Bridging the Gap: An Interface Architecture for Integrating CAN and TSN Networks

Aldin Berisa[1], Benjamin Kraljusic[1,2], Nejla Zahirovic[1,3], Mohammad Ashjaei[1], Masoud Daneshtalab[1], Mikael Sjödin[1], and Saad Mubeen[1]

[1]Mälardalen University, Västerås, Sweden
[2]Arcticus Systems, Järfälla, Sweden
[3]HIAB, Hudiksvall, Sweden
{firstname.lastname}@mdu.se

Abstract. The increasing complexity of modern embedded systems highlights the limitations of CAN networks in data handling and transmission speed. The IEEE Time-Sensitive Networking (TSN) task group developed standards to enhance switched Ethernet with high bandwidth, low jitter, and deterministic communication. Despite these advances, CAN and TSN will likely coexist in the automotive industry due to the cost-effectiveness of CAN. This paper explores the development and evaluation of an interface architecture that integrates CAN and TSN networks. Using realistic scenarios, we evaluate various gateway forwarding strategies and how they affect the delays of CAN frames passing through the TSN network. We also show how these strategies impact the bandwidth utilization of the TSN network by these frames. Our experiments demonstrate that encapsulating a single CAN frame within a TSN frame effectively minimizes the end-to-end delay of CAN frames, in particular when a high-speed TSN network is used. Furthermore, we compare the performance of Time-Aware Shaper (TAS) and Weighted Round Robin (WRR) on the TSN network, finding that WRR offers improved delays for CAN frames compared to TAS due to the lack of synchronization between CAN and TSN.

Keywords: Controller Area Network · Time-sensitive Network · Gateway · Automotive embedded systems.

1 Introduction

The automotive industry has significantly advanced through the adoption of embedded systems. As vehicles incorporate more features, the number of Electronic Control Units (ECUs) has increased to several tens per vehicle [10]. This large number of ECUs requires a network capable of handling higher data throughput with low latency and real-time requirements.

The Controller Area Network (CAN) [18], referred to as classical CAN in this paper, has been the most widely used in-vehicle network due to its simplicity and reliability. However, with an 8-byte data size limit and a maximum speed of 1 Mbit/s, classical CAN cannot meet the high data-rate demands in modern vehicles. In response, CAN Flexible Data-rate (FD) [5] was developed to increase the payload size and data rates. Despite these enhancements, CAN FD still falls short of meeting the high data-rate requirements in modern vehicles.

Switched Ethernet, offering speeds over 10 Gbit/s, can address these demands but lacks low-jitter and timing-predictable communication [9]. These limitations in the traditional switched Ethernet are addressed by a set of Time-Sensitive Networking (TSN) standards developed by the IEEE TSN task group. These standards provide numerous features such as high-bandwidth, low-latency, low-jitter, and timing-predictable communication, among others [2].

While real-time Ethernet networks like TSN are expected to eventually replace CAN, this transition will be gradual due to the continued use of low-cost legacy networks like CAN [22]. During this transition, CAN and TSN will need to communicate via a gateway node [4]. Several gateway techniques enable communication between CAN and TSN, allowing multiple CAN frames to be encapsulated in a single TSN frame for efficient bandwidth use [17]. However, this can cause delays for CAN frames awaiting encapsulation. Gateway techniques can minimize these delays using timers and marking frames as "urgent" [7, 8, 19].

This paper presents an experimental evaluation of gateway techniques explored by Berisa et al. [4] on a CAN-to-TSN interface architecture. Many TSN networks use unscheduled and unsynchronized off-the-shelf end-systems, such as cameras and LiDARs [3]. We investigate the impact of different traffic shaping mechanisms, such as Time-Aware Shaper (TAS) and Weighted Round Robin (WRR) traffic schedulers, on encapsulated CAN frames in an unsynchronized CAN-to-TSN network. These two schedulers are selected because TAS provides deterministic scheduling which is crucial for time-sensitive applications, while WRR offers a balance between fairness and efficiency, making it suitable for mixed-criticality traffic. The key contributions in this paper are as follows.

- We develop an interface architecture for CAN-to-TSN communication using existing gateway techniques.
- We conduct comparative evaluation of the impact of TAS and WRR schedulers on TSN frames that are transmitted from the gateway in an unsynchronized TSN network.
- We evaluate the CAN-to-TSN gateway and the effect of traffic shapers on TSN frames that are transmitted from the gateway in a realistic automotive use case. Our experiments conclude that encapsulating a CAN frame in a TSN frame is preferable on a 1 Gbit/s TSN network. Additionally, TSN frames transmitted from the gateway experience increased delays using TAS compared to a WRR shaper in an unsynchronized TSN network.

2 Background and Related Work

2.1 Controller Area Network (CAN)

In 1985, Bosch developed the CAN [18] to reduce vehicle weight by decreasing the number of cables needed to connect various ECUs. CAN connects multiple nodes to a single network, simplifying architecture and control. It is an asynchronous multi-master serial data bus that uses fixed-priority non-preemptive scheduling, meaning once a frame starts transmission, it cannot be aborted, and the highest priority frame is transmitted first. Classical CAN operates at speeds up to 1 Mbit/s with frame payloads holding up to 8 bytes.

2.2 CAN Flexible Data-Rate (FD)

CAN FD [5] is an ISO standard that improves on the classical CAN protocol by allowing higher data throughput with payloads up to 64 bytes and data rates up to 8 Mbit/s. Its main advantages are reduced frame transmission times and support for larger frame formats. CAN FD can coexist with classical CAN on the same network by distinguishing transmission bit rates between arbitration and data bits. During arbitration, arbitration bits are transmitted at rates compatible with classical CAN, while data bits are transmitted at higher rates during the data phase.

2.3 Time-Sensitive Networking (TSN)

TSN is a set of IEEE 802.1 standards supporting high-bandwidth, time-critical, and low-latency communication over switched Ethernet [1]. TSN relies on a common notion of time and traffic scheduling. IEEE 802.1AS provides precise clock synchronization with sub-microsecond accuracy. TSN supports both offline and online traffic scheduling. IEEE 802.1Qbv introduces a transmission gate mechanism known as Time-Aware Shaper (TAS) for switch port egress queues, allowing traffic to transmit according to a pre-set schedule, which is known as Gate Control List (GCL), enabling latency-free offline scheduled traffic (ST). TSN, building on IEEE 802.1AVB (Audio-Video Bridging), also supports the credit-based shaper (CBS) for real-time rate-constrained traffic, scheduled online. TSN hardware can also support the Weighted Round Robin (WRR) scheduler instead of CBS. WRR is an online scheduling mechanism sharing bandwidth according to predefined proportions. Each queue is assigned a weight, determining bandwidth allocation, and the scheduler cycles through queues in a round-robin fashion, serving each based on its weight.

2.4 Related Work

Various design strategies for connecting CAN and Ethernet gateways have been explored in prior studies, primarily utilizing Ethernet to facilitate communication between multiple CAN domains.

Early research on gateways between CAN and Ethernet networks was conducted by Scharbarg et al. [17]. They presented techniques for encapsulating CAN frames into Ethernet frames, including single CAN frame encapsulation and multiple CAN frames within a single Ethernet frame (buffered approach). To minimize buffering delays, they introduced a timer mechanism that triggers encapsulation and transmission of the buffered CAN frames.

Kern et al. [8] introduced a dynamically reduced timer upon the arrival of high-priority CAN frames and proposed “urgent” CAN frames, allowing immediate encapsulation and transmission of high-priority frames. Nacer et al. [13] demonstrated that traffic shaping can reduce the load on the receiving CAN bus caused by frames released by the CAN-Ethernet gateway.

Herber et al. [7] introduced a CAN-AVB gateway approach that evaluated different queuing techniques for forwarding CAN messages to the AVB network. AVB frames are transmitted cyclically, with periods determined by the number of CAN frames encapsulated in each AVB frame. Berisa et al. [4] were the first to investigate CAN-to-TSN gateway solutions, showing that encapsulating multiple CAN frames into a single TSN frame may not be optimal for high-speed TSN networks.

Xie et al. [21] developed a CAN-TSN gateway prototype, proposing a low-congestion scheduler using a TSN-to-CAN gateway to forward encapsulated CAN frames. Their scheduler prioritizes frames based on a “Maximum Awaiting Time” (MAT) strategy, giving priority to frames with the least time until their deadline. Their experiments showed improved schedulability of CAN frames but required modifications to the TSN frame header and did not consider the effect of different traffic shapers on the TSN network.

To the best of our knowledge, there is no existing work that has experimentally evaluated a CAN-to-TSN gateway and the effect of different traffic shapers on an unsynchronized TSN network. This paper demonstrates that using TAS on an unsynchronized TSN network may not be the most efficient approach concerning the delays experienced by encapsulated CAN frames.

3 CAN-to-TSN Gateway Architecture and Forwarding Techniques

This section presents the proposed CAN-to-TSN gateway including the gateway architecture and forwarding techniques used by the gateway.

3.1 Interface Architecture of the Gateway

Communication between CAN and TSN networks is facilitated through a gateway node that interfaces with both networks. In this work, we focus on a CAN-to-TSN gateway where CAN frames received at the gateway are transmitted to the TSN network using encapsulation and forwarding techniques. The maximum number of CAN frames that can be encapsulated in a single TSN frame is limited by the maximum payload size of 1500 bytes. The maximum size of a classical CAN frame is 17 bytes, while a CAN FD frame can be up to 74 bytes.

The gateway can generate periodic TSN frames of class AVB but does not support class ST due to the event-driven nature of CAN, which uses online scheduling, whereas class ST is scheduled offline. Sporadic CAN frames are also not supported, as they would require a more sophisticated encapsulation technique for sporadic TSN frames.

A high-level interface architecture of the CAN-to-TSN gateway is shown in Figure 1. When a CAN frame is received at the CAN PHY of the gateway, it is stored in the receive buffer, and an interrupt is generated to the dispatcher. The dispatcher then forwards the frame to the appropriate memory queue based on the destination in the TSN network. The order in which frames are stored in the queues depends on the queuing technique used, which can be First-In-First-Out (FIFO), Fixed-Priority (FP), or one-to-one.

A TSN frame is generated from the memory queues by forwarding a specified number of CAN frames from the queue to the Ethernet MAC. After an encapsulation delay, the generated TSN frame remains in the buffer until the specified period for the TSN frame. This cyclic transmission of frames allows for transmission predictability while limiting the delay experienced by the CAN frame. Finally, the frame is sent to the Ethernet PHY for transmission across the TSN network.

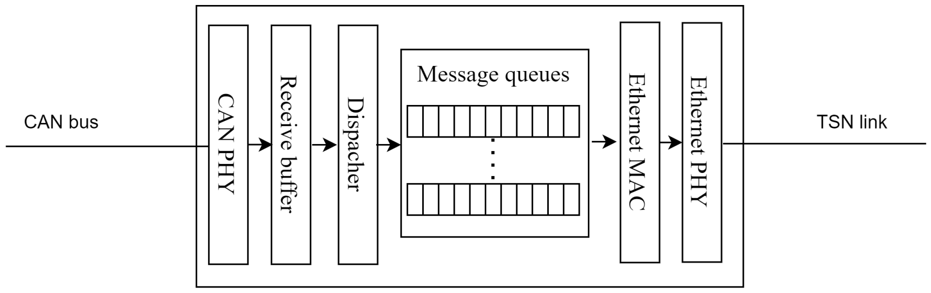

Fig. 1: High-level interface architecture of a CAN-to-TSN gateway.

3.2 Gateway Forwarding Techniques

One-to-one Technique: The one-to-one mapping technique is the simplest approach for encapsulating CAN frames into TSN frames. In this technique, a CAN frame is encapsulated into a TSN frame as soon as it arrives at the gateway. This minimizes delays at the gateway since there is no queuing of frames. However, TSN frames experience overhead due to the small size of CAN frames (up to 17 bytes for CAN and up to 74 bytes for CAN FD), and the minimum size required for a TSN frame is 64 bytes. Padding is necessary to meet the minimum size of a TSN frame. Additionally, creating a TSN frame for each CAN frame utilizes more bandwidth compared to encapsulating multiple CAN frames in a TSN frame.

First-In-First-Out (FIFO) Technique: When utilizing the FIFO forwarding technique, CAN frames are dequeued in the order in which they were added to the message queue. This technique ensures fair forwarding of CAN frames regardless of their priority, but it may result in significant delays for higher-priority CAN frames.

Fixed-Priority (FP) Technique: With the FP forwarding technique, CAN frames are forwarded for encapsulation into TSN frames based on priority, which is determined by their ID. The lower the ID, the higher the priority of the CAN frame. The advantage of this technique is that high-priority frames experience significantly less forwarding delay, as they are prioritized. However, implementing the FP technique is more complex compared to other strategies, and lower-priority frames may experience significantly larger forwarding delays.

4 Configuration of TSN Network

The configuration of the TSN network can impact the delays experienced by the encapsulated CAN frames. In this section, we discuss how to configure the TAS and WRR traffic schedulers in the TSN network and define the periods of the TSN frames transmitted by the gateway.

4.1 Period of TSN Frames Transmitted by the Gateway

To determine the period of TSN frames transmitted from the gateway, we will use Equation (1) proposed by Herber et al [7]:

$$T_{TSN}(q) = \beta / \sum_{\forall i \in fwd(q)} \frac{1}{T_i} \tag{1}$$

Where β is the number of CAN frames encapsulated into a single TSN frame, and $fwd(q)$ represents the set of CAN frames forwarded to queue q.

It's important to consider the period of TSN frames transmitted from the gateway when using the one-to-one forwarding technique. This technique involves immediately encapsulating and transmitting a CAN frame upon its arrival at the gateway. Depending on the CPU of the gateway, a large number of CAN frames arriving at the gateway and requiring immediate encapsulation and transmission can lead to loss of the CAN frames if not handled properly. To address this, we suggest using a one-to-one mapping technique that involves periodic encapsulation of CAN frames. The period for encapsulation should besmaller than the smallest period of CAN frames being forwarded to the queue q as shown in Equation (2).

$$T_{cycle} \leq \min_{\forall i \in fwd(q)} (T_i) \tag{2}$$

The forwarding period should be large enough to ensure timely encapsulation and transmission of the CAN frame to the TSN network. A short period for TSN frames encapsulating CAN frames helps maintain the short end-to-end delays characteristic of one-to-one mapping.

4.2 Configuration of TSN Schedulers

Time-Aware Shaper (TAS): The TAS operates by opening the gate at queue q of a TSN output port based on the GCL schedule. This schedule repeats in a cycle with a predefined period denoted as T_{cycle}. To determine the gate opening time $O(q)$, shown in Equation (3), we must first find the transmission time of the largest frame being transmitted to queue q. The transmission time is calculated by dividing the frame length L by the link transmission rate R.

$$O(q) = \frac{\max_{\forall i \in q}(L_i)}{R} \tag{3}$$

For the GCL schedule to be feasible, the sum of all gate opening times of each queue q_i should be less than or equal to the GCL schedule cycle T_{cycle}, as shown in Equation (4), where Q represents the set of queues of a TSN output port.

$$T_{cycle} \geq \sum_{\forall i \in Q} O(q_i) \tag{4}$$

Weighted Round Robin (WRR) Scheduler: The WRR scheduler ensures that transmission queues receive available bandwidth based on their assigned weights. The scheduler processes the queues in a round-robin fashion, but instead of treating all queues equally in one cycle, it allocates bandwidth to each queue based on its weight. This means that a queue with a higher weight receives a larger share of the bandwidth.

As explained by Walrand et al. [20], the long-term transmission rate of queue q_i can be calculated using the WRR scheduler with Equation (5). In this equation, R_i represents the long-term transmission rate of q_i, w_i is the weight of queue q_i, and S_w is the sum of all weights assigned to the queues.

$$R_i = R\frac{w_i}{S_w} \tag{5}$$

In real-life scenarios, where the long-term transmission rate of the devices sending to queue q_i is known, we can derive the weight of the queue needed to guarantee the necessary bandwidth for the queue using Equation (6).

$$w_i = S_w\frac{R_i}{R} \tag{6}$$

5 Experimental Setup of the CAN-to-TSN Network

This section provides a detailed description of the experimental setup, including insights into the architecture shown in Figure 2. The experimental setup is based on a realistic automotive system provided by our industrial partners where control signals are transmitted over CAN to the TSN network.

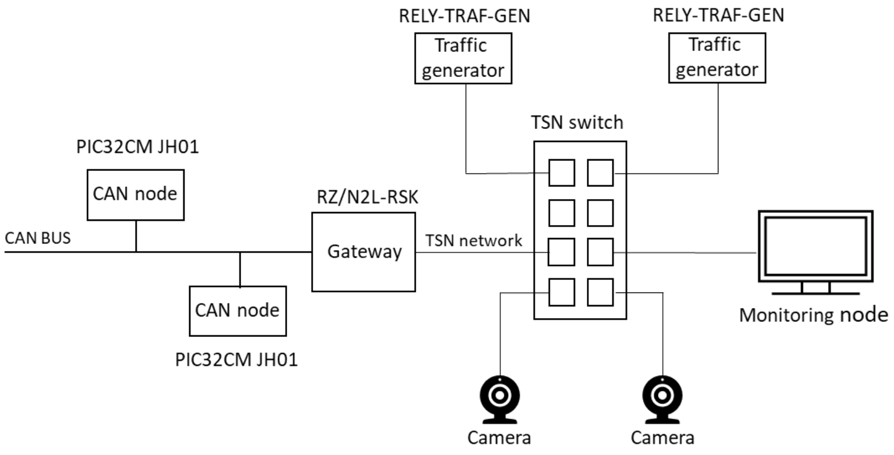

Fig. 2: Model of the experimental set-up of CAN-to-TSN network.

5.1 Implementation of the CAN Network

The CAN network consists of two nodes and a gateway. The CAN and CAN FD nodes are implemented using two Microchip PIC32CMJH01 boards [11]. These boards feature a 48 MHz Arm Cortex M0+ Core microcontroller unit with 512 KB Flash memory, 64 KB SRAM, and two CAN controllers that support classical CAN and CAN FD. CAN frames are transmitted using tasks in FreeRTOS [6] that operate with a period equal to the frame it transmits. To ensure consistency and repeatability in our experimental setup, both CAN nodes start their schedulers only upon simultaneously receiving an external signal and stop frame transmission after a predefined number of hyperperiods This approach ensured that CAN messages always appear on the bus in the same order, which is crucial for comparing different gateway strategies under consistent conditions.

5.2 Implementation of the TSN Network

The TSN network consists of two cameras, a TSN switch, two nodes that generate traffic, and one node that serves as the destination for the encapsulated CAN frames and the gateway. The TSN switch and cameras are provided by our industrial partners and support features such as TAS, WRR, and, in the case of cameras, the ability to stream video feed at 18 Mbit/s. We are using RELY-TRAF-GEN [14] traffic generators capable of producing TSN traffic up to 3 Gbit/s. The destination is a Monitoring node, which is a PC that receives the encapsulated CAN frames. Since the end-stations were not initially intended for integration in a TSN network, they are unable to synchronize with the TSN network.

5.3 Implementation of the CAN-to-TSN Gateway

The CAN-to-TSN gateway is implemented on the Renesas RZ/N2L RSK development board [16]. The board is equipped with an Arm Cortex processor running at 400 MHz, 256 KB flash memory, and 1.5 MB RAM. It also features a TSN switch that supports various TSN mechanisms including IEEE 802.1Qbv and 802.1AS, as well as a CAN FD controller. Message buffers for storing CAN messages are implemented as software buffers, constructed using C arrays. The interrupt routine, CAN ISR, is activated upon the arrival of each CAN frame. During this phase, the interrupt routine stores each frame in the designated buffer. The encapsulation routine is a periodic task executed on the gateway CPU, with the predefined period determined by Equations (1) and (2). This function extracts β CAN frames from the reception buffer and encapsulates them into a data field of an encapsulating TSN frame. The TSN frame is then forwarded to the Ethernet PHY as shown in Figure 1 and transmitted to the network.

5.4 Measuring the End-to-End Delays

Measuring the end-to-end delays of CAN messages requires a precise technique due to the lack of synchronization between devices on the CAN bus and the TSN network. This lack of a unified time reference introduces difficulties in accurately determining end-to-end delays. To address these limitations, we propose a technique that decomposes the end-to-end delay measurement of CAN/CAN FD messages into three distinct components: delays on the CAN/CAN FD network, delays through the gateway, and delays across the TSN network.

Delays on the CAN/CAN FD bus are obtained using the worst-case response time for each message on the CAN bus, calculated using the MPS-CAN Analyzer [12]. The delay experienced at the gateway is measured internally by the gateway itself. To measure delays within the TSN network, we use the RELY-TSN-LAB device [15], which measures network delay by timestamping packets at the input and output of the network.

6 Evaluation

In this section, we evaluate the CAN-to-TSN gateway implementation using an automotive use case. The experimental evaluation involves various gateway techniques discussed in Section 3. We encapsulate CAN frames using different values of β (1, 3, 6, 9, 12) and both FP and FIFO enqueueing of frames. For $\beta = 1$, we have selected T_{TSN} to be 100 μs based on Equation (2). Additionally, we evaluate the impact of the TAS and WRR traffic schedulers in the TSN network. All experiments are conducted in a controlled setting to measure the worst-case end-to-end delay experienced by the encapsulated CAN frames. To ensure the validity of the experiments, each experiment is run for 4 hyperperiods of the CAN frames being forwarded to the gateway.

6.1 Evaluation Scenarios Description

In the first scenario, we assess the delays experienced by the CAN frames using different gateway forwarding techniques and increasing encapsulation size β. Fifteen CAN frames are sent from the CAN nodes to the Monitoring node. The properties of the CAN frames are shown in Table 1. We conduct the evaluation for both classical CAN and CAN FD. Classical CAN operates at 500 Kbit/s. For CAN FD, the message size is increased by a factor of 8 to accommodate the larger supported payloads. The arbitration phase of CAN FD also runs at 500 Kbit/s, while the data phase runs at 2 Mbit/s. The links on the TSN network operate at speeds of 100 Mbit/s and 1 Gbit/s.

Table 1: CAN frames ID 0-14 used in the automotive use case. Periods (T) and their Data Length Code (DLC).

ID	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
T (ms)	50	100	500	20	20	20	20	50	100	100	500	20	1000	40	500
DLC	7	8	3	2	2	8	8	8	4	8	3	8	2	6	8

In the second scenario, we examine the effects of the TAS and WRR traffic schedulers on the delays of encapsulated CAN frames in Table 1 with different gateway forwarding techniques. In this scenario, the cameras stream data at rates of 18 Mbit/s to the Monitoring node. Additionally, the traffic generators also stream data at rates of 40 Mbit/s. The TSN network operates at 100 Mbit/s to effectively demonstrate the impact of the traffic shapers on a high-load TSN network. Because the CAN-to-TSN gateway transmits the highest priority data in the TSN network, the GCL cycle time of the TAS in our experiments is set to be the transmission period of TSN frames originating from the gateway. The weights for the WRR are set as described in Section 4 to ensure the necessary bandwidth for the end-systems. The CAN frames are then transmitted to the Monitoring node, with the CAN network utilizing classical CAN operating at 500 Kbit/s.

6.2 Evaluation Results: Gateway Forwarding Techniques

The experimental evaluation for the first scenario for classical CAN is depicted in Figure 3 while for CAN FD, it is depicted in Figure 4. The graphs show that the encapsulation size β significantly impacts the delays experienced by the CAN frames. When β is set to 1, the frames experience the least amount of delay even though the full one-to-one mapping is not supported by the gateway processor. As we increase β the delays experienced by frames also increase significantly, especially when $\beta = 15$. This is because the period of the TSN frame encapsulating the CAN frames is calculated using β and the period of the CAN frames being transmitted to the gateway. As β increases, the period of the TSN frame becomes larger, resulting in less frequent transmission. Consequently, CAN frames wait longer in the queue for encapsulation.

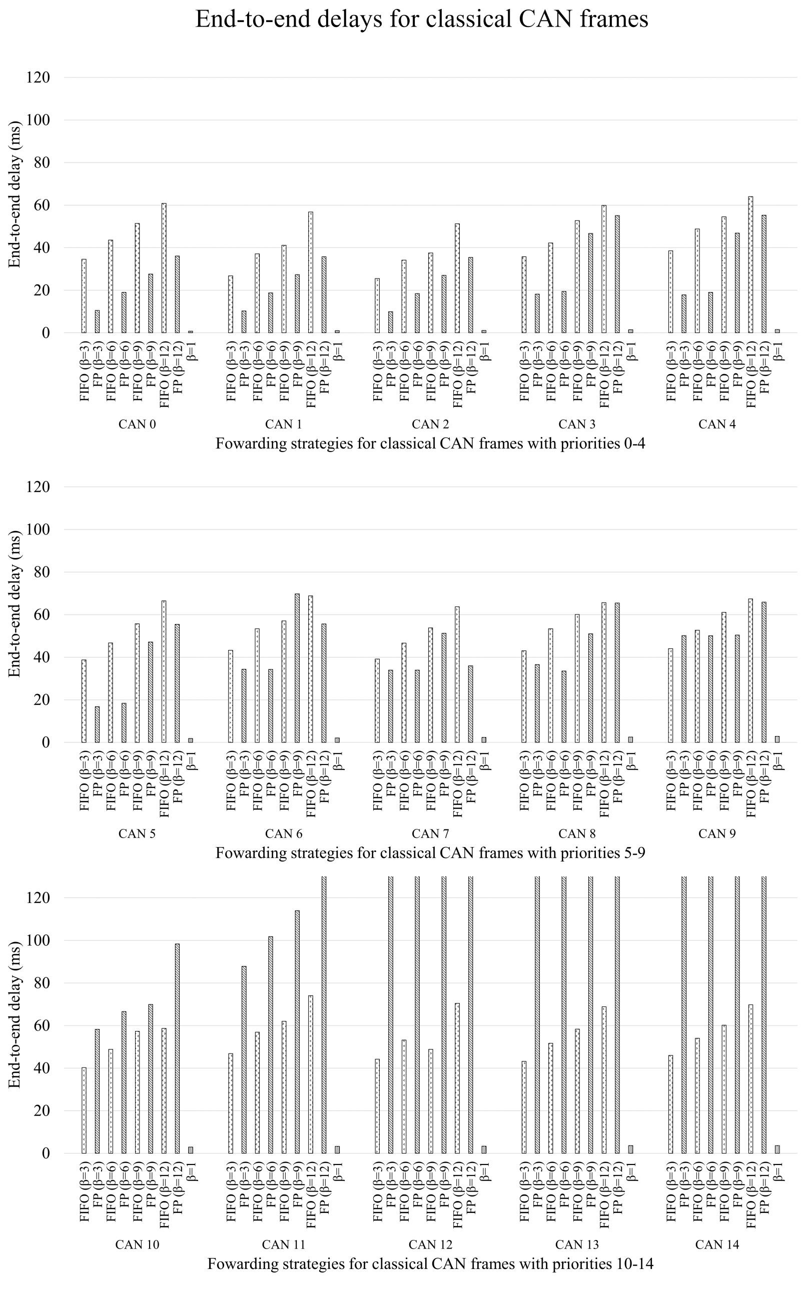

Fig. 3: Evaluation results for various gateway forwarding techniques when using classical CAN.

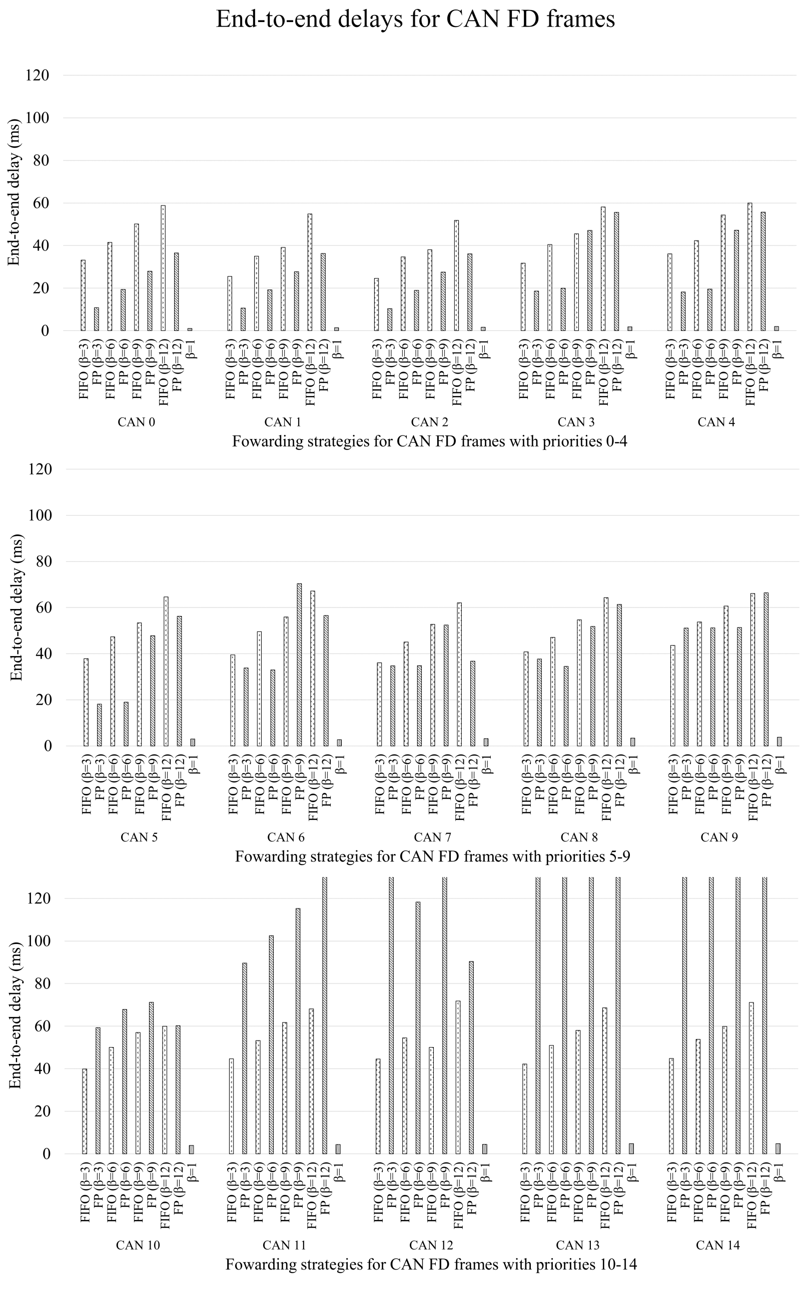

Fig. 4: Evaluation results for various gateway forwarding techniques when using CAN FD.

The delays are also affected by the gateway forwarding technique used. Using FP provides smaller delays for high-priority frames, while low-priority frames experience significantly larger delays, especially with higher values of β. Thus, FIFO might be preferred for lower-priority frames. Lower-priority frames experience high delays with FP because they must wait longer due to the high frequency of high-priority frames arriving in the queue.

When using CAN FD instead of classical CAN, we observed similar trends, with frames experiencing the least delays when $\beta = 1$, and low-priority frames experiencing significantly larger delays with FP.

Lastly, we evaluate the bandwidth utilization on the TSN network depending on the gateway technique used. The graphs depicted in Figure 5 show the TSN network running at 100 Mbit/s and 1000 Mbit/s with different encapsulation sizes for both CAN FD and classical CAN. It is noticeable that using a 1000 Mbit/s TSN network with classical CAN does not show a significant difference when encapsulating multiple CAN frames compared to encapsulating only one CAN frame, with bandwidth usage around 0.5 % and 1.5 %, respectively. For CAN FD, we see 1.5 % bandwidth usage when encapsulating multiple CAN frames and 2.5 % when encapsulating only one. When reducing the TSN link speed to 100 Mbit/s, the difference becomes more noticeable. With $\beta = 1$ and classical CAN, bandwidth usage is 15 %, which can be reduced to 3 % with $\beta = 15$. For CAN FD at 100 Mbit/s, bandwidth usage is 25 % with $\beta = 1$ and 15 % with $\beta = 15$. The larger bandwidth utilization with CAN FD is due to the larger frame sizes, making the TSN frames larger and increasing transmission times.

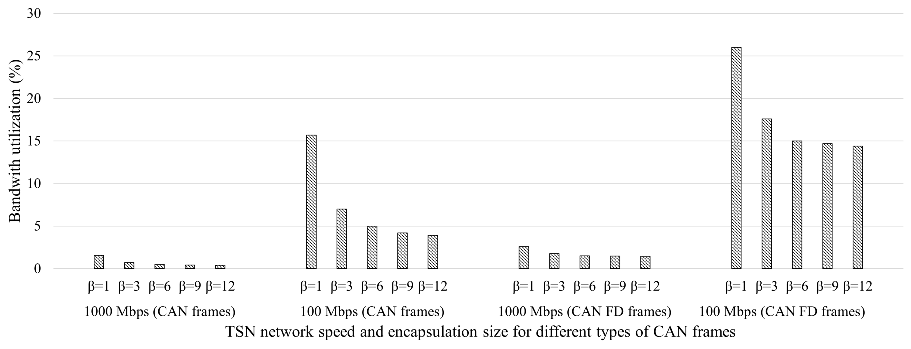

Fig. 5: Evaluation of TSN network bandwidth utilization of TSN frames transmitted from the gateway

Even though using $\beta = 1$ consumes more bandwidth in the TSN network than using larger β values, the lower delays experienced by the CAN frames make it preferable, especially with a 100 Mbit/s TSN network. For slower networks, such as 100 Mbit/s or 10 Mbit/s, encapsulating multiple CAN frames makes sense. Selecting the gateway technique depends on whether high-priority CAN frames can tolerate delays. If immediate transfer is needed, FP is preferred; otherwise, FIFO is recommended to limit delays for lower-priority frames.

6.3 Evaluation Results: TAS and WRR Traffic Shapers

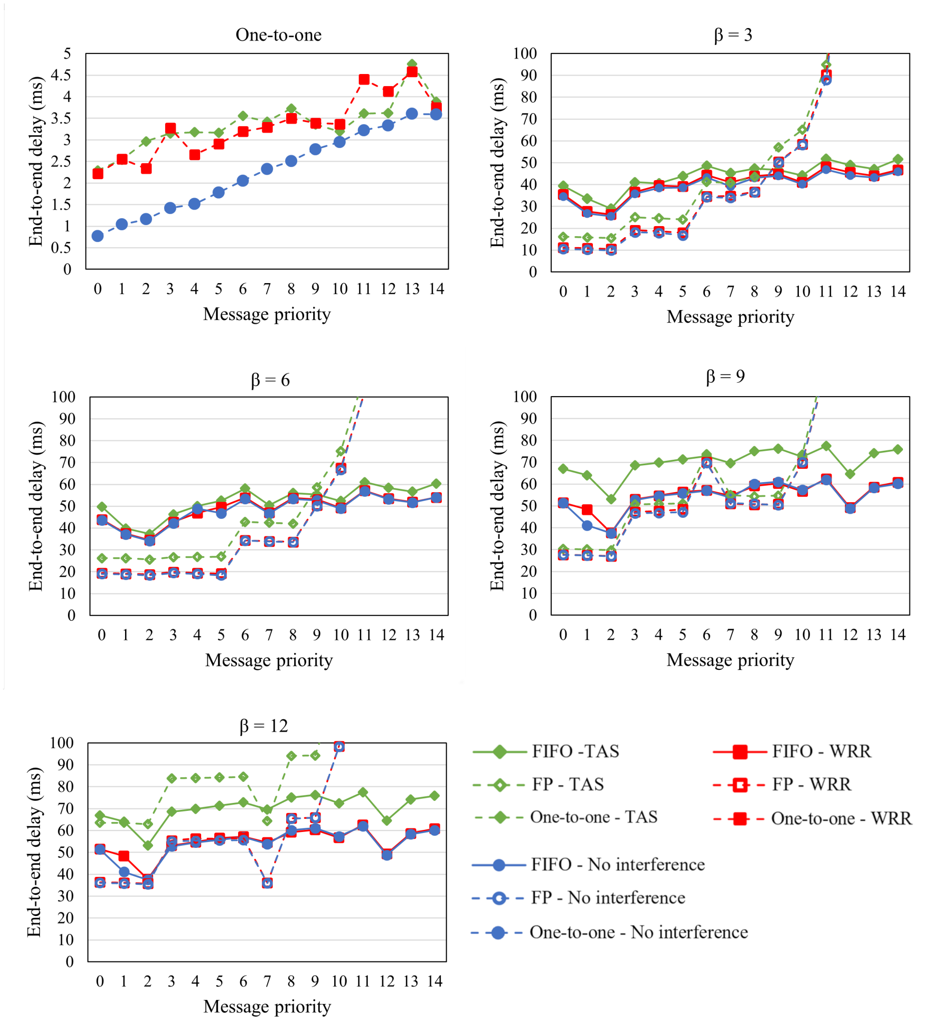

Fig. 6: Comparison of end-to-end delays of encapsulated CAN messages for different TSN traffic schedulers

In the second evaluation scenario, we compared the performance of TAS and WRR traffic shapers, as well as no interference, for the encapsulated CAN frames in the TSN network. The results are presented in Figure 6. TAS performed worse compared to WRR due to the lack of synchronization between the TSN and CAN networks and the devices on the TSN network. TAS delays increased up to one cycle of the GCL because frames could arrive at any point before the next GCL cycle. Sudden increases and decreases in frame delays, particularly noticeable in FIFO forwarding, occur when a frame arrives just after encapsulation, causing it to wait until the next cycle, increasing delay. Conversely, a decrease in delay occurs when a frame arrives just before encapsulation. WRR performed better with lower delays because frames did not have to wait for an extra cycle while ensuring the necessary bandwidth for encapsulated CAN frames. The notable exception is the one-to-one technique, where TSN frames are transmitted at a high frequency of 100 μs. In this case, TAS and WRR had comparable effects.

Although these findings suggest that WRR is preferable to TAS in this specific experimental case, it is important to note that TAS is designed for synchronized networks with dedicated offline scheduling to achieve optimal performance. In such cases, TAS is expected to outperform WRR.

7 Conclusion

In this paper, we developed a prototype of an interface architecture for a CAN-to-TSN gateway and evaluated various gateway forwarding techniques. Moreover, we also investigated the impact of Time-Aware Shaper (TAS) and Weighted Round Robin (WRR) traffic schedulers on the encapsulated CAN frames in an unsynchronized TSN network. The techniques assessed included First-In-First-Out (FIFO), Fixed-Priority (FP), and one-to-one mapping. Our experiments, conducted using an experimental hardware setup based on a real-world automotive use case, demonstrated that combining multiple CAN frames into a single TSN frame does not significantly increase bandwidth usage on high-speed links compared to encapsulating a single CAN frame. In terms of delay, the one-to-one forwarding technique is preferred, as it simplifies the gateway design and provides lower delays. Additionally, in our experiments, we found that due to the lack of synchronization in the TSN network, the WRR scheduler is more effective than TAS. WRR provided lower delays and ensured the necessary bandwidth for encapsulated CAN frames, making it a preferable choice in unsynchronized TSN networks.

Acknowledgments. This work is supported by the Swedish Governmental Agency for Innovation Systems (VINNOVA) via the INTERCONNECT and PROVIDENT projects and by the Swedish Knowledge Foundation via the projects SEINE. We would like to thank our industrial partners HIAB, ABB and Arcticus Systems.

References

1. IEEE Time-Sensitive Networking (TSN) Task Group, 2016, http://www.ieee802.org/1/pages/tsn.html
2. Ashjaei, M., Lo Bello, L., Daneshtalab, M., Patti, G., Saponara, S., Mubeen, S.: Time-sensitive networking in automotive embedded systems: State of the art and research opportunities. J. Syst. Archit. **117**(C) (aug 2021)
3. Barzegaran, M., Reusch, N., Zhao, L., Craciunas, S.S., Pop, P.: Real-time guarantees for critical traffic in IEEE 802.1 qbv TSN networks with unscheduled and unsynchronized end-systems. arXiv preprint arXiv:2105.01641 (2021)

4. Berisa, A., Ashjaei, M., Daneshtalab, M., Sjödin, M., Mubeen, S.: Investigating and analyzing CAN-to-TSN gateway forwarding techniques. In: IEEE ISORC. pp. 136–145 (2023)
5. Bosch, R.: CAN with flexible data-rate specification. Robert Bosch GmbH, Stuttgart (2012)
6. FreeRTOS: Freertos: Real-time operating system for microcontrollers. `https://www.freertos.org`
7. Herber, C., Richter, A., Wild, T., Herkersdorf, A.: Real-time capable can to avb ethernet gateway using frame aggregation and scheduling. In: DATE. pp. 61–66. IEEE (2015)
8. Kern, A., Reinhard, D., Streichert, T., Teich, J.: Gateway strategies for embedding of automotive CAN-frames into ethernet-packets and vice versa. In: ARCS. pp. 259–270. Springer (2011)
9. Lim, H.T., Völker, L., Herrscher, D.: Challenges in a future IP/Ethernet-based in-car network for real-time applications. In: DAC. pp. 7–12 (2011)
10. Lo Bello, L., Mariani, R., Mubeen, S., Saponara, S.: Recent advances and trends in on-board embedded and networked automotive systems. IEEE Transactions on Industrial Informatics **15**(2) (2019)
11. Microchip: PIC32CM JH family of microcontrollers (2024), `https://www.microchip.com/en-us/products/microcontrollers-and-microprocessors/32-bit-mcus/pic32-32-bit-mcus/pic32cm-jh`
12. Mubeen, S., Mäki-Turja, J., Sjödin, M.: MPS-CAN analyzer: Integrated implementation of response-time analyses for controller area network. Journal of Systems architecture **60**(10) (2014)
13. Nacer, A.A., Jaffres-Runser, K., Scharbarg, J.L., Fraboul, C.: Strategies for the interconnection of CAN buses through an ethernet switch. In: 2013 8th IEEE SIES. IEEE (2013)
14. Relyum: RELY-TRAF-GEN, Time-Sensitive Traffic Generator (2023), `https://soc-e.com/rely-traf-gen/`, accessed: 2024-05-13
15. Relyum: RELY-TSN-LAB, Time-Sensitive Networking Testing Tool (2023), `https://www.relyum.com/web/rely-tsn-lab/`, accessed: 2024-05-13
16. Renesas: RZ/N2L RSK, renesas starter kit for RZ/N2L (2024), `https://www.renesas.com/us/en/products/microcontrollers-microprocessors/rz-mpus/rzn2l-rsk-renesas-starter-kit-rzn2l`
17. Scharbarg, J.L., Boyer, M., Fraboul, C.: CAN-Ethernet architectures for real-time applications. In: IEEE ETFA. vol. 2, pp. 8–pp. IEEE (2005)
18. Standard, I.: 11898: Road vehicles—interchange of digital information—controller area network (CAN) for high-speed communication. International Standards Organization, Switzerland (1993)
19. Thiele, D., Schlatow, J., Axer, P., Ernst, R.: Formal timing analysis of CAN-to-Ethernet gateway strategies in automotive networks. Real-time systems **52**(1), 88–112 (2016)
20. Walrand, J.: A concise tutorial on traffic shaping and scheduling in time-sensitive networks. IEEE Communications Surveys & Tutorials (2023)
21. Xie, G., Zhang, Y., Chen, N., Chang, W.: 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)
22. Zinner, H., Noebauer, J., Gallner, T., Seitz, J., Waas, T.: Application and realization of gateways between conventional automotive and IP/Ethernet-based networks. In: DAC. pp. 1–6 (2011)