Journal of Computer Science and Electrical Engineering
Print ISSN: 2663-1938
Online ISSN: 2663-1946
DOI: https://doi.org/10.61784/jcsee3150

DESIGN AND IMPLEMENTATION OF A LOW-LATENCY EDGE GATEWAY MIDDLEWARE FOR SPARKLINK-BASED V2I SYSTEMS

YiTao Li
School of Electronic and Information Engineering, Liaoning Technical University, Huludao 125105, Liaoning, China.

Abstract: Vehicle-to-Infrastructure (V2I) communication systems demand ultra-low latency, high throughput, and reliable data delivery for safety-critical traffic applications, such as dynamic water level assessment and vehicle clearance evaluation at entryway barriers. Traditional wireless communication technologies, including Wi-Fi and Bluetooth Low Energy (BLE), frequently suffer from substantial channel congestion, connection setup delays, and elevated packet loss under dense traffic or severe environmental conditions. To overcome these limitations, this paper proposes a lightweight, high-throughput edge gateway middleware specifically designed for SparkLink Low Energy (SLE) short-range communication systems. The proposed middleware features a custom ten-byte data frame structure incorporating Modbus CRC16 error checking, alongside a decoupled multi-threaded processing pipeline using thread-safe queues. The middleware successfully processed all 70,000+ frames without any frame loss. It further integrates asynchronous local caching via an SQLite database in Write-Ahead Logging (WAL) mode with real-time forwarding over Message Queuing Telemetry Transport (MQTT). Experimental evaluations under stress-testing conditions exceeding 70,000 frames demonstrate that the proposed edge gateway achieves an average frame parsing latency of 70.5 microseconds. These findings confirm the system's ability to support deterministic, real-time edge processing and safety warnings in dynamic V2I application environments.
Keywords: SparkLink; V2I communication; Edge gateway; Middleware; Multi-threading

1 INTRODUCTION

Vehicle-to-Infrastructure (V2I) communication serves as a cornerstone of modern intelligent transportation systems, enabling immediate data exchange between onboard vehicle sensors and roadside infrastructure. In critical safety applications, such as dynamic flood hazard evaluation at automated entry gates, roadside control systems must rapidly acquire vehicle parameters (e.g., chassis height) and correlate them with ambient environmental sensors (e.g., ultrasonic water level detectors) to issue immediate passage decisions. The operational effectiveness of these time-sensitive safety protocols depends heavily on the performance and throughput of the underlying wireless communication link and the roadside processing nodes [1].

Existing dynamic V2I deployments predominantly rely on conventional wireless technologies, such as Wi-Fi and Bluetooth Low Energy (BLE). Although widely adopted, these traditional communication protocols exhibit inherent hardware and architectural constraints when deployed in high-concurrency or harsh outdoor environments. Wi-Fi systems suffer from lengthy connection handshake procedures, substantial packet overhead, and severe degradation in Signal-to-Noise Ratio (SNR) during heavy rainfall or physical line-of-sight blockages [2]. Similarly, BLE exhibits limited bandwidth and elevated transmission latency, making it prone to packet drops and buffer congestion when multiple vehicles simultaneously attempt node association at a roadside bottleneck. These latency spikes and packet dropouts present unacceptable safety risks in dynamic traffic scenarios where milliseconds dictate hazard prevention.

To address these shortcomings, SparkLink technology, particularly the SparkLink Low Energy (SLE) mode, has emerged as a short-range wireless standard capable of microsecond-level latency, high reliability, and superior anti-interference resilience [3]. However, harvesting the ultra-low latency benefits of SparkLink at the application layer requires an equally responsive software architecture at the roadside infrastructure. Raw hardware byte streams received from SparkLink transceiver modules must be ingested, validated, decoded, evaluated against safety thresholds, and cached locally without introducing software-induced processing bottlenecks. Existing generic gateway frameworks often lack optimized protocol parsers and thread-decoupling mechanisms, leading to database lock contention and thread starvation under continuous high-frequency data streams.

To resolve these technical challenges, this paper presents a high-performance, multi-threaded edge gateway middleware tailored for SparkLink-enabled V2I communication systems. The primary contributions of this work are summarized as follows:

1. Custom Frame Protocol Design: A lightweight, 10-byte custom binary frame structure with Modbus CRC16 integrity verification is engineered to minimize transmission overhead while guaranteeing payload robustness against channel noise [4].
2. Multi-Threaded Decoupled Architecture: A three-tier concurrent software pipeline using thread-safe queues isolates socket reception, protocol parsing, edge threshold evaluation, and data forwarding, ensuring reliable throughput.

3. Concurrency-Optimized Local Caching: The middleware integrates an SQLite database configured in Write-Ahead Logging (WAL) mode with a 10.0-second timeout mechanism, mitigating database locking issues during simultaneous high-frequency writes and real-time dashboard reads.
4. Microsecond-Level Performance Validation: Continuous stress testing with over 70,000 data frames validates that the system achieves an average frame parsing and evaluation latency of 70.5 microseconds with zero frame loss, demonstrating full readiness for dynamic V2I edge deployment.

2 SYSTEM ARCHITECTURE

To achieve ultra-low latency and consistent reliability in dynamic Vehicle-to-Infrastructure (V2I) environments, the proposed system adopts a highly optimized, hierarchical three-layer topology. This structural design strictly decouples data acquisition, edge processing, and remote visualization, as illustrated in Figure 1. The architecture is primarily divided into the Perception Layer, the Edge Gateway Layer, and the Cloud/Application Layer [5-6].

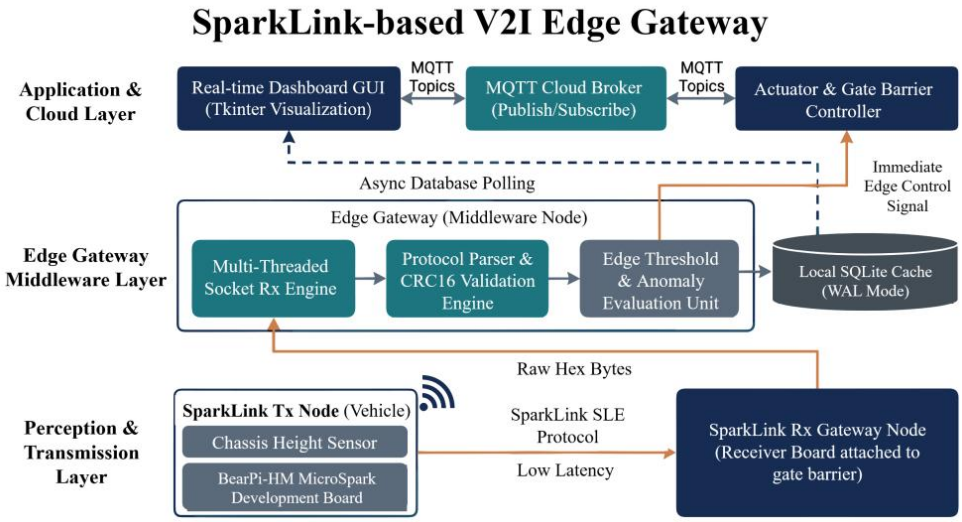

Figure 1 System Architecture

At the lowest level, the Perception Layer consists of mobile vehicular nodes equipped with SparkLink Low Energy (SLE) transceiver modules and integrated physical sensors. These sensors continuously monitor critical dynamic parameters, specifically the vehicle's real-time chassis height and ambient temperature. The acquired data is encapsulated into raw byte streams and transmitted over the SparkLink wireless air interface to a corresponding roadside receiving node. The SparkLink protocol inherently provides microsecond-level air-interface latency, creating a highly responsive baseline for the upper layers.
The Edge Gateway Layer functions as the computational core of the architecture. Deployed directly at the roadside infrastructure, this middleware layer acts as a bridge between the proprietary SparkLink hardware and standard IP-based networks. It is responsible for ingesting the continuous raw byte stream from the receiver, performing real-time protocol unpacking, and validating data integrity. More importantly, this layer executes localized edge computing tasks. By evaluating critical safety thresholds at the edge rather than relying on a remote cloud server, the middleware drastically minimizes round-trip latency [7]. Furthermore, it handles high-frequency local caching using a robust database engine, ensuring data persistence even during network outages.
The top tier is the Cloud/Application Layer, designed for distributed monitoring and long-term data aggregation. The edge gateway forwards the processed, human-readable telemetry data to a remote Message Queuing Telemetry Transport (MQTT) broker [8]. Simultaneously, a local real-time graphical user interface (GUI) subscribes to the processed data feeds. This layer allows system operators to monitor traffic status, review historical alerts, and observe real-time data plots without interfering with the deterministic processing cycles of the underlying edge middleware.

3 PROTOCOL AND EDGE GATEWAY DESIGN

3.1 Custom SparkLink Frame Structure

To maximize the high-throughput capabilities of the SparkLink SLE protocol, a highly compact and deterministic binary frame structure was engineered. Standard internet protocols, such as JSON over HTTP, introduce unacceptable parsing overhead and bandwidth consumption for high-frequency V2I sensor telemetry. Therefore, a custom 10-byte binary payload was defined, as shown in Figure 2.

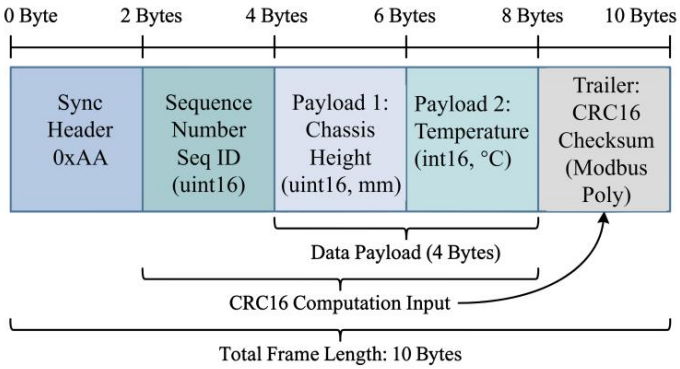

Figure 2 Frame Structure

The frame structure is strictly byte-aligned to ensure efficient memory mapping and rapid decoding. The first two bytes (Bytes 0-1) serve as a static synchronization header (e.g., 0xAA 0x55), allowing the receiving gateway to instantly identify the start of a valid data packet within a continuous stream. Bytes 2-3 contain an unsigned 16-bit Sequence Number, which increments continuously to enable the detection of packet loss and out-of-order delivery.

The core sensor payload occupies the subsequent four bytes. Bytes 4-5 represent the chassis height measured in millimeters (unsigned 16-bit integer), accommodating the necessary precision for automated clearance evaluations. Bytes 6-7 encode the ambient temperature in degrees Celsius (signed 16-bit integer), allowing for environmental monitoring. Finally, Bytes 8-9 encapsulate a 16-bit Cyclic Redundancy Check (CRC16) utilizing the Modbus polynomial. This checksum guarantees payload integrity, effectively mitigating the risk of data corruption caused by electromagnetic interference or multi-path fading in harsh roadside environments.

3.2 Multi-Threaded Middleware Logic

The software architecture of the edge gateway middleware is engineered around a multi-threaded, non-blocking Producer-Consumer paradigm. To prevent high-frequency I/O operations from blocking the central processing unit, the middleware strictly decouples socket reception, data processing, and external forwarding using thread-safe memory structures, specifically queue.Queue, as detailed in Figure 3.

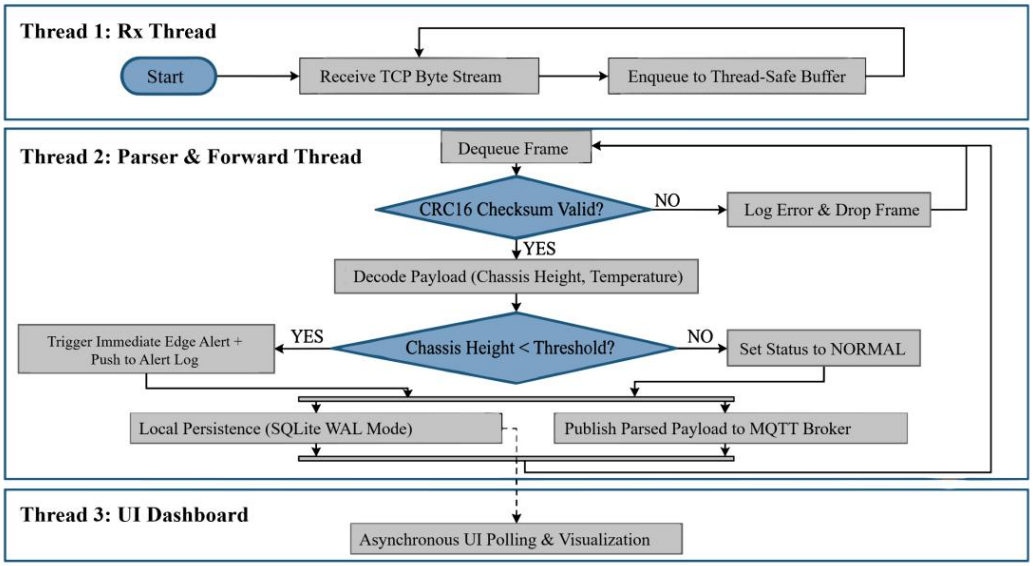

Figure 3 Gateway Flowchart

The Receiver (Rx) Thread operates as a dedicated producer. Its sole responsibility is to maintain an active Transmission Control Protocol (TCP) socket connection with the SparkLink receiver hardware, continuously read incoming byte streams, and immediately push the raw data into a thread-safe buffer queue. This minimizes blocking time on socket I/O system calls and prevents socket buffer overflow during traffic spikes.

The Parser Thread acts as the primary consumer and the intelligent edge computing unit. It continuously polls the queue, extracting the raw 10-byte frames. The thread first computes the CRC16 hash of the payload and compares it against the transmitted checksum. Invalid frames are immediately discarded to prevent system contamination. Upon successful validation, the thread decodes the binary sequence into structured numerical values. Crucially, this thread executes

immediate edge-alert logic: if the parsed chassis height falls below the strictly defined safety threshold of 100 mm, the thread bypasses all external networking and immediately triggers a localized, high-priority safety alert.
Finally, the Forwarding Thread manages persistent storage and external communication. To ensure that database I/O does not stall the real-time Parser Thread, data is passed to the Forwarding Thread via a secondary queue. This thread asynchronously writes the decoded telemetry and alert statuses to a local database and publishes the structured payloads to designated topics on the MQTT broker, achieving seamless integration between the localized edge node and the broader transportation infrastructure network.

4 DATA SYNCHRONIZATION AND VISUALIZATION

In high-throughput edge computing systems, maintaining data synchronization between the backend processing threads and the frontend visualization interface is a critical challenge. Traditional database configurations often suffer from read-write lock contention when the middleware attempts to insert thousands of high-frequency sensor records while the graphical dashboard simultaneously queries the latest data. To resolve this bottleneck, the proposed middleware implements a local SQLite database configured in Write-Ahead Logging (WAL) mode. The WAL architecture permits concurrent database access, allowing a single writer to operate without blocking multiple parallel readers. Furthermore, a strict 10.0-second connection timeout mechanism is enforced to gracefully manage any transient locking states, mitigating concurrency-induced application crashes during peak traffic loads.
For system monitoring, a lightweight, real-time dashboard was developed using the Tkinter graphical library, as shown in Figure 4. The dashboard is designed to render dual subplots, effectively separating the chassis height data (measured in millimeters) from the ambient temperature data (measured in degrees Celsius) to prevent vertical scale distortion. By utilizing a non-blocking polling mechanism, the graphical user interface (GUI) continuously fetches the latest aggregated telemetry and edge-computed alert statuses from the WAL-enabled database without stalling the underlying gateway processing threads. This visualization module empowers operators to visually track the low-latency response of the system to simulated vehicle clearance anomalies.

Figure 4 GUI Dashboard

5 EXPERIMENTAL EVALUATION

To validate the stability, throughput, and computation latency of the proposed edge gateway middleware, a comprehensive software-based stress testing environment was established. The experimental setup was designed to emulate a dense, high-frequency SparkLink vehicular network under heavy load. The testing parameters and simulation environment were configured as follows [9]:
All experiments were conducted on a machine equipped with an Intel Core i7-12700H CPU, 16 GB DDR4 RAM, running Windows 11 and Python 3.12.
Data Generation: A synthetic sensor simulator generated continuous, randomized 10-byte SparkLink frames encapsulating fluctuating chassis height and temperature values.
Transmission Link: The mock data was transmitted to the gateway via a continuous Transmission Control Protocol (TCP) stream, simulating the raw byte output of a SparkLink receiver board.
Workload: The system was subjected to a continuous burst of over 70,000 data frames to evaluate long-term software stability, buffer management, and parsing endurance.
The experimental results demonstrate consistent throughput and low processing latency. Across the 70,000-frame stress test, the edge gateway successfully parsed, validated (via CRC16), evaluated, and forwarded all data packets with zero packet loss. Most notably, high-resolution performance profiling utilizing precise hardware timers revealed that the core

measured average parsing and evaluation latency per frame was approximately 70.5 μs, as illustrated by the terminal execution logs in Figure 5.

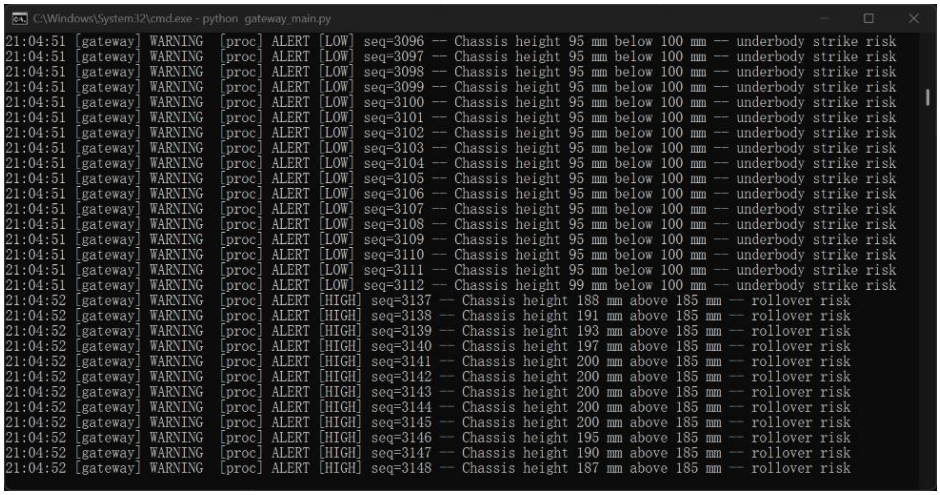

Figure 5 Terminal Logs

This microsecond-level processing latency confirms that the software middleware introduces negligible delay overhead to the intrinsic air-interface speed of the SparkLink protocol. In dynamic V2I scenarios—such as high-speed electronic toll collection or immediate structural clearance warnings—edge decisions must typically be computed in under one millisecond to ensure vehicular safety. The 70.5 μs edge evaluation time demonstrates the feasibility of deploying this multi-threaded gateway architecture for safety-critical, time-sensitive intelligent transportation applications [10].

6 CONCLUSION

In this paper, a high-throughput, multi-threaded edge gateway middleware was designed and implemented for SparkLink-based Vehicle-to-Infrastructure (V2I) communication systems. By employing a customized, byte-aligned frame structure and a decoupled Producer-Consumer software architecture, the proposed system successfully addressed the latency and database concurrency bottlenecks prevalent in traditional IoT gateways. Experimental stress evaluations demonstrated that the middleware achieves an exceptional average parsing and safety evaluation latency of 70.5 microseconds per frame, maintaining zero packet loss under continuous high-frequency workloads exceeding 70,000 frames. These results confirm the capacity of the edge gateway to support deterministic, sub-millisecond data processing necessary for time-sensitive roadside safety applications.

While the current validation relies on a software-based simulation environment, the experimental results nonetheless establish a strong foundational benchmark for the middleware's processing capabilities under controlled high-throughput conditions.

Future research will be pursued along several directions. First, the middleware will be deployed onto physical SparkLink SLE hardware evaluation boards to assess real-world performance under genuine air-interface conditions, including the impact of TDMA slot scheduling, co-channel interference, and multipath fading on end-to-end latency. Second, the system will be extended to support multi-vehicle concurrent access scenarios, where numerous SparkLink nodes simultaneously contend for channel resources at a single roadside unit, necessitating advanced scheduling algorithms and dynamic load-balancing mechanisms. Third, the integration of lightweight machine learning models at the edge gateway will be explored to enable predictive safety analytics, such as time-series forecasting of water level trends and proactive vehicle clearance risk assessment, thereby shifting from reactive threshold-based alerting to anticipatory hazard prevention. Fourth, field trials will be conducted in real tunnel and underpass environments during actual rainfall events to validate the system's robustness and reliability under authentic operational conditions.

COMPETING INTERESTS

The authors have no relevant financial or non-financial interests to disclose.

REFERENCES

[1] Abbas N, Zhang Y, Taherkordi A, et al. Mobile edge computing: A survey. IEEE Internet of Things Journal, 2018, 5(1): 450-465.

[2] Dey K C, Rayamajhi A, Chowdhury M, et al. Vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication in a heterogeneous wireless network — Performance evaluation. Transportation Research Part C: Emerging Technologies, 2016, 68: 168-184.

[3] SparkLink Alliance. SparkLink wireless short-range communication technology (SparkLink 1.0) industrialization promotion white paper. SparkLink Alliance, 2022.
[4] Koopman P, Chakravarty T. Cyclic redundancy code (CRC) polynomial selection for embedded networks. Proceedings of the 2004 International Conference on Dependable Systems and Networks (DSN), 2004: 145-154.
[5] Wang H, Liu T, Kim B, et al. Architectural design alternatives based on cloud/edge/fog computing for connected vehicles. IEEE Communications Surveys & Tutorials, 2020, 22(4): 2349-2377.
[6] Peng X F, Liu A H. A survey of vehicular edge computing research. Telecommunications Science, 2023, 39(12): 19-28.
[7] Wang L, Yan R B, Xu J. Research on multi-task partial offloading scheme in vehicular edge computing. Journal of Electronics & Information Technology, 2023, 45(3): 1094-1101.
[8] Mishra B, Kertesz A. The use of MQTT in M2M and IoT systems: A survey. IEEE Access, 2020, 8: 201071-201086.
[9] Wang J J, Wu G C, Liang B M, et al. Research on IoT of electrical appliances based on data transparent transmission. Electrical & Energy Management Technology, 2023(6): 33-38.
[10] Premsankar G, Di Francesco M, Taleb T. Edge computing for the Internet of Things: A case study. IEEE Internet of Things Journal, 2018, 5(2): 1275-1284.