INSIM: A Modular Simulation Platform for TSN-based In-Vehicle Networks

Mohammadparsa Karimi, Majid Nabi, Andrew Nelson, Kees Goossens, and Twan Basten
Electronic Systems (ES) group, Department of Electrical Engineering, Eindhoven University of Technology
The Netherlands
{m.karimi, m.nabi, a.t.nelson, k.g.w.goossens, and a.a.basten}@tue.nl

***Abstract*—In-vehicle networks (IVNs) are rapidly evolving to support increasingly complex automotive applications, demanding higher bandwidth and deterministic timing bounds. Time-Sensitive Networking (TSN) has emerged as a promising Ethernet-based technology that addresses these stringent requirements. However, evaluating TSN-based IVN strategies remains a challenge due to the lack of standardized benchmarks and simulation tools. This paper introduces INSIM, a modular simulation platform specifically designed for TSN-based IVNs, providing an intuitive graphical interface, an extensible plug-in architecture, and integrated benchmarking features. INSIM integrates analytical performance models and discrete-event simulations (as plug-ins), enhancing the workflow for engineers by refining topology design, adjusting parameters, conducting simulations, and assessing performance, while providing researchers with a flexible platform to plug in, analyze, and compare custom network resource managers or analytical performance models.**

***Index Terms*—In-Vehicle Networking, TSN, Simulation Platform, INSIM**

I. Introduction

The automotive industry is transitioning to software-defined vehicles to facilitate advanced driver assistance systems and autonomous capabilities [1]. Although conventional in-vehicle networks (CAN [2] and FlexRay [3]) have historically offered reliable real-time performance, their bandwidth limitations are now apparent due to the increased amount of data introduced by modern applications [4]. Consequently, it is crucial to enhance communication bandwidth without compromising the deterministic timing necessary for safety-critical applications. Meeting these constraints requires networking technologies that can provide high bandwidth and stringent timing assurances [5].

Time-Sensitive Networking (TSN) [6] addresses these requirements by introducing a suite of standards that, when configured effectively, ensure bounded latency and reliable data delivery over Ethernet-based infrastructures. TSN includes features such as time synchronization (IEEE 802.1AS) [7], traffic scheduling (IEEE 802.1Qbv) [8], frame preemption (IEEE 802.1Qbu) [9] and per-stream filtering and policing (IEEE 802.1Qci) [10]. Together, these ensure that data communication complies with strict timing requirements while maintaining optimal network performance. Although TSN has significant potential for automotive applications, determining which features to enable and how to configure them for optimal bandwidth utilization, bounded low latency, and reduced jitter remains a dynamic research field [11]. Developing robust scheduling and configuration strategies is crucial for maximizing TSN's potential and ensuring deterministic results within the context of complex vehicle requirements [6].

Despite considerable research focused on TSN configuration strategies, scheduling algorithms (as network resource manager), and performance modeling, the practical validation of these approaches remains challenging [14]. One of the primary barriers is the absence of stream and topology benchmarks, which makes objective and repeatable evaluations difficult. Moreover, existing simulation environments, such as OMNeT++ [12], typically rely on script-based configurations, which pose usability and scalability issues for automotive researchers and network engineers. These complexities limit the ability to comprehensively analyze network performance under diverse and realistic conditions.

To address these limitations, we introduce the **In-Vehicle Network Simulation (INSIM)** platform, a modular framework designed for TSN-based IVNs. INSIM is built on a flexible plug-in-based architecture enabling straightforward integration of simulation tools, analytical models, and resource managers. The platform includes a graphical user interface (GUI) that facilitates the configuration of IVN topologies and traffic streams and also offers comprehensive visualizations of network configurations and performance results. In addition, INSIM provides benchmarking modules, comprising predefined traffic streams and network topologies. INSIM integrates OMNeT++ as a simulation tool in the backend via its plug-in architecture. The **contributions** of this work are as follows:

- **INSIM Framework:** INSIM introduces a framework for IVNs that integrates a user-friendly GUI and both simulation and analytical analysis components. Using an extensible plug-in mechanism, INSIM enables straightforward incorporation of custom network schedulers and resource managers, analytical models, and simulation tools, offering flexibility and ease of adaptation for diverse automotive IVN needs.
- **Benchmark Suite:** We introduce a collection of traffic streams and representative network topologies for various automotive applications. Although not an industry standard, these benchmarks offer a convenient starting point for evaluating new approaches without building scenarios from scratch.

- **Open-Source Release:** All components, including GUI, benchmarks, and plug-in modules, are released under an open-source license. The source code is available as open source via the TU/e ES GitHub repository (https://github.com/TUE-EE-ES) and the repository's README file includes a direct link to the INSIM web application for immediate access.

INSIM offers a unified approach to investigate TSN-based IVNs. For engineers, it optimizes the process from topology design and parameter configuration to simulation and performance evaluation, and researchers can plug in and validate their own schedulers and network resource management techniques or performance models, using INSIM's user-friendly environment to evaluate innovative TSN solutions in a realistic automotive context. The remainder of this paper is organized as follows. Section II reviews the existing tools for TSN simulation and highlights the gaps in their capabilities. Section III introduces a simplified user work flow for INSIM. Section IV presents the architecture of INSIM and describes key implementation details for each module. Section V demonstrates how the platform can be deployed in a representative use case. Finally, Section VI concludes with a summary of the features of our tool and outlines potential directions for future research.

II. Related Work

Many existing tools for evaluating in-vehicle TSN rely on hardware testbeds or simulation tools, but often lack analytical methods for evaluations in the early stages of design. Hardware-based approaches are for instance reported in [13], [14]. Both the TSN testbed of [13] and EnGINE [14] harness Commercial Off The-Shelf (COTS) Ethernet equipment and open-source orchestration software for TSN measurements. These setups usually require extensive (re-)configuration for each scheduling variant and offer limited scalability when dealing with larger topologies or higher traffic loads. They also do not include standard benchmarks, making it difficult to systematically compare performance in various scenarios. Simulation frameworks (e.g., OMNeT++ with TSN plug-ins) [15] can handle larger topologies and diverse traffic configurations, though they generally need significant setup and produce a longer runtime when investigating multiple scheduling or topology scenarios. Moreover, they fail to offer an extensible plug-in framework that allows individuals to integrate their own network resource management algorithms or performance models, enabling analysis and comparison. Meanwhile, comprehensive automotive platforms such as EDGAR [16] are oriented toward complete self-driving pipelines, treating networking as a secondary concern and forgoing specialized TSN benchmarking or early-stage latency modeling.

In contrast to all of the above, INSIM merges simulation-based and analytical features for in-vehicle TSN, providing pre-defined benchmarks and benchmark generation functionality. This enables rapid performance estimates and detailed simulations. A user-friendly GUI and a modular plug-in framework further streamline the evaluation of novel algorithms and models.

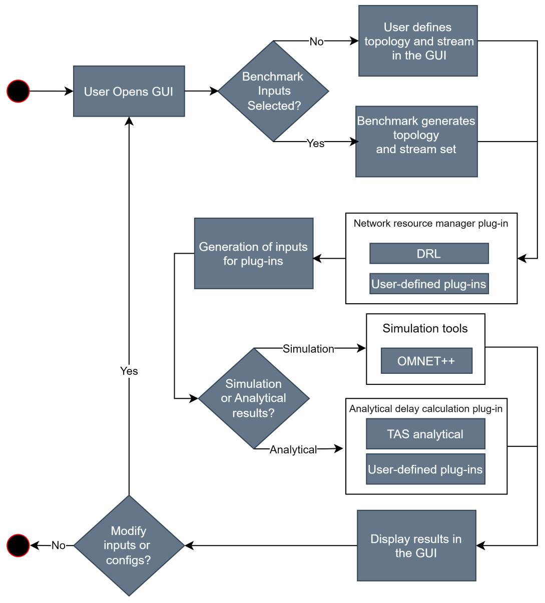
User Opens GUI
Benchmark Inputs Selected?
No
Yes
User defines topology and stream in the GUI
Benchmark generates topology and stream set
Network resource manager plug-in
DRL
User-defined plug-ins
Generation of inputs for plug-ins
Simulation tools
OMNET++
Simulation
Simulation or Analytical results?
Analytical
Analytical delay calculation plug-in
TAS analytical
User-defined plug-ins
Yes
Modify inputs or configs?
No
Display results in the GUI

Fig. 1. Typical INSIM user flow

III. INSIM User Flow

Figure 1 illustrates the typical user flow for INSIM, focusing on the most straightforward usage scenarios. Some advanced features, such as creating custom plug-ins, manually configuring TSN parameters, or editing the benchmark database, are not included and are discussed in Section IV. The following steps outline how a typical user configures, runs, and analyzes an INSIM evaluation under default conditions.

1) **GUI Launch:** The user starts by opening the INSIM graphical interface. This interface acts as the main control hub for tasks ranging from topology and traffic stream configuration to simulation and result visualization.
2) **Benchmark Selection or Custom Inputs:** When entering the GUI, the user can either select benchmark inputs (reference topologies and stream sets) or provide custom topologies and streams through the interface. In the former case, INSIM automatically loads predefined benchmark data; in the latter case, the user can specify node counts, link capacities, and traffic characteristics.
3) **TSN Configuration Generation:** Once network topology and traffic input are finalized, the INSIM network resource manager automatically generates a TSN configuration, including scheduling modes. This network resource manager is also implemented as a plug-in. INSIM provides two default schedulers, but users can easily integrate a custom network resource manager via the same plug-in mechanism.
4) **OMNeT++ and JSON Model Creation:** Next, INSIM produces configuration files that encode both the network topology and the TSN setup. These artifacts are passed either to OMNeT++ (if a full simulation run is desired) or to the analytical delay calculation module.

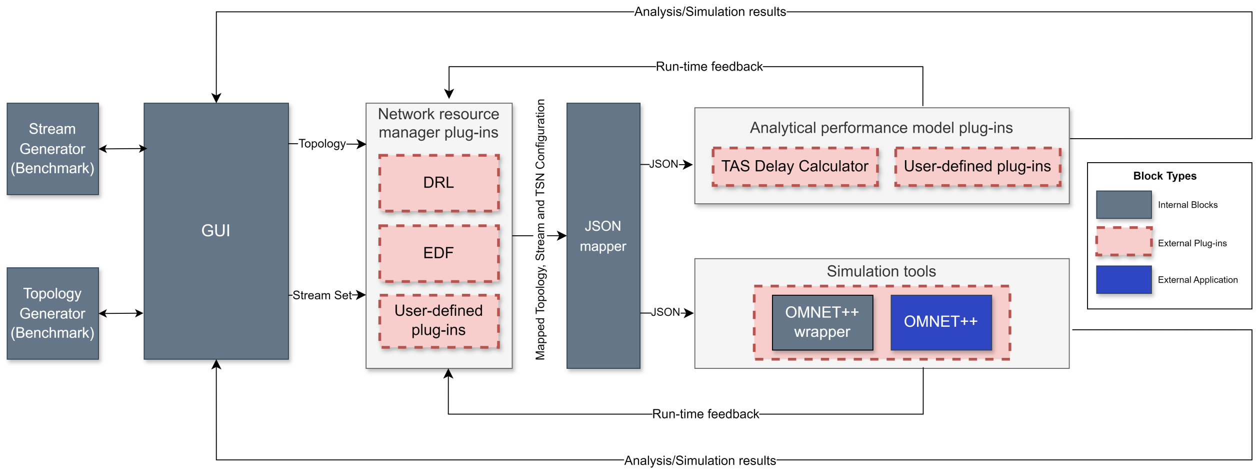

Fig. 2. The block diagram of INSIM

5) **Simulation or Analytical Evaluation:** Depending on the user's choice, INSIM invokes OMNeT++ to run a simulation-based experiment or a plugin-based analytical model to provide a faster approximation of delay. INSIM supports an extensible plugin architecture, allowing users to integrate custom analytical or simulation models alongside the built-in options.
6) **Results Display:** After completion, INSIM presents the resulting performance data in the GUI. If further refinements or parameter changes are required, the user may revisit the configuration, modify inputs, and re-run the analysis. Otherwise, if no additional adjustments are needed, the user can exit the tool.

As mentioned, the steps above sketch a straightforward user flow. INSIM also provides more advanced functionality, including a plug-in mechanism for custom network resource managers, analytical modeling, and detailed manual TSN configuration.

IV. INSIM Architecture

Figure 2 shows the block diagram of the INSIM architecture, highlighting both the core components and the external plug-in or external application blocks. The following subsections describe each block in detail.

A. *Topology Generator (Benchmark)*

The Topology Generator is responsible for producing IVN topologies. Benchmark topologies, derived from known automotive designs, are preloaded, but users can also modify them. Once a topology is selected or created, it is forwarded to the GUI for further adjustment and enhancement. Currently, we employ a static database for this block, meaning that each application has a predefined topology that can be selected and applied through the GUI. The overarching concept for the topology is derived from the zonal architecture for the automotive domain [17]. This is only one possible approach to modeling in-vehicle topologies. Since software-defined vehicles remain an emerging and developing concept [18], we do not claim that these benchmarks reflect future industry standards. INSIM is sufficiently flexible though to include newly developed benchmarks.

B. *Stream Generator (Benchmark)*

Similar to the topology generator, the stream generator creates traffic streams that represent in-vehicle applications. The Stream Generator draws on a static database of traffic streams. These streams may range from low-rate sensor data to high-bandwidth camera feeds, complete with timing deadlines or priority requirements. The benchmarks for typical automotive streams can be loaded by default, or users can specify custom traffic patterns. As with the topologies, these traffic definitions represent just one possible approach. They serve as a baseline for initial experimentation.

C. *GUI*

The GUI is the main entry point for user interaction. Through this interface, users can select or modify topologies, define traffic streams, configure key TSN parameters, and visualize the results. Currently, it supports only TAS scheduling configurations, though additional TSN standards may be integrated in the future. The GUI also collects user preferences, such as whether to run an analytical latency estimate or a full OMNeT++ simulation. Once the inputs are finalized, the GUI relays them to the network resource manager along with any selected benchmark data. Additionally, the GUI supports a plug-in interface that enables users to integrate custom algorithms without modifying the core INSIM code.

D. *Network Resource Manager (Plug-Ins)*

At the core of the system, the network resource manager creates schedules for the provided stream set and network topology, checks the feasibility of the resources, and prepares a final TSN configuration for deployment. By default, INSIM provides two scheduling options (exclusively for TAS in the current version): Earliest Deadline First (EDF) scheduling [19] and a Deep Reinforcement Learning (DRL) approach [20]. In addition, this module offers a plug-in mechanism for incorporating custom scheduling policies. Researchers can

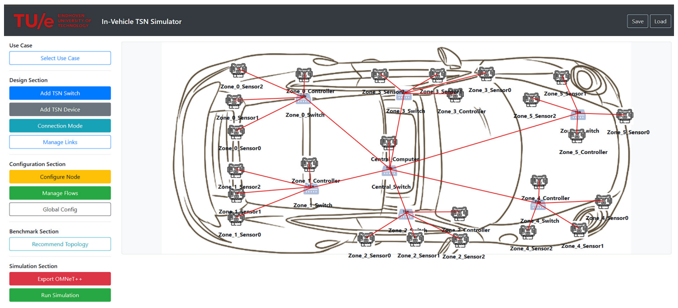

Fig. 3. LKA use-case topology in the INSIM GUI

simply implement the required interface functions and then register their plug-in with the network resource manager. Once loaded, the new algorithm appears in INSIM as an additional resource management option.

E. JSON Mapper

After the network resource manager finalizes the TSN configuration and traffic assignments, the JSON Mapper translates these specifications into a structured format for the analytical delay calculator and the OMNeT++ wrapper converts this JSON output to an OMNeT++ compatible format. This JSON-encoded representation captures critical details, such as node connections, traffic class definitions, scheduling windows, and link capacities, ensuring that each external module can execute the same network model consistently.

F. Analytical Performance model (Plug-Ins)

The Analytical Performance model provides a fast approximate delay estimate for configured in-vehicle traffic streams. Currently, INSIM includes a built-in calculation model for TAS [21], which allows a quick assessment of end-to-end delays without running a full simulation. Beyond this default, users can extend or replace the analytical model through an open plug-in interface by implementing the required APIs.

G. OMNeT++ Wrapper

If the user prefers a more detailed simulation, the OMNeT++ wrapper also parses the JSON model and then translates it into the NED files (topology descriptions), INI files (initialization parameters), and C++ code (simulation logic) required by OMNeT++. During the simulation run, it captures intermediate statistics (for example, queue utilization and packet latencies) and returns them to the network resource manager. If a dynamic resource manager plug-in is present, these run-time data can further refine decisions on the fly.

H. OMNeT++

Finally, OMNeT++ is the external discrete-event simulator where the configured TSN network is executed (automatically via INSIM). OMNeT++ logs and metrics are then redirected through the simulation wrapper, allowing INSIM to display simulation results such as latency distributions within the GUI.

V. Practical Use Case

In this section, we demonstrate how INSIM can be used in a practical scenario. We present a simplified Lane Keeping Assistant (LKA) to show INSIM's use. LKA helps a vehicle maintain its lane by continuously monitoring the lane boundaries and providing corrective signals and steering commands. For ease of illustration, we represent only a subset of the typical LKA data flows.

A. Topology

For our LKA use case, we consider a zonal architecture comprising six distinct zones. Each zone contains a set of sensors and actuators. To efficiently collect, process, and transmit data within each zone, a dedicated zone controller is used. Each zone also includes a TSN switch. These six TSN switches are arranged to connect to a central switch, forming a star topology. The central switch is directly linked to the central computer, which executes the primary LKA algorithms. Figure 3 presents the LKA topology in the INSIM GUI.

B. Stream Sets

For simplicity, we define only three deterministic traffic streams in our LKA use case, each with a specific period and packet size. The first is a camera stream that originates in Zone 0 and is directed to the central computer and that is assumed to contain periodic lane image data for lane detection. The second is an ultrasound sensor stream from Zone 2 to the central computer, which transmits crucial proximity measurements for hazard detection. Finally, a steering command stream travels from the central computer to the Zone 1 controller, delivering real-time steering adjustments derived from the central LKA application.

C. Scheduling

To facilitate a meaningful comparison between simulation and analytical modeling, we configure the TAS on both the zonal switches and the central switch. Specifically, each stream is assigned a different gating window within every scheduling period, ensuring that only the gate corresponding to a particular stream is open during its allocated transmission time. Consequently, if a packet arrives during a closed gate interval, it must wait until the next opening for that stream. This setup creates quantifiable delay scenarios, thereby allowing us to compare delay predictions obtained through fast analytical models against the results from OMNeT++ simulations.

D. Analysis results

Figures 4 and 5 show the delay distribution histograms obtained from the INSIM default analytical performance model and OMNeT++, respectively, for the simplified LKA traffic scenario. As the figures illustrate, the computed delays align closely between the two approaches, albeit with minor discrepancies, an expected outcome given the inherent differences between a discrete-event simulation and a calculation-based model. This example illustrates the use of INSIM for performance analysis. Researchers can seamlessly plug in their own

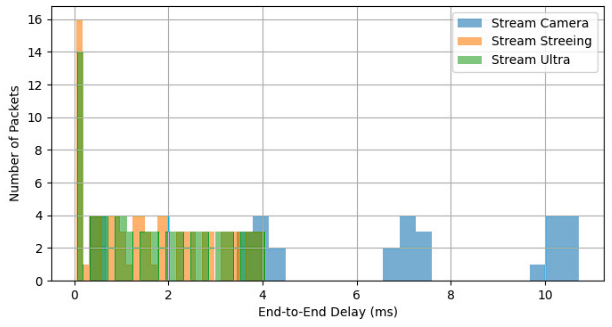

Fig. 4. Delay distribution histogram from INSIM's default analytical performance model

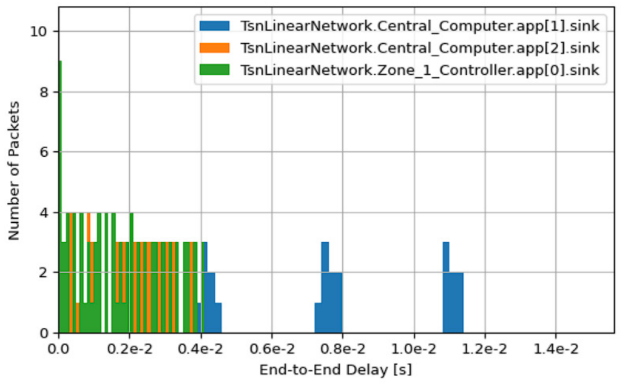

Fig. 5. Delay distribution from OMNeT++ simulation

analytical models, network simulators, or network resource managers and compare results in various network scenarios and use cases.

VI. Conclusion and Future work

This paper introduced **INSIM**, a modular framework for analyzing and simulating in-vehicle TSN networks. By offering an extensible plug-in mechanism, INSIM allows researchers to easily integrate their own resource management strategies or analytical performance models, and compare them against baseline implementations. INSIM enables engineers to optimize their design and validation processes. Additionally, INSIM's built-in stream and topology benchmarks provide a convenient starting point for examining various TSN design choices without the need to recreate scenarios from scratch.

Despite its utility, INSIM remains a prototype. Future work will include extending the framework beyond the current TAS-focused APIs and adding plug-ins to support additional TSN standards (such as the Credit-Based Shaper). In addition, we plan to extend existing benchmarks to reflect more vehicle-scale applications. Finally, continued extensions in supporting static and dynamic network resource managers will further reinforce the role of INSIM as a comprehensive platform to advance in-vehicle TSN research.

Acknowledgment

This work has received funding from the European Chips Joint Undertaking under Framework Partnership Agreement No 101139789 (HAL4SDV).

References

[1] M. T. SaiLakshmi, J. B. Vishnu, and S. Madhuri, "Software-defined vehicles: The future of the automobile industry," in *2025 COMSNETS*, pp. 192–197, IEEE, 2025.

[2] W. Voss, "A Comprehensible Guide to Controller Area Network", *Copperhill Media*, 2008.

[3] R. Makowitz and C. Temple, "FlexRay–a communication network for automotive control systems," in *2006 IEEE International Workshop on Factory Communication Systems*, pp. 207-212, IEEE, 2006.

[4] C. Zhang *et al.*, "Deterministic communications for in-vehicle network: Overview and challenges," in *ICAIIS*, 2021, pp. 207-212, ACM, 2021.

[5] Z. Zhou *et al.*, "Simulating TSN Traffic Scheduling and Shaping for Future Automotive Ethernet," *Journal of Communications and Networks*, vol. 23, no. 1, pp. 53–62, 2021.

[6] Y. Peng *et al.*, "A Survey on In-Vehicle Time-Sensitive Networking," *IEEE Internet of Things Journal*, vol. 10, no. 16, pp. 14375–14396, 2023.

[7] G. Garner, "Standard for Local and Metropolitan Area Networks—Timing and Synchronization for Time-Sensitive Applications," *IEEE Draft Standard 802.1AS*, 2016.

[8] "IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic," *IEEE Std 802.1Qbv-2015* (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd-2015, and IEEE Std 802.1Q-2014/Cor 1-2015), pp. 1–57, 2016.

[9] "IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 26: Frame Preemption," *IEEE Std 802.1Qbu-2016* (Amendment to IEEE Std 802.1Q-2014), pp. 1–52, 2016.

[10] "IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 28: Per-Stream Filtering and Policing," *IEEE Std 802.1Qci-2017* (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd-2015, IEEE Std 802.1Q-2014/Cor 1-2015, IEEE Std 802.1Qbv-2015, IEEE Std 802.1Qbu-2016, and IEEE Std 802.1Qbz-2016), pp. 1–65, 2017.

[11] C. Xue *et al.*, "Real-Time Scheduling for 802.1Qbv Time-Sensitive Networking (TSN): A Systematic Review and Experimental Study," in *RTAS*, IEEE, 2024, pp. 108–121.

[12] A. Varga, "OMNeT++," *Modeling and Tools for Network Simulation*, pp. 35–59, Springer Berlin Heidelberg, 2010.

[13] D. Andronovici *et al.*, "Cross-validating open source in-vehicle TSN simulation models with a COTS hardware testbed," in *IEEE VNC*, pp. 172–179, IEEE, 2024.

[14] F. Rezabek *et al.*, "Engine: Flexible research infrastructure for reliable and scalable time sensitive networks," *Journal of Network and Systems Management*, vol. 30, no. 4, pp. 74, 2022.

[15] M. Bosk *et al.*, "Simulation and practice: A hybrid experimentation platform for TSN," in *IFIP Networking Conf.*, pp. 1–9, IEEE, 2023.

[16] P. Karle *et al.*, "EDGAR: An autonomous driving research platform–from feature development to real-world application," *arXiv preprint arXiv:2309.15492*, 2023.

[17] T. Stüber *et al.*, "A survey of scheduling algorithms for the time-aware shaper in time-sensitive networking (TSN)," *IEEE Access*, vol. 11, pp. 61192–61233, 2023.

[18] U. Bordoloi *et al.*, "Autonomy-driven emerging directions in software-defined vehicles," in *DATE*, pp. 1–9, IEEE, 2023.

[19] L. Leonardi, L. Lo Bello, and G. Patti, "Combining Earliest Deadline First Scheduling with Scheduled Traffic Support in Automotive TSN-Based Networks," *Appl. Syst. Innov.*, vol. 5, no. 6, p. 125, 2022.

[20] M. Karimi *et al.*, "Deep-Reinforcement-Learning-based Scheduler for Time-Aware Shaper in In-Vehicle Networks," in *2025 VTC-Spring*, IEEE, 2025.

[21] P. Pop, K. Alexandris, and T. Wang, "Configuration of multi-shaper Time-Sensitive Networking for industrial applications," *IET Networks*, vol. 13, no. 5–6, pp. 434–454, 2024.