Demo: Interactive Off-the-Shelf In-Car TSN Testbed

Darinela Andronovici*, Damien Nicolas*, Ion Turcanu*, and Christoph Sommer†
*Luxembourg Institute of Science and Technology (LIST), Luxembourg
†TU Dresden, Faculty of Computer Science, Germany
{ darinela.andronovici, damien.nicolas, ion.turcanu }@list.lu
https://www.cms-labs.org/people/sommer

***Abstract*—Modern automotive networks based on Time-Sensitive Networking (TSN) are becoming increasingly complex. While hands-on experience is critical to understanding these concepts, the complexity and cost associated with TSN technologies often make practical training inaccessible. As an alternative, network simulation tools have been widely adopted, but they lack interactivity and immediate feedback. To bridge this gap, we propose an interactive and affordable TSN testbed built using off-the-shelf hardware. Our solution provides a user-friendly interface for configuring the testbed and experiencing real-time interactions, such as assessing the impact of background noise traffic on automotive LiDAR sensor data. We demonstrate the functionality of our testbed and provide open-source access to the source code, aiming to improve the quality of TSN training and live experimentation.**

I. Introduction

Hands-on experience is generally considered to be a crucial part of gaining a deep understanding of networking [1] – be it for students, researchers, or practitioners. However, as essential as it is, practical experience is often hard to come by. This is particularly true for technologies that are complex, expensive, or both, such as Time-Sensitive Networking (TSN) as deployed for in-car networking of modern and future vehicles [2], [3].

By necessity, many therefore turn to simulations of TSN networks as a substitute for hands-on experience. There is a wide range of excellent simulators or simulation frameworks available for this purpose, such as ns-3 [4] and OMNeT++ [5], the latter if paired with module libraries such as the INET Framework [6], NesTing [7], or CoRE4INET [8]. These simulators offer tremendous flexibility and are widely used in research and teaching. However, they are limited in their interactivity because they are designed to run simulations and only then present results – often after considerable computation time – rather than allowing users to interact with the system and get immediate feedback on their actions, thus reducing effectiveness [9].

Therefore, there is a real need for an interactive, affordable, real-world TSN testbed with a flat learning curve for training and live experimentation. In this demo paper, we describe how such a testbed can be built using Commercial Off-the-Shelf (COTS) hardware and demonstrate its functionality in terms of configuration capabilities and user interaction aiming to enhance the quality of experience. In particular, our solution provides an easy-to-use interface that allows users to interact with and configure an in-car TSN testbed and to experience the impact of background noise traffic on the data generated by an automotive LiDAR sensor. We provide open-source access to the source code of the proposed solution.[1]

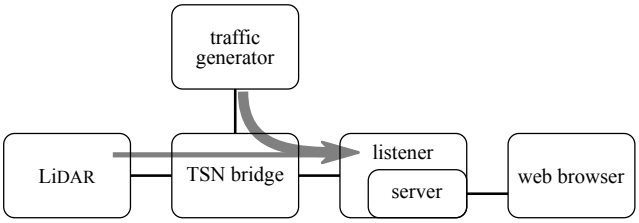

Figure 1. Testbed architecture.

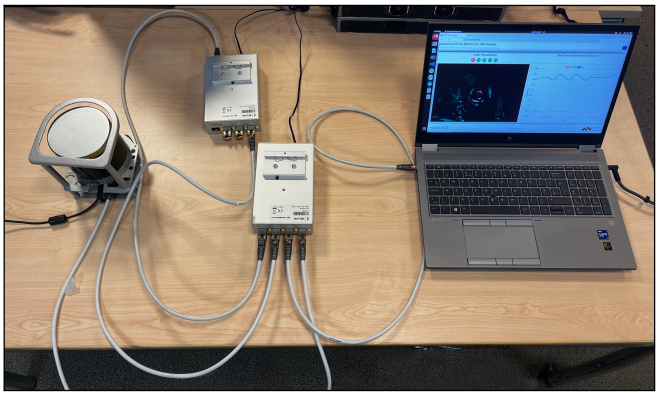

Figure 2. Photo of the demo.

II. System Design

Figure 1 presents the architecture of the proposed system, which has the following hardware components: a Robosense *Helios 16* high-precision 3D LiDAR, a TSN bridge (RELY-TSN-Bridge v20.1.11), a traffic generator (RELY-TRAF-GEN v20.1.0), and a listener, which is an off-the-shelf Linux machine. The hardware components are connected using 1 Gbit/s Cat5 Ethernet cables. A photo of the demo setup is shown in Figure 2.

The LiDAR sensor supports GPS, PTP, and gPTP as synchronization mechanisms. It transmits data over Ethernet using User Datagram Protocol (UDP) datagrams that use both the Main data Stream Output Protocol (MSOP) and the Device Information Output Protocol (DIFOP). The DIFOP packets

[1]Available at https://github.com/LIST-LUXEMBOURG/TSNavig8

IEEE Vehicular Networking Conference (VNC 2024)

contain various information about the sensor configuration and its current state, while the MSOP packets contain the laser scanning data, including distance, angle, and reflection intensity. The traffic generator is a device capable of sending Layer 2 traffic to the network with configurable parameters such as packet size, transmission rate, VLAN ID (VID), and Priority Code Point (PCP).

The software architecture of our proposed demo runs entirely on the listener and includes a server written in Python and a web client application developed using the *Vue.js* framework.

The server consists of three distinct processes: (i) a process dedicated to receiving data from the LiDAR sensor and transmitting it to the web client, (ii) a process dedicated to monitoring bandwidth, and (iii) a process dedicated to configuring the hardware components. In particular, the first process running on the server listens on UDP port 6699 where MSOP packets are sent from the LiDAR. This UDP listener processes the payload of the datagrams and converts the polar coordinates (angle and distance) to Cartesian coordinates. Once completed, the messages containing the $\langle x, y, z, distance \rangle$ tuple are sent to the web client application. The distance measured in centimeters is used to compute the color of each point. Points closer to the LiDAR appear red, while those further away appear blue. The second process is responsible for computing the real-time bandwidth utilization every second and sending this data to the web client application. Finally, the third process is used to perform hardware configuration over the Secure Shell (SSH) protocol based on input parameters provided by the user.

The web client application consists of several *Vue.js* components that are connected to the aforementioned server processes via WebSockets. Specifically, the *PlotLidarComponent* uses the *Three.js* library to render cloud points, allowing the user to visualize scanned data from the LiDAR sensor. Next, the *PlotBandwidthComponent* uses *Chart.js* to plot the bandwidth utilization measured in Mbit/s. In addition to the visualization components, the demo includes two configuration components, one for the traffic generator and one for the Time-Aware Shaper (TAS) mechanism in the TSN bridge.

III. Demonstration

Figure 3 illustrates the interface of our proposed solution. The web interface features two interactive graphs: one displaying LiDAR data with live rendering and one displaying the real-time visualization of bandwidth utilization.

The row of action buttons positioned above the LiDAR plot provides the following key functions: starting and stopping the LiDAR data capture, controlling traffic generation with Start and Stop commands, enabling and disabling the Time-Aware Shaper (TAS) mechanism, running a Negative Test which aims to highlight the limitations of standard Ethernet, and resetting the TAS configuration if necessary.

The button in the upper right corner opens a sidebar that allows the user to configure key parameters of the frames sent by the traffic generator: frame size, Priority Code Point (PCP), and bandwidth usage as a fraction of available bandwidth. It is also possible to freely define the TAS configuration by specifying the number of slots, their duration, and which priority queues are enabled.

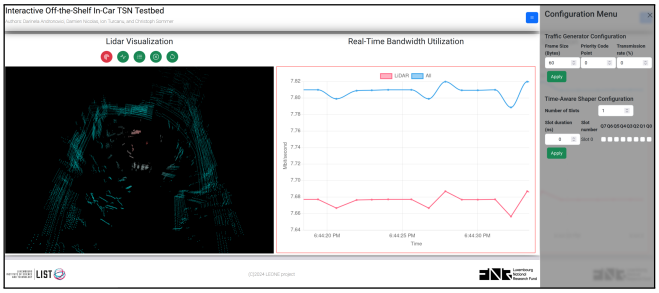

Figure 3. Screenshot of the user interface.

Taken together, this allows users to test different traffic patterns and TAS configurations and to see, live, their effect on a real sensor. This, in turn, allows users to directly experience – and thus get an intuitive understanding of – the interrelation of network configuration, network performance, and quality of sensor data.

Acknowledgments

This research was funded in whole, or in part, by the Luxembourg National Research Fund (FNR), grant reference DEFENCE22/IS/17800623/LEONE. For the purpose of open access, and in fulfilment of the obligations arising from the grant agreement, the authors have applied a Creative Commons Attribution 4.0 International (CC BY 4.0) license to any Author Accepted Manuscript version arising from this submission.

References

[1] M. Prvan and J. Ožegović, "Methods in Teaching Computer Networks: A Literature Review," *ACM Transactions on Computing Education*, vol. 20, no. 3, Jun. 2020.

[2] Y. Peng, B. Shi, T. Jiang, X. Tu, D. Xu, and K. Hua, "A Survey on In-Vehicle Time-Sensitive Networking," *IEEE Internet of Things Journal*, vol. 10, no. 16, pp. 14 375–14 396, Aug. 2023.

[3] I. Turcanu and C. Sommer, "Poster: Potentials of Mixing TSN Wired Networks and Best-Effort Wireless Networks for V2X," in *13th IEEE Vehicular Networking Conference (VNC 2021), Poster Session*, Virtual Conference: IEEE, Nov. 2021, pp. 135–136.

[4] G. F. Riley and T. R. Henderson, "The ns-3 Network Simulator," in *Modeling and Tools for Network Simulation*. Springer, 2010, pp. 15–34.

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

[6] L. Mészáros, A. Varga, and M. Kirsche, "INET Framework," in *Recent Advances in Network Simulation: The OMNeT++ Environment and its Ecosystem*. Springer, 2019, pp. 55–106.

[7] J. Falk et al., "NeSTiNg: Simulating IEEE Time-sensitive Networking (TSN) in OMNeT++," in *2019 International Conference on Networked Systems (NetSys)*, IEEE, Mar. 2019.

[8] T. Steinbach, H. D. Kenfack, F. Korf, and T. C. Schmidt, "An extension of the OMNeT++ INET framework for simulating real-time Ethernet with high accuracy," in *4th International ICST Conference on Simulation Tools and Techniques*, Barcelona, Spain: ICST, 2011, pp. 375–382.

[9] F. M. van der Kleij, R. C. W. Feskens, and T. J. H. M. Eggen, "Effects of Feedback in a Computer-Based Learning Environment on Students' Learning Outcomes: A Meta-Analysis," *AERA Review of Educational Research (RER)*, vol. 85, no. 4, pp. 475–511, Dec. 2015.