**ACTA IMEKO**
**ISSN: 2221-870X**
*September 2025, Volume 14, Number 3, 1 – 10*

# Bridging real-time CAN networks with in-vehicle single pair ethernet: A time-sensitive networking approach

**Alberto Morato[1], Federico Tramarin[2]**

[1] *National Research Council of Italy, CNR-IEIIT, Padova, Italy*
[2] *Department of Engineering "Enzo Ferrari", University of Modena and Reggio Emilia, Modena, Italy*

ABSTRACT
This paper explores the challenges of integrating existing in-vehicle networks (IVNs) with Ethernet, explicitly focusing on the coexistence of traditional Controller Area Network (CAN) systems with Single Pair Ethernet (SPE) and Time-Sensitive Networking (TSN). The research presents an experimental setup using real hardware to simulate a zonal vehicle architecture, where CAN nodes are integrated into the in-vehicle Ethernet network.
The approach utilizes a gateway implemented on a TSN-enabled switch, facilitating seamless connectivity between the real-time CAN network and the TSN-enabled in-vehicle Ethernet. The study demonstrates the feasibility of combining real-time CAN traffic with high-bandwidth Ethernet traffic through the integrated gateway. It also emphasizes the crucial role of TSN features in maintaining the real-time properties of the CAN network, even in the presence of additional traffic loads.

**Section:** RESEARCH PAPER

**Keywords:** In-Vehicle Networks (IVNs); Controller Area Network (CAN); Single Pair Ethernet (SPE); Time-Sensitive Networking (TSN); latency measurements

**Citation:** A. Morato, F. Tramarin, Bridging real-time CAN networks with in-vehicle single pair ethernet: A time-sensitive networking approach, Acta IMEKO, vol. 14 (2025) no. 3, pp. 1 – 10, DOI: 10.21014/actaimeko.v14i3.2073

**Section Editor:** Francesco Lamonaca, University of Calabria, Italy

**Received:** February 10, 2025; **In Final Form:** June 25, 2025; **Published:** August, 2025.



**Funding:** This research has been supported in part within the activities of the project "Time Sensitive Networking for future enabling technologies: measurement methods and metrological characterization in hybrid wired/wireless scenarios" CUP E53D23014650001, funded by EU in NextGenerationEU plan through the Italian "Bando Prin 2022 PNRR - D.D. 1409 del 14-09-2022" by MUR.
This research has been also carried out within the PNRR research activities of the consortium iNEST (Interconnected North-Est Innovation Ecosystem) funded by the European Union Next-GenerationEU (Piano Nazionale di Ripresa e Resilienza (PNRR) - Missione 4 Componente 2, Investimento 1.5 - D.D. 1058 23/06/2022, ECS_00000043)

**Corresponding Author:** Alberto Morato, e-mail: alberto.morato@cnr.it

## 1. INTRODUCTION

In recent years, the automotive industry has experienced a profound transformation, with vehicles advancing into complex assemblies of interconnected sensors, actuators, and Electronic Control Units (ECUs). These developments have fundamentally altered the processes of vehicle design, manufacturing, and operation. Nevertheless, as vehicles become more dependent on sensor data for a wide array of functionalities, in-vehicle data exchange technologies are becoming crucial in the creation of new features and services, underscoring the critical significance of accurate and efficient measurement methods.

Traditionally, automotive networks have relied on technologies such as Controller Area Network (CAN), Local Interconnect Network (LIN), and Media Oriented Systems Transport (MOST) to facilitate communication between sensors and ECUs. Although these technologies serve their purpose well, they face limitations in meeting the demands of emerging features and services, primarily due to their restricted bandwidth and limited data transmission capabilities. Despite the predominant use of CAN for powertrain, vehicle dynamics, and low-level diagnostics, there is a growing demand for higher bandwidth and lower latency in In-Vehicle Networks (IVN) [1].

The landscape is evolving rapidly through the integration and enhancement of low-level systems with new features provided by ever more complex Advanced Driver Assistance Systems (ADAS), which encompass a diverse array of sensors, such as cameras, LIDARs, and radars, producing a substantial volume of measurement data at a high rate. Other emerging features are arising in automotive, like IoT [2]–[4] or vehicle-to-vehicle (V2V) communications [5]. Given the safety-critical nature of ADAS systems, it is imperative to ensure reliable and efficient data exchange.

In pursuit of enhanced performance and functionality, the

automotive industry is moving towards the adoption of Ethernet and its real-time extension with Time-Sensitive Networking (TSN) as the in-vehicle communication network [6]–[8].

Ethernet is a widely accepted standard for wired digital communications that is defined in IEEE 802.3. This standard defines a set of profiles tailored to meet the needs of various communication scenarios, ranging from high-speed data transfer to robust, lightweight solutions for constrained environments. These profiles are differentiated on the basis of factors such as data rate, cabling requirements, and intended use cases. For example, 1000Base-TX and 2.5GBase-T are common profiles that are now implemented in most modern consumer and office-oriented equipment to support high-bandwidth data streams that are typical of office environments, home electronics, live video streaming, and similar applications. These two profiles support 1 Gbps and 2.5 Gbps speeds, respectively, over two pairs of twisted pair cables. However, the equipment required to operate these networks can be relatively expensive and often requires the use of shielded cables to prevent crosstalk and interference from other pairs or external sources.

Although these profiles are the most popular for high-speed data transfer in consumer electronics, the automotive and industrial sectors are moving in a slightly different direction. In these industries, several factors need to be considered beyond the data rate. For example, reliability, robustness, compatibility with other protocols (such as Modbus TCP/IP, Ethernet/IP, and PROFINET in industrial applications), maximum cable length, cost, and weight are all critical factors that often outweigh the sole focus on data rate. For this reason, 100 Mbps and lower speed profiles are still widely used in these fields.

These requirements are so significant that, in addition to high data rate profiles, the IEEE 802.3 standard is also developing parallel profiles to address these needs. For example, the IEEE 802.3 standard family has recently been enhanced with the IEEE 802.3cg-2019 standard, which defines new profiles such as 10BASE-T1L (industrial) and 10BASE-T1S (automotive). These profiles are part of what is commonly referred to as Single-Pair Ethernet (SPE).

SPE introduces an entirely new physical layer with impressive capabilities, including coverage of up to 1,000 meters, full-duplex data rates of up to 10 Mbps, data and power over a single twisted pair and bus topology, such as CAN bus.

The use of a single twisted pair makes SPE a lightweight and cost-effective alternative, particularly suitable for modern vehicles and industrial environments. It also allows for the reuse of existing cables and topologies, which further reduces deployment costs.

In automotive applications, SPE can be employed to replace specific sections of the CAN bus network while maintaining the same wiring harness. This transition offers several advantages, including the adoption of a more modern and faster network, while significantly reducing the weight of the wiring harness and the overall cost of the vehicle.

Focusing specifically on the automotive sector, the transition to fully SPE networks will be gradual and is expected to take a significant amount of time. In more realistic short- to medium-term scenarios, SPE will coexist with traditional automotive networks [9]. In this context, IVNs are shifting towards zone-based architectures, where different zones, based on CAN bus or other traditional networks, are interconnected through SPE as, for example, shown in Figure 1. Essentially, SPE will serve as the backbone for transferring data between these zones or domains, enabling seamless communication throughout the vehicle [10].

However, this architecture can present certain challenges. Specifically, since all traffic from different zones is aggregated onto the SPE backbone, a high volume of traffic, such as that generated by

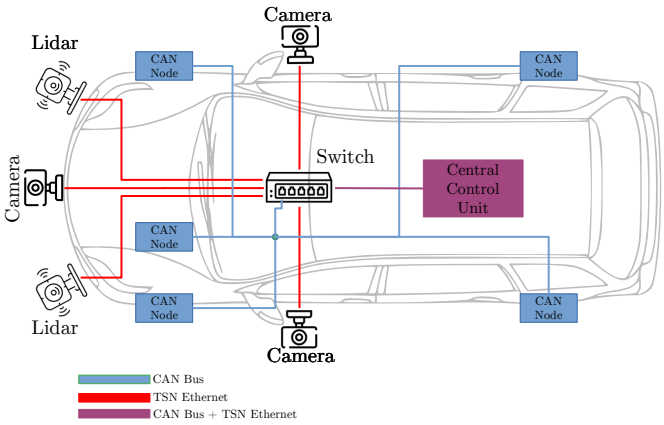


Figure 1. Example of a TSN-enabled in-vehicle network topology comprising CAN bus nodes.

ADAS, like cameras or LiDAR, can result in increased latency for some data streams. The issue becomes critical when the delayed data is time-sensitive. Such delays can potentially lead to safety concerns, for example, in collision avoidance ADAS systems where timely data delivery is crucial. Similarly, delays in data related to powertrain or vehicle dynamics can cause general malfunctions or poor responsiveness, affecting the overall performance and safety of the vehicle. It is evident that critical data, especially in communication systems with a relatively low data rate such as SPE, must be handled with mechanisms for stream reservation and prioritization. These measures are essential to ensure reliability and bounded latency for time-sensitive and safety-critical data streams.

In this direction, Time Sensitive Networking (TSN) is progressively gaining interest in the automotive sector as a solution to address the challenges posed by integrating various communication technologies [8], [11]. TSN, an extension of the IEEE 802.1 standards, introduces mechanisms for time synchronisation, traffic scheduling, and bounded latency, ensuring the reliable delivery of time-critical messages over Ethernet. It is important to note that TSN defines these mechanisms at a high level and is not tied to a specific physical or MAC layer. In recent years for example, TSN capabilities have been integrated into Wi-Fi (IEEE 802.11) [12], demonstrating its flexibility in different networking technologies. This flexibility ensures that TSN is fully compatible with SPE, although SPE does not share the same physical layer as traditional Ethernet. This compatibility is further reinforced by the development of TSN-enabled SPE controllers from major chip manufacturers, providing robust support for time-critical communication in automotive and industrial applications [13].

In this paper, we address the challenges of integrating existing IVNs with Ethernet, focusing on the coexistence of traditional CAN networks with SPE and Time-Sensitive Networking. Specifically, we propose an experimental setup based on real hardware that mimics a zone-based vehicle architecture, where CAN nodes are integrated within the in-vehicle Ethernet network. The proposed approach utilizes a gateway implemented on a TSN-enabled switch, capable of bridging the real-time CAN network with the TSN-enabled in-vehicle Ethernet network. Although this work does not specifically address 10BASE-T1S, we consider 10BASE-TX, which provides the same data rate and exhibits similar characteristics. We assume that data originating from the CAN network is time-critical and must be transmitted with minimal latency. To evaluate the performance of the proposed system, we measure the latency of real-time CAN traffic under different traffic profiles and TSN configurations. For completeness, we also consider

other data rates to assess the scalability of the system and highlight the importance of TSN at lower data rates. The experimental results demonstrate the effectiveness of the proposed approach in meeting time-critical communication requirements, ensuring the real-time properties of the CAN network are preserved, while leveraging the advantages of TSN-enabled Ethernet.

In detail, the paper is organized as follows. Section 2. provides an overview of the related work. Section 3. introduces the proposed approach, the experimental setup, and the measurement methodology. Section 4. presents the experimental results. Finally, Section 5. concludes the paper.

## 2. RELATED WORKS AND CONTRIBUTION

The integration of CAN and Ethernet in the context of vehicles is progressing through various strategies, each with its own trade-offs in terms of latency, network efficiency, and cost.

In this regard, [14] introduces CAN XL as an evolution of CAN that is capable of encapsulating Ethernet frames, and it explores how composite CAN XL-Ethernet networks can operate effectively. The paper describes two encapsulation techniques: Ethernet over CAN (EoC), which encapsulates Ethernet frames into CAN XL frames, and IP over CAN (I4oC), which directly maps IP datagrams to CAN. Although the results are promising, implementing this approach can be complex, as it requires dedicated hardware that differs significantly from existing automotive systems and standards. Additionally, the need for a fragmentation protocol in CAN XL to handle large Ethernet payloads is also discussed. Consequently, it seems natural to move toward the convergence of CAN over Ethernet rather than proposing a completely new version of CAN [15].

The most common approach is the use of gateways to interconnect the two networks. However, the main challenges in designing these interfaces arise from differences in communication protocols and timing requirements. A significant amount of research has focused on finding innovative solutions to improve the efficiency of the gateway. One major challenge is the disparity in operating speeds between the two networks. CAN typically operates at lower speeds (up to 1 Mbps), while Ethernet operates at much higher speeds. Encapsulating a CAN frame within an Ethernet frame often results in reduced efficiency [16]. To address this, [17] examines gateway strategies for embedding CAN frames in Ethernet/IP packets, proposing various transformation techniques that optimize protocol overhead and message latency. Beyond the standard 1:1 mapping method, buffer-based, time-based, and priority-based approaches are discussed. Similarly, [18] focuses on optimizing the design of the gateway and implements a frame-packing strategy to reduce bandwidth consumption. It introduces the Fixed Waiting Time (FWT) strategy, where CAN messages are accumulated for a fixed duration before being sent as a single Ethernet frame.

Evidently, the key to higher efficiency lies in the aggregation of multiple CAN frames into a single Ethernet frame. However, this introduces additional latency, which may be unacceptable for certain applications [19], [20].

To minimize latency, CAN frames should be transmitted as soon as they are available at the gateway, even at the cost of efficiency. However, achieving the lowest possible latency is challenging due to competing Real-Time (RT) or Best-Effort (BE) traffic within the Ethernet network. In such cases, two primary strategies can be employed to reduce latency:

1) **Optimized Medium Access Control (MAC) Protocols:** For instance, [21] proposes enhancements to 10BASE-T1S Ethernet, including the Physical Layer Collision Avoidance (PLCA) protocol and a priority mechanism based on the Inter-Packet Gap (IPG) to allow nodes with urgent data to transmit with higher priority. Similarly, [22] further explores the use of 10BASE-T1S in delay-sensitive vehicular networks, introducing an interrupt mechanism for urgent packet transmission. This mechanism allows express packets to interrupt ongoing transmissions, enabling very low-latency delivery.
2) **Time-Sensitive Networking (TSN):** TSN is used to address latency-related issues and mitigate interference between data streams. In [23], a time-aware CAN-to-Ethernet gateway is developed, utilizing TSN features such as time synchronisation (802.1AS). The gateway, implemented on an STM32 microcontroller, acquires CAN frames and transmits them to an Ethernet network connected to a TSN-enabled switch. Frames are then transmitted on the basis of a scheduler that considers maximum waiting times, thereby optimizing latency performance. Furthermore, [24] integrates the Time-Aware Shaper (TAS) of TSN and demonstrates that this approach can reduce the maximum latency. However, a drawback of using TAS is that traffic is "windowed", meaning that in the worst case, traffic can be delayed by the cycle time. This poses challenges for applications requiring the lowest possible latency.

Therefore, in this paper, we adopt a similar, yet fundamentally different approach. Specifically, we leverage the capabilities of modern TSN-enabled switches to provide integrated support for both CAN and Ethernet on a unified hardware platform. Instead of relying on external hardware to convert CAN frames into Ethernet frames, we utilize a TSN-enabled gateway that seamlessly bridges the two networks directly on the TSN switch.

For traffic prioritization, rather than using TAS, we rely on IEEE 802.1Qav (credit-based fair queuing). This approach simplifies the integration of CAN and Ethernet on a shared TSN-enabled hardware platform, while achieving efficient traffic management. The proposed method minimizes complexity, while maintaining a balance between latency and efficiency, making it well suited for modern vehicular communication systems.

## 3. EXPERIMENTAL SETUP AND MEASUREMENT METHODOLOGY

Figure 1 presents a sample configuration of an in-vehicle network topology that seamlessly integrates both CAN and Ethernet networks. In this configuration, the topology starts by being divided into two separate network domains. The first domain comprises multiple nodes that are tasked with processing data from various sensors and actuators; these include components such as powertrain systems and wheel-related sensors. The second domain is designated for the Ethernet network, which supports modern sensors capable of generating high-volume data streams, such as cameras, LiDARs, and radars.

A TSN-enabled switch serves as the intermediary bridge between these two network domains and can also operate as a CAN-to-Ethernet gateway. In particular, this switch functions as a real-time embedded Linux device furnished with real-time networking capabilities and the potential for custom software execution. This configuration allows for the direct realization of CAN-to-Ethernet bridging on the switch itself. Ultimately, the principal central controller accepts the data streams from the TSN switch, processes these data, and makes decisions to manipulate the actuators.

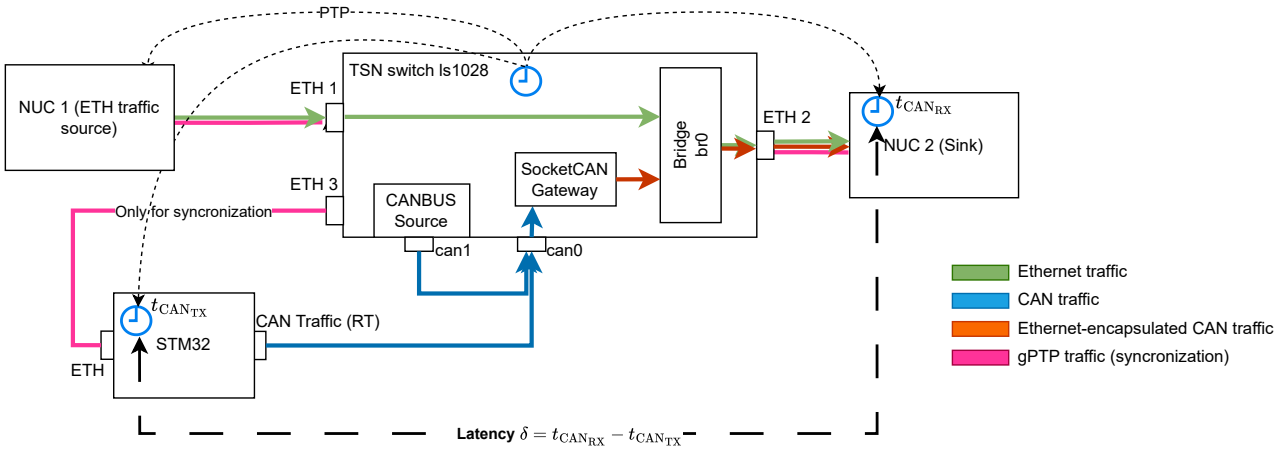


Figure 2. Scheme of the experimental setup.

In our experimental setup, a prototype system was effectively constructed to replicate the topology showcased in Figure 1. For simulation purposes, a single Ethernet device was used to emulate multiple connected devices and generate network traffic. The setup features two Next Unit of Computing (NUC) devices, specifically the TNKi5000 Intel NUCs, both powered by an Intel Core i5-1135G7 CPU and operating on Ubuntu 22.04 with a standard non-real-time kernel (version 6.3), as shown in Figure 2 and Figure 3. One of these NUCs is tasked with generating a traffic profile that simulates ADAS scenarios, while the other, maintaining the same software and hardware configurations, acts as the primary central controller. The TSN switch, an NXP LS1028A model running the Real-Time Edge Linux distribution, incorporates five Ethernet interfaces and two CAN interfaces.

To mimic the attributes of a 10BASE-T1S Single-Pair Ethernet network, the Ethernet interfaces on the switch and the NUCs were configured to operate at a speed of 10 Mbps utilizing the native Linux `ethtool` utility.

Regarding the CAN bus network, the primary component involved is an STMicroelectronics STM32F767ZI microcontroller and the Ethernet switch. Specifically, the NUCLEO-F767ZI development board, which is based on the STM32F767ZI microcontroller, was utilized in this setup. This microcontroller is powered by an ARM Cortex-M7 core and incorporates an integrated CAN interface along with a hardware-supported IEEE 1588-2008 Precision Time Protocol (PTP) clock. As we will elaborate later, the PTP clock is integral in our setup for the precise measurement of CAN frame latency.

The microcontroller operates at a peak frequency of 216 MHz, which is derived from a PLL driven by an 8 MHz crystal oscillator. The PTP clock is directly derived from the base clock with the assistance of a prescaler, which dictates its increment rate. To accommodate any variations, the PTP clock can be modified for offset, frequency, and phase through dedicated software registers.

On the software side, we employ Zephyr RTOS, an open source real-time operating system that provides an abstraction layer that supports a wide array of platforms and silicon manufacturers. Among its numerous attributes, Zephyr extends support for TSN, specifically IEEE 802.1AS (gPTP). In this study, we used the upstream version of Zephyr available on GitHub [25], using it without particular custom changes.

Essentially, the STM32 board emulates one of the time-critical nodes linked to the vehicle's CAN bus network. Both CAN interfaces present on the switch are implemented in the following manner:

- The `can0` interface operates as a "collector" port, capturing all CAN traffic and forwarding it to the Ethernet network. The STM32 CAN interface is connected to this particular port.
- The `can1` interface serves to emulate the rest of the CAN bus network, incorporating various other devices, sensors, and actuators that may use CAN communication. A CAN traffic generator is used to control this port, dispatching CAN frames with various periodicities and priorities.

This configuration enables efficient simulation of a real-world CAN bus network without the need for a large number of physical CAN devices. Given that both the STM32 device and the traffic generator on `can1` transmit frames to `can0`, we can account for the delays resulting from the CAN controllers and transceivers, as well as the potential delays arising from bus collisions that can manifest in a real network.

In the subsequent sections, we will dive into the detailed architecture and operational principles of each subsystem, along with the methodological approach employed to measure CAN frame latency.

### 3.1. Time synchronisation (802.1AS)

The primary objectives of this experimental campaign are threefold: i) to examine the latency experienced by CAN frames in a mixed CAN-TSN domain, ii) to evaluate the impact of interfering traffic on the time-sensitive CAN subsystem, and iii) to assess the effectiveness of TSN features in prioritizing real-time and time-sensitive traffic.

A crucial aspect of these experiments is the ability to perform precise time measurements with minimal uncertainty, ensuring an accurate evaluation of the system performance.

To this end, sharing the same time reference among all the devices involved is crucial. To achieve this, we used the IEEE 802.1AS standard, a profile of the IEEE 1588 PTP, specifically designed for time-sensitive networking applications and capable of providing a sub-microsecond synchronisation error [26].

In essence, IEEE 802.1AS, also known as the Generalized PTP (gPTP), establishes a common time reference across all time-aware devices in a network. The protocol operates on the basis of a hierarchical structure, where a single Grandmaster clock (GM), also called Grand Leader (GL), serves as the main time source, while other devices, called Followers, synchronise their local clocks to match the GM's reference time. The synchronisation process involves the periodic exchange of timestamped messages, including

*Sync*, *Follow_Up*, *Delay_Req*, and *Delay_Resp* frames, which allow each Follower to estimate both propagation delay and clock offset relative to the GM. The local clocks then apply corrections to align with the reference time, ensuring precise synchronisation throughout the system.

As depicted in Figure 2, the TSN switch functions as the Grandmaster Clock in our setup, while the two NUCs and the STM32 microcontroller act as Followers. It is important to note that, in the case of the STM32, the Ethernet interface is used solely for time synchronisation purposes, whereas actual data transmission occurs over the CAN interface. This setup ensures that all devices in the system share a common notion of time, which is crucial for accurately measuring the CAN frame latencies and analyzing network performance.

### 3.2. Latency measurement methodology

The measurement system utilized in this experimental campaign is based on the methodology proposed in [27]. This approach eliminates the need for external measuring instruments or systems. Instead, data packets serve as probe frames, encapsulating transmission and reception timestamps, which are subsequently extracted and analyzed to obtain latency values.

To ensure the significance of the measured latency and minimize uncertainty, it is assumed that the devices generating probe frames are accurately time-synchronised. This method has already been validated as both reliable and accurate [27], [28], guaranteeing the quality of the results obtained.

Although time synchronisation is straightforward for Ethernet devices within the TSN domain, it presents challenges for CAN nodes, as CAN lacks a native time-synchronisation mechanism or timestamping capability. However, if the CAN frames originate from a time-aware device within the TSN domain, it is possible to adopt the same technique proposed in [27]. This involves encapsulating a synchronised timestamp within the CAN frame: upon transmission, the system clock's current timestamp is embedded in the frame, and upon reception, this timestamp is extracted and compared against the receiver's clock to compute latency. Let $t_{\mathrm{CAN_{TX}}}$ denote the transmission timestamp and $t_{\mathrm{RX}}$ the reception timestamp. The latency $\delta$ is then calculated as

$$\delta = t_{\mathrm{RX}} - t_{\mathrm{CAN_{TX}}}. \tag{1}$$

In the setup, the presence of two CAN interfaces on the switch, along with a hardware-based IEEE 1588-2008 PTP clock-enabled Ethernet port on the STM32, facilitates the development and implementation of the latency measurement methodology.

For traffic generated by the switch to simulate other devices on the CAN bus, we generate CAN frames containing the current time of the TSN switch, which are transmitted through the `can1` interface. Subsequently, these frames are received at the `can0` interface and forwarded to the CAN-to-Ethernet gateway. A similar procedure is followed in the case of the STM32. As previously described, the STM32 is connected to the TSN switch via Ethernet exclusively for synchronisation purposes, ensuring that its PTP clock is aligned with that of the TSN switch. The gPTP core of Zephyr OS provides access to the current synchronised time for other OS modules.

On the STM32, we implemented a dedicated thread that periodically transmits CAN frames. Each time a frame is generated, the system retrieves the current synchronised timestamp from the OS and embeds it into the frame before sending it via the CAN interface to the switch. The switch then receives the frame and forwards it to the CAN-to-Ethernet gateway.

It is important to emphasize that the use of multiple physical CAN interfaces connected to a single reception port, rather than a single port in loopback mode, provides a more accurate estimation of real-world latencies. This setup accounts for delays introduced by multiple CAN controllers, better reflecting the conditions of a real deployment scenario. In contrast, when frames are transmitted from a single port in loopback mode without multiple CAN devices, there are no bus collisions, which could lead to an artificially lower latency estimation.

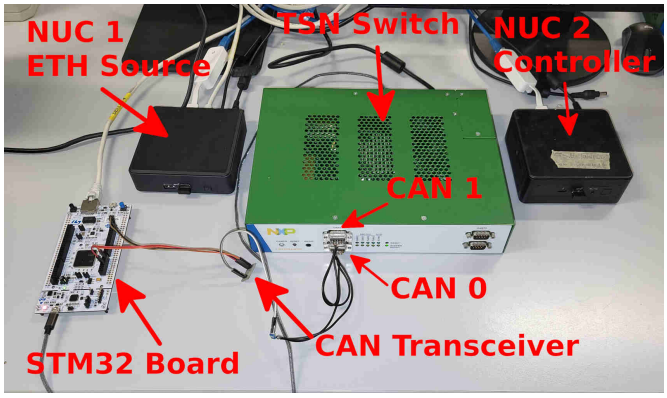


Figure 3. Experimental setup.

Another important point to consider is that although we encapsulate timestamps in the frames generated directly by the switch, the analyses presented later focus exclusively on the measurements taken from frames generated by the STM32 microcontroller. These frames best represent real CAN network traffic, as they are affected by delays introduced by transceivers and potential collisions. The traffic generated by the switch serves solely as interference traffic.

### 3.3. CAN traffic generator

In order to generate CAN traffic effectively and simulate communications from multiple devices, we employ custom-built software that operates directly on the TSN switch. This specialized software is programmed to consistently generate CAN frames at a steady and uniform rate, featuring a predetermined CAN ID alongside a payload comprising 8 data Bytes, which is fully populated. For the purpose of closely mimicking real-world vehicular environments, we derive CAN frame IDs from real automotive implementations. Specifically, our solution is capable of interpreting and replaying CAN database files, known as DBCs, which we have sourced from an open source repository, as cited in [29]. From all available databases, we selected one that comprises a total of 92 unique frames, with their IDs spanning from `0x109` to `0x651`. In particular, since the database lacks detailed information concerning the periodicity of these frames, we made assumptions based on an even distribution, categorizing them into three distinct groups with transmission intervals set at 1 ms, 10 ms, and 100 ms, respectively. To more accurately simulate the asynchronous nature of real-world transmission scenarios, the software is configured to introduce a deliberate delay for the initial transmission of the first frame in each group. This delay is a random period between 0 and 100 ms, ensuring variability in start times.

Regarding the STM32, as briefly mentioned earlier, we implemented a dedicated thread that periodically transmits CAN frames with a data size of 8 Bytes. Each frame embeds a gPTP timestamp retrieved from the gPTP core of the Zephyr OS kernel. The transmission rate is set to 1 ms, using the CAN ID `0x101`. In particular, this ID is lower than the smallest ID found in the database. This choice is intentional because we are primarily interested in the latency of time-critical frames. In real-world applications,

it is common for time-critical frames to have lower IDs than other frames, ensuring that they are prioritized in the event of collisions.

In general, the CAN bus was set to operate at 500 kbps, which is a common speed for INVs. The occupancy of the bus was 100%, verified with `canbusload` of the `can-utils` suite, to simulate the worst scenario.

### 3.4. CAN-to-Ethernet gateway

The gateway utilizes an adapted version of the SocketCAN utilities offered for the Linux kernel, which have been improved to accommodate the traffic priority features required by the 802.1Qav standard. Operationally, the gateway functions by acquiring CAN frames through the `can1` interface. These frames are subsequently encapsulated within User Datagram Protocol (UDP) frames, which are then directed to the internal software bridge of the TSN system. Following this processing step, the encapsulated frames are transmitted over Ethernet to reach the main controller.

### 3.5. Stream reservation (802.1Qav)

In the experimental configuration under consideration, it is imperative to prioritize two data streams: i) the PTP synchronisation traffic and ii), the CAN traffic. During initial trials, it was evident that when the Ethernet communication channel was heavily loaded, there were instances where PTP frames were either intermittently dropped or experienced significant delays. This caused a serious interruption in synchronisation. Consequently, it became apparent that giving precedence to PTP traffic over all other data types was of crucial importance. To effectively implement this priority scheme, Virtual Local Area Networks (VLANs) were set up according to the 802.1Q standard on all devices that participate in the synchronisation process.

The standard employs the Priority Code Point (PCP) field located within the VLAN tag of an Ethernet frame to offer a range of eight distinct priority levels, numbered sequentially from 0 to 7. This allows for the categorization and distinction of frames according to their assigned importance or the urgency of their forwarding priority. In the configuration that has been proposed, we have designated priority level 5 specifically for PTP traffic, while priority level 3 has been allocated to CAN traffic. All other Ethernet traffic types are assigned a lower priority level of 1.

To effectively support this priority scheme, the switch takes advantage of the capabilities of IEEE 802.1Qav, the internal configuration of which is illustrated in Figure 4. In particular, the Linux kernel integrates a specific traffic queue discipline, named Multiqueue Priority Qdisc `MQPRIO`, which enqueues frames into physical hardware queues and forwards them based on their priority.

Finally, to fine-tune the maximum bandwidth allocated for each stream, the queues handling CAN and Ethernet traffic are then managed by the Credit-Based Shaper (CBS) mechanism. The mechanism is realized through increasing/decreasing the credit value of the specific queues, that is, the credit for high-priority queues increases faster and therefore reaches the transmitting threshold more frequently than the low-priority queues. This strategy, together with the queue discipline, allows traffic access bandwidth to be controlled based on traffic priorities.

### 3.6. Time-Aware Shaping (IEEE 802.1Qbv)

Time-Aware Shaping (TAS), standardized by IEEE 802.1Qbv, enables the deterministic scheduling of Ethernet traffic by dividing time into recurring cycles with predefined transmission windows.

At the core of 802.1Qbv is the Gate Control List (GCL), which manages transmission gates associated with each traffic class queue.

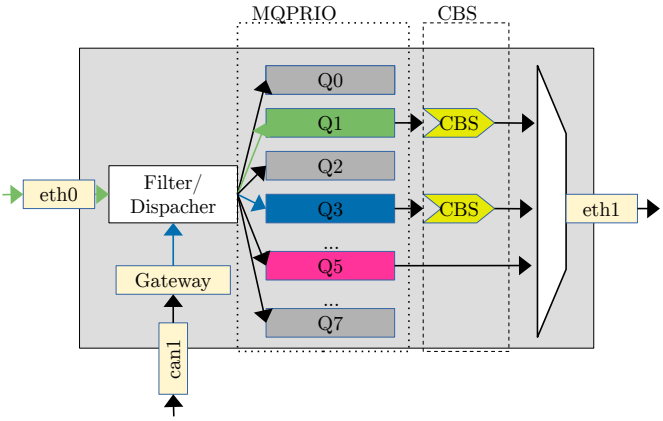


Figure 4. 802.11Qav configuration of the TSN switch.

These gates are opened and closed based on a time schedule, effectively implementing a Time-Division Multiple Access (TDMA) scheme. During each cycle, only selected queues are permitted to transmit, ensuring that high-priority traffic is not delayed by lower-priority or best-effort traffic. `TAPRIO` is the Linux implementation of IEEE 802.1Qbv. It leverages this concept by allowing users to define time-aware transmission schedules directly in the operating system. It configures the gates of hardware (or software) queues to open and close according to a cyclic GCL, ensuring that high-priority frames are transmitted within their reserved time slots.

### 3.7. Ethernet traffic generator

To simulate the presence of ADAS or other devices in the Ethernet network, we employed the `iperf3` utility, which is capable of generating traffic at the maximum rate supported by the network. The utility was configured to generate a continuous stream of UDP packets with a payload size of 500 Bytes. This configuration was chosen to ensure that the Ethernet network was fully saturated, thereby creating a worst-case scenario for the CAN traffic. The utility was executed on the NUC acting as the Ethernet traffic generator, which was connected to the TSN switch via an Ethernet interface. The switch was configured to prioritize the CAN traffic over the Ethernet traffic, as previously described. So when stream reservation (802.1Qav) was enabled, Ethernet traffic was classified as best-effort traffic with priority level 1, while CAN traffic was classified as real-time traffic with priority level 3.

## 4. EXPERIMENTAL RESULTS

We conducted multiple sets of experiments to evaluate the latency of real-time CAN traffic under different traffic profiles and TSN configurations. For each experiment, 100,000 samples were collected from frames originating from the STM32.

In the first set of experiments, we measured the latency of the CAN connection without interference from any other traffic, establishing a baseline latency. Next, we introduced Ethernet traffic simulating data streams from sensors, without enabling any TSN features. As mentioned above, the Ethernet traffic stream was generated using the `iperf3` utility in a way that fully saturated the available bandwidth.

With Ethernet traffic enabled, we then performed two additional tests, where we purposely enabled the mentioned specific TSN features. In particular, we first enabled `MQPRIO`, which assigned CAN bus traffic to a transmission queue of higher priority than Ethernet traffic. Next, we conducted another experiment where both `MQPRIO` and `CBS` (that is, bandwidth stream reservation) were enabled (see Sec. 3.5.).

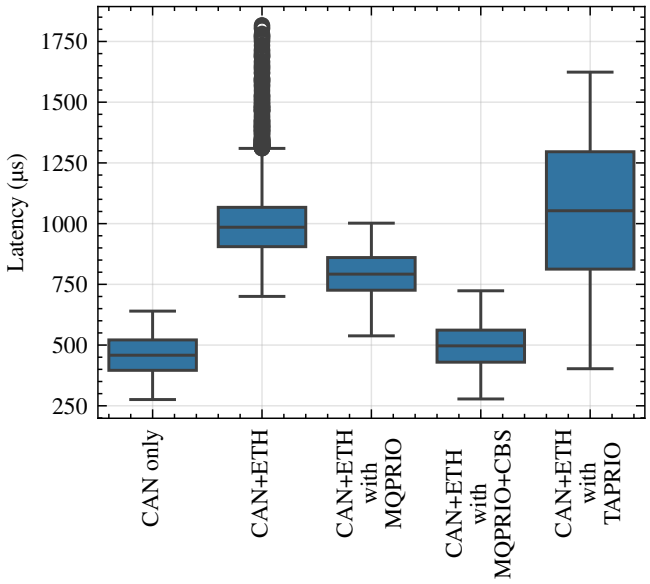


Figure 5. Latency of real-time CAN traffic transmitted the STM32 with the Ethernet network set at 10 Mbps with various traffic profiles and TSN configurations.

Table 1. Statistics of the latencies of real-time CAN traffic transmitted the STM32 with the Ethernet network set at 10 Mbps with various traffic profiles and TSN configurations.

| Scenario | Mean (µs) | Std (µs) | Min (µs) | Max (µs) |
|---|---|---|---|---|
| CAN only | 459 | 80 | 276 | 640 |
| CAN+ETH | 988 | 115 | 700 | 1817 |
| CAN+ETH with MQPRIO | 792 | 91 | 538 | 1002 |
| CAN+ETH with MQPRIO+CBS | 497 | 88 | 278 | 723 |
| CAN+ETH with TAPRIO | 1053 | 289 | 403 | 1624 |

Moreover, to allow a thorough comparison of the different TSN strategies, we also performed a final test in which, instead of using `MQPRIO` and `CBS`, we enabled only `TAPRIO` (see Sec. 3.6.).

### 4.1. Results at 10 Mbit/s

The results of the first set of experiments with the Ethernet network set to 10 Mbit/s are presented in Figure 5, while Table 1 provides more detailed statistical analysis.

As a primary outcome, the baseline latency of CAN traffic, including transmission through the CAN interfaces and processing on the gateway, is approximately 459 µs on average, with a relatively low standard deviation of 80 µs. A particularly interesting result concerns the minimum latency, which is 276 µs. This implies that the total latency overhead introduced by the software components in the STM32, the gateway, and transmission over the Ethernet network is approximately 52 µs. In fact, considering that a full CAN frame is 112 bits long and the CAN bus operates at 500 Kbps, the transmission time for a single frame is 224 µs. The remaining time is therefore attributable to the additional processing and network transmission. This is a positive result as it demonstrates that in the best-case scenario, the gateway does not introduce a significant latency overhead for CAN traffic.

On the other hand, the baseline latency can extend up to a maximum of 640 µs. This variability is primarily due to other traffic on the CAN bus, which can cause delays, collisions, and retransmissions.

When interfering traffic is introduced on the Ethernet network without any TSN features enabled, latency increases significantly. The average latency rises to 988 µs, with a maximum of 1817 µs. This result is expected, as the Ethernet traffic saturates the link and the CAN traffic is not prioritized. Furthermore, the second column of the box plot in Figure 5 shows a significant number of outliers, probably caused by Ethernet traffic interference. The presence of these outliers highlights the lack of determinism in the system, making it unsuitable for hard real-time and time-critical applications.

The introduction of `MQPRIO`, which prioritizes CAN traffic over Ethernet traffic without reserving bandwidth, results in a slight overall reduction in latency, with an average of 792 µs. In this setup, CAN traffic is assigned to a higher priority transmission queue than Ethernet traffic. Consequently, when the switch processes queued frames, it prioritizes those from the CAN bus. Despite this prioritization, it is still not possible to fully restore latency to baseline levels. However, the improvement in determinism is evident as the standard deviation decreases and the presence of outliers is reduced.

Finally, the introduction of `CBS` allows for dedicated bandwidth reservation for CAN traffic. Specifically, the shaper was configured to ensure that the reserved bandwidth supports the same worst-case packet delivery rate as a CAN bus operating at 500 Kbps. For reference, let us consider the following.

- Total length of a CAN frame: 112 bits
- Bitrate of CAN: 500 Kbps
- Packet rate for CAN: $\frac{500000}{112} \approx 4464$ packets per second (pps)

After UDP encapsulation on the gateway, the length of the encapsulated packet becomes 400 bits. To maintain the same packet delivery rate, the minimum bitrate required would be:

$$\text{Minimum bitrate} = 4464 \cdot 400 \approx 1\,785\,600 \text{ bps} \approx 2 \text{ Mbps}. \quad (2)$$

We configured the CBS to reserve 2 Mbps for CAN traffic and 8 Mbps for Ethernet traffic. As shown in Figure 5 and Table 1, the benefits of introducing 802.1Qav are evident. With TSN features enabled (`MQPRIO+CBS`), the latency of CAN traffic is approximately 497 µs. In terms of standard deviation, the bandwidth reservation effectively restores the latency to its original level, i.e. the latency observed without additional Ethernet traffic, at the cost of reserving 2 Mbps of bandwidth. Although a degradation of approximately 10 % still remains in the mean and maximum latency values compared to the baseline (no background traffic), this represents a substantial improvement over the case without TSN features enabled.

For reference, we also conducted an additional test in which, instead of using `MQPRIO` and `CBS`, we only enabled `TAPRIO`, the time-aware shaper of TSN, which regulates traffic based on time windows. As illustrated in Figure 6, we allocated a total transmission window of 1 ms, with 200 µs reserved for CAN traffic and 800 µs for Ethernet traffic. This setup is functionally equivalent to the `MQPRIO+CBS` configuration, but uses a different approach. Specifically, reserving 200 µs from a 1 ms window on a 10 Mbps link effectively corresponds to reserving 2 Mbps bandwidth. However, unlike `CBS`, which reserves bandwidth continuously, `TAPRIO` reserves bandwidth only during the designated time window. This means that traffic with a certain priority can be transmitted only when the corresponding window is open.

As shown in Figure 5 and Table 1, the latency with `TAPRIO` is consistently higher than in the baseline and `CBS` cases. This

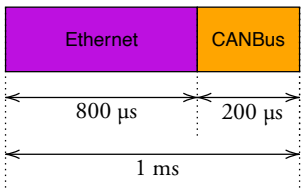


Figure 6. IEEE 802.1Qbv gates schedule.

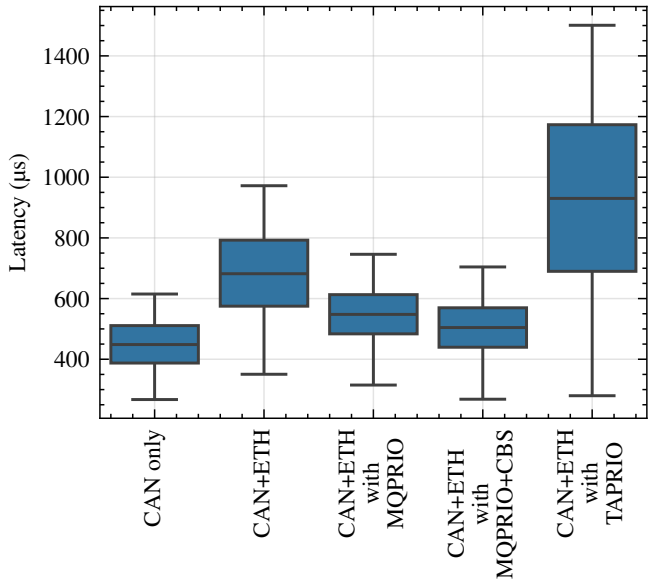


Figure 7. Latency of real-time CAN traffic transmitted the STM32 with the Ethernet network set at 100 Mbps with various traffic profiles and TSN configurations.

Table 2. Statistics of the latencies of real-time CAN traffic transmitted the STM32 with the Ethernet network set at 100 Mbps with various traffic profiles and TSN configurations.

| Scenario | Mean (µs) | Std (µs) | Min (µs) | Max (µs) |
|---|---|---|---|---|
| CAN only | 449 | 78 | 267 | 615 |
| CAN+ETH | 684 | 135 | 351 | 972 |
| CAN+ETH with MQPRIO | 548 | 86 | 315 | 746 |
| CAN+ETH with MQPRIO+CBS | 504 | 87 | 268 | 704 |
| CAN+ETH with TAPRIO | 931 | 289 | 280 | 1501 |

behavior is expected: in the best case, a frame arriving from the CAN bus finds the transmission window already open and is offloaded immediately. However, in the worst case, the frame must wait for the next available window, leading to higher latency and a higher standard deviation. On the other hand, as with CBS, the number of outliers is reduced compared to the scenario without TSN features. This makes TAPRIO a suitable solution for real-time applications where determinism is more critical than absolute latency.

### 4.2. Results at 100 Mbit/s

Based on these results, we performed an additional set of experiments with the Ethernet network set to 100 Mbps. The results of these experiments are shown in Figure 7, while Table 2 provides a more detailed statistical analysis.

As observed, the results confirm the behavior previously discussed with respect to TSN configurations. However, the higher bandwidth reduces the impact of the various TSN features. This is because, at 100 Mbps, the CAN bus traffic represents only 2 % of the total traffic, whereas in the previous case of 10 Mbps, it represented 20 %.

Examining the experiment without the TSN features enabled, there is an increase in overall latency, but it is significantly lower than in the 10 Mbps case. Enabling MQPRIO alone reduces the latency to near-baseline levels. As in the previous scenario, CBS fully restores the original latency, that is, the latency observed without additional traffic.

Special attention should be paid to the case where only TAPRIO is enabled. Here, as observed, there is practically no difference compared to the experiment at 10 Mbps. This is expected, as the presence of time windows produces the same effects as in the previous case.

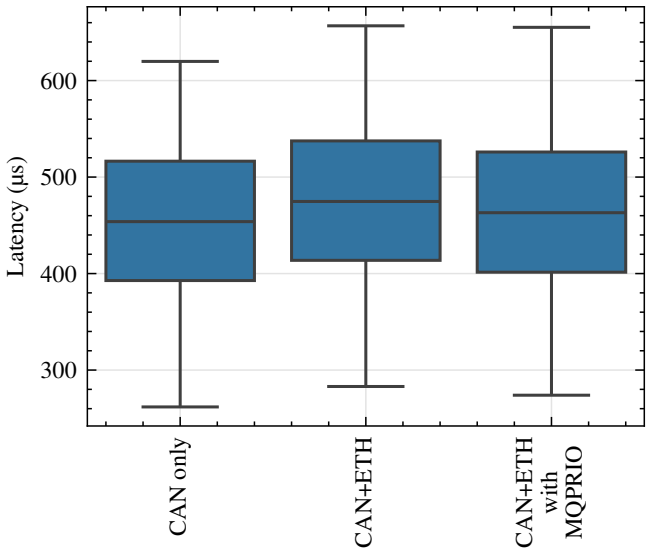


Figure 8. Latency of real-time CAN traffic transmitted the STM32 with the Ethernet network set at 1 Gbps with various traffic profiles and TSN configurations.

Table 3. Statistics of the latencies of real-time CAN traffic transmitted the STM32 with the Ethernet network set at 1 Gbps with various traffic profiles and TSN configurations.

| Scenario | Mean (µs) | Std (µs) | Min (µs) | Max (µs) |
|---|---|---|---|---|
| CAN only | 454 | 76 | 262 | 620 |
| CAN+ETH | 475 | 76 | 283 | 657 |
| CAN+ETH with MQPRIO | 464 | 78 | 274 | 655 |

### 4.3. Results at 1 Gbit/s

Finally, to validate our observations regarding the impact of increased bandwidth, we performed an additional set of experiments with the Ethernet network set to 1 Gbps. The results of these experiments are presented in Figure 8, while Table 3 provides a more detailed statistical analysis.

As observed, when Ethernet traffic is enabled, the increase in latency is marginal, as the high bandwidth effectively supports CAN traffic even without TSN features enabled. Furthermore, the introduction of MQPRIO alone successfully reduces the latency to its baseline value.

### 4.4. Discussion of the results

The results obtained provide valuable information on the impact of TSN features on real-time traffic latency. Firstly, it becomes evident that a well-configured setup of shapers and priorities is crucial for ensuring the operational efficiency and stability of the TSN domain, particularly when the Ethernet link operates at lower bandwidths. In such situations, if time-critical traffic shares bandwidth with high-bandwidth best-effort traffic, there is a risk of frame drops or delays for time-critical transmissions. This issue arises because all frames share the same hardware queues, potentially leading to buffer resource starvation.

Therefore, careful consideration and optimization of priority assignments are essential to maintain the integrity and reliability of real-time communication in TSN-enabled networks.

The feasibility of combining real-time CAN traffic with high-bandwidth Ethernet traffic using an on-board gateway has been demonstrated. When the Ethernet link operates at 10 Mbps, the additional latency introduced by Ethernet traffic is substantial compared to the baseline latency observed with CAN traffic alone. However, the introduction of TSN features, such as 802.1Qav, successfully restores the original latency, even in the presence of other high-bandwidth traffic. This effect becomes progressively less significant as the Ethernet link bandwidth increases, with minimal impact at 100 Mbps and virtually no effect at 1 Gbps.

These findings highlight the importance of TSN features, particularly in Single-Pair Ethernet applications where bandwidth is limited. In such cases, the Credit-Based Shaper is essential to ensure the network's real-time properties. Additionally, the Time-Aware Shaper proves to be a valuable solution for real-time applications that prioritize determinism, even if absolute latency is less critical.

## 5. CONCLUSION

In this paper, we explored the integration of CAN bus communication with in-vehicle 10BASE-T1S single pair ethernet using TSN. The proposed approach leverages TSN switches that support both interfaces on a single hardware platform while allowing custom software execution. The analysis focused on a worst-case bandwidth efficiency scenario, where the CAN-to-Ethernet gateway operates without aggregating CAN frames, thereby avoiding high-level queuing at the application layer. We demonstrated that the proposed implementation integrates seamlessly into a TSN-enabled system and benefits from the TSN functionalities. Specifically, measurements from our prototype established a baseline for CAN bus traffic latency and confirmed that TSN features are crucial at low bandwidths to restore original latency levels, when high-bandwidth traffic is present. Furthermore, the findings showed that other reservation strategies, such as the Time-Aware Shaper, despite being less efficient than the Credit-Based Shaper, can be valuable for real-time applications that prioritize determinism only.

This setup serves as a foundation for more complex scenarios. Future research will extend the experimental setup to incorporate additional TSN switches and devices, allowing an assessment of performance when multiple CAN bus networks coexist within a single TSN domain. In addition, future work will introduce more intricate traffic patterns, such as burst traffic, to examine its impact and address the associated challenges.

## AUTHORS' CONTRIBUTION

A. Morato led the research and was responsible for all core aspects of the work, including: conceptualization, methodology, software, validation, formal analysis, investigation, data curation, visualization, and writing (original draft).

F. Tramarin supported the work through supervision, resources, funding acquisitions, and writing (review and editing).

All authors have read and agreed to the published version of the manuscript.

## REFERENCES


[1] B. Deng, J. Nan, W. Cao, W. Wang, A Survey on Integration of Network Communication into Vehicle Real-Time Motion Control, IEEE Communications Surveys & Tutorials, vol. 25, 2023, no. 4, pp. 2755–2790.
DOI: 10.1109/COMST.2023.3295384

[2] F. Lamonaca, D. L. Carnì, Evaluation of the Effects of Mobile Smart Object to Boost IoT Network Synchronization, Sensors, vol. 21, Jan. 2021, no. 12, p. 3957.
DOI: 10.3390/s21123957

[3] I. Ahmed, E. Balestrieri, P. Daponte, R. Imperatore, F. Lamonaca, M. Paolucci, F. Picariello, Morphometric Measurement of Fish Blood Cell: An Image Processing and Ellipse Fitting Technique, IEEE Transactions on Instrumentation and Measurement, vol. 73, 2024, pp. 1–12.
DOI: 10.1109/TIM.2024.3353280

[4] A. F. Gentile, D. Macrì, D. L. Carnì, E. Greco, F. Lamonaca, A Performance Analysis of Security Protocols for Distributed Measurement Systems Based on Internet of Things with Constrained Hardware and Open Source Infrastructures, Sensors, vol. 24, Jan. 2024, no. 9, p. 2781.
DOI: 10.3390/s24092781

[5] L. Lusvarghi, C. A. Grazia, M. Klapez, M. Casoni, M. L. Merani, Awareness messages by vulnerable road users and vehicles: Field tests via lte-v2x, IEEE Transactions on Intelligent Vehicles, vol. 8, 2023, no. 10, pp. 4418–4433.
DOI: 10.1109/TIV.2023.3280744

[6] L. Leonardi, L. L. Bello, G. Patti, Bandwidth partitioning for time-sensitive networking flows in automotive communications, IEEE Communications Letters, vol. 25, 2021, no. 10, pp. 3258–3261.
DOI: 10.1109/LCOMM.2021.3103004

[7] L. Leonardi, L. L. Bello, G. Patti, Towards time-sensitive networking in heterogeneous platforms with virtualization, 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 2020, vol. 1, pp. 1155–1158.
DOI: 10.1109/ETFA46521.2020.9212116

[8] L. Lo Bello, G. Patti, L. Leonardi, A perspective on ethernet in automotive Communications—Current status and future trends, Applied Sciences, vol. 13, 2023, no. 1278.
DOI: 10.3390/app13031278

[9] K. Appajosyula, Ethernet-based ids for zonal architecture, SAE Technical Paper, 2024, pp. 28–0121.

[10] J. Lim, J. Lee, Y. S. Hong, C. Kang, A Framework for Designing Zonal Architectures for In-Vehicle Networks: Balancing Communication Load and Wiring Length, IEEE Transactions on Vehicular Technology, 2024, pp. 1–14.
DOI: 10.1109/TVT.2024.3522659

[11] T. Fedullo, A. Morato, F. Tramarin, L. Rovati, S. Vitturi, A comprehensive review on time sensitive networks with a special focus on its applicability to industrial smart and distributed measurement systems, Sensors, vol. 22, 2022, no. 4, p. 1638.
DOI: 10.3390/s22041638

[12] S. Sudhakaran, K. Montgomery, M. Kashef, D. Cavalcanti, R. Candell, Wireless time sensitive networking for industrial collaborative robotic workcells, 2021 17th IEEE International Conference on Factory Communication Systems (WFCS), 2021, pp. 91–94.
DOI: 10.1109/WFCS46889.2021.9483447

[13] Single Pair Ethernet (SPE) 10BASE-T1S and 100BASE-T1 Devices Transform IIoT at the Edge and in Higher-Speed Applications. Online. [Accessed: 21 January 2025].
https://www.microchip.com/en-us/about/news-releases/products/single-pair-ethernet-10base-t1s-and-100base-t1-devices-transform

[14] G. Cena, S. Scanzio, A. Valenzano, Composite CAN XL-Ethernet Networks for Next-Gen Automotive and Automation Systems, 2023 IEEE 19th International Conference on Factory Communication Systems (WFCS), Apr. 2023, pp. 1–8. DOI: 10.1109/WFCS57264.2023.10144116

[15] H. Ordouei, F. Gerfers, S. Waldmann, In-Vehicle Network Standards - Overview and Implementation Examples, 2022 IEEE International Symposium on Circuits and Systems (ISCAS), May 2022, pp. 1023–1027. DOI: 10.1109/ISCAS48785.2022.9938007

[16] J. H. Kim, S.-H. Seo, N.-T. Hai, B. M. Cheon, Y. S. Lee, J. W. Jeon, Gateway framework for in-vehicle networks based on CAN, FlexRay, and Ethernet, IEEE transactions on vehicular technology, vol. 64, 2014, no. 10, pp. 4472–4486. DOI: 10.1109/TVT.2014.2371470

[17] A. Kern, D. Reinhard, T. Streichert, J. Teich, Gateway strategies for embedding of automotive CAN-frames into ethernet-packets and vice versa, Architecture of Computing Systems-ARCS 2011: 24th International Conference, Como, Italy, February 24-25, 2011. Proceedings 24, Springer, 2011, pp. 259–270. DOI: 10.1007/978-3-642-19137-4_22

[18] H. Ayed, A. Mifdaoui, C. Fraboul, Frame packing strategy within gateways for multi-cluster avionics embedded networks, Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation (ETFA 2012), IEEE, 2012, pp. 1–8. DOI: 10.1109/ETFA.2012.6489577

[19] S. Bae, H. Park, K. Ko, J.-W. Choi, Trade-off Analysis of Gateway Strategies for Embedding of CAN Frames into Ethernet Packets, 2024 15th International Conference on Information and Communication Technology Convergence (ICTC), Oct. 2024, pp. 611–616. DOI: 10.1109/ICTC62082.2024.10826684

[20] H. Kim, W. Yoo, S. Ha, J.-M. Chung, In-Vehicle Network Average Response Time Analysis for CAN-FD and Automotive Ethernet, IEEE Transactions on Vehicular Technology, vol. 72, June 2023, no. 6, pp. 6916–6932. DOI: 10.1109/TVT.2023.3236593

[21] J. Min, Y. Park, Performance Enhancement of In-Vehicle 10BASE-T1S Ethernet Using Node Prioritization and Packet Segmentation, IEEE Access, vol. 10, 2022, pp. 103 286–103 295. DOI: 10.1109/ACCESS.2022.3209824

[22] J. Min, Y. Park, Application of 10BASE-T1S Ethernet in Delay-Sensitive Vehicular Networks, 2023 IEEE Vehicular Networking Conference (VNC), Apr. 2023, pp. 57–60. DOI: 10.1109/VNC57357.2023.10136336

[23] G. Xie, Y. Zhang, N. Chen, W. Chang, A high-flexibility CAN-TSN gateway with a low-congestion TSN-to-CAN scheduler, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2023. DOI: 10.1109/TCAD.2023.3277812

[24] D. A. Nascimento, S. Bondorf, D. R. Campelo, Modeling and Analysis of Time-Aware Shaper on Half-Duplex Ethernet PLCA Multidrop, IEEE Transactions on Communications, vol. 71, Apr. 2023, no. 4, pp. 2216–2229. DOI: 10.1109/TCOMM.2023.3246080

[25] Zephyr project, Jan. 2025. Online. [Accessed: 29 January 2025]. https://github.com/zephyrproject-rtos/zephyr

[26] I. Val, Ó. Seijo, R. Torrego, A. Astarloa, IEEE 802.1AS clock synchronization performance evaluation of an integrated wired–wireless TSN architecture, IEEE Transactions on Industrial Informatics, vol. 18, 2022, no. 5, pp. 2986–2999. DOI: 10.1109/TII.2021.3106568

[27] A. Morato, C. Zunino, M. Cheminod, S. Vitturi, F. Tramarin, A TSN-based Technique for Real-Time Latency Evaluation in Communication Networks, 2024 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), IEEE, Glasgow, United Kingdom, May 2024, pp. 1–6. DOI: 10.1109/I2MTC60896.2024.10560981

[28] S. Sudhakaran, C. Hall, D. Cavalcanti, A. Morato, C. Zunino, F. Tramarin, Measurement method for end-to-end time synchronization of wired and wireless tsn, 2023 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), 2023, pp. 1–6. DOI: 10.1109/I2MTC53148.2023.10176093

[29] Commaai/opendbc, comma.ai, Mar. 2024. Online. [Accessed: 11 March 2025]. https://github.com/commaai/opendbc