Dual-Path V2N/V2V OTA Update Distribution for SDVs: Design and Experimental Evaluation

Khaoula Sghaier[1,2,3] Arslane Hamza-Cherif[1] Pierre Merdrignac[1]
Ghada Gharbi[2] Chahrazed Ksouri[1] Bechir Yengui[1]
Badis Hammi[3] Pierre Parrend[2,4] Didier Verna[2]

[1] VEDECOM, Versailles, France
[2] LRE, EPITA, Le Kremlin-Bicêtre, France
[3] SAMOVAR, Télécom SudParis, Institut Polytechnique de Paris, Palaiseau, France
[4] University of Strasbourg, CNRS, ICube, Strasbourg, France

April 20, 2026

Abstract

With the rapid emergence of Software-Defined Vehicles (SDVs), efficient and secure Over-The-Air (OTA) updates have become essential to support continuous software evolution. This paper presents a distributed OTA architecture combining differential updates, IPFS-based peer-to-peer dissemination, and centralized orchestration via ThingsBoard. Differential updates reduce bandwidth consumption by transmitting only version deltas, IPFS provides decentralized content addressing and redundancy elimination, and ThingsBoard ensures consistent edge-to-vehicle coordination. Experimental results show that the distributed architecture significantly reduces last-hop delivery latency by leveraging Vehicle-to-Network (V2N) and Vehicle-to-Vehicle (V2V) communications, achieving up to a 81% reduction compared to centralized cloud-based updates, and can be scaled for fleet of vehicles. Furthermore, resilience evaluation under intermittent connectivity demonstrates up to 85.7% improvement in update success rate for distributed delivery, underscoring the relevance and potential of the proposed solution in the SDV context.

Keywords

Software-defined vehicles, OTA updates, distributed architecture, V2X

1 Introduction

The automotive industry is experiencing an unprecedented connectivity revolution. Modern vehicles are no longer isolated mechanical systems but increasingly sophisticated connected platforms, integrating cellular networks, short-range Vehicle-to-Everything (V2X) communications, and cloud services to deliver safety, efficiency, and infotainment applications [10]. This evolution is further accelerated by the emergence of Software-Defined Vehicles (SDVs), a new class of connected vehicles in which hardware capabilities are increasingly abstracted and vehicle functions are delivered, remotely updated, and reconfigured through software [13]. With the demand for new features deployment, software bugs fixing and timely security remediation, Over-The-Air (OTA) updates are fundamental capabilities of SDVs [14]. Due to frequent updates and increasing size of software packages, conventional OTA update processes face significant challenges. First, traditional approach relies on centralized architecture requiring each vehicle to independently download firmware from backend servers. This results in O(N) bandwidth consumption that scales linearly with fleet size, creating severe backend bottlenecks for large-scale deployments (e.g., 10,000+ vehicles) [5]. Second, conventional OTA platforms require vehicles to maintain a stable and persistent internet connection throughout the update process. This requirement is poorly aligned with vehicular mobility [8], and often necessitates that vehicles remain stationary to avoid failure due to insufficient internet connectivity [4]. Critically, firmware download and installation compete for the same limited stationary window. The limited time available to obtain large and frequent updates by SDVs may defer installation and become problematic in certain situations as UNECE R156 mandates timely vulnerability remediation [18]. As a result, slow or inefficient OTA processes directly extend the exposure window between patch release and deployment, creating both security and compliance risks. Third, with frequent firmware updates, consecutive versions typically contain about 80% of identical code, with only bug fixes or feature additions constituting the actual changes. Hence, distributing full firmware images for each update results in unnecessary bandwidth consumption and storage overhead, imposing higher computational costs during content verification and reassembly. Although traditional content delivery networks can reduce latency through geographic distribution, they do not fundamentally address the bandwidth scaling problem, as each vehicle still requires a complete independent download. To address the above-mentioned challenges, peer-to-peer distribution protocols such as InterPlanetary File System (IPFS) offer autonomous redistribution among

vehicles, reducing backend bandwidth to be independent of fleet size under sufficient connectivity assumptions [1]. Complementary, orchestration based on edge-cloud synchronization can support intermittent connectivity during network partitions thanks to metadata buffering on edge nodes. Finally, binary differential updates that encode only the delta between firmware versions using algorithms like xdelta and bsdiff to achieve high compression ratios [3]. Within this study, we design and evaluate an OTA update system that integrates IPFS-based content distribution, ThingsBoard orchestration [16] for state management across distributed nodes, and differential update mechanisms. This approach aims to ensure security throughout the OTA update download chain by supporting end-to-end software integrity according to ISO 24089 [7]. While IPFS [11], ThingsBoard [6], and differential update [9] algorithms are individually established technologies, the novelty of this work lies in: (i) their unified co-design into a dual-path: Vehicle-to-Network (V2N) links using WiFi-based internet access, and Vehicle-to-Vehicle (V2V) links leveraging ITS-G5; (ii) an original two-tier coordination design in which ThingsBoard exclusively manages update orchestration, announcement, and vehicle state tracking, while IPFS handles firmware content delivery and integrity verification through Content Identifiers (CIDs); and (iii) an experimental validation of this combined architecture on a real physical testbed with industry-grade hardware under intermittent connectivity constraints. The experimental results demonstrate the effectiveness of the solution based on representative metrics such as bandwidth consumption time, end-to-end latency and success rate of OTA updates. The remainder of this paper is organized as follows. Section 2 surveys the related work, section 3 presents our proposed distributed OTA architecture and section 4 details the practical implementation. Section 5 evaluates the solution performances from real tests. Finally, Section 6 concludes the paper and discusses future research directions.

2 Related work

Among content-distribution protocols, IPFS has emerged as particularly suitable for vehicular OTA applications [12] [11], providing uniqueness, preserving integrity, and ensuring decentralized verifiability through cryptographic CIDs. In [12], the authors propose utilizing IPFS to create a peer-to-peer content-delivery network between vehicles and Road-Side Units (RSUs). This work demonstrates IPFS viability for distributing small semantic data packets (typically 1kB) with low latency (appx 20ms per content

request) in a vehicular network. However, their approach does not incorporate differential update techniques or address fleet-wide version management and orchestration challenges. In [11], Oliveira et al. enabled IPFS based OTA update delivery by leveraging Mobile Edge Computing (MEC) through distributed RSUs in an urban infrastructure, thus, supporting download update blocks with intermittent communication. However, it does not fully assess fleet-wide dissemination and bandwidth efficiency of IPFS-based distribution for large numbers of vehicles. ThingsBoard provides orchestration of IoT devices through containerized microservices, supporting features such as real-time telemetry and fleet-wide OTA monitoring [16]. Its edge extension allows autonomous operation and synchronization upon network availability and uniquely combines full OTA lifecycle management with flexible edge deployment and scalability to thousands of devices [16] [15]. Bhattacharjee et al. [2] confirm the capability of edge orchestration to improve resilience, reduce latency, and alleviate network load.

From a delivery perspective, [14] highlights fundamental limitations of cellular and Wi-Fi OTA. Cellular networks suffer from physical resource block scarcity and limited vehicle connectivity windows (1 hour/day), which can extend large update delivery to several days. Wi-Fi remains geographically constrained and less secure. To mitigate these issues, ScalOTA leverages electric charging infrastructure to complete downloads during charging sessions, but at the cost of dedicated deployment. In contrast, our approach removes this dependency by pre-positioning firmware during normal driving via a dual-path V2N/V2V design, ensuring that limited stationary windows are reserved for installation rather than download.

The performances of using different communication technologies have been studied in [9], demonstrating that C-V2X (5G) can achieve 30× faster transmission than WiFi. Besides, incremental differential updates can reduce transmission times by up to 5000× compared to full package updates for YOLO (object detection) and UFAST (lane detection) software packages. Yet, this work does not address the scalability challenges of distributing OTA updates across multiple vehicles simultaneously. Moreover, it relies on a centralized dissemination model and does not consider fleet-level bandwidth efficiency or deployment constraints of real-world SDV OTA systems.

Our work goes beyond the state of the art by unifying IPFS-based decentralized distribution, ThingsBoard edge orchestration, and differential updates into a dual-path architecture combining V2N and V2V delivery. Unlike prior approaches that rely on infrastructure or treat these components in isolation, our design decouples firmware dissemination from installation, enabling pre-positioning during normal vehicle operation and pre-

serving limited stationary windows for safe installation. We further validate this architecture on a real testbed under intermittent connectivity, demonstrating improvements in bandwidth efficiency, latency, and update success rate at scale.

3 System architecture

3.1 Distributed architecture

The proposed distributed OTA system adopts a three-layer hierarchical architecture, as illustrated in Fig. 1, where each layer fulfills distinct responsibilities through well-defined interfaces, enabling efficient update propagation while maintaining security guarantees and system scalability.

Cloud layer: the cloud layer represents the root authority, comprising the OEM cloud server responsible for hosting authenticated firmware packages and managing the cryptographic trust chain. This layer maintains the canonical state of all fleet deployments, performing cryptographic operations (RSA/SHA-256 digital signatures and certificate authority coordination), policy enforcement (rollout strategies, compatibility matrices, and authorization frameworks), and global state management (version distribution tracking and update status monitoring).

Edge layer: edge nodes, deployed as MEC servers and anchored by the 5G User Plane Function (UPF), act as intelligent regional gateways between the cloud infrastructure and the vehicular fleet. These nodes maintain synchronized IPFS replicas of signed OTA update packages, enabling localized content delivery that reduces latency and improves resilience. By coordinating update distribution within their coverage area and serving vehicle requests through nearby MEC instances, the edge layer offloads core network traffic and preserves update availability under intermittent cloud connectivity.

Vehicle layer: the vehicular layer comprises connected vehicles equipped with embedded IPFS nodes and V2V communication capabilities. Vehicles function as both consumers and distributors within a dynamic peer-to-peer mesh, capable of retrieving updates from edge nodes or neighboring vehicles. Vehicle responsibilities include autonomous update discovery, local cryptographic verification (digital signatures and hash validation), active IPFS participation to acquire updates and redistribute them.

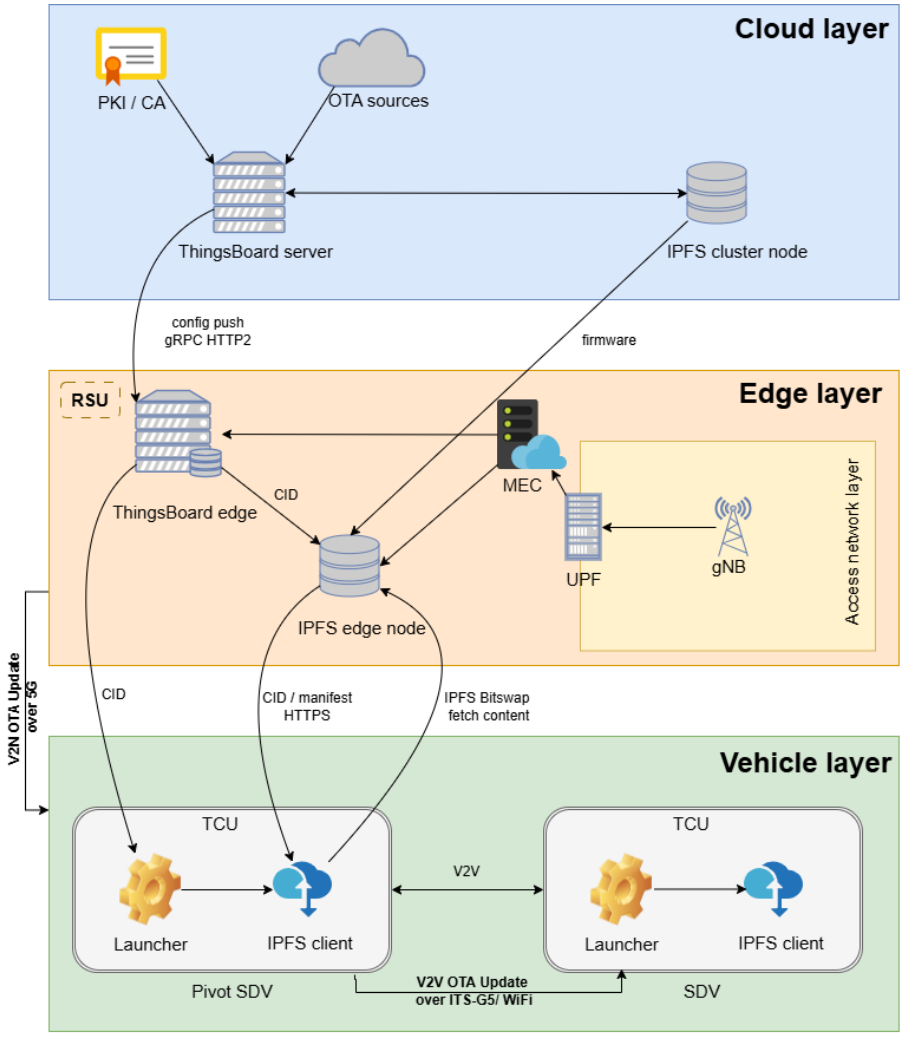

Figure 1: Three-layer distributed OTA architecture

4 Platform implementation

4.1 OTA update distribution pipeline

The update distribution follows four sequential phases illustrated in Figure 2, progressing from delta update generation to decentralized vehicular propagation. Each phase addresses distinct operational concerns while maintaining end-to-end cryptographic integrity.

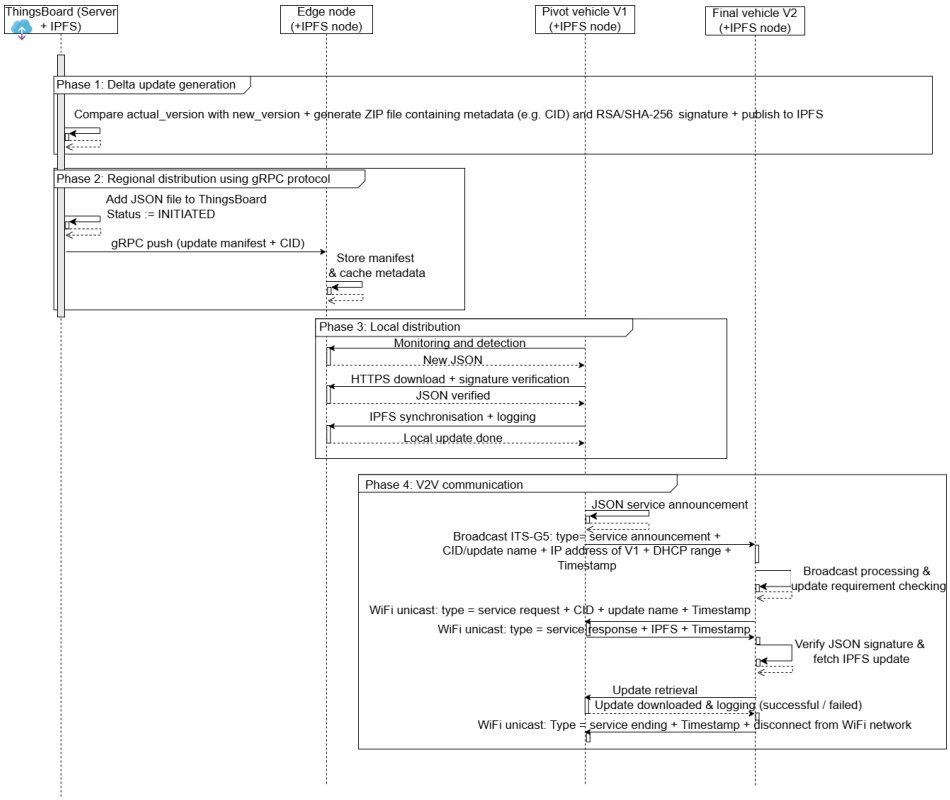

Figure 2: Sequence diagram of distributed OTA updates

Phase 1: delta generation and IPFS publication
The OTA update server generates binary differential updates between firmware versions using bsdiff. The update package is signed using RSA with SHA-256 to ensure integrity and authenticity and a zip file containing metadata (e.g. CID) and the signature is created. The signed package is then published to IPFS, yielding a CID that deterministically addresses the artifact and enables content verification at every hop.

Phase 2: cloud-to-edge propagation
The signed package enters ThingsBoard's cloud repository and is assigned to target vehicles, triggering automatic device notifications. Edge servers synchronize with the cloud backend, replicating firmware packages locally. Upon receiving update requests from vehicles in `INITIATED` state, the edge server's rule chain extracts the CID from the manifest, retrieves the package from its local IPFS node, and prepares it for distribution.

Phase 3: local distribution
Upon update detection, the TCU establishes a secure HTTPS connection with the ThingsBoard Edge server to retrieve the firmware package. The download is performed in successive blocks with automatic interruption handling, ensuring robustness against transient network failures. After reception, firmware integrity is validated from SHA-256 hash computation and checksum verification. In case of download or verification failure, the TCU reports the update status through telemetry, and a new download attempt is automatically scheduled.

Phase 4: V2V communication
A pivot vehicle periodically broadcasts an OTA service announcement over ITS-G5, advertising the availability of firmware updates through a JSON message containing the CID, update name, network access information, and a timestamp. Upon reception, neighboring vehicles process the broadcast, verify update requirements, and initiate a service request via a WiFi unicast message specifying the desired CID. After obtaining response from pivot vehicle and retrieving the update content from IPFS, firmware is downloaded, and the transaction outcome is logged as successful or failed.

4.2 Experimental deployment

The proposed OTA update framework is implemented in a real-world testbed environment using physical hardware components as depicted in Fig 3. The end-to-end architecture comprises a ThingsBoard and IPFS cloud server managing the central firmware repository, a ThingsBoard edge server orchestrating regional updates, and a TCU-equipped vehicle establishing V2N and V2V connectivity. The cloud server hosts firmware payloads in an IPFS repository where delta updates are generated. Thus, only the changes between consecutive firmware versions are encapsulated and identified by CIDs. The pivot vehicle, denoted V1, serves simultaneously as an edge infrastructure client and peer relay node with a TCU equipped both by cellular and WiFi interfaces. An NXP i.MX8M Plus-based TCU operates as a V2V-only client in the client vehicle V2. TCUs from V1 and V2 are each connected to a UNEX OBU-201 ITS-G5 communication unit (IEEE 802.11p, 5.9 GHz band, 6 Mbps physical data rate), supporting V2V broadcast and unicast communication. The implemented platform handles OTA updates orchestration and distribution in two distinct pathways, namely centralized V2N and distributed V2V pathways. In addition, to evaluate scalability beyond the physical hardware constraints, we extend the testbed using Docker containerization, dynamically instantiating fleet sizes ranging from 5 to 30 vehi-

cles as isolated IPFS nodes within a controlled bridge network environment. This configuration emulates early adopter vehicles that receive updates directly from infrastructure and subsequently redistribute them to neighboring peers.

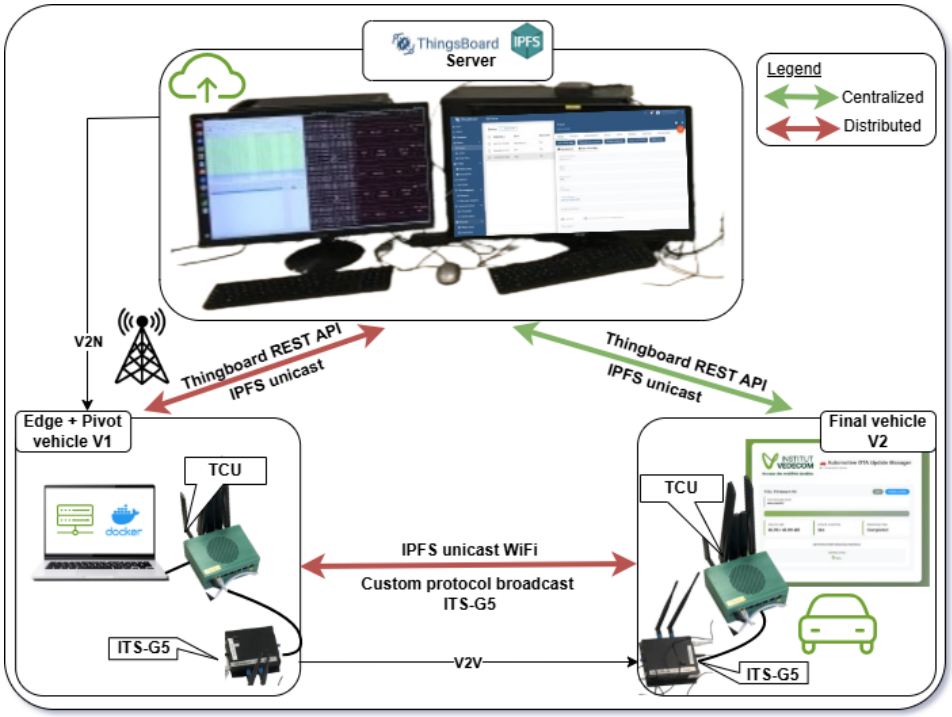

Figure 3: Experimental platform for OTA update distribution

5 Performance evaluation

5.1 Evaluation scenario

To conduct a comprehensive evaluation of the proposed framework, we performed multi-hop firmware distribution experiments using realistic payload sizes representative of automotive TCU deployments. Each experiment involved propagating updates from the cloud server through the edge orchestration layer to target vehicle V2, with pivot vehicle V1 serving as an intermediate relay node. Throughout the distribution pipeline, Python-based monitoring agents executed on each architectural component (cloud server,

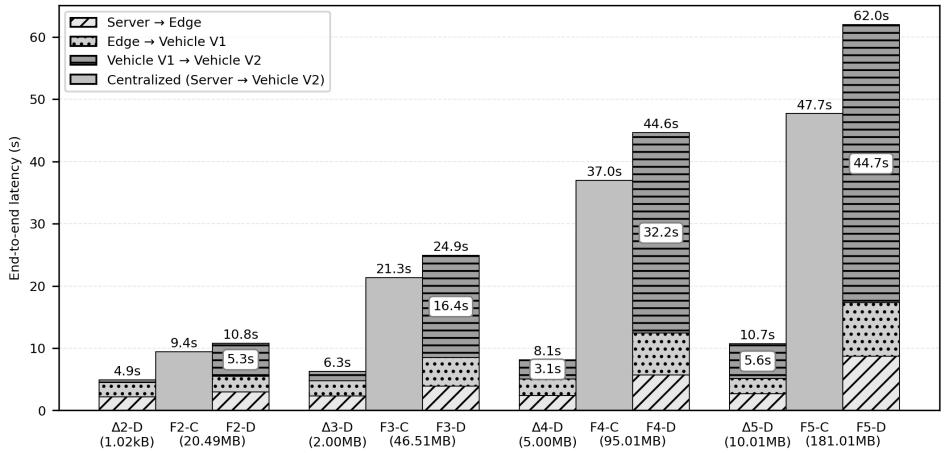

Figure 4: Comparison of end-to-end latency of delta and full firmwares in distributed and centralized architectures

edge node, V1, V2) capture key performance metrics, including latency, per-hop transfer times, and download status (`SUCCESSFUL`/`FAILED`), further exploited for performance assessment.

Fleet deployment was managed through a custom orchestration script that instantiates vehicle-specific containers with dedicated configurations. The automated test harness provisions fleets of $N = 5, 10, 20$ and 30 vehicle endpoints, each running the IPFS daemon `kubo v0.25.0`. After initialization, the orchestration script establishes IPFS swarm connections between all vehicle nodes and the edge server peer. Vehicles then concurrently retrieve the target firmware CID, with per-vehicle download durations and completion status logged to CSV files.

To assess resilience under V2X constraints, we conducted interrupted download experiments within a 20-second communication window. We tested three interruption durations: $3s$, $5s$, and $10s$, representing light, medium, and heavy connectivity disruptions characteristic of mobile vehicular scenarios. For each architecture (centralized, distributed) and interruption level, we performed $N = 20$ independent trials. Between trials, a cold-start protocol was enforced, resetting all cached data and system state to ensure measurement independence.

5.2 End-to-end latency of OTA distribution

End-to-end latency is defined as the effective time required to download update chunks across each component of the architecture $T_{\text{end-to-end}} = T_{\text{server-edge}} + T_{\text{edge-V1}} + T_{\text{V1-V2}}$. The time during which update packages remain cached at the edge in a standby state is intentionally excluded from this metric, as it does not contribute to active transmission or processing delay.

Fig. 4 depicts end-to-end latency for different size-compressed full firmware updates in both centralized and distributed architectures and delta updates in distributed architecture (referenced respectively by `F[update_version]-C`, `F[update_version]-D`, and $\Delta_{update-version}$). Fig. 4 shows that the end-to-end latency in the distributed architecture is slightly higher than in the centralized approach. This increase is primarily due to the additional integrity verification operations performed at each hop of the distributed pipeline. Specifically, checksum and signature verification are executed at the edge and at each vehicle (V1 and V2) to ensure firmware authenticity and integrity, thereby preventing any content tampering during propagation. Despite this overhead, the last-hop propagation in the distributed architecture —whether via V2N (Edge→V1) or V2V (V1→V2)—substantially mitigates transmission latency by up to 43.52% and 81.9% respectively- than direct cloud-to-vehicle delivery in the centralized model. Vehicles dynamically retrieve updates either from the edge or from a pivot vehicle based on optimal connectivity and proximity conditions. Furthermore, the use of delta updates significantly reduces the volume of transmitted data by an average of 96.2%. Since deltas are content-aware and encode only the binary differences between firmware versions, even substantial firmware growth does not lead to proportional transmission costs. Consequently, when combined with distributed dissemination, delta updates achieve up to 4.45× speedup over centralized full-firmware delivery.

5.3 Bandwidth consumption evaluation

Fig. 5 compares firmware dissemination time for Firmware_V2 (20.49 MB) under centralized and distributed architectures as fleet size increases. In the centralized case, fleet completion time scales linearly as $T_{\text{cent}}(N) = T_{\text{single}} \times N$, since each vehicle independently downloads the firmware from the cloud. In contrast, the distributed architecture was evaluated using Docker-based emulation, where an edge server disseminates the firmware to N containerized vehicles via IPFS parallel block transfer. The resulting fleet completion time $T_{\text{dist}}(N)$ reflects amortized bandwidth usage across vehicles.

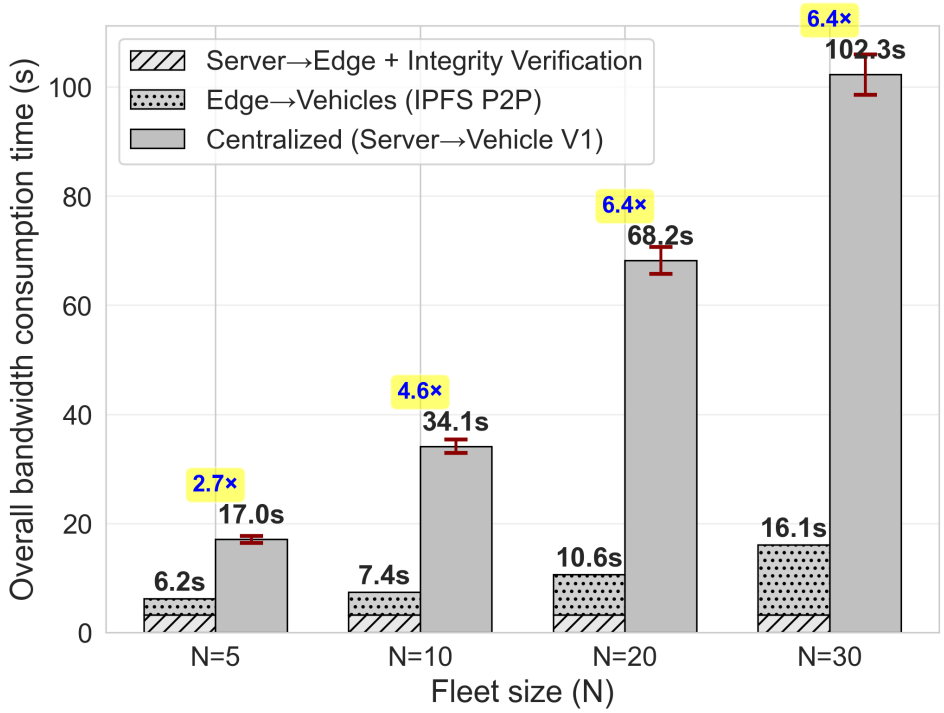

Figure 5: Bandwidth time consumption over fleet sizes

The bandwidth optimization factor $S(N) = \frac{T_{\text{cent}}(N)}{T_{\text{dist}}(N)}$ quantifies the benefit of edge-based parallel distribution. For $N = 30$ centralized delivery projects $102.3s$, while the distributed approach completes in $16.1s$, yielding a speedup of $S(30) = 6.4$. The speedup increases from $S(5) = 2.7$ to $S(30) = 6.4$. This optimization directly lowers backhaul load and OEM backend egress traffic, enabling OTA deployment to scale to larger fleets and denser update campaigns without proportional increases in network capacity—an essential capability for SDVs.

5.4 Success rate of OTA updates

Success rate of OTA update download has been evaluated using Firmware_V2 (20.49 MB) under different connectivity conditions. Results are reported in Table 1, where success rate (SR) is computed by $SR = \frac{N_{success}}{N_{total}} \times 100\%$. Here, $N_{success}$ represents full successful downloads completing within the 20-second timeout constraint. While all architectures achieve 100% success under baseline conditions, connectivity interruptions reveal critical differ-

ences. Under light interruption (3s, 15% of window), centralized delivery (F2-C) achieves 70% SR while distributed delivery (F2-D) reaches 85%—a 15 percentage point improvement. This gap widens progressively with interruption severity: at medium disruption (5s, 25% of window), F2-C drops to 55% while F2-D maintains 80% (25-point gap), and under heavy interruption (10s, 50% of window), F2-C achieves only 35% success compared to F-D's 65%—an 85.7% relative improvement. This near-doubling of success rate under severe disruption validates our hypothesis: proximity-based distributed delivery fundamentally transforms OTA viability in time-constrained mobile scenarios.

Table 1: OTA update success rate under varying network interruption conditions

Interruption scenario	**F2-C**	**F2-D**
Baseline (no interruption)	100%	100%
Light interruption (3 s)	70%	85%
Medium interruption (5 s)	55%	80%
Heavy interruption (10 s)	35%	65%

5.5 Discussion on architecture scalability

While experiments were conducted on an emulated fleet of 30 vehicles, the scalability of the proposed architecture extends well beyond this bound by design. In traditional centralized models, simultaneous update campaigns result in infrastructure saturation that scales linearly with fleet size [5]. In contrast, our framework restricts core network exposure to a limited set of edge gateways. Once a firmware CID is propagated to the edge layer, all subsequent dissemination is offloaded to the local vehicular network via V2V relay over ITS-G5, eliminating further backend connectivity requirements for the remaining fleet. As each vehicle acquiring the update becomes a content provider for its neighbors within its 150-300m broadcast range [17], the system enables autonomous redistribution that transforms high traffic density into a distribution asset rather than a bottleneck.

6 Conclusion

This paper presents a distributed OTA update architecture combining V2N wireless connectivity and V2V relay for software-defined vehicles. While individual update latency increases due to multi-hop propagation, the distributed approach demonstrates superior fleet-scale performance: reducing last-hop transmission time by 81.9% and significantly lowering aggregate bandwidth consumption compared to centralized delivery. Experimental validation under intermittent connectivity scenarios reveals that proximity-based V2V delivery achieves 65% success rate under 50% communication window disruption, compared to 35% for cloud-based centralized delivery—an 85.7% relative improvement. These results demonstrate that distributed architecture addresses critical scalability, resilience, and bandwidth efficiency challenges for vehicular OTA deployments. Future work will extend the latency evaluation across heterogeneous network configurations enabling network-aware adaptive path selection between V2N and V2V delivery modes. Network configurations will include comparative assessment of 4G LTE and 5G NR cellular backhaul links, as well as variable ITS-G5 physical data rates to characterize the sensitivity of end-to-end OTA delivery performance to last-hop channel conditions. Additionally, we will leverage these insights to strengthen the security of the proposed OTA update architecture, tailoring lightweight security measures compliant with ISO/SAE 21434 and UNECE WP.29.

Acknowledgement

Badis Hammi has received funding from the European Commission under the Horizon Europe project AI4CCAM (Grant Agreement No. 101076911).

References

[1] Juan Benet. Ipfs - content addressed, versioned, p2p file system, 2014.

[2] Arpan Bhattacharjee, Hamza Mahmood, Sidi Lu, Nejib Ammar, Akila Ganlath, and Weisong Shi. Edge-assisted over-the-air software updates. In *2023 IEEE 9th International Conference on Collaboration and Internet Computing (CIC)*, pages 18–27, 2023.

[3] Russ Bielawski, Ron Gaynier, Di Ma, Sam Lauzon, and André Weimerskirch. Cybersecurity of firmware updates. Technical Report DOT HS 812 807, National Highway Traffic Safety Administration, U.S. Department of Transportation, October 2020.

[4] Zeb Bowden and Jacob Chandler. Current implementations of over-the-air updates. Technical report, National Surface Transportation Safety Center for Excellence, Jul 2024.

[5] Stephen Cawley. Changing vehicle architecture and the role of ota, 2021. Accessed: 2025-12-28.

[6] Pablo Gimeno Hermán. Implementación de actualizaciones ota en dispositivos iot para la plataforma thingsboard. 2025.

[7] ISO. ISO 24089:2023 — road vehicles — cybersecurity of over-the-air (ota) updates, 2023.

[8] Lei Liu, Ming Zhao, Miao Yu, Mian Ahmad Jan, Dapeng Lan, and Amirhosein Taherkordi. Mobility-aware multi-hop task offloading for autonomous driving in vehicular edge computing and networks. *IEEE Transactions on Intelligent Transportation Systems*, 24(2):2169–2182, 2023.

[9] Yichen Luo and Sidi Lu. Incremental over-the-air update for software-defined vehicles with c-v2x. In *2025 IEEE 101st Vehicular Technology Conference (VTC2025-Spring)*, pages 1–7, 2025.

[10] Md. Noor-A-Rahim, Zilong Liu, Haeyoung Lee, Mohammad Omar Khyam, Jianhua He, Dirk Pesch, Klaus Moessner, Walid Saad, and H. Vincent Poor. 6g for vehicle-to-everything (v2x) communications: Enabling technologies, challenges, and opportunities. *Proceedings of the IEEE*, 110(6):712–734, 2022.

[11] José Oliveira, Pedro Almeida, Pedro Rito, Duarte Raposo, and Susana Sargento. Over-the-air updates for software defined vehicle services with ipfs. In *NOMS 2024-2024 IEEE Network Operations and Management Symposium*, pages 1–9. IEEE, 2024.

[12] Victor Ortega and Jose F. Monserrat. Semantic distributed data for vehicular networks using the inter-planetary file system. *Sensors*, 20(22), 2020.

[13] Khaoula Sghaier, Badis Hammi, Ghada Gharbi, Pierre Merdrignac, Pierre Parrend, and Didier Verna. Advancing security in software-defined vehicles: A comprehensive survey and taxonomy. *arXiv preprint arXiv:2510.09675*, 2025.

[14] Ali Shoker, Fernando Alves, and Paulo Esteves-Verissimo. Scalota: Scalable secure over-the-air software updates for vehicles. In *2023 42nd International Symposium on Reliable Distributed Systems (SRDS)*, pages 151–161, 2023.

[15] D.I. Shvaika, A.I. Shvaika, and V.O. Artemchuk. Advancing iot interoperability: dynamic data serialization using thingsboard. *Journal of Edge Computing*, 3(2):126–135, 2024.

[16] ThingsBoard. Over-the-air firmware and software updates, 2025. Accessed: Nov. 2025.

[17] Unex Technology Corporation. V2x system-on-module: Communication range analysis. `https://unex.com.tw/en/faq/som/v2x_communication_range/`, 2024. Accessed: 2026.

[18] United Nations Economic Commission for Europe (UNECE). Un regulation no. 156: Software update and software update management system. `https://unece.org/sites/default/files/2024-03/R156e%20%282%29.pdf`, 2024. UNECE WP.29, latest consolidated version.