Experimental Evaluation of a CAN-to-TSN Gateway Implementation

*Aldin Berisa, †Benjamin Kraljusic, ‡Nejla Zahirovic, *Mohammad Ashjaei,
*Masoud Daneshtalab, *Mikael Sjödin,*Saad Mubeen
*Mälardalen University, Västerås, Sweden;
†Arcticus Systems, Järfälla, Sweden;
‡HIAB, Hudiksvall, Sweden;
firstname.lastname@mdu.se

***Abstract*—The increasing complexity of modern embedded systems highlights the limitations of Controller Area Network (CAN) in terms of transmission speed and scalability. The IEEE Time-Sensitive Networking (TSN) task group developed a set of standards to enhance switched Ethernet with high bandwidth, low jitter, and deterministic communication. Despite these advances, CAN will likely co-exist with TSN in, e.g., the automotive industry due to factors such as cost-effectiveness and legacy of CAN. This paper presents an experimental evaluation of a CAN-to-TSN gateway implementation, focusing on the impact of different forwarding and scheduling strategies on network performance. We analyze various queuing techniques and scheduling mechanisms in a realistic experimental setup and assess their impact on end-to-end delay and TSN bandwidth utilization. The evaluation results demonstrate that encapsulating only 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 perform a comparative evaluation of the Time-Aware Shaper (TAS) and Weighted Round Robin (WRR) mechanisms in the TSN network. Interestingly, WRR leads to lower delays for CAN frames in the TSN network compared to TAS, which we attribute to the lack of synchronization between CAN and TSN.**

***Index Terms*—Controller Area Network, CAN, Time-sensitive Network, TSN, Gateway, Automotive embedded systems.**

I. 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)[1] has increased to several tens per vehicle [2]. 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) [3], referred to as classical CAN in this paper, is the most widely used onboard network due to its simplicity, reliability, and low cost. However, classical CAN, with an 8-byte data payload limit and a maximum speed of 1 Mbit/s, cannot meet the high data-rate demands of modern vehicles. To address this, the CAN Flexible Data-rate (FD) [4] was developed to increase both the payload size and the data rates. Despite these enhancements, CAN FD still does not meet the high data-rate requirements of modern vehicles.

Switched Ethernet, offering speeds over 10 Gbit/s, can address the high data-rate requirements but lacks the support for low-jitter and timing-predictable communication [5]. These limitations in the traditional switched Ethernet are addressed by the set of Time-Sensitive Networking (TSN) standards that are 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 [6].

Although real-time Ethernet networks such as TSN are expected to eventually replace CAN, the transition will be gradual due to the continued use of low-cost legacy CAN networks [7]. During this transition, CAN and TSN will need to communicate through a gateway [8]. 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 [9]. However, this can cause delays for CAN frames awaiting encapsulation. Gateway techniques can minimize these delays using timers and marking frames as "urgent" [10]–[12].

While existing studies have analyzed different CAN-to-TSN forwarding strategies, there is still a lack of experimental evaluation regarding their performance in unsynchronized TSN networks. Many real-world TSN networks operate without global synchronization and rely on unscheduled off-the-shelf end-systems, such as cameras and LiDARs, which introduce unpredictability into the network [13]. Most previous work has focused on theoretical analysis or synchronized network environments [8], [14] and has not fully explored how traffic shaping mechanisms such as Time-Aware Shaper (TAS) and Weighted Round Robin (WRR) affect CAN-to-TSN communication in such scenarios. A deeper investigation is necessary to understand the impact of queuing, forwarding, and traffic shaping mechanisms on latency and bandwidth utilization in unsynchronized networks.

In this paper, we present an experimental evaluation of a CAN-to-TSN gateway implementation, focusing on the effects of different forwarding and scheduling techniques on network performance. We evaluate the impact of various traffic shaping mechanisms, including TAS and WRR, on encapsulated CAN frames in an unsynchronized CAN-to-TSN network. These two traffic shaping mechanisms are selected because TAS provides deterministic scheduling which is crucial to time-sensitive applications, while WRR offers a balance between fairness and efficiency, making it suitable for mixed-criticality traffic. Additionally, we compare the measured forwarding delays from our experiments with the theoretical analysis in [8]

A version of this work has been made available as a technical report for indexing [1]. This report does not constitute published work.

[1]We will use the terms ECU and node interchangeably throughout the paper.

979-8-3315-9984-3/25/$31.00 ©2025 IEEE

to validate the accuracy of the analytical model. The key contributions of this paper are as follows:

- We implement and experimentally evaluate a CAN-to-TSN gateway in a realistic automotive use case, assessing how different queuing and forwarding strategies affect transmission latency and network utilization.
- We conduct a comparative evaluation of the impact of TAS and WRR schedulers on TSN frames that are transmitted from the gateway.
- Our experiments demonstrate that encapsulating a single 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. Furthermore, a comparison between the measured forwarding delays and the theoretical analysis from [8] confirms that the analytical model provides accurate upper-bound estimates.

II. Background and Related Work

A. Controller Area Network (CAN)

In 1985, Robert Bosch developed the CAN protocol to reduce vehicle weight by decreasing the number of cables needed to connect various ECUs. The protocol was later standardized in ISO 11898 [3]. CAN connects multiple nodes to a single network, simplifying architecture and control. It is an asynchronous multi-master serial data network 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 up to 8 bytes.

B. CAN Flexible Data-Rate (FD)

CAN FD [4] is an ISO standard that improves on 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.

C. 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 [15]. TSN leverages features such as a common notion of time and traffic shaping to enable deterministic networking.The IEEE 802.1AS standard enables precise clock synchronization with sub-microsecond accuracy, which is essential for time-triggered scheduling mechanisms in TSN. Among these, the IEEE 802.1Qbv standard introduces the Time-Aware Shaper (TAS), which controls the transmission of traffic at switch port egress queues using a gate control mechanism. This mechanism allows traffic to be transmitted according to a pre-set schedule, which is known as the Gate Control List (GCL), enabling latency-free offline scheduled traffic (ST). TSN, which is built upon the IEEE 802.1AVB (Audio-Video Bridging) standards, also supports the credit-based shaper (CBS) for real-time rate-constrained traffic scheduled online. TSN hardware can support the Weighted Round Robin (WRR) scheduler, and according to IEEE 802.1Qaz, WRR can coexist with CBS on the same output port. 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 queue based on its weight.

D. Related Work

Several techniques have been proposed for enabling communication between CAN and Ethernet domains via gateways. Early work by Scharbarg et al. [9] and Kern et al. [10] explored basic encapsulation strategies and timer-based release mechanisms to balance bandwidth efficiency and latency. To improve responsiveness, urgent-frame handling and dynamic timers were introduced.

Subsequent work focused on scheduling and traffic shaping. Nacer et al. [16] proposed shaping outgoing traffic to reduce the burst-induced load on the receiving CAN bus. Herber et al. [11] studied queuing strategies in CAN-to-AVB gateways using cyclic AVB transmission but only evaluated gateway delays. More recent work by Thiele et al. [12] provided event-model abstractions for worst-case delay analysis.

In the context of TSN, several studies have focused on improving schedulability. Xie et al. [17] and Wu et al. [18] proposed low-latency scheduling strategies, such as Maximum Awaiting Time (MAT) and a high response ratio priority scheduling algorithm (HRRP) for CAN-to-TSN gateways. However, these approaches often rely on header modifications or focus only on best-case latencies. Yan et al. [19] proposed offline optimization methods for encapsulation and scheduling, though their assumptions (e.g., capped TSN payload, deadlines larger than periods) limit generality. Morato et al. [14] evaluated CBS shaping in a one-to-one mapping scheme but assumed synchronized CAN and TSN nodes.

To the best of our knowledge, no prior work has experimentally evaluated the effect of queuing and traffic shaping strategies such as TAS and WRR on a gateway implementation under an unsynchronized CAN-to-TSN setup. This paper addresses this gap through a prototype-based evaluation of real-time performance across varying queuing techniques and TSN traffic shapers.

III. CAN-to-TSN Gateway Architecture

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

A. Architecture of the Gateway

Communication between CAN and TSN networks is facilitated through a gateway node that interfaces both networks. In this work, we focus on a CAN-to-TSN gateway based on the architecture presented in [8], 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 generates periodic TSN frames but does not implement the ST class as defined in TSN. While CAN traffic is event-driven and scheduled dynamically, the gateway buffers and transmits frames at periodic intervals, without strict offline scheduling requirements. The periodic forwarding from the gateway ensures bounded delays without explicit synchronization with a global schedule. While this study focuses on periodic CAN frames, the existing gateway techniques can also handle sporadic CAN frames. In such cases, sporadic arrivals may introduce additional queuing delays, particularly under high network load. Future work could explore the impact of sporadic CAN traffic and evaluate the effectiveness of different scheduling policies in such scenarios.

A high-level architecture of the CAN-to-TSN gateway is shown in Figure 1. When a CAN frame is received at the CAN Physical Layer (PHY), which is responsible for the physical transmission of data over the network medium, it is stored in the receive buffer, and an interrupt is generated to the dispatcher. The dispatcher assigns incoming CAN frames to memory queues based on their TSN destination. In the current implementation, each destination in the TSN network is assigned a separate queue, ensuring that frames are destined for the same endpoint. This approach simplifies scheduling but may require careful memory management when handling a large number of TSN destinations. 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. In the one-to-one approach, each incoming CAN frame is forwarded immediately for encapsulation without additional queuing.

A TSN frame is generated from the memory queues by forwarding a specified number (β) of CAN frames from the queue to the Ethernet Media Access Control (MAC) layer, which manages access to the physical network medium and performs frame encapsulation. The parameter β determines how many CAN frames must be accumulated before encapsulation into a TSN frame, directly impacting latency and bandwidth efficiency. After an encapsulation delay, the generated TSN frame remains in the buffer until the specified period for the TSN frame. This cyclic transmission of TSN frames allows predictable transmission of the TSN frames while limiting the delay experienced by the CAN frame. Finally, the frame is sent to the Ethernet PHY for transmission across the TSN network. It is important to note that traffic shaping does not occur within the gateway itself. Instead, shaping is applied at the TSN switch, where TAS and WRR schedulers regulate the transmission of TSN frames. The gateway primarily functions as a bridge, encapsulating CAN frames into TSN frames and forwarding them according to the selected queuing policy.

B. Gateway Forwarding Techniques

1) One-to-one Technique: The one-to-one mapping is the simplest technique 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 for the received CAN frames at the gateway since this technique does not require any queuing of the frames. However, TSN frames experience overhead due to the small

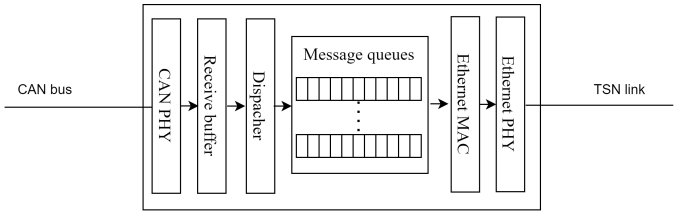

Fig. 1: High-level architecture of a CAN-to-TSN gateway.

size of CAN frames (up to 17 bytes for CAN and up to 74 bytes for CAN FD). The minimum Ethernet frame size is 64 bytes as per IEEE 802.3. However, for real-time analysis, the total transmission overhead must also include the preamble, start frame delimiter, and interframe gap, leading to a total minimum transmission size of 84 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 as compared to encapsulating multiple CAN frames in a TSN frame.

2) 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 frame 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 that arrive later than the lower-priority CAN frames.

3) 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 techniques, and lower-priority frames may experience significantly larger forwarding delays at the gateway.

IV. Configuration of the 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.

A. 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. [11]:

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

Where $T_{TSN}(q)$ represents the transmission period of TSN frames from queue q, 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 with T_i being the period of the forwarded CAN frame i.

It is 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 frame loss if not efficiently managed. One commonly used approach is to encapsulate multiple CAN frames into a single TSN frame, which reduces CPU processing overhead and improves transmission efficiency. However, in this study, we implemented a one-to-one mapping approach to evaluate its impact on queuing and forwarding behavior. While this method may require additional CPU processing and bandwidth on the TSN network, it reduces buffering delays experienced by the CAN frames. This one-to-one mapping technique involves periodic encapsulation of CAN frames. In general, the encapsulation period should be no greater than the smallest period among all CAN frames forwarded to a queue to ensure timely transmission:

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

However, in the specific case of one-to-one mapping, where each CAN frame is encapsulated individually upon arrival, the encapsulation period should instead cover the sum of the frequencies of all CAN frames in the queue. This stricter condition ensures that all queued frames are encapsulated at the required rate, preventing queue buildup and maintaining the required real-time constraints:

$$T_{TSN}(q) \leq \frac{1}{\sum_{\forall i \in fwd(q)} T_i} \tag{3}$$

It is important to note that T_{TSN} is independent of T_{cycle}, which refers to the cycle time of the GCL in TAS mechanism. Unlike T_{cycle}, which is determined by the TSN schedule, T_{TSN} is dictated by the encapsulation strategy applied to CAN frames in the gateway.

B. Configuration of TSN Schedulers

1) Time-Aware Shaper (TAS): The TAS operates by opening the gate at a TSN output port queue q based on the GCL schedule. In this work, we consider multiple queues per TSN output port, as different devices transmitting through the switch may require distinct traffic classes. While TAS is typically applied only to the highest priority, our setup considers scheduling multiple queues to accommodate different traffic priorities in the system.

The GCL schedule repeats in a cycle with a predefined period denoted by T_{cycle}. To determine the gate opening time $O(q)$, shown in Equation (4), 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.

$$O(q) = \frac{\max_{\forall i \in q}(L_i)}{R} \tag{4}$$

Since we consider a single TSN switch and a single scheduled link, we do not explicitly define queue offsets, as all transmissions are scheduled relative to the same cycle start time. However, in multi-hop networks, queue offsets would need to be carefully managed to avoid contention.

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 (5), where Q represents the set of queues of a TSN output port.

$$T_{cycle} \geq \sum_{\forall i \in Q} O(q_i) \tag{5}$$

2) 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, which refers to the average rate at which data is transmitted from the queue over an extended period of time, can be calculated using the WRR scheduler with Equation (6). 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.

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

In real-life scenarios, where the long-term transmission rate of the devices sending to queue q_i is known. The weight of the queue needed to guarantee the necessary bandwidth for the queue can be derived using Equation (7).

$$w_i = S_w\frac{R_i}{R} \tag{7}$$

V. Experimental Setup

This section provides a detailed description of the experimental setup of the CAN-to-TSN network as 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.

A. Implementation of the CAN Network

The CAN network consists of two nodes and a gateway. One node transmits CAN frames while the other node transmits CAN FD frames. The CAN and CAN FD nodes are implemented using two Microchip PIC32CMJH01 evaluation boards [21]. Each evaluation board features a 48 MHz Arm Cortex M0+ Core microcontroller with 512 KB Flash memory, 64 KB SRAM, and two CAN controllers that support classical CAN and CAN FD. The microcontroller runs FreeRTOS [22] real-time operating system.

Each CAN or CAN FD frame inherits its period from the period of its transmitting task. To ensure consistency and repeatability in our experimental setup, both nodes start their schedulers simultaneously and stop frame transmission after a predefined number of hyperperiods in the CAN schedule. A hyperperiod is defined as the least common multiple (LCM) of the periods of all CAN frames transmitted across the entire CAN network. This approach ensures that CAN frames always appear on the network in the same order, which is crucial

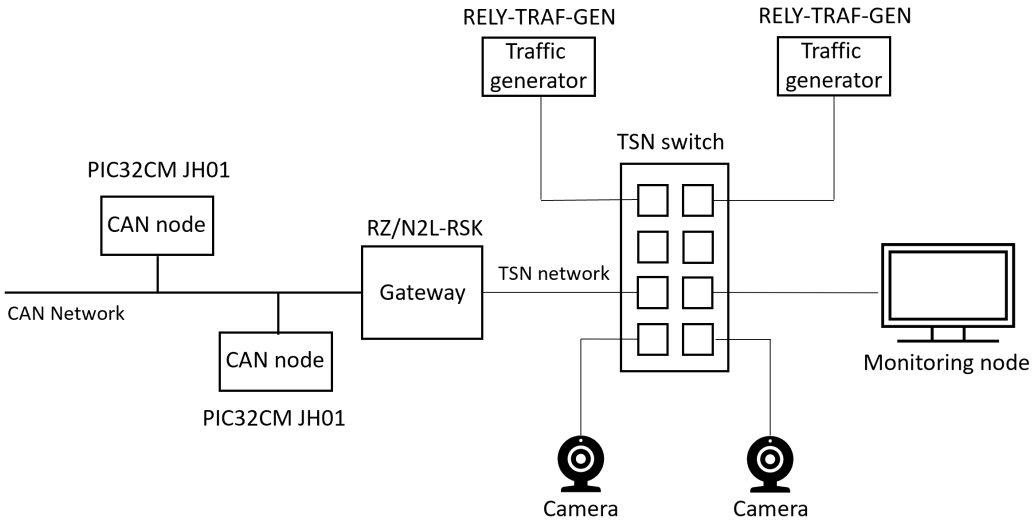

Fig. 2: Experimental hardware set-up of CAN-to-TSN network.

TABLE I: 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

for comparing different gateway strategies under consistent conditions.

B. Implementation of the TSN Network

The TSN network consists of a TSN switch connecting two video cameras, two traffic-generating nodes, a destination node, and the CAN-to-TSN gateway as shown in Figure 2. The TSN switch and cameras are industrial-grade devices used in construction vehicles. However, due to confidentiality agreements, details regarding the specific models of the TSN switch and cameras cannot be disclosed. The cameras stream constant bitrate (CBR) video traffic at 18 Mbit/s each and are assigned to output port queues 6 and 5 in the TSN switch. Traffic is generated using RELY-TRAF-GEN [23], capable of producing TSN traffic up to 3 Gbit/s, with generated flows ranging from 256 bytes to 1518 bytes and interarrival times between 10–200 μs, assigned to queue 0. The encapsulated CAN frames are transmitted at data rates aligned with automotive sensor traffic and are assigned to the highest-priority queue 7 in the TSN output port. Since the nodes in the TSN network are legacy nodes, they are not synchronized with the network.

The TSN switch used in this study was provided by our industrial partner and supports TAS and WRR. WRR was selected as the traffic shaping mechanism because it is widely supported across TSN-capable hardware, including the switch used in this study. We acknowledge that Interleaved WRR (IWRR) has been shown to provide improved real-time performance over classical WRR in some scenarios [24]. However, since our TSN switch only supports standard WRR, we focus our evaluation on this mechanism.

C. Implementation of the CAN-to-TSN Gateway

The CAN-to-TSN gateway is implemented on the Renesas RZ/N2L RSK development board [25]. 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. Frame buffers for storing CAN frames are implemented as software buffers.

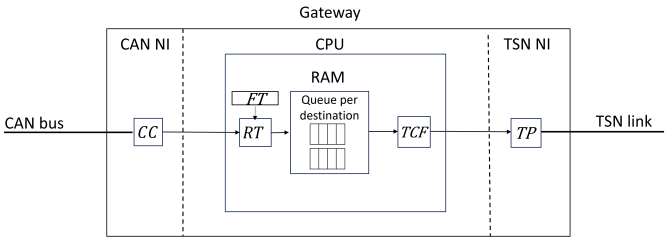

Fig. 3: Implementation of the CAN-to-TSN gateway

The frame encapsulation pipeline is illustrated in Figure 3, where CAN frames arriving at the CAN Network Interface (CAN NI) are first processed by the CAN Controller (CC), which extracts basic frame information and ensures error-free reception. The Receive Task (RT) is then triggered to classify the frame based on its ID. The Forwarding Table (FT) is used to determine the TSN destination of the incoming CAN frame. The RT checks the FT to identify whether the frame has a known TSN destination and assigns it to its corresponding queue in RAM. A periodic TSN Creation and Forwarding (TCF) task is executed on the gateway CPU at intervals defined

by Equations (1) and (2). The TCF task extracts β CAN frames from the buffer, encapsulates them into a TSN frame, and forwards them to the TSN Port (TP) of the TSN Network Interface (TSN NI) for transmission.

In our experimental setup, all encapsulated CAN frames are forwarded to the highest-priority TSN queue (queue 7) in the TSN switch to ensure minimal transmission latency. Therefore, queue selection does not impact performance, as all CAN frames are treated with the highest priority. While it would be possible to distribute CAN frame priorities across multiple TSN queues, this would introduce additional transmission delays, as lower-priority queues may experience waiting times due to traffic shaping mechanisms such as TAS and WRR. Since the focus of this study is on evaluating queuing and traffic shaping strategies rather than queue mapping effects, we chose to use queue 7 for all frames.

The detailed frame processing steps of the gateway are outlined in Algorithm 1, which formalizes the frame reception, queuing, encapsulation, and forwarding process.

Algorithm 1 CAN-to-TSN Gateway Frame Processing

Input: Incoming CAN frame F_{CAN}
Output: Encapsulated and transmitted TSN frame F_{TSN}

// Step 1: Receive CAN Frame
CAN frame F_{CAN} arrives at CAN Network Interface (CAN NI)
CAN Controller (CC) processes the frame and triggers the Receive Task (RT)
RT extracts data (ID, payload)

// Step 2: Frame Classification and Queue Assignment
RT consults the Forwarding Table (FT) to determine TSN destination
if F_{CAN} has a known TSN destination **then**
    Assign F_{CAN} to its respective queue in RAM
else
    Discard F_{CAN} (no valid TSN mapping)
end if

// Step 3: Periodic Encapsulation and TSN Frame Transmission
TSN Creation and Forwarding (TCF) executes at time intervals defined by Eq. (1) and (2)
Extract β CAN frames from the assigned queue
Form a TSN frame F_{TSN} with encapsulated CAN frames
Forward F_{TSN} to TSN Port (TP) of the TSN Network Interface (TSN NI)
TSN NI transmits F_{TSN} over the TSN link

D. End-to-End Delay Measurement

Measuring the end-to-end delays of CAN frames requires a precise technique due to the lack of synchronization between devices on the CAN network 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 frames into three distinct components: delays on the CAN/CAN FD network (D_{CAN}), delays induced by the gateway (D_{GW}), and delays on the TSN network (D_{TSN}). The total end-to-end delay (D_{E2E}) can be expressed as:

$$D_{\text{E2E}} = D_{\text{CAN}} + D_{\text{GW}} + D_{\text{TSN}} \tag{8}$$

Delays on the CAN/CAN FD network correspond to the worst-case response times of the frames and are calculated using the MPS-CAN Analyzer [26], [27]. 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 [28], which measures network delay by timestamping packets at the input and output of the network.

VI. 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 III. We encapsulate CAN frames using different values of β (1, 3, 6, 9, 12) and both FP and FIFO enqueueing of frames. For β =1, the period T_{TSN} is equal to 100 μs, which is significantly lower than the required encapsulation period (2.84 ms) calculated from the CAN frame frequencies in Table I based on Equation (3). This allows the system to approximate one-to-one mapping, minimizing queuing delays and ensuring timely forwarding. However, this choice utilizes more computational and network resources than strictly necessary, as encapsulating more frequently increases CPU usage and TSN frame generation. Additionally, we evaluate the impact of the TAS and WRR traffic schedulers in the TSN network. Furthermore, to validate the accuracy of the experimental results, we compare the measured forwarding delays against the theoretical analysis presented in [8]. To ensure the validity of the experiments, each experiment is run for 4 hyperperiods of the CAN frames being forwarded to the gateway.

A. Evaluation Scenarios

We consider two scenarios in the evaluation.

1) First Scenario: In this scenario, we evaluate 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 I. We conduct the evaluation for both classical CAN and CAN FD. Classical CAN operates at 500 Kbit/s. For CAN FD, the frame 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.

2) Second Scenario: In this scenario, we evaluate the effects of the TAS and WRR traffic schedulers on the delays of encapsulated CAN frames in Table I with different gateway forwarding techniques. In this scenario, the cameras stream data at 18 Mbit/s to the monitoring node. Additionally, the

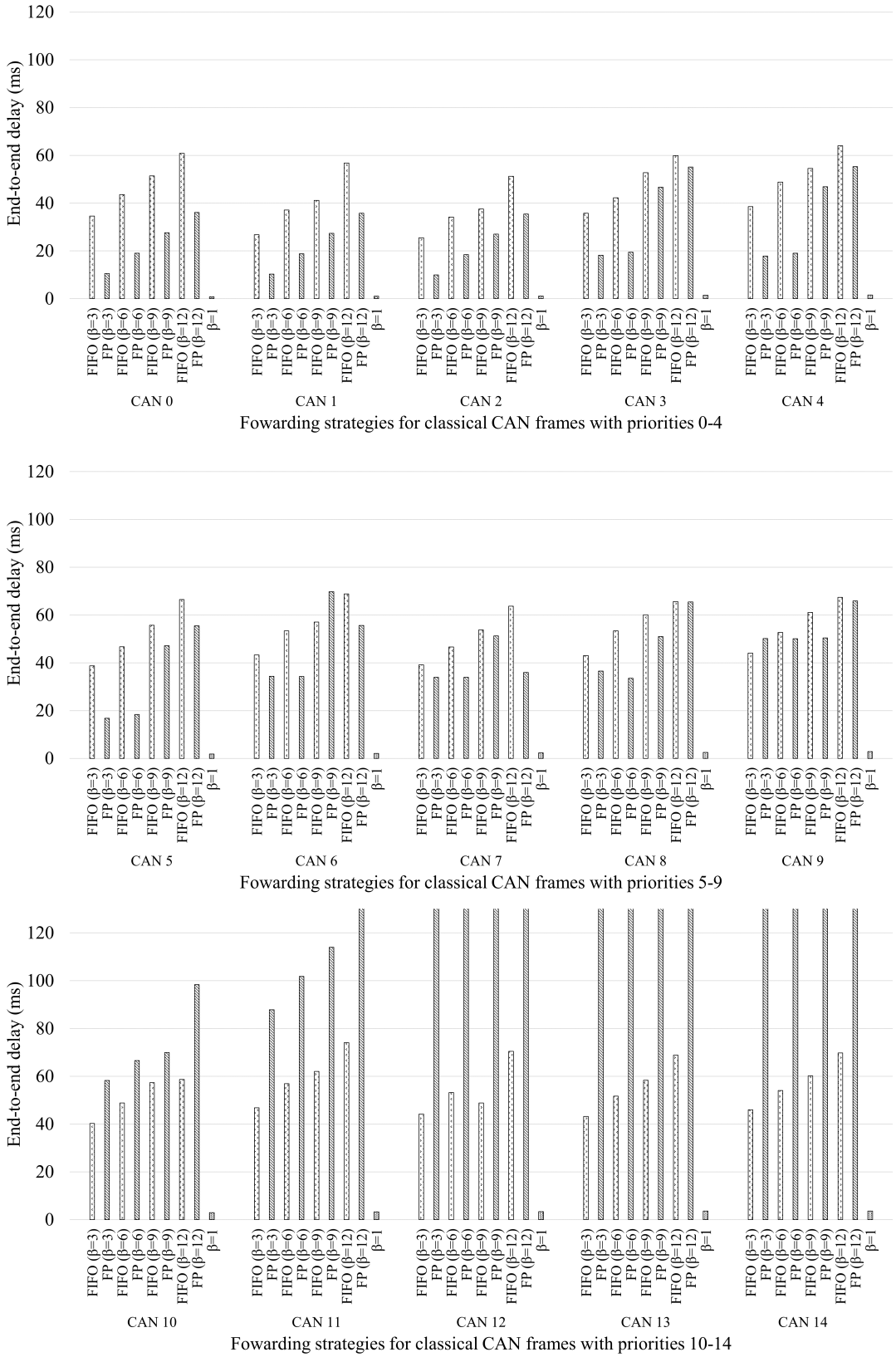

Fig. 4: Evaluation results for various gateway forwarding techniques when using classical CAN.

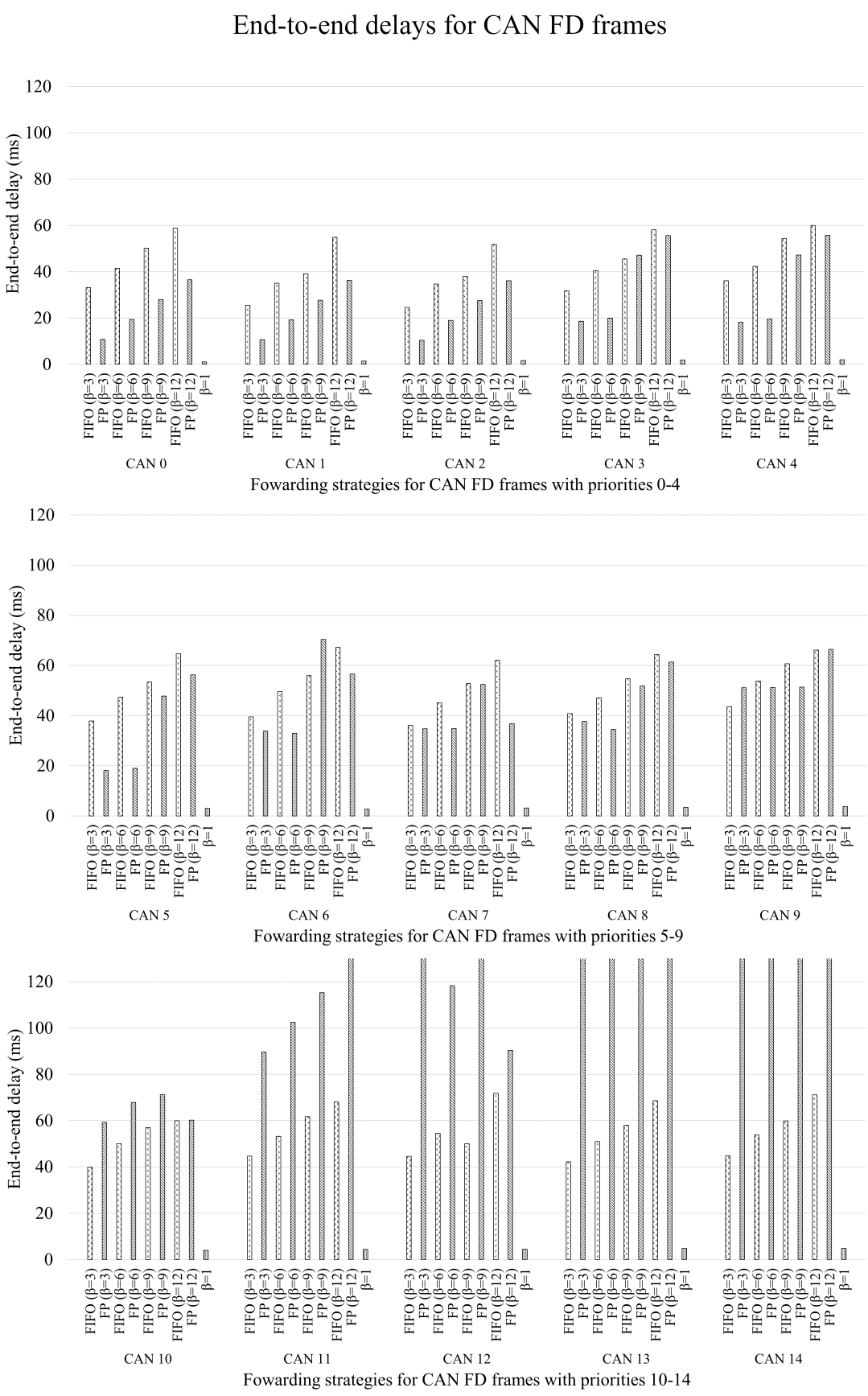

Fig. 5: Evaluation results for various gateway forwarding techniques when using CAN FD.

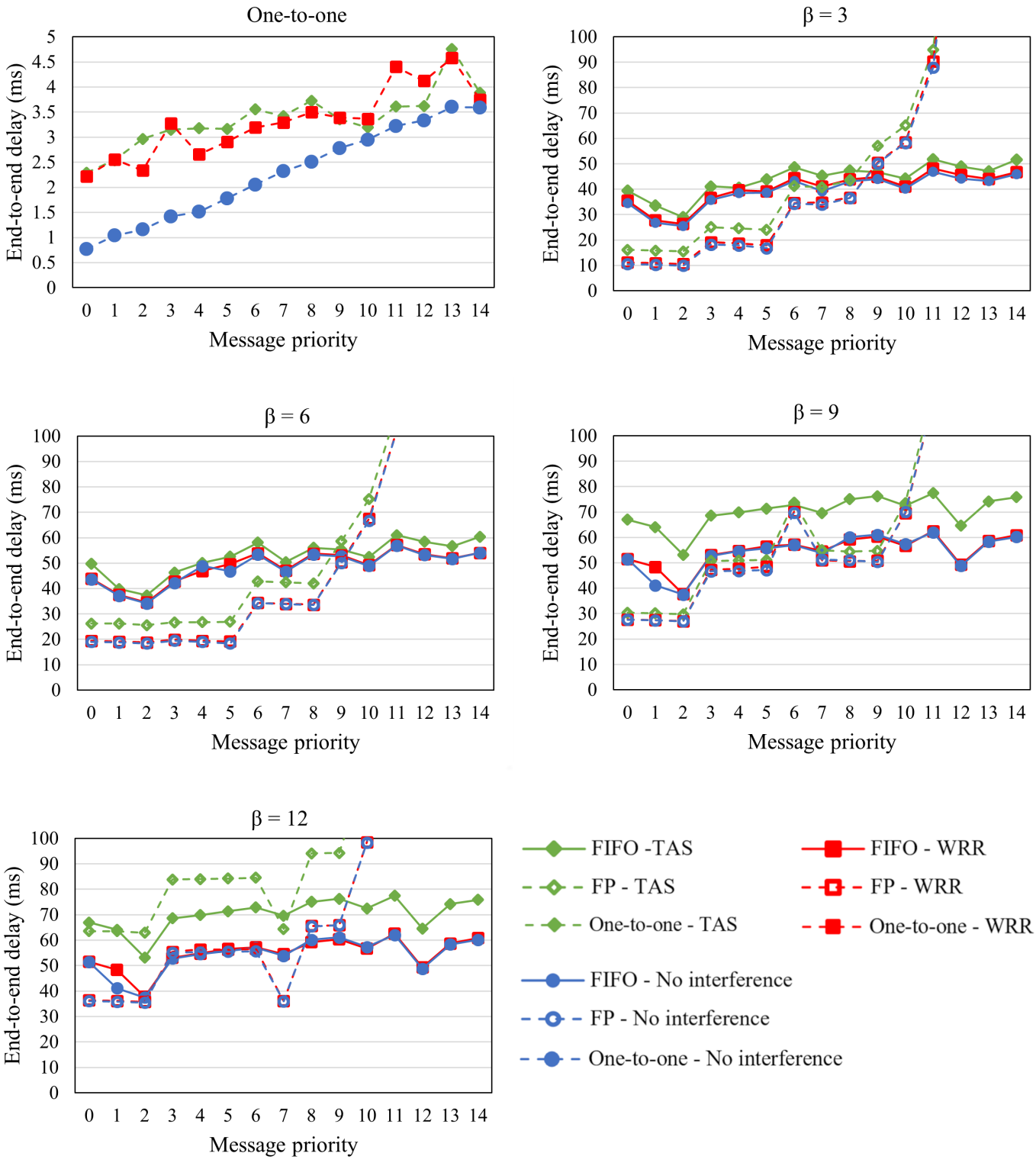

Fig. 6: Comparison of end-to-end delays of encapsulated CAN frames for different TSN traffic schedulers

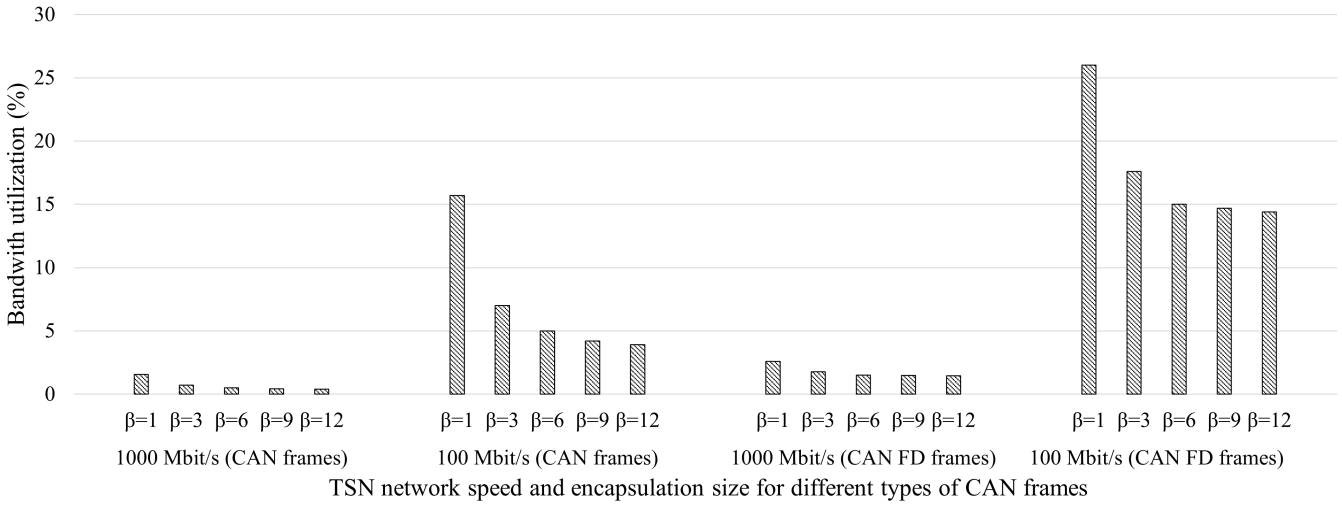

Fig. 7: Evaluation of TSN network bandwidth utilization of TSN frames transmitted from the gateway

traffic generators also stream data at 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. Since the encapsulated CAN frames are assigned the highest priority in the TSN network, the GCL cycle time of the TAS in our experiments is configured to match the transmission period of TSN frames originating from the gateway. The weights for the WRR are set as described in Section IV 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.

B. Evaluation Results: Gateway Forwarding Techniques

The experimental evaluation for the first scenario for classical CAN is depicted in Figure 4 while for CAN FD, it is depicted in Figure 5. The graphs show that the encapsulation size β significantly impacts the delays experienced by the CAN frames. When β is set to 1, each TSN frame contains exactly one CAN frame, minimizing the transmission delay experienced by the CAN frame. However, due to CPU constraints, the gateway processor may not always encapsulate each arriving CAN frame immediately upon arrival. As a result, while the system is configured for one-to-one mapping, occasional buffering may still occur before encapsulation, leading to slight deviations from an ideal one-to-one mapping strategy. As we increase β the delays experienced by frames also increase significantly, especially when β = 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.

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 β = 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 7 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 frame. When reducing the TSN link speed to 100 Mbit/s, the difference becomes more noticeable. With β = 1 and classical CAN, bandwidth usage is 15 %, which can be reduced to 5 % with β = 12. For CAN FD at 100 Mbit/s, bandwidth usage is 25 % with β = 1 and 15 % with β = 12. The larger bandwidth utilization with CAN FD is due to the larger frame sizes, making the TSN frames larger and increasing transmission times.

Even though using β = 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 1000 Mbit/s TSN network. However, increased bandwidth usage may lead to congestion on the TSN network, potentially impacting other TSN traffic. Unlike techniques that aggregate multiple CAN frames into a single TSN frame, one-to-one forwarding generates a higher number of TSN frames, which could introduce congestion and increase delays for other traffic classes in the TSN network.In our setup, the TSN network was not fully saturated, minimizing this effect, but in lower-speed (100 Mbit/s or 10 Mbit/s) networks, increased TSN frame generation could impact delays for existing TSN

traffic. For slower network speeds, encapsulating multiple CAN frames results in relatively lower delays. 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.

C. Evaluation Results: TAS and WRR Traffic Shapers

In the second evaluation scenario, we compared the performance of TAS and WRR traffic shapers, as well as the no-interference case, for encapsulated CAN frames in the TSN network. The no-interference case represents a scenario where only encapsulated CAN frames are transmitted, serving as a baseline for comparison against TAS and WRR traffic shaping mechanisms. The results are presented in Figure 6, which illustrates the end-to-end delays of CAN frames under different traffic shaping techniques. Figure 6 shows that TAS results in significantly higher delay compared to WRR due to the lack of synchronization between the CAN and TSN networks. The delay can increase by up to one full GCL cycle, as frames may arrive at any arbitrary point before the next GCL cycle begins. Conversely, a delay decrease is observed when a frame arrives just before encapsulation, leading to immediate transmission. This pattern of fluctuations is directly tied to the periodic nature of TAS gating cycles.

WRR exhibits lower delays because frames do not have to wait for a complete cycle before transmission. Instead, WRR ensures that a portion of the bandwidth is always allocated to encapsulated CAN frames, allowing them to be transmitted without waiting for a pre-defined time slot. The exception to this trend is the one-to-one forwarding technique, where TSN frames are transmitted every 100 μs, reducing the impact of the gating delay in TAS. In this case, TAS and WRR exhibit comparable performance since the short encapsulation period ensures that CAN frames are forwarded with minimal waiting time.

These results highlight the impact of synchronization on traffic shaping performance. Although TAS performed worse in this specific unsynchronized scenario, it is important to emphasize that TAS is designed for fully synchronized networks with offline scheduling. In such cases, TAS is expected to outperform WRR by providing guaranteed transmission slots for critical traffic and reducing jitter. However, in networks where synchronization cannot be guaranteed, WRR offers a more adaptable and predictable scheduling mechanism, making it preferable for handling encapsulated CAN frames under dynamic conditions.

D. Evaluation Results: Comparative analysis

The measured forwarding delays in this study align closely with the theoretical predictions presented in [8], which analyze the impact of different forwarding techniques in a CAN-to-TSN gateway. The theoretical model in [8] predicts that one-to-one forwarding results in the lowest forwarding delay, as each CAN frame is immediately encapsulated and transmitted without waiting for additional frames to be aggregated. Our experimental results confirm this behavior, with one-to-one forwarding achieving the lowest observed worst-case forwarding delay of 3 ms, closely matching the expected theoretical upper-bound estimate of 5 ms from [8]. Similarly, the FIFO and fixed-priority (FP) queuing techniques exhibit higher forwarding delays due to frames waiting in the queue before encapsulation. In our experiments, FIFO forwarding with $\beta = 10$ exhibited a worst-case forwarding delay of 54 ms, while the theoretical analysis in [8] estimated a delay of 63-65 ms under similar conditions. Additionally, for FP forwarding with $\beta = 10$, the worst-case measured forwarding delay of the highest-priority frame was 25 ms, compared to the theoretical estimate of 30 ms. The observed discrepancies between measured and theoretical values can be attributed to the pessimistic nature of the analysis, which provides worst-case delay estimates rather than exact predictions. Despite these minor deviations, the experimental findings strongly support the validity of the theoretical predictions in [8], confirming the relative behavior of different forwarding techniques and their expected performance trends in a CAN-to-TSN gateway.

VII. Conclusion

In this paper, we implemented and experimentally evaluated a CAN-to-TSN gateway, analyzing the impact of various gateway forwarding techniques. Moreover, we investigated the impact of the Time-Aware Shaper (TAS) and the Weighted Round Robin (WRR) traffic schedulers on the encapsulated CAN frames in an unsynchronized TSN network. The evaluated techniques included First-In-First-Out (FIFO), Fixed-Priority (FP), and the 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 end-to-end delay, the one-to-one forwarding technique is preferred, as it simplifies the gateway design and provides lower delays. Additionally, a comparative analysis between the measured forwarding delays and the existing theoretical analysis confirmed that the analytical model provides accurate upper-bound estimates. Furthermore, 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.

Acknowledgment

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

References

[1] A. Berisa, S. Mubeen, M. Daneshtalab, M. Ashjaei, M. Sjödin, B. Kraljusic, and N. Zahirovic, "Bridging the gap: An interface architecture for integrating CAN and TSN networks," Mälardalen Real-Time Research Centre, Mälardalen University, Tech. Rep., 2024.

[2] L. Lo Bello, R. Mariani, S. Mubeen, and S. Saponara, "Recent advances and trends in on-board embedded and networked automotive systems," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 2, 2019.

[3] International Standards Organization (ISO), "ISO 11898-1: Road vehicles—interchange of digital information—controller area network (CAN) for high-speed communication," 1993.

[4] "CAN with Flexible Data-Rate Specification," Robert Bosch GmbH, Stuttgart, Tech. Rep., 2012.

[5] H.-T. Lim, L. Völker, and D. Herrscher, "Challenges in a future IP/Ethernet-based in-car network for real-time applications," in *Design Automation Conference (DAC)*, 2011, pp. 7–12.

[6] M. Ashjaei, L. Lo Bello, M. Daneshtalab, G. Patti, S. Saponara, and S. Mubeen, "Time-sensitive networking in automotive embedded systems: State of the art and research opportunities," *Journal of Systems Architecture (JSA)*, vol. 117, no. C, 2021.

[7] H. Zinner, J. Noebauer, T. Gallner, J. Seitz, and T. Waas, "Application and realization of gateways between conventional automotive and IP/Ethernet-based networks," in *Design Automation Conference (DAC)*, 2011, pp. 1–6.

[8] A. Berisa, M. Ashjaei, M. Daneshtalab, M. Sjödin, and S. Mubeen, "Investigating and analyzing CAN-to-TSN gateway forwarding techniques," in *IEEE International Symposium on Real-Time Distributed Computing (ISORC)*, 2023, pp. 136–145.

[9] J.-L. Scharbarg, M. Boyer, and C. Fraboul, "CAN-Ethernet architectures for real-time applications," in *IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 2, 2005, pp. 8–pp.

[10] A. Kern, D. Reinhard, T. Streichert, and J. Teich, "Gateway strategies for embedding of automotive CAN-frames into ethernet-packets and vice versa," in *International Conference on Architecture of Computing Systems (ARCS)*. Springer, 2011, pp. 259–270.

[11] C. Herber, A. Richter, T. Wild, and A. Herkersdorf, "Real-time capable CAN to AVB ethernet gateway using frame aggregation and scheduling," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2015, pp. 61–66.

[12] D. Thiele, J. Schlatow, P. Axer, and R. Ernst, "Formal timing analysis of CAN-to-Ethernet gateway strategies in automotive networks," *Real-time systems*, vol. 52, no. 1, pp. 88–112, 2016.

[13] M. Barzegaran, N. Reusch, L. Zhao, S. S. Craciunas, and P. Pop, "Real-time traffic guarantees in heterogeneous time-sensitive networks," in *in International Conference on Real-Time Networks and Systems (RTNS)*, 2022, pp. 46–57.

[14] A. Morato, E. Ferrari, S. Vitturi, F. Tramarin, C. Zunino, and M. Cheminod, "A TSN-based approach to combine real-time CAN network with in-vehicle ethernet," in *MetroAutomotive*. IEEE, 2024.

[15] IEEE, "Time-Sensitive Networking (TSN) Task Group," 2016, Available: http://www.ieee802.org/1/pages/tsn.html. Accessed: 2024-05-13.

[16] A. A. Nacer, K. Jaffres-Runser, J.-L. Scharbarg, and C. Fraboul, "Strategies for the interconnection of CAN buses through an ethernet switch," in *IEEE International Symposium on Industrial Embedded Systems (SIES)*, 2013.

[17] G. Xie, Y. Zhang, N. Chen, and 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.

[18] W. Wu, H. Huang, W. Li, R. Liu, Y. Xie, and S. Long, "Real-time analysis and message priority assignment for TSN-CAN gateway," *IEEE Transactions on Intelligent Transportation Systems*, 2024.

[19] W. Yan, D. Wei, B. Fu, R. Li, and G. Xie, "A mixed-criticality traffic scheduler with mitigating congestion for CAN-to-TSN gateway," *ACM Transactions on Design Automation of Electronic Systems*, 2024.

[20] J. Walrand, "A concise tutorial on traffic shaping and scheduling in time-sensitive networks," *IEEE Communications Surveys & Tutorials*, 2023.

[21] Microchip, "PIC32CM JH Family of Microcontrollers," 2024, Available: https://www.microchip.com/en-us/products/microcontrollers-and-microprocessors/32-bit-mcus/pic32-32-bit-mcus/pic32cm-jh. Accessed: 2024-05-13.

[22] FreeRTOS, "FreeRTOS: Real-Time Operating System for Microcontrollers," Available: https://www.freertos.org. Accessed: 2024-05-13.

[23] Relyum, "RELY-TRAF-GEN: Time-Sensitive Traffic Generator," 2023, Available: https://soc-e.com/rely-traf-gen/. Accessed: 2024-05-13.

[24] S. M. Tabatabaee, J.-Y. Le Boudec, and M. Boyer, "Interleaved weighted round-robin: A network calculus analysis," *IEICE Transactions on Communications*, 2021.

[25] Renesas, "RZ/N2L RSK: Renesas Starter Kit for RZ/N2L," 2024, Available: https://www.renesas.com/us/en/products/microcontrollers-microprocessors/rz-mpus/rzn2l-rsk-renesas-starter-kit-rzn2l. Accessed: 2024-05-13.

[26] S. Mubeen, J. Mäki-Turja, and M. Sjödin, "MPS-CAN analyzer: Integrated implementation of response-time analyses for controller area network," *Journal of Systems architecture*, vol. 60, no. 10, 2014.

[27] S. Mubeen, J. Mäki-Turja, and M. Sjödin, "Integrating mixed transmission and practical limitations with the worst-case response-time analysis for controller area network," *Journal of Systems and Software*, vol. 99, 2015.

[28] Relyum, "RELY-TSN-LAB: Time-Sensitive Networking Testing Tool," 2023, Available: https://www.relyum.com/web/rely-tsn-lab/. Accessed: 2024-05-13.