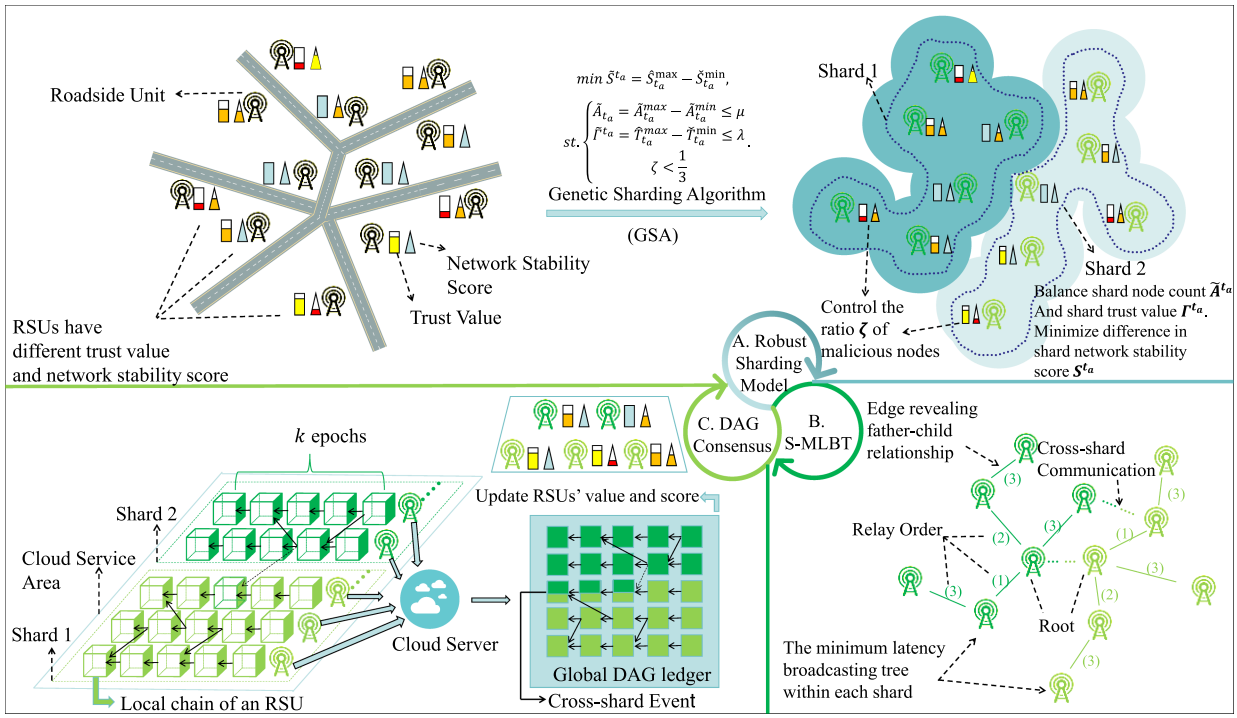

Fig. 2. Overall DRDST architecture: the Robust Sharding model, the S-MLBT broadcasting, and the DAG consensus. RSUs are first allocated to different shards, then establish tree-broadcasting structure, and eventually execute consensus.

TABLE I
KEY NOTATIONS

Notations	Definitions
X, Y	Set of nodes and set of shards, respectively
$\tau_i^{t_a}, s_i^{t_a}$	Trust value and network stability score of node x_i in time t_y, respectively
$\Gamma_j^{t_a}, S_j^{t_a}$	Trust value and the network stability score of shard j, respectively
$\theta(t_i^o)$	Trust reduction of node i communicating with node role o
$\hat{k}_i^{t_a}, \check{k}_i^{t_a}$	Number of successfully and unsuccessfully processed transactions by nodes i in term t_y, respectively
$t_{i,t_a}^o, T_{t_a}^o$	Actual and required processing time by node i to validate and package transactions into a micro block, respectively
T_{online}	Continuous online time of a node
C, η	Computational capacity and failure probability of a node
$\tilde{A}_{t_a}^{max}, \tilde{A}_{t_a}^{min}$	The maximum and minimum number of RSUs among s shards, respectively
$\hat{\Gamma}_{t_a}^{max}, \hat{\Gamma}_{t_a}^{max}$	The highest and lowest trust value among all the shards, respectively
$\hat{S}_{t_a}^{\max}, \check{S}_{t_a}^{min}$	The highest and lowest network stability scores among all the shards, respectively
$\tilde{A}_{t_a}, \tilde{\Gamma}^{t_a}, \tilde{S}^{t_a}$	The difference in node counts, trust value, and network stability score among all the shards, respectively
P_a, P_b, P_c	The three candidate solutions selected from the population
$M[j], T[j]$	The mutated and trial individual, respectively
$R_{(m,n)}, d_{(m,n)}$	The data transmission rate and the physical distance between RSU m and n, respectively

establish an evaluation system that assesses nodes based on their behavior during the consensus process. This system helps identify malicious nodes, and determine the extent of a node's influence within the network and whether it should be excluded. To ensure the robustness and reliability of the network, we have formed a new scoring criterion for the trust value and network stability score of nodes as key evaluation metrics. The trust value reflects the behavioral performance of nodes during consensus formation, while the network stability score provides

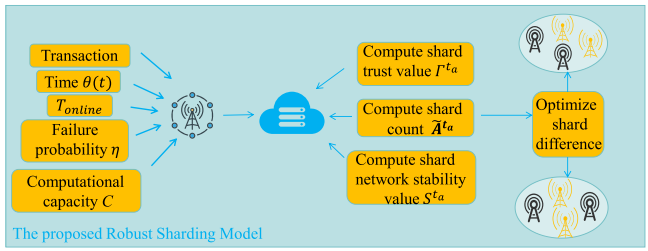

Fig. 3. The workflow of our robust sharding model.

a comprehensive assessment of node performance in the network environment.

The workflow of our model is illustrated in Fig. 3. Initially, the model calculates three critical metrics: node count, shard trust value, and shard network stability score. Subsequently, the model determines the differences in these metrics among all the shards. Finally, some related optimization objectives are utilized to guide the sharding algorithm. The detailed concepts and design of the model are provided as follows.

Let X Denote the Set of p nodes[1] in a Blockchain system

$$X = \{x_i | i \in \{1, 2, \ldots, p\}\} \tag{1}$$

where a node can transmit and validate transactions.

In a time range with a series of time slots $(t_0, \ldots, t_a, \ldots)$ node x_i in time t_a has the form

$$x_i = \left\langle Id, \tau_i^{t_a}, s_i^{t_a} \right\rangle \tag{2}$$

[1]The nodes involved in our model are all RSUs.

where Id, $\tau_i^{t_a}$ and $s_i^{t_a}$ are the node identification number, trust value, network stability score of node x_i, respectively.

Considering that each node will communicate with three types of nodes, namely Root (Ro), Father (Fa), and Child (Ch), according to the tree-broadcasting protocol which would be later introduced in Section IV, we compute $\tau_i^{t_a}$ as follows:

$$\tau_i^{t_a} = \alpha\left(\hat{k}_i^{t_a} - \check{k}_i^{t_a}\right) - \beta\left(\frac{\theta\left(t_i^{Ro}\right)}{T_{t_a}^R} + \frac{\theta\left(t_i^{Fa}\right)}{T_{t_a}^F} + \frac{\theta\left(t_i^{Ch}\right)}{T_{t_a}^C}\right) \tag{3}$$

$$\theta\left(t_{i,t_a}^o\right) = \begin{cases} 0, & t_{i,t_a}^o \le T_{t_a}^o,\ o\epsilon\{Ro,\ Fa,\ Ch\} \\ t_{i,t_a}^o, & \text{otherwise} \end{cases}, \tag{4}$$

where α and β are the trust reward and reduction coefficients, respectively; and $\theta(t_i^o)$denotes the trust reduction of node i communicating with node role o. $\hat{k}_i^{t_a}$ and $\check{k}_i^{t_a}$ are the number of successfully and unsuccessfully processed transactions by nodes i in term t_a. t_{i,t_a}^o and $T_{t_a}^o$ are the actual and required processing time by node i to generate a micro block in t_a, respectively.

In our scheme, the CS manages the global DAG ledger, which allows it to have a comprehensive view of each RSU's transaction processing status. Besides, the timestamps recorded in each event provide the actual time taken to process transactions, and required time is determined by the average processing time across all RSUs for generating a block. The CS utilizes them to accurately compute the trust values based on (3). Moreover, since each RSU maintains its own local chain, it enables RSUs to verify the honesty of the trust values calculated by the CS. In the event of discrepancies, RSUs can directly send a feedback message to the CS requesting a re-calculation. This bilateral verification mechanism, combining both local and global ledgers, facilitates mutual inspection and consensus on trust values.

The main idea behind (3) and (4) is that reaching consensus should be rewarded, while stalling time should be punished. Those who mine selfishly and do not want to forward the message, or directly forward the tampered message will all suffer from a decrease in trust value. The network stability score is directly related to the reliability and efficiency of nodes, data transmission, and verification in the sharded network. A node with poor network stability has a shorter continuous online time T_{online}, a lower computational capacity C and a higher failure probability η. We compute $s_i^{t_a}$ by

$$s_i^{t_a} = \sum_{k=1}^{b} w_k f_k\left(\Omega_k^{t_a}\right) \tag{5}$$

where $w_k > 0$ is the weight coefficient determined through Analytic Hierarchy Process (AHP) [42], corresponding to the $b$indicators, namely T_{online}, C, and η. Ω_k^{ta} represents the value of the k-th indicator at time t_a, and $f_k(\Omega_k^{t_a})$ is the nonlinear function that maps $\Omega_k^{t_a}$ to the score range of [0, [1]. We show f_k functions for different values of k in next three equations.

The score judging from $T_{online}$is calculated through the Sigmoid function, which provides a smooth output, mapping linear online data to non-linear values in the interval [0, [1].

$$f_1\left(T_{online}\right) = \frac{1}{1 + e^{-(T_{online} - T_0)}} \tag{6}$$

where T_0 is the average online time.

When an RSU joins the communication tree, it and its connected nodes send an online message to the CS with the join timestamp. Upon disconnection, they notify the CS with an offline timestamp. With these timestamps, the CS calculates each node's online duration and the overall T_0 across the network. Additionally, when an RSU leaves, the CS resets its T_{online} to zero.

From a statistical point of view, since RSU devices may be due to differences in cost, design, or functional requirements, the distribution of computing resources c is uneven, subject to a positive skew distribution. We use logarithmic functions to process the data, reduce the influence of extreme values in the distribution, and use min-max normalization to calculate the score judging from C.

$$f_2\left(C\right) = \frac{log\left(C\right) - min\left(log\left(C\right)\right)}{max\left(log\left(C\right)\right) - min\left(log\left(C\right)\right)}. \tag{7}$$

Assuming in the IoV, the probability of each RSU being attacked is independent of time, that is, each attack is an independent event [40]. In our standard, if the RSU has never been paralyzed ($\eta = 0$), the score is 1, indicating that the node is very stable. As failure probability η increases, the score gradually decreases, approaching 0. This feature can be characterized by an exponential decay function as follows:

$$f_3\ (\eta) = e^{-\gamma\eta} \tag{8}$$

where η represents the failure probability of the RSU, and γ is a positive adjustment parameter used to control the rate of decline of the scoring function.

This flexibility is essential in a network where conditions can change rapidly. For instance, a higher γ would lead to a faster decrease in score as the failure probability η increases, which could be desirable in environments where quick detection and response to potential security threats are critical.

B. Constrained Optimization Problem

Firstly, when sharding, the number of nodes within each shard must be taken into account. If two shards have the same shard trustworthiness values but different numbers of nodes, the shard with more nodes might have lower individual node trustworthiness. Let $\tilde{A}_{t_a}$represent the difference in node counts among shards as follows:

$$\tilde{A}_{t_a} = \tilde{A}_{t_a}^{max} - \tilde{A}_{t_a}^{min}, \tag{9}$$

where $\tilde{A}_{t_a}^{max}$and $\tilde{A}_{t_a}^{min}$are the maximum and minimum RSU counts among $q$shards. By minimizing $\tilde{A}_{t_a}$,we strive to achieve balanced consensus speeds across all shards.

Secondly, to prevent the trust value and network stability of some shards too low, we need to allocate nodes to shards as reasonably as possible. Otherwise, the shard may be a single shard taken over by malicious nodes, resulting in poor reliability and security of the entire blockchain sharding system. Let the trust value and network stability score of a shard be represented

by the average value of all the nodes in the shard, then, we have

$$\Gamma_j^{t_a} = \frac{\sum_{x_i \in \delta_j} \tau_i^{t_a}}{|\delta_j|} \tag{10}$$

$$S_j^{t_a} = \frac{\sum_{x_i \in \delta_j} s_i^{t_a}}{|\delta_j|} \tag{11}$$

where $\Gamma_j^{t_a}$ is the trust value of shard j, δ_j is the set of nodes in shard j and $S_j^{t_a}$ is the network stability score of shard j. Therefore, we have

$$\tilde{\Gamma}^{t_a} = \hat{T}_{t_a}^{max} - \check{T}_{t_a}^{\min} \tag{12}$$

$$\tilde{S}^{t_a} = \hat{S}_{t_a}^{\max} - \check{S}_{t_a}^{\min}, \tag{13}$$

where $\hat{T}_{t_a}^{max}$ and $\check{T}_{t_a}^{\min}$ are the highest and lowest trust values in q shards, respectively; $\hat{S}_{t_a}^{\max}$ and $\check{S}_{t_a}^{\min}$ are the highest and lowest network stability score in q shards, respectively. Through minimizing $\tilde{\Gamma}^{t_a}$ and $\tilde{S}^{t_a}$, we prevent security bottlenecks in the sharding system.

Since robustness is crucial for blockchain systems, we choose $\tilde{S}^{t_a}$ as the primary optimization target, while converting $\tilde{\Gamma}^{t_a}$ and $\tilde{A}_{t_a}$ into constraints.

$$\min \quad \tilde{S}^{t_a} = \hat{S}_{t_a}^{\max} - \check{S}_{t_a}^{\min},$$
$$\text{s. t.} \quad \begin{cases} \tilde{A}_{t_a} = \tilde{A}_{t_a}^{\max} - \tilde{A}_{t_a}^{\min} \le \mu \\ \tilde{\Gamma}^{t_a} = \hat{T}_{t_a}^{\max} - \check{T}_{t_a}^{\min} \le \lambda \\ \zeta < \frac{1}{3} \end{cases}, \tag{14}$$

where μ represents the acceptable difference in RSU shard node counts, λ represents the trust value threshold, and ζ represents the ratio of malicious nodes in the shard.

C. Design of Genetic Sharding Algorithm

Considering the Vast Number of Nodes and the Diversity of Trust Values and Network Stability Scores Among nodes, It is imperative to employ an algorithm capable of handling large-scale, multi-objective optimization challenges. The Genetic Algorithm (GA) [43], a heuristic search algorithm that simulates the process of biological evolution, stands out for its global search capability and robustness against variations in problem scale. Since fast response to a highly dynamic network is especially significant in IoV, we form our GSA to enable rapid and accurate sharding.

As depicted in Fig. 4, the workflow of the GSA, inspired by genetic principles, is designed to accommodate RSU sharding system and encompasses several critical steps.

1) Initialization of the Population: Let $P = \{P_1, P_2, \ldots, P_{population\ size}\}$ be the current population, where P_u represents a candidate solution, i.e., the mapping scheme from RSUs to shards. The initial candidate solution population is generated through a randomization mechanism, ensuring the diversity of the population and providing a broad exploration space for the evolution of the algorithm. Specifically, for an individual P_u in the population, where u indicates the index in the population. Each element $P_u[i]$ for nodes, where i is the index of the node, is randomly selected from the network. This initialization strategy not only ensures the diversity of the population but also provides a broad exploration space for the evolution of the algorithm. Then, we can express the initialization process as a mapping relationship from the node set X to the shard set Y. Let $Y = \{y_j | j \in \{1, 2, \ldots, q\}\}$. be the shard set, and the mapping function for initializing the population $f : X \leftarrow Y$ can be represented as:

$$x_i \mapsto y_j = f(x_i), \ where\ i \in \{1, 2, \ldots, p\}, j \in \{1, 2, \ldots, q\}. \tag{15}$$

Algorithm 1: Fitness Function.

Input: Set of nodes X, shard assignment T
Output: Fitness score f_s

$S \leftarrow$ default dict (list)
Step1: Assign all nodes to their corresponding shards and form all the shards.
for each (i, shard) in enumerate (T) **do**
  S[shard] $\leftarrow x_i$
end for
Step2: Calculate trust value, network stability score and node count of each shard.
for s_j in S **do**
  Calculate $\Gamma_j^{t_a}$ using Eq. (10)
  Calculate $S_j^{t_a}$ using Eq. (11)
  $\tilde{A}_i^{t_y} \leftarrow |y_j|$
end for
Step3: Calculate the difference of the three metrics among all the shards.
Calculate $\tilde{\Gamma}^{t_a}$ using Eq. (12)
Calculate $\tilde{S}^{t_a}$ using Eq. (13)
Calculate $\tilde{A}_{t_a}$ using Eq. (9)
Step4: Apply penalty value to judge whether any shard exceeds predefined threshold on these metrics.
penalty $\psi \leftarrow 0$
if $\tilde{\Gamma}^{t_a} > \lambda$ or $\tilde{A}_{t_a} > \mu$ or $\zeta > \frac{1}{3}$ or $\tilde{S}^{t_a} > 0.15$ **then**
  $\psi \leftarrow \psi + 1000$
  $f_s \leftarrow \Gamma_j^{t_a} + \tilde{A}_{t_a} + \tilde{S}^{t_a} + \zeta + \psi$
end if
return f_s

2) Iteration: The iterative process is the driving force behind genetic algorithms, which continuously optimizes the population through selection, crossover, and mutation operations. The iterative process can be described as follows.

- *Selection operation:* Randomly select three different candidate solutions P_a, P_b and P_c from the current population P as samples. This can ensure each candidate solution has an equal probability be selected.
- *Mutation operation:* We denote the network stability score of an RSU at gene position (located shard) j in P_a as $s_a[j]$ and in P_b as $s_b[j]$. With the Mutation Factor (MF) as the

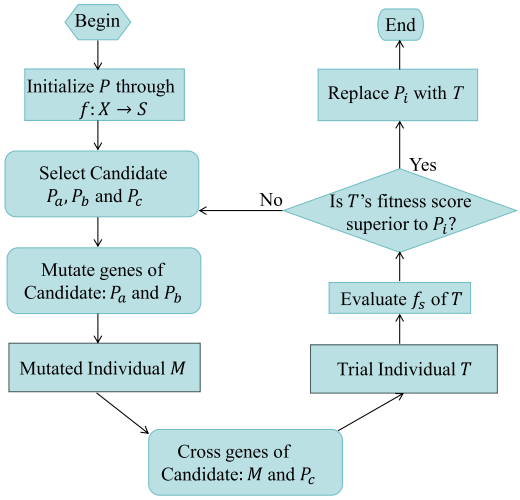

Fig. 4. The workflow of genetic sharding algorithm.

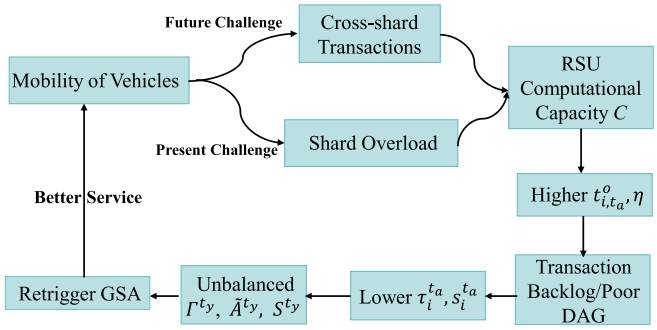

Fig. 5. The impacts of mobility of vehicles on blockchain consensus and shard reconfiguration.

condition, choose genes from P_b to replace corresponding positions in P_a, generating a mutated individual M.

$$M[j] = \begin{cases} P_b[j] & if\ rand(0,1) < MF\left(1 - \frac{s_a[j]+s_b[j]}{2s_{max}}\right) \\ P_a[j] & otherwise \end{cases} \tag{16}$$

where s_{max} is the maximum network stability score among all RSUs. The MF is here modified to be inversely proportional to the network stability score, encouraging the retention of more stable RSUs.

- *Crossover operation:* Let $s_M[j]$ and $s_{P_c}[j]$ denote the network stability scores of the RSUs at position j in M and P_c, respectively. With the crossover probability (CP) as the condition, choose genes from the mutated individual M to replace corresponding positions in P_c, generating a trial individual T.

$$T[j] = \begin{cases} M[j] & if\ rand(0,1) < CP\left(1 - \frac{s_M[j]+s_{P_c}[j]}{2s_{max}}\right) \\ P_c[j] & otherwise \end{cases} \tag{17}$$

This modified CP ensures that RSUs with higher network stability are more likely to be selected for the new sharding configuration.

3) Fitness Evaluation: It is crucial for evaluating the quality of candidate solutions. In GSA, the fitness function takes into account the trust value of nodes within each shard $\tilde{T}_j^{ta}$, network stability score S_j^{ta}, and the size of the shard $\tilde{A}_i^{ta}$.If any shard exceeds the predefined threshold on these metrics, a penalty term Ψ will be applied to the fitness score f_s of the trial individual, guiding the algorithm toward better shard structures. This process is summarized in Algorithm 1.

4) Selection Operation: If T's fitness is superior to P_u, then use T to replace P_u to form a new population P'. The next iteration will operate on this new population.

Since the trust value and network stability score of each node are updated regularly, the CS continuously monitors the difference of these metrics across shards. If the differences between shards exceed the predefined thresholds (see Algorithm 1, Line 18), the CS triggers a re-sharding process. This dynamic adjustment ensures that the influence of malicious nodes on consensus results is minimized, thereby alleviating the challenges posed by the Byzantine problem.

In addition to Byzantine problems, the impacts of the mobility of vehicles can also be minimized. As shown in Fig. 5, when vehicles move, they introduce challenges to the network. Shard overload occurs when the computational capacity C of RSUs is exceeded, leading to higher processing time t^o_{i,t_a} and failure probability η, consequently causing transaction backlogs and poor DAG performance. This overload is further exacerbated by the future challenge of cross-shard transactions, which become more frequent as vehicles switch between different RSUs. These transactions require additional computational resources and coordination across shards, increasing the complexity and potential for overload. However, when CS retriggers GSA to reallocate resources, the sharding structure can better accommodate the dynamic nature of vehicle mobility, thereby improving service quality.

D. Complexity Analysis

We consider both the time complexity and the space complexity of the GSA. The time complexity primarily focuses on the time required for the execution of the algorithm, while the space complexity is concerned with the storage space needed during the execution process. *N* is defined as the population size, which is the total number of individuals considered in each generation; *p* as the total number of nodes, representing the complexity of the problem space; and *I* as the length of each individual's genome, indicating the number of genes or parameters to be optimized.

1) Time Complexity Analysis: The algorithm initiates with a random selection of three unique individuals from the population, incurring $O(N)$ complexity due to the population-wide iteration to prevent duplicates. Mutation follows, comparing genes of two individuals per genome, resulting in $O(I)$ complexity as each gene must be examined. Crossover mirrors this with $O(I)$ complexity, iterating over genes to form trial individuals. Fitness evaluation incurs $O(p)$ complexity, requiring a scan of all nodes and shards. The final selection and replacement operation, assessing and potentially updating individuals in the population, carries $O(Np)$ complexity.

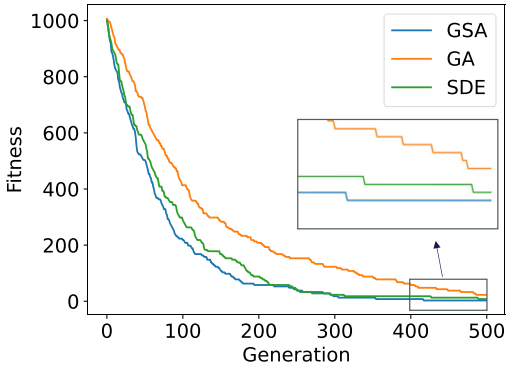

Fig. 6. The convergence trend comparison.

Collectively, these operations yield a total time complexity of $O(GNp)$ for the GSA, with G denoting generations. While this complexity may appear significant at first glance, it reflects the algorithm's ability to explore a large solution space based on cloud server's abundant computational resources. Importantly, it is a necessary trade-off for achieving the robustness, adaptability, and high performance required in dynamic IoV networks.

2) Space Complexity Analysis: GSA requires storage for the population and any auxiliary structures used during the fitness evaluation. The population storage alone demands $O(NI)$ space, as each individual in the population must be represented along with their corresponding genes. The new population generated in each iteration also requires the same amount of space, leading to a combined space complexity for population storage of $O(NI)$. Additionally, the auxiliary space for calculating trust scores, stability scores, and shard sizes during the fitness evaluation requires $O(p)$ space, as it involves maintaining lists, dictionaries, sets and tuples for all nodes. Consequently, the total space complexity of the GSA is $O(NI + p)$. This analysis underscores the algorithm's spatial efficiency and the resources needed to support its operations, which is essential for practical applications where memory and storage are critical considerations.

E. Quality of Solutions

To analyze the quality of solutions produced by the GSA, the convergence trend of the average fitness over generations is employed as a critical performance indicator. This metric evolves towards optimal solutions with each successive generation. In comparison, we compare GSA with Zhang et al.'s [44] sharding method based on Differential Evolution (SDE) and traditional GA [43]. Specifically, we execute each algorithm 1000 times, and for each run, the best fitness value at every generation is recorded. We then aggregate the 1000 best values to obtain the mean fitness for each generation.

The convergence trend of the average fitness over generations, as depicted in Fig. 6, demonstrates effectiveness in identifying and refining high-quality solutions. The GSA exhibits the steepest decline in the early generations, underscoring its most rapid improvement in solution quality during this critical phase. Furthermore, the GSA maintains the lowest average fitness as the algorithms progress into the gradual plateau phase. This suggests that the GSA is not only efficient in the initial stages but also continues to refine solutions effectively, achieving a near-optimal state in a limited number of generations. This convergence trend is a testament to GSA's potential for real-world applications in optimizing sharding in blockchain-enabled IoV systems.

V. S-Minimum Latency Broadcast Tree

Based on GSA, the RSU network is divided into multiple shards to ensure robustness and low latency of the system.

We construct an MLBT within each shard to further optimize latency, adopting a tree-based communication pattern instead of gossip. To optimize the latency of the entire RSU network, it is ultimately necessary to optimize the MLBT with the maximum latency among the constructed q shards. The problem is formally defined as follows. To build an MLBT for each shard, we need to determine for each root node its set of paths to the other all nodes within the same shard.

The latency is the sum of data transmission latency and propagation latency. The former refers to the latency caused by data transmission over wired channels between microservices. Microservices are the fine-grained, independently deployable service components within the IoV's edge computing environment. The latter refers to the time it takes for a signal to travel through the propagation medium. Thus, the total latency between RSU m and n is

$$t_{wired} = \begin{cases} \frac{L}{R_{m,n}} + \frac{d_{m,n}}{v}, & m \neq n \\ 0, & otherwise \end{cases} \tag{18}$$

where L is the size of the data packet transmitted, $R_{m,n}$ is the data transmission rate between RSU m and n, $d_{m,n}$ is the physical distance between RSU m and n, and v is the propagation speed of the signal in the medium. For each shard, let W_j be the set of all paths p_j^g within the shard j.

$$W_j = \left\{ p_j^g | g \in \{1, 2, \ldots, h\} \right\} \tag{19}$$

where h denotes the number of paths connected by the root node $r_j$in shard j. Our goal is to find a set of paths W_j that satisfies

$$W_j = \arg\min_{W_j} \max_{p_j^g \in W_j} \sum_{E \in p_j^g} t_{E\_wired}. \tag{20}$$

where E is an edge representing one hop from one node to another on the path p_j^g, and t_{E_wired} denotes the latency from RSU m to n on edge E.Thus, the optimal solution for the broadcast latency of the entire network should satisfy

$$\min \max_{1 \le i \le q} \left(\max_{p_j^g \in W_j} \sum_{E \in p_j^g} t_{E\_wired} \right). \tag{21}$$

Although (21) as a NP-complete problem lacks exact polynomial-time solutions [45], it can be solved by a common heuristic or approximate algorithm. And the heuristic algorithm, which involves evaluating multiple candidate solutions and iterating over different nodes, can be executed in parallel, leveraging multi-threading to significantly reduce computation time.

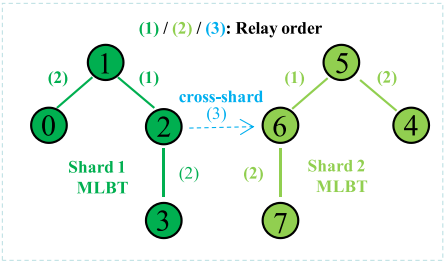

Fig. 7. MLBT Structure of Shard 1 and Shard 2 ($H = 2$). The arrow indicates the cross-shard communication direction of parent and child nodes.

Specifically, the construction and evaluation of candidate broadcast trees can be distributed across multiple processing units. By employing these parallelization strategies, we can ensure that the S-MLBT protocol remains computationally efficient, thereby maintaining the overall performance and scalability of our proposed DRDST scheme.

After solving the problem, we can establish MLBT in each shard. For instance, as is shown in Fig. 7, there are two MLBTs with height $H = 2$ originating from shard 1 and shard 2, respectively. RSU 1 and RSU 5 are root nodes for each MLBT. When RSU 2 from shard 1 initiates consensus, it will broadcast the events to its father node RSU 1 and child node RSU 3. When synchronizing an event, RSU 2 will broadcast the event to fan-out node RSU 6 via cross-shard communication. In Sections VII and VIII, we will further demonstrate the performance of our S-MLBT, including the effects on consensus success rate and propagation speed. Through simulations, a balance between the robustness and low latency of our model could be justified.

VI. DAG Consensus

A. Improved Hashgraph

Gossip faces high redundancy and difficulty in ensuring consistency when handling cross-shard transactions. Therefore, the traditional gossip-based hashgraph [46] for DAG consensus makes it difficult to achieve efficient transaction ranking. To avoid these problems, we propose an improved hashgraph to enable DAG consensus. It optimizes the efficiency and security of cross-shard transactions, fundamentally addressing the complexities introduced by the mobility of vehicles in the IoV.

In the hashgraph architecture, each node maintains a local chain that is part of the global DAG. This DAG is built through inter-node references, establishing parent-child relationships between events. Each event contains two cryptographic hashes: one for its self-parent and another for its other-parent. This interlinked structure forms an immutable chain, where each event is cryptographically bound to others, enhancing the system's resistance to tampering. Hashgraph ensures that all nodes eventually see every transaction through a consensus that maintains the immutability and order of the ledger.

In our DRDST, nodes propagate new events via the S-MLBT protocol, which surpasses gossip in network-wide information dissemination. An event achieves consensus and is recorded on an RSU's local chain once it garners over two-thirds YES votes within its shard [46], and is then broadcast to other

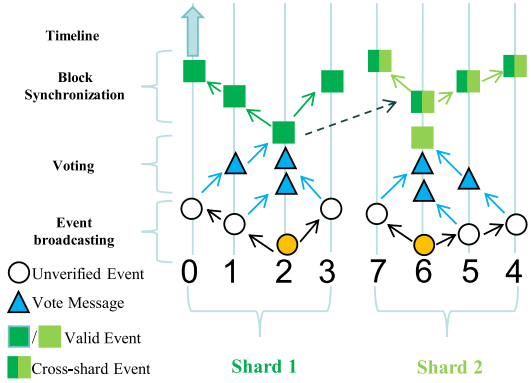

Fig. 8. DAG consensus based on improved hashgraph.

shards through cross-tree communication. Utilizing hashgraph, receiving nodes integrate new events with existing transactions and continue broadcasting, ensuring cross-shard transactions are processed and confirmed within the global DAG, as shown in Fig. 8.

To further illustrate our sharded DAG scenario, a global DAG ledger stored in the cloud server is depicted in Fig. 9. During the first two epochs, there are no cross-shard transactions on the DAG ledger, as no vehicles have switched between RSUs from different shards. For instance, Event *A* is solely the result of intra-shard communication, where RSU 1 broadcasts known transactions to RSU 2, and RSU 2 subsequently packages them.

The first event containing cross-shard transactions on the ledger is Event *B*, which is generated by RSU 3 after RSU 2 synchronizes the transactions with it via cross-tree communication. This indicates that during epoch 3, a vehicle is moving between shard 1 and shard 2. Although no cross-shard communication occurs between shard 2 and shard 3 during the first four epochs, upon the retriggering of GSA, RSU 5 is reassigned to shard 3, leading to the formation of Event *D* through intra-shard communication. Concurrently, under the new sharded network configuration, if a vehicle moves between RSU 4 and RSU 5, this will again result in the generation of Event *E* via cross-tree communication. It is important to note that Event *E* is the first event to have visibility across all RSUs.

All events visible to Event *E* are enclosed within purple boxes in Fig. 9, signifying that as Event *E* is synchronized across the network, the events formed in earlier epochs will also be seen by the majority of RSUs. For example, by the end of epoch 8, Event *A* can be seen by the most recent events packaged by RSUs 1-5, while Event *C* can be seen by the most recent events packaged by RSUs 3-7. Therefore, all transactions will eventually be strongly seen by RSUs across all involved shards, with the hashgraph protocol securing a global order for all events through its DAG structure and consensus. Cryptographic linkage among events deters tampering, ensuring transaction security. Consequently, when vehicles frequently switch between RSUs across different shards, our sharded DAG ensures visibility of cross-shard transactions among RSUs, enabling rapid processing of subsequent transactions upon a vehicle's cross-shard movement, thus maintaining efficient, resource-light network operation.

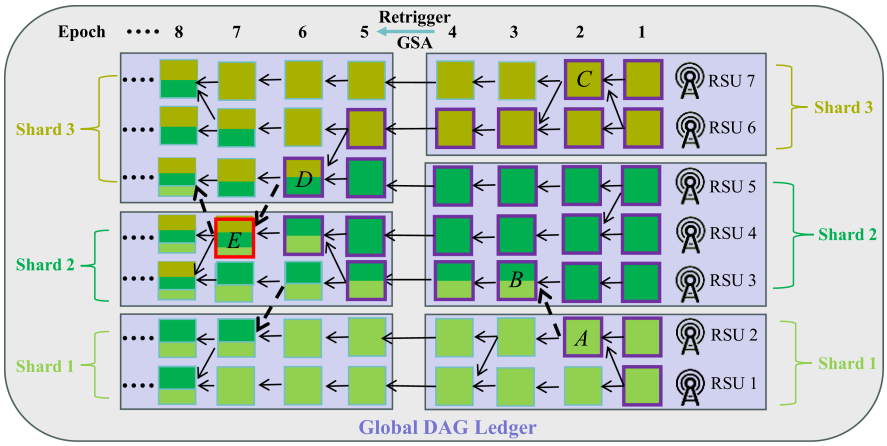

Fig. 9. The global DAG ledger stored by the CS.

B. Security Analysis

Sybil attack: This attack in distributed systems refers to a malicious strategy where an adversary creates a multitude of pseudonymous identities to gain undue influence over a system that relies on democratic processes or consensus mechanisms. Such attacks threaten the integrity and security of blockchain networks. Traditional hashgraph's resistance to Sybil attacks is derived from its non-permissioned structure, which makes it economically prohibitive for an adversary to control a significant portion of the network with fake identities. Thus, as long as our design combines a secure identity authentication mechanism, such as [1], it can easily resist this attack.

View split attack: This attack exploits nodes' limited visibility of transactions to deceive them into validating and spreading fraudulent transactions. Fortunately, our DAG consensus has resistance to view split attacks.

We set each shard to have u RSUs, and f as the maximum number of faulty or Byzantine nodes, where $u \geq 3f + 1$. Consider two conflicting cross-shard transactions involving these q shards, Tx_1 and Tx_2. Let U_1 and U_2 represent the sets of RSUs that strongly see these transactions. According to [40], the cardinality of these sets in the sharded network must satisfy

$$\begin{aligned} |U_1| &> 2qf + q, \\ |U_2| &> 2qf + q. \end{aligned} \tag{22}$$

Given the total number of RSUs $qu \geq 3qf + q$, the intersection of U_1 and U_2 must contain at least $q(f + 1)$ RSUs.

$$|U_1 \cap U_2| \geq q\,(f + 1)\,, \tag{23}$$

which implies that at least $q(f + 1)$ RSUs can concurrently observe both Tx_1 and Tx_2. Since the maximum number of malicious nodes in the whole network is qf, at least q honest RSUs can accurately discern the legality of the transactions and vote accordingly. An illegal cross-shard transaction will not receive more than $2qf$ votes, thus failing to achieve strong visibility and preventing the view split attack.

TABLE II
SIMULATIONS PARAMETERS

Parameters	Values
Research area	100/200 km^2
Number of RSUs	100/200
Number of vehicles	[1000, 10000]
Speed of vehicles	20/40/60/80 km/h
Client request speed	2000/3000 transactions per second
Distance between two RSUs	[500,1500] m
RSU bandwidth	[10,30] Mb/s
Maximum transactions within one event	1024
Shard count	4/6/8/10/12
Trust value of each RSU	[0,10]
Network stability score of each RSU	[0,1]
Number of fan-out nodes of each RSU	8
RSU Byzantine fault rate	2%-15%
RSU offline/inactive rate	2%-10%

VII. PERFORMANCE SIMULATION

Our DRDST contains many functional components. To verify their effectiveness of them, the propagation time of S-MLBT, and the trust value and stability score of RSU are simulated in this section. In addition, to verify the collaboration of these components, we also conducted ablation tests.

A. Simulation Setup

Our computer is equipped with an Intel i9-14900HX processor, a 2.2 GHz CPU, and an NVIDIA GeForce RTX 4070 GPU with 8 GB memory. We implement a simulation of the network and consensus layer for the IoV based on Python 3.0, which accommodates a customizable number of participating clients and nodes. To ensure that our configura-tion is close to real-world conditions, we incorporate the data transmission rate between RSUs and propagation speed to be the same as [49]. Furthermore, motivated by the need to reflect a realistic deployment scenario, we have considered two areas: one with 100 RSUs and another with 200 RSUs, allowing for a comprehensive evaluation of the network performance. All the parameters are listed in Table II.

B. Propagation Time of S-MLBT

In this part, we analyze the propagation speed of our S-MLBT. To demonstrate this performance, we characterize by

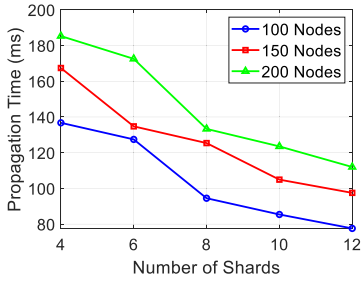

Fig. 10. Propagation time of S-MLBT.

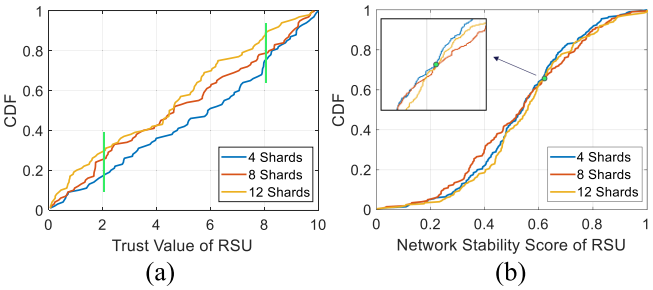

Fig. 11. (a) CDF of RSUs' trust values. (b) CDF of RSUs' network stability scores.

the propagation time it takes for the last leaf node among all the trees to receive the request message from its corresponding root node. The client request speed is set to be 3000 (tps, *transactions per second*).

As shown in Fig. 10, the propagation time decreases as the number of shards increases. This reduction is due to transactions being distributed across more shards, thereby reducing the broadcast load on each individual tree. Notably, when the number of shards increases from 6 to 8, the propagation time for both 100-node and 200-node networks experiences a more significant drop. This is because, inside a tree with height of $H \geq 2$, the number of RSUs should range between 2^H and $2^{H+1} - 1$. A lower tree height implies shorter paths for message propagation. These findings highlight the efficient propagation speed of our S-MLBT.

C. Trust Value and Stability Score of RSU

In this part, we utilize the Cumulative Distribution Function (CDF), corresponding to the trust value and network stability score of RSUs, to provide deeper insights into the performance of DRDST. Specifically, we deploy 200 RSUs and set the client request speed to 3000 *tps*.

Fig. 11 shows that under 4 shards, the proportion of RSUs with a trust value below 2 is merely approximately 18%, with the majority of RSUs exhibiting trust values ranging between 0.2 and 0.6. Notably, nearly 24% of the nodes achieve a trust score exceeding 0.8. Conversely, in the scenario with 12 shards, the proportion of RSUs with a trust value below 0.2 approaches 30%. Additionally, only about 10% of nodes reaching a trust score above 8. The 8-shard network presents an intermediate distribution, where approximately 24% of RSUs have trust values below 0.2, and around 20% exceed 8.

However, as depicted in Fig. 11, the CDF of network stability scores for all sharding scenarios exhibit similar distributions. Approximately 5% of RSUs have scores between 0 and 0.2, the majority fall within the 0.3 to 0.6, and about 35% surpass 0.6. Notably, post the intersection at a stability score around 0.63, the curve representing 4-shard network remains at the top. It has been revealed that in tree structures, the detachment of nodes closer to the root can cause a greater number of leaf nodes to become disconnected, thereby leading to fewer nodes with high stability scores and affecting consensus.

D. Ablation Study

In this part, we perform an ablation study to empirically confirm the necessity and synergistic benefits of each component to the overall superiority of DRDST. In subsequent simulations, the system without our DAG consensus is represented as w/o DAG. It only permits one leader to initiate consensus within a shard at any given time, while the remaining nodes act solely as participants in the consensus process. The system without the S-MLBT protocol is represented as w/o S-MLBT. It employs gossip-based communication for message dissemination among nodes. The system without the robust sharding model is represented as w/o Sharding. It will utilize a random sharding method.

We extensively evaluated the ablation test performance of the DRDST scheme by the following four metrics.

Latency (ϑ):

$$\vartheta = \frac{1}{Z}\sum_{c=1}^{Z}\left(T_{\text{con},c} - T_{\text{sub},c}\right). \tag{24}$$

where Z is the total number of transactions, $T_{\text{con},\,c}$ is the time when the c-th transaction reaches consensus and $T_{\text{sub},c}$ is the time when the c-th transaction is submitted by the client.

Throughput ($\mathbf{\Phi}$):

$$\Phi = \frac{\sum_{i=1}^{q} T_{Xi}}{t}, \tag{25}$$

where T_{Xi} is the total number of transactions processed by RSU i in the period t.

Consensus Success Rate ($\mathbf{\Upsilon}$):

$$\Upsilon = \frac{T_{Xc}}{T_{X\text{r}}}, \tag{26}$$

where T_{Xc} is the number of transactions reached consensus and T_{Xr} is the total number of requested transactions.

Node Traffic Load ($\boldsymbol{\omega}$):

$$\omega = \frac{\sum_{i=1}^{q} D_i}{q}, \tag{27}$$

where D_i is the total volume of data transmitted by RSU i in the network. The unit for ω is MegaByte MB.

Latency: Fig. 12(a) shows that system latency rises with more RSUs due to prolonged message propagation. The full DRDST model consistently shows the lowest latency, which moderates as RSUs increase, demonstrating effective manage-ment at scale. Removing DAG leads to escalating latency, underscoring its role

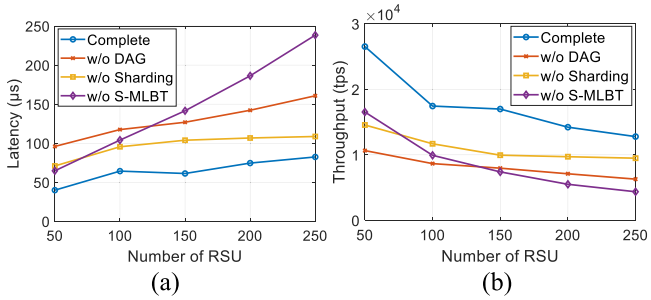

Fig. 12. (a) Ablation study for latency. (b) Ablation study for throughput.

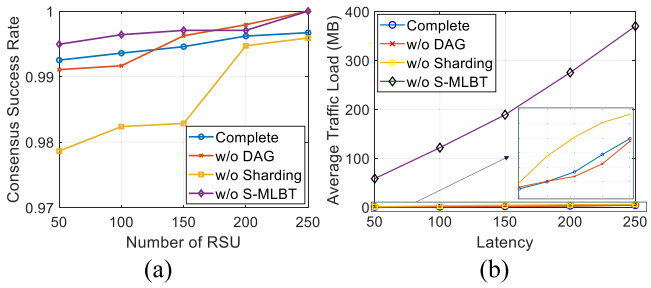

Fig. 13. (a) Ablation study for consensus success rate. (b) Ablation study for node traffic load.

in transaction ordering and cross-shard efficiency. The higher latency in the system without sharding highlights the importance of parallel processing for scalable networks. Model without S-MLBT sees a sharp latency increase, confirming the protocol's critical role in broadcast optimization. The DAG-lacking model consistently shows higher latency than the one without sharding, indicating DAG's greater contribution to network scalability and the importance of managing cross-shard transactions.

Throughput: Fig. 12(b) shows that all systems' throughput decreases as RSUs increase due to growing communication demands. The complete DRDST model retains the highest throughput, preventing hot shards and transaction backlogs effectively. The DAG-lacking model has lower throughput due to impaired high-concurrency transaction processing. Model without sharding, lacking parallel processing, also shows reduced throughput. The S-MLBT-deficient model, initially second in throughput with 50 RSUs, declines sharply to the lowest at 250 RSUs, highlighting S-MLBT's critical role in network broadcast latency and communication efficiency.

Consensus Success Rate: Fig. 13(a) shows that the consensus success rate rises with increasing RSUs due to redundancy enhancing alternative consensus paths, even amidst RSU failures. The DRDST model sustains high success rates, affirming its effectiveness. The system lacking sharding suffers the lowest rates, vulnerable to malicious nodes in shards. The DAG-lacking model, despite an initial lower rate, shows an upward trend with network expansion, suggesting DAG's throughput advantages might sacrifice robustness. This highlights the importance of balancing performance and robustness in IoV consensus mechanism design. Despite potential trade-offs in robustness associated with S-MLBT, the complete DRDST model consistently achieves a highly acceptable consensus success rate, which will be verified in the next section of comparison simulations.

Node Traffic Load: Fig. 13(b) shows that node traffic load increases with more RSUs due to heightened communication demands for synchronization and consensus. The full DRDST model shows lower traffic loads, efficiently managing transactions and avoiding central node congestion. The DAG-lacking model exhibits decreased traffic due to reduced communication complexity from concurrent consensus initiation. Model without sharding has higher traffic loads, as sharding is key to distributing transactions and preventing overload. The S-MLBT-deficient model tops traffic loads, underscoring the inefficiency of gossip-based communication, which leads to redundant transmissions and higher load.

VIII. Comparison Simulation

In this section, we compare DRDST with two methods through simulations: the Tree-Based Gossip Protocol (TBGP) [47], and the Graphical Consensus-Based Sharding (GCS) [48].

TBGP: It constructs the tree-based broadcasting protocol to optimize the randomness in traditional gossip, thereby reducing redundancy in message dissemination and improving consensus efficiency. It also employs federated learning to select nodes for constructing the TBGN, ensuring network stability and efficiency. Then, it introduces a block weight-based chain selection algorithm to achieve effective global ordering of blocks.

GCS: It uses graphical consensus within the shard, forming consensus groups through maximum connected subgraphs and electing leaders based on node reliability. Data within each shard is stored locally on the chain, while the main chain adopts a DAG structure, to support cross-shard sharing. Additionally, the GCS scheme incorporates shard backup and node scheduling strategies to address shard failures and overheating issues.

In this comparison, we also extensively evaluate TBGP, GCS, and DRDST schemes by the four metrics in Section VII-D. Simulation parameters are also referred to Table II. Through comparative analysis, we prove that the DRDST scheme meets the real-time requirements of vehicle networking while achieving high throughput and consensus success rate, and has efficiency advantages in transaction processing.

A. Latency

Initially, we set up 100 remote RSUs within a 100 km^2 area with a client request speed of 3000 tps. Fig. 14(a) shows that the latency decreases as the shard count increases. DRDST achieves the lowest latency in all cases by balancing latency among shards and utilizing S-MLBT for efficient communication. TBGP has the highest latency since it focuses more on node stability to ensure robustness. GCS has lower latency than TBGP because it uses asymmetric encryption to protect off-chain data, thus improving computing latency.

Subsequently, we expand the scope to encompass an area with 200 RSUs, while maintaining the same client request speed. This extension enables us to investigate the network's scalability and performance with heightened service unit density. Fig. 14(b) shows that latency increases for all schemes due to the growing

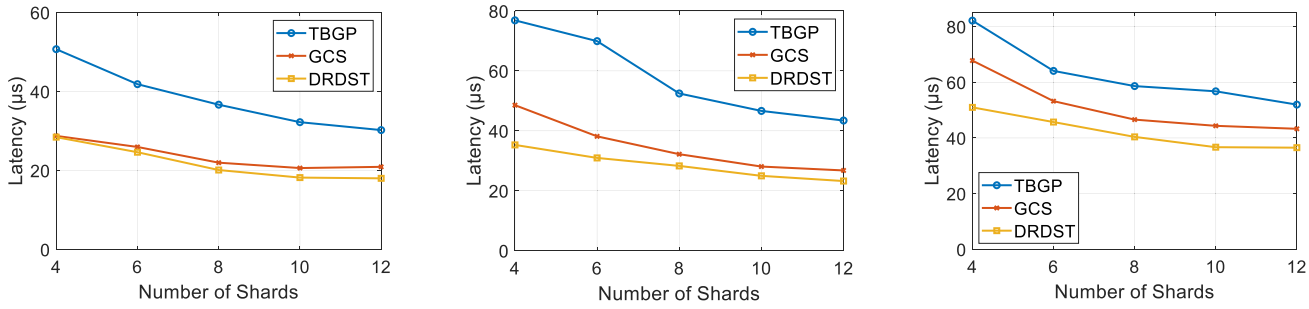

Fig. 14. Latency comparison (a) 100 RSUs and 3000 tps. (b) 200 RSUs and 3000 tps. (c) 200 RSUs and 2000 tps.

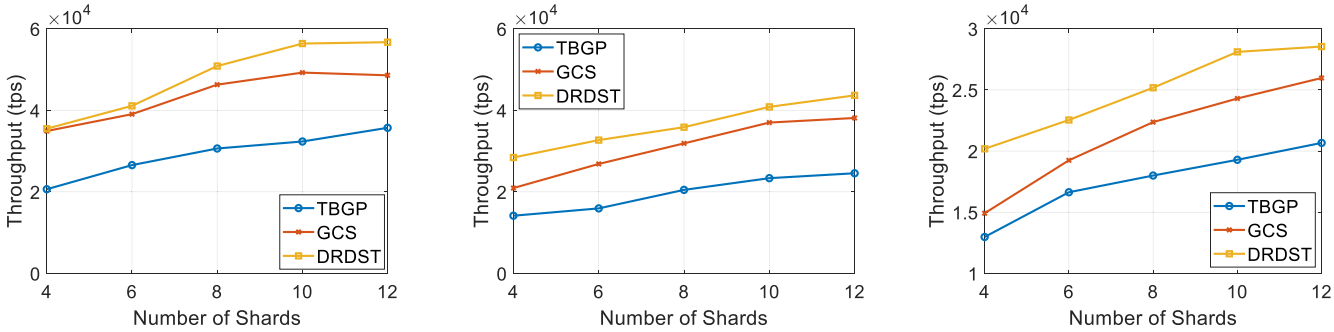

Fig. 15. Throughput comparison (a) 100 RSUs and 3000 tps. (b) 200 RSUs and 3000 tps. (c) 200 RSUs and 2000 tps.

number of nodes in each shard, which demands more extensive communication. DRDST continues to showcase the lowest latency in all cases. However, TBGP's focus on ensuring shard stability and GCS's focus on geographical location may both lead to a single shard becoming a bottleneck.

Ultimately, we decrease the client request speed to 2000 tps and maintained 200 RSUs in the network. As shown in Fig. 14(c), the latency for all three schemes experiences a further increase under various shard configurations, suggesting that the allocation of resources across shards may not be optimally adapted to the decreased speed. For instance, if certain RSUs become bottlenecks, latency at these points may continue to rise even with an overall reduction in load. Nonetheless, DRDST consistently exhibits the lowest latency in all cases, indicating that the system is capable of reconfiguring resources and adjusting task distribution when shifting from a high-load to a low-load state. This validates the dynamic efficacy of our sharding approach.

B. Throughput

First, we set up 100 RSUs, with the client request speed set at 3000 tps. Fig. 15(a) shows that as shard count increases, the parallelism is enhanced, leading to an increase in throughput. DRDST achieves the highest throughput through a balanced sharding strategy and tree-broadcasting protocol. GCS outperforms TBGP by incorporating node scheduling to handle shard overload and expedite the verification of backlog transactions. GCS's lower throughput relative to DRDST is likely due to PBFT's requirement for multiple message exchange rounds, leading to significant communication overhead.

Next, we extend our analysis to include an area with 200 RSUs, while maintaining the same client request speed. Fig. 15(b) indicates that throughput decreases for all schemes due to the increased node count per shard, which slows transaction propagation and complicates consensus. Despite this, DRDST maintains superior throughput. As shard count increases, GCS's throughput peaks early, TBGP's grows slowly, and DRDST shows consistent improvement. This demonstrates DRDST's superior adaptability to network size changes, optimizing communication routes and resource allocation for sustained high performance.

Finally, we reduced the client request speed to 2000 tps while maintaining a network with 200 RSUs. Fig. 15(c) reveals a throughput reduction for all schemes, likely due to a mismatch between request speed and network capacity. While higher client request speeds do not guarantee increased throughput, the drop to 2,000 tps may significantly underutilize the network's capacity, leading to unrealized throughput gains. Despite this, DRDST consistently outperforms in throughput, highlighting the benefits of dynamic sharding for efficient network resource utilization.

C. Consensus Success Rate

Commencing our analysis, we set up 100 RSUs under the client request rate of 3000 tps. Fig. 16(a) shows that the consensus success rate decreases as the shard count increases, as malicious nodes exploit the smaller group to their advantage. DRDST achieves the highest rate and minimal decline, due to sharding through node trust and network stability score. As the

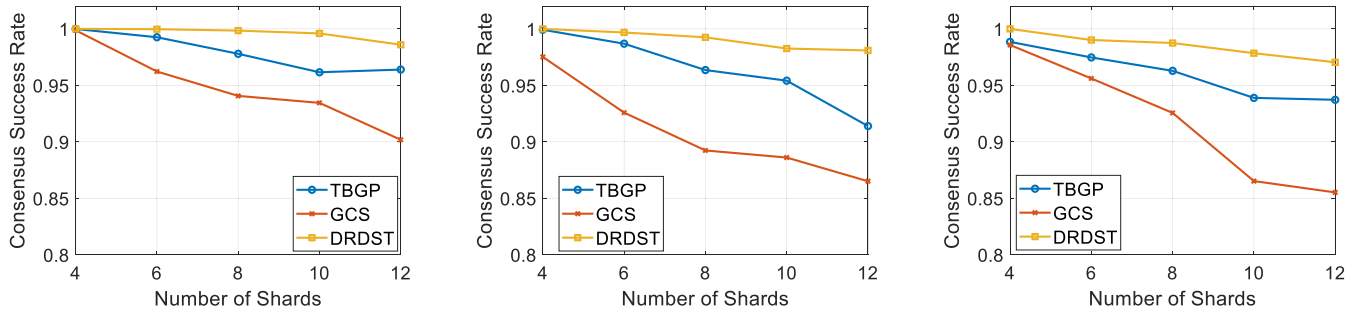

Fig. 16. Consensus success rate (a) 100 RSUs and 3000 tps. (b) 200 RSUs and 3000 tps. (c) 200 RSUs and 2000 tps.

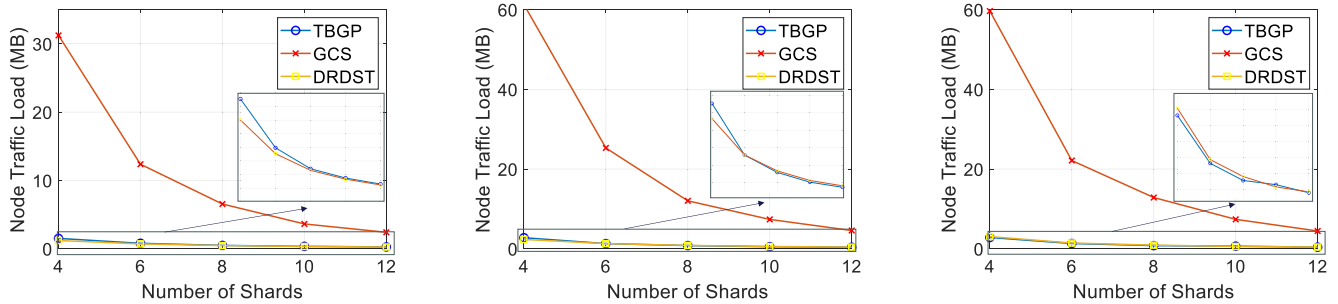

Fig. 17. Node traffic load (a) 100 RSUs and 3000 tps. (b) 200 RSUs and 3000 tps. (c) 200 RSUs and 2000 tps.

shard count rises, GCS suffers the worst decrease in rate, due to its reliance on RSU geography rather than performance metrics. Compared with GCS, TBGP has a higher consensus success rate since it classifies nodes according to their stability.

Following the initial setup, we expand the network to 200 RSUs, persisting with the client request speed at 3000 tps. We elevate the Byzantine fault rate to better assess system behavior under a high fault rate, essential for evaluating network robustness at scale. Fig. 16(b) shows that the consensus success rate declines for all schemes, reflecting that a higher Byzantine fault rate leads to fewer commit messages or YES votes for consensus initiators. Initially, with 4 shards, all schemes achieve high consensus success rates due to a lower proportion of malicious nodes in each shard. However, as shard count increases, GCS experiences the most significant drop, attributed to its reliance on RSU geography rather than performance, potentially compromising shard integrity. In contrast, TBGP consistently outperforms GCS, as FL categorizes nodes by performance and stability.

Ultimately, we opt to reduce the client request speed to 2000 tps while maintaining a consistent 200 RSUs. In Fig. 16(c), DRDST still demonstrates the highest consensus success rate across all shard counts. Besides, as the shard count increases, the DRDST scheme exhibits the smallest decrease at 2.95%, compared to 5.19% for TBGP and 13.25% for GCS, reaffirming the effectiveness of our robust sharding model.

D. Node Traffic Load

Initiating our analysis, we consistently set up 100 RSUs under the client request speed of 3000 tps. Fig. 17(a) shows that as shard count increases, the traffic load per node diminishes due to the broader distribution of transactions. In GCS, nodes bear the heaviest traffic load due to the communication-intensive consensus process in PBFT [50], [51], [52]. In contrast, TBGP and DRDST use a structured tree-based topology to optimize data flow, limiting each node to communicate only with its parent and children nodes, thus avoiding extensive pairwise messaging. Notably, DRDST exhibits the lowest node traffic load, validating the enhanced efficiency of our S-MLBT protocol.

Expanding the network to 200 RSUs with a client request speed of 3000 *tps*, Fig. 17(b) shows that while traffic load nearly doubles across all schemes, those using the tree-broadcasting protocol, notably DRDST, maintain significantly lower node loads. Specifically, DRDST's sharding model, which balances shard disparities, more effectively prevents hot shards, unlike GCS's geographical sharding, which can result in uneven load distribution and increased node loads.

Adjusting the client request speed to 2000 *tps* with 200 RSUs, Fig. 17(c) indicates that nodes in PBFT consensus experience reduced traffic due to smaller packets with fewer transactions, while tree-broadcasting protocols face increased loads due to necessary dynamic adjustments for rate changes, especially if root node adjustments are not timely, leading to load imbalances. At lower shard counts, DRDST's frequent link reconfigurations seeking for minimum latency slightly raise node traffic over TBGP, due to the overhead of tree detachment and reconnection.

E. Mobility of Vehicles

When vehicles switch between RSUs across shards, efficiently managing cross-shard transactions is essential for a

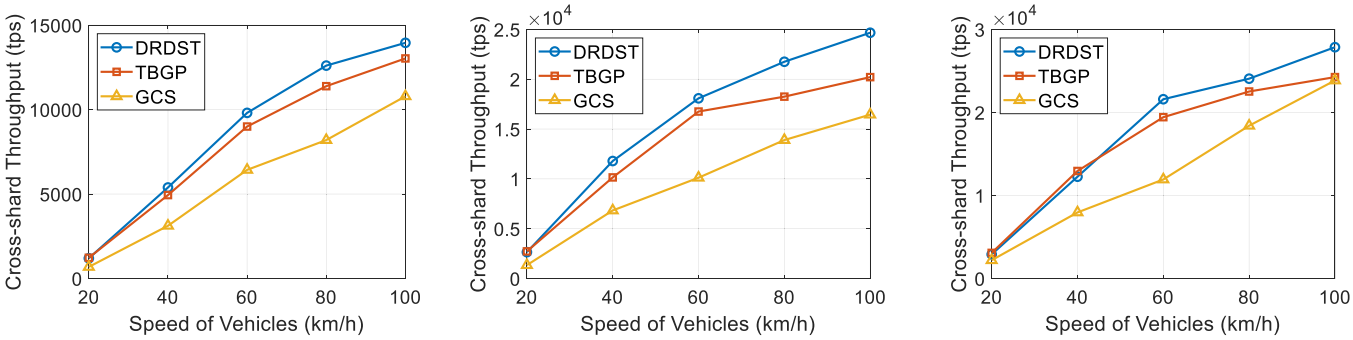

Fig. 18. Cross-shard throughput (a) 4 shards (100 RSUs). (b) 8 shards (100 RSUs). (c) 12 shards (100 RSUs).

sharding system to maintain blockchain consensus. In this part, we specifically assess how the mobility of vehicles affects these schemes' throughput of cross-shard transactions in a 100 km^2 area with 100 RSUs by varying speeds of vehicles and shard counts.

Fig. 18(a) shows that throughput increases with higher mobility in a 4-shard setup, with DRDST outperforming others due to its S-MLBT mechanism for efficient cross-shard communication. TBGP has lower throughput due to its non-latency-optimized broadcasting tree. Besides, PBFT's communication efficiency in GCS is also poor, hindering cross-shard transaction processing.

When the shard count is 8, Fig. 18(b) shows that throughput continues to rise for all schemes since vehicles encounter more shards at the same speed, leading to increased cross-shard transactions. However, TBGP's throughput grows slowly beyond 60 km/h, possibly due to FL's slow adaptation to network dynamics, leading to transaction backlogs. DRDST and GCS maintain steady growth, but sharding by RSU geography rather than performance poorly manages cross-shard transactions.

At 12 shards, Fig. 18(c) shows only a slight further gain in throughput for all schemes, regardless of vehicle speed. Beyond 60 km/h, although DRDST maintains the highest throughput, the growth slackens, indicating the increasing difficulty in managing cross-shard transactions. The throughput gap between GCS and TBGP initially widens and then narrows. This may be because, at lower speeds, TBGP's communication efficiency is more apparent, while at higher speeds, GCS's shard backup and node scheduling improve resource allocation for consensus, enhancing performance.

IX. Conclusion

In this paper, we have proposed a low-latency and robust DAG consensus to address the efficiency challenges for the IoV. We have presented a comprehensive approach that encompasses the development of a robust sharding model, the establishment of S-MLBT for optimized broadcast latency, and the implementation of an improved hashgraph protocol to efficiently manage cross-shard transactions. In this way, the impacts of vehicle mobility on blockchain consensus are well alleviated. Simulation results and ablation studies confirm the critical role of each component in achieving optimal network performance and its superiority over other advanced schemes, underscoring the importance of a balanced approach that considers both performance and robustness. Based on these, blockchain can now truly serve for low-latency data management in IoV. Through deployment and implementation of efficient consensus, a trustworthy framework for traffic information sharing is established. Users will actively engage in vehicle connectivity and contribute to IoV development. In future work, while achieving rapid sharding, we will explore the feasibility of integrating intelligent learning algorithms suitable for IoV to further optimize sharding.

References

[1] H. Luo et al., "ESIA: An efficient and stable identity authentication for Internet of Vehicles," *IEEE Trans. Veh. Technol.*, vol. 73, no. 4, pp. 5602–5615, Apr. 2024.

[2] W. Chen, W. Yang, M. Xiao, L. Xue, and S. Wang, "LBDT: A lightweight blockchain-based data trading scheme in Internet of Vehicles using proof-of-reputation," *IEEE Trans. Mobile Comput.*, vol. 24, no. 4, pp. 2800–2816, Apr. 2025.

[3] M. Kamal, G. Srivastava, and M. Tariq, "Blockchain-based lightweight and secured V2V communication in the Internet of Vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 3997–4004, Jul. 2021.

[4] Y. Xu, S. Zhang, C. Lyu, J. Liu, T. Taleb, and S. Norio, "TRIMP: Three-sided stable matching for distributed vehicle sharing system using Stackelberg Game," *IEEE Trans. Mobile Comput.*, vol. 24, no. 2, pp. 1132–1148, Feb. 2025.

[5] H. Luo, G. Sun, H. Yu, B. Lei, and M. Guizani, "An energy-efficient wireless blockchain sharding scheme for PBFT consensus," *IEEE Trans. Netw. Sci. Eng.*, vol. 11, no. 3, pp. 3015–3027, May/Jun. 2024.

[6] L. Luu et al., "A secure sharding protocol for open blockchains," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*., 2016, pp. 17–30.

[7] E. Kokoris-Kogias et al., "OmniLedger: A secure, scale-out, decentralized ledger via Sharding," in *Proc. IEEE Symp. Secur. Privacy*, 2018, pp. 583–598.

[8] M. Zamani, M. Movahedi, and M. Raykova, "RapidChain: Scaling blockchain via full sharding," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 931–948.

[9] J. Yun, Y. Goh, and J.-M. Chung, "Trust-based shard distribution scheme for fault-tolerant shard blockchain networks," *IEEE Access*, vol. 7, pp. 135164–135175, 2019.

[10] P. K. Singh, R. Singh, S. K. Nandi, K. Z. Ghafoor, D. B. Rawat, and S. Nandi, "Blockchain-based adaptive trust management in internet of vehicles using smart contract," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 6, pp. 3616–3630, Jun. 2021.

[11] X. Cai et al., "A sharding scheme-based many-objective optimization algorithm for enhancing security in blockchain-enabled Industrial Internet of Things," *IEEE Trans. Ind. Inform.*, vol. 17, no. 11, pp. 7650–7658, Nov. 2021.

[12] S. Li, M. Yu, C.-S. Yang, A. S. Avestimehr, S. Kannan, and P. Viswanath, "PolyShard: Coded sharding achieves linearly scaling efficiency and security simultaneously," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 249–261, 2021.

[13] A. Liu et al., "DynaShard: Secure and adaptive blockchain sharding protocol with hybrid consensus and dynamic shard management," *IEEE Internet Things J.*, vol. 12, no. 5, pp. 5462–5475, Mar. 2025.
[14] N. Gao, R. Huo, S. Wang, T. Huang, and Y. Liu, "Sharding-hashgraph: A high-performance blockchain-based framework for Industrial Internet of Things with hashgraph mechanism," *IEEE Internet Things J.*, vol. 9, no. 18, pp. 17070–17079, Sep. 2022.
[15] X. Zhang, R. Li, and H. Zhao, "A parallel consensus mechanism using PBFT based on DAG-lattice structure in the Internet of Vehicles," *IEEE Internet Things J.*, vol. 10, no. 6, pp. 5418–5433, Mar. 2023.
[16] C. Li et al., "Scaling Nakamoto consensus to thousands of transactions per second," 2018, *arXiv:1805.03870*.
[17] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "Spectre: A fast and scalable cryptocurrency protocol," *Cryptol. ePrint Arch.*, Report 2016/1159, 2016. [Online]. Available: https://eprint.iacr.org/2016/1159
[18] Y. Sompolinsky, S. Wyborski, and A. Zohar, "PHANTOM: GHOSTDAG—A scalable generalization of Nakamoto consensus," in *Proc. 3rd ACM Conf. Adv. Financial Technol.*, New York, NY, USA, 2021, pp. 57–70.
[19] E. Rohrer and F. Tschorsch, "Kadcast: A structured approach to broadcast in blockchain networks," in *Proc. 2021 ACM Conf. Adv. Financial Technol.*, 2019, pp. 199–213.
[20] E. Rohrer and F. Tschorsch, "Kadcast-NG: A structured broadcast protocol for blockchain networks," *IEEE/ACM Trans. Netw.*, vol. 31, no. 6, pp. 3269–3283, Dec. 2023.
[21] Y. Zhu et al., "Design of low-latency overlay protocol for blockchain delivery networks," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2022, pp. 1182–1187.
[22] R. Zheng, H. Luo, G. Su, and H. Yu, "DHBN: An efficient broadcast protocol for blockchain networks in highly dynamic heterogeneous environment," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2024, pp. 1–6.
[23] X. Chen, G. Xue, R. Yu, H. Wu, and D. Wang, "A vehicular trust blockchain framework with scalable byzantine consensus," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 4440–4452, May 2024.
[24] A. Demers et al., "Epidemic algorithms for replicated database maintenance," in *Proc. ACM Symp. Princ. Distrib. Comput.*, 1987, pp. 1–12.
[25] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the XOR metric," in *Proc. Int. Workshop Peer-to-Peer Syst.*, 2002, pp. 53–65.
[26] C. E. Palazzi et al., "How do you quickly choreograph inter-vehicular communications? A fast Vehicle-to-Vehicle multi-hop broadcast algorithm, explained," in *Proc. 4th IEEE Consum. Commun. Netw. Conf.*, 2007, pp. 960–964.
[27] D. Tian et al., "A distributed position-based protocol for emergency messages broadcasting in vehicular ad hoc networks," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1218–1227, 2018.
[28] X. Wang et al., "Optimizing content dissemination for real-time traffic management in large-scale Internet of Vehicle systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1093–1105, Feb. 2019.
[29] X. Qu, S. Wang, K. Li, J. Huang, and X. Cheng, "TidyBlock: A novel consensus mechanism for DAG-based blockchain in IoT," *IEEE Trans. Mobile Comput.*, vol. 24, no. 2, pp. 722–735, Feb. 2025.
[30] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in Bitcoin," in *Proc. 19th Int. Conf. Financial Cryptogr. Data Secur.*, 2015, pp. 507–527.
[31] S. Popov, "The tangle," *IOTA Found., White paper*, vol. 1, 2016, Art. no. 3.
[32] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Blockchain empowered asynchronous federated learning for secure data sharing in Internet of Vehicles," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4298–4311, Apr. 2020.
[33] L. Cui et al., "A blockchain-based containerized edge computing platform for the Internet of Vehicles," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2395–2408, Feb. 2021.
[34] Y. Fu, S. Wang, Q. Zhan, and D. Zhang, "Game model of optimal quality experience strategy for Internet of Vehicles bandwidth service based on DAG blockchain," *IEEE Trans. Veh. Technol.*, vol. 72, no. 7, pp. 8898–8913, Jul. 2023.
[35] K. N. Qureshi, S. Din, G. Jeo, and F. Piccialli, "Internet of Vehicles: Key technologies, network model, solutions and challenges with future aspects," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 3, pp. 1777–1786, Mar. 2021.
[36] D. Abbasinezhad-Mood and H. Ghaemi, "Dual-signature blockchain-based key sharing protocol for secure V2V communications in multi-domain IoV environments," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 10, pp. 13407–13416, Oct. 2024.
[37] M. Shen, H. Lu, F. Wang, H. Liu, and L. Zhu, "Secure and efficient blockchain-assisted authentication for edge-integrated Internet-of-Vehicles," *IEEE Trans. Veh. Technol.*, vol. 71, no. 11, pp. 12250–12263, Nov. 2022.
[38] Z. Yang et al., "QBMA-BIV: Quantum-key-distribution (QKD)-based multi-server authentication scheme for blockchain-enabled Internet of Vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 11, pp. 18433–18448, Nov. 2024.
[39] S. Bojjagani, Y. C. A. P. Reddy, T. Anuradha, P. V. V. Rao, B. R. Reddy, and M. K. Khan, "Secure authentication and key management protocol for deployment of Internet of Vehicles (IoV) concerning intelligent transport systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 12, pp. 24698–24713, Dec. 2022.
[40] Y. Liu and H. Gao, "Stability, scalability, speedability, and string Stability of connected vehicle systems," *IEEE Trans. Syst., Man, Cybern.: Syst.*, vol. 52, no. 5, pp. 2819–2832, May 2022.
[41] H. Luo, Y. Wu, G. Sun, H. Yu, and M. Guizani, "ESCM: An efficient and secure communication mechanism for UAV networks," *IEEE Trans. Netw. Service Manage.*, vol. 21, no. 3, pp. 3124–3139, Jun. 2024.
[42] T. L. Saaty, *The Analytic Hierarchy Process*, NewYork, New York, NY, USA, USA, USA: McGraw-Hill, 1980.
[43] J. H. Holland, "Reproductive plans and genetic operators," in *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, Cambridge, MA, USA: MIT Press, 1992, pp. 89–120.
[44] P. Zhang, W. Guo, Z. Liu, M. Zhou, B. Huang, and K. Sedraoui, "Optimized blockchain sharding model based on node trust and allocation," *IEEE Trans. Netw. Service Manage.*, vol. 20, no. 3, pp. 2804–2816, Sep. 2023.
[45] P. J. Slater, E. J. Cockayne, and S. T. Hedetniemi, "Information dissemination in trees," *SIAM J. Comput.*, vol. 10, pp. 692–701, 1981.
[46] L. Baird, "Hashgraph consensus: Detailed examples," Swirlds Technical Report SWIRLDS-TR-2016-02, Dec. 11, 2016. [Online]. Available: https://www.swirlds.com/
[47] L. Li, D. Huang, and C. Zhang, "An efficient DAG blockchain architecture for IoT," *IEEE Internet Things J.*, vol. 10, no. 2, pp. 1286–1296, Jan. 2023.
[48] W. Li et al., "Graphical consensus-based sharding for efficient and secure sharings in blockchain-enabled Internet of Vehicles," *IEEE Trans. Veh. Technol.*, vol. 73, no. 2, pp. 1991–2002, Feb. 2024.
[49] Z. Qu et al., "Community and priority-based microservice placement in collaborative vehicular edge computing networks," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2024, pp. 1–6.
[50] H. Luo, X. Yang, H. Yu, G. Sun, B. Lei, and M. Guizani, "Performance analysis and comparison of nonideal Wireless PBFT and RAFT Consensus networks in 6G communications," *IEEE Internet Things J.*, vol. 11, no. 6, pp. 9752–9765, Mar. 2024.
[51] H. Luo, Q. Zhang, G. Sun, H. Yu, and D. Niyato, "Symbiotic blockchain consensus: Cognitive backscatter communications-enabled wireless blockchain consensus," *IEEE/ACM Trans. Netw.*, vol. 32, no. 6, pp. 5372–5387, Dec. 2024.
[52] H. Luo, G. Sun, C. Chi, H. Yu, and M. Guizani, "Convergence of symbiotic communications and blockchain for sustainable and trustworthy 6G wireless networks," *IEEE Wireless Commun.*, vol. 32, no. 2, pp. 18–25, Apr. 2025.

Runhua Chen is currently working toward the bachelor's degree in electronic information engineering with the Glasgow College, University of Electronic Science and Technology of China (UESTC), and in electronics and electrical engineering with the University of Glasgow, Glasgow, U.K. His research interests include the Internet of Things, Internet of Vehicles, blockchain consensus, and AI.

Haoxiang Luo (Graduate Student Member, IEEE) received the BEng degree in communication engineering from the University of Electronic Science and Technology of China (UESTC), China, in 2021. He is currently working toward the PhD degree with the Key Laboratory of Optical Fiber Sensing and Communications (Ministry of Education), UESTC, and also a visiting student with the Nanyang Technological University (NTU), Singapore, since May 2025. His research interests include communication networks, IoT technology, trustworthy AI models, and blockchain. He was the session chair of the 2023 IEEE MetaCom and a TPC member of the 2024/2025 IEEE VTC Spring/Fall, 2024/2025/2026 IEEE WCNC, 2025 IEEE ICC, and so on. He is a reviewer of *IEEE Wireless Communications*, *IEEE Transactions on Wireless Communications*, *IEEE Transactions on Information Forensics and Security*, *IEEE Transactions on Dependable and Secure Computing*, *IEEE Transactions on Network Science and Engineering*, *IEEE Transactions on Computational Social Systems*, *IEEE Transactions on Artificial Intelligence*, *IEEE Internet of Things Journal*, and others.

Gang Sun (Senior Member, IEEE) received the PhD degree from the University of Electronic Science and Technology of China (UESTC), China, in 2012. He was a visiting professor with The Australian National University, in 2015. He is currently a full professor with the Key Laboratory of Optical Fiber Sensing and Communications (Ministry of Education), UESTC. His research interests include network virtualization, high-performance computing, parallel and distributed systems, and blockchain. He has edited many special issues in top journals, such as *IEEE Journal of Selected Areas in Sensors*, *IEEE Vehicular Technology Magazine*, *IEEE Internet of Things Magazine*, *Future Generation Computer Systems*, and so on.

Hongfang Yu (Senior Member, IEEE) received the BS degree from the Xidian University in 1996, the MS and PhD degrees from the University of Electronic Science and Technology of China (UESTC), China, in 1999 and 2006, respectively. She is currently a full professor with the Key Laboratory of Optical Fiber Sensing and Communications (Ministry of Education), UESTC, and the vice dean with the School of Information and Communication Engineering, UESTC. Her research interests include data center networking, network function virtualization, software-defined networking, blockchain technology, and federated learning. She is an area editor of the *IEEE Internet of Things Journal*, a technical editor of the *IEEE Network*, an associate editor of the *IEEE Communications Surveys & Tutorials*, and *Digital Communications and Networks*.

Dusit Niyato (Fellow, IEEE) received the BEng degree from the King Mongkuts Institute of Technology Ladkrabang (KMITL), Thailand, and the PhD degree in electrical and computer engineering from the University of Manitoba, Canada. He is a full professor with the College of Computing and Data Science, Nanyang Technological University, Singapore. His research interests include the areas of mobile generative AI, edge general intelligence, quantum computing and networking, and incentive mechanism design.

Schahram Dustdar (Fellow, IEEE) received the PhD degree in business informatics from the University of Linz, Austria, in 1992. He is currently a full professor heading the Distributed Systems Group, Technische Universität Wien (TU Wien), Vienna, Austria. He was the founding co-editor-in-chief of *ACM Transactions on Internet of Things* as well as editor-in-chief of Computing (Springer). He is an associate editor of *IEEE Transactions on Services Computing*, *IEEE Transactions on Cloud Computing*, *ACM Computing Surveys*, *ACM Transactions on the Web*, and *ACM Transactions on Internet Technology*, as well as on the editorial board of *IEEE Internet Computing* and *IEEE Computer*. He is an elected member of the Academia Europaea: The Academy of Europe, where he is Chairman of the Informatics Section, an AAIA president (2021–2023) and fellow (2021).