DEMO: TSN Reservation Framework for Legacy Apps – Signaling, Shaping, Monitoring, and Measuring

Alexej Grigorjew, Tobias Hoßfeld
University of Würzburg, Germany

David Hock, Fabian Lipp
Infosim GmbH & Co. KG, Germany

ABSTRACT

This demo presents a framework that enables predictable real-time guarantees for legacy TCP applications over Ethernet. It includes four components: (i) a lightweight signaling system that implements a simplified TSN Resource Allocation Protocol with minimal dependencies, (ii) a proxy that terminates TCP connections locally, initializes resource reservations, and enforces the traffic specification through shaping, (iii) a monitoring system that tracks overall network utilization and status, and (iv) an accurate live measurement and visualization of one-way delays for all reserved streams. The components are focused on simplicity to facilitate deployment and encourage innovations.

CCS CONCEPTS

• **Networks** → *Network measurement*; *Network monitoring*;

KEYWORDS

Time-Sensitive Networking, Resource-Allocation Protocol, TSN Monitoring, TSN Testbed, TSN Controller

ACM Reference Format:
Alexej Grigorjew, Tobias Hoßfeld and David Hock, Fabian Lipp. 2025. DEMO: TSN Reservation Framework for Legacy Apps – Signaling, Shaping, Monitoring, and Measuring. In *ACM SIGCOMM 2025 Posters and Demos (SIGCOMM Posters and Demos '25), September 8–11, 2025, Coimbra, Portugal.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3744969.3748423

1 INTRODUCTION

Time-Sensitive Networking (TSN) [7] enables predictable real-time communication and is increasingly adopted in domains like industrial networks [4] and in-vehicle communication systems [6], replacing legacy field buses with Ethernet.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SIGCOMM Posters and Demos '25, September 8–11, 2025, Coimbra, Portugal
© 2025 Association for Computing Machinery.
ACM ISBN 979-8-4007-2026-0/25/09.
https://doi.org/10.1145/3744969.3748423

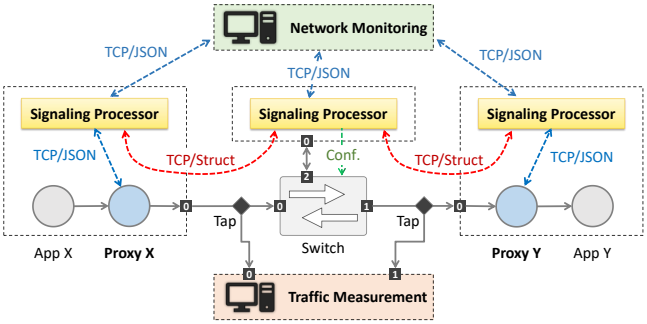

Figure 1: TSN framework: signaling processor, TCP-TSN proxy, monitoring, and traffic measurement

While TSN integration into areas such as 5G radio systems [10, 13] has gained attention, practical testing remains difficult. Existing tools [11] often cover only parts of the stack, e.g., focusing on signaling while omitting control logic or application interfaces. Historically, extensible baseline software has driven the adoption and understanding of new technologies. In Software-Defined Networking (SDN), tools like Open vSwitch [12], the Ryu controller [1], and Mininet [9] have been widely used in research, teaching, and development – thanks to their ease of deployment and extensibility.

Contribution: In this demo, we encourage more practical TSN implementations. A framework for TSN resource reservation with Strict Priority transmission selection is presented. It can handle legacy TCP applications by redirecting them through a dedicated proxy, which also handles traffic shaping for end devices. Further, it provides APIs for monitoring and configuration, and an accurate off-the-shelf measurement methodology for one-way delays and traffic volume. It can be used as-is for education, or extended to implement new functions, such as novel redundancy mechanisms. To this end, the key components are designed to be simple and modular, with very little dependencies.

2 FRAMEWORK COMPONENTS

The framework is comprised of four components: a signaling processor and controller, a data proxy to support legacy TCP apps, a network monitor for resource utilization visualization, and a live traffic measurement framework with hardware time stamping. An overview of all components

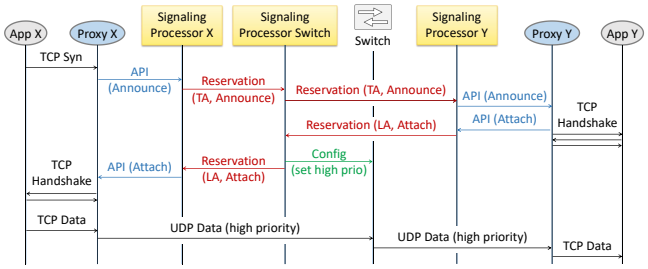

Figure 2: Connection establishment with reservation

can be seen in Figure 1. A sequence diagram of the signaling process for a new connection is presented in Figure 2.

Signaling Processor: The core component is the signaling processor, implemented in plain Python based on the Resource Allocation Protocol (RAP) specification [5]. Beyond signaling traffic requirements, it handles control tasks such as latency computation and admission control. To simplify deployment and extensions, we implemented a streamlined RAP variant that removes many dependencies and introduces direct Layer 4 addressing. While not fully standard-compliant, it preserves RAP's data structures and control logic. From an application's perspective, the resulting configuration mirrors standard RAP behavior, offering a lightweight alternative. Extensions validated in this version are typically portable to the full RAP specification.

The signaling processor supports three interface types, shown in different colors in Figure 2. First, the JSON API (blue) is used to issue and monitor new reservations. Second, the simplified RAP protocol (red, TA/LA) distributes and processes reservations, including latency calculations and RAP protocol field updates. Third, once a reservation is accepted, the switch is configured accordingly – here, the priority of the corresponding UDP connection is increased (green). The reservation completes when Proxy X finalizes the TCP handshake.

Proxy Application: The proxy, written in C for high performance, enables legacy TCP applications to use UDP with resource reservations for predictable latency guarantees. As shown in Figure 1, it triggers a reservation via the signaling processor's JSON interface upon receiving a new TCP connection. Each reservation includes a traffic specification, e.g., burst size and data rate, which is enforced by the proxy using token-bucket shaping.

Network Monitoring: Monitoring is handled by a Network Management System (NMS), represented by StableNet [8] in this demo. Since the control decisions are decentralized, the NMS collects data from all host and switch processors via the JSON interface. It provides detailed insights into nodes, resources, and streams, enabling visualization, bottleneck detection, configuration optimization, and root cause analysis. For instance, if a stream reservation fails due to delay constraints, the NMS can pinpoint the exact link responsible.

Traffic Measurement: The final component is an accurate, live traffic measurement framework. As shown in Figure 1, traffic from both reserved and best-effort streams is mirrored before and after the switch using Layer 1 network taps, avoiding added jitter. The mirrored streams are received on two ports of an Intel i350 NIC, which provides 125 MHz hardware time stamping [2, 3]. Both ports share a controller, so all time stamps are based on the same clock, enabling delay measurements with sub-microsecond accuracy. In our tests, this was sufficiently accurate to detect different cable lengths based on propagation delays. The system works in four stages: (i) an eBPF filter removes irrelevant traffic, (ii) a packet sniffer matches incoming packets and computes delays, (iii) a Python tool aggregates the results, and (iv) a web interface visualizes delay updates in real time.

Together, these components can not only kick-start TSN resource reservation for legacy apps, they can also monitor and measure the underlying control plane and data plane.

3 DEMONSTRATION

The demo topology is the one depicted in Figure 1. Two end devices each host the respective apps, the signaling processor, and the proxy. Another Linux machine will host the signaling processor for the switch. The switch itself can be any managed switch, such as the Kontron D10, which supports several TSN standards in the data plane. The monitoring and measurement tools will be hosted on additional, individual PCs. The monitoring PC will physically be connected to the same switch as the end devices – the TCP/JSON streams will be transmitted with best-effort performance on the same infrastructure.

The demonstration will include various scenarios with and without traffic reservation, for comparison. Cross-traffic will be introduced to fill the switch queues for unreserved, low priority traffic. After successful reservation, the respective streams receive a higher priority. Apps can range from synthetic data, such as iPerf tests, to sensor data from TinkerForge bricklets [14]. In addition to the physical testbed, the signaling and monitoring components can be demonstrated for a bigger, virtual network.

ACKNOWLEDGMENTS

This work was funded by the Bavarian Ministry of Economic Affairs, Regional Development and Energy under grant number DIK0250/02 (project KOSINU5).

REFERENCES

[1] Ryu SDN Framework Community. 2025. Ryu component-based software defined networking framework. (2025). https://ryu-sdn.org/

[2] Alexej Grigorjew, Philip Diederich, Tobias Hoßfeld, and Wolfgang Kellerer. 2022. Affordable Measurement Setups for Networking Device Latency with Sub-Microsecond Accuracy. In *KuVS Fachgespräch - Würzburg Workshop on Modeling, Analysis and Simulation of Next-Generation Communication Networks 2022 (WueWoWAS'22)*.

[3] Alexej Grigorjew, Lukas Kilian Schumann, Philip Diederich, Tobias Hoßfeld, and Wolfgang Kellerer. 2023. Understanding the Performance of Different Packet Reception and Timestamping Methods in Linux. In *KuVS Fachgespräch - Würzburg Workshop on Modeling, Analysis and Simulation of Next-Generation Communication Networks 2023 (WueWoWAS'23)*.

[4] IEC/IEEE. 2025. IEC/IEEE 60802 TSN Profile for Industrial Automation. (2025). https://1.ieee802.org/tsn/iec-ieee-60802/

[5] IEEE. 2025. IEEE Standard Draft P802.1DD – Resource Allocation Protocol (RAP). *IEEE Std P802.1DD D1.1* (2025).

[6] IEEE. 2025. P802.1DG – TSN Profile for Automotive In-Vehicle Ethernet Communications. (2025). https://1.ieee802.org/tsn/802-1dg/

[7] IEEE. 2025. Time-Sensitive Networking (TSN) Task Group. (2025). https://1.ieee802.org/tsn/

[8] infosim.net. 2025. StableNet Automated Network & Service Management. (2025). https://www.infosim.net/stablenet/

[9] Bob Lantz, Brandon Heller, and Nick McKeown. 2010. A network in a laptop: rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. 1–6.

[10] Ahmed Nasrallah, Akhilesh S Thyagaturu, Ziyad Alharbi, Cuixiang Wang, Xing Shao, Martin Reisslein, and Hesham ElBakoury. 2018. Ultra-low latency (ULL) networks: The IEEE TSN and IETF DetNet standards and related 5G ULL research. *IEEE Communications Surveys & Tutorials* 21, 1 (2018), 88–145.

[11] Lukas Osswald, Steffen Lindner, Lukas Wüsteney, and Michael Menth. 2021. Rap extensions for the hybrid configuration model. In *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 1–8.

[12] Ben Pfaff, Justin Pettit, Teemu Koponen, Ethan Jackson, Andy Zhou, Jarno Rajahalme, Jesse Gross, Alex Wang, Joe Stringer, Pravin Shelar, et al. 2015. The design and implementation of open {vSwitch}. In *12th USENIX symposium on networked systems design and implementation (NSDI 15)*. 117–130.

[13] Zenepe Satka, Mohammad Ashjaei, Hossein Fotouhi, Masoud Daneshtalab, Mikael Sjödin, and Saad Mubeen. 2023. A comprehensive systematic review of integration of time sensitive networking and 5G communication. *Journal of systems architecture* 138 (2023), 102852.

[14] tinkerforge.com. 2025. TinkerForge Bricklets. (2025). https://www.tinkerforge.com/en/doc/Hardware/Bricklets/Bricklets.html