Towards the Integration of TAPRIO-based Scheduling with Centralized TSN Control

George Papathanail, Lefteris Mamatas, and Panagiotis Papadimitriou
University of Macedonia, Greece
{papathanail, emamatas, papadimitriou}@uom.edu.gr

***Abstract*—Time-Sensitive Networking (TSN) aspires to provide a solid underpinning for meeting latency guarantees across traffic classes with diverse requirements. Despite the emergence of various specifications from the TSN family of standards, there is a shortage of prototype implementations with support for both TSN-based scheduling and its configuration by a Centralized Network Controller (CNC).**

To this end, we present the design and implementation of a TSN platform that couples Time-Aware Priority Shaper (TAPRIO) with CNC. The CNC is utilized to convey TSN schedules into TAPRIO using NETCONF and YANG. We employ this platform for assessing the impact of TAPRIO on the latency and jitter experienced by scheduled traffic. Furthermore, we quantify the TAPRIO configuration overhead by populating TSN schedule configurations by the CNC. Our experimental results corroborate the proper operation of TAPRIO and its efficient interaction with the CNC.

I. Introduction

Deterministic networking has evolved into a crucial research topic and is constantly gaining traction and appeal across various types of networks, such as industrial, vehicular, wireless, as well as service provider networks [1]–[4]. In this respect, deterministic networking introduces mechanisms either at the network or the data-link layer towards traffic prioritization with the aim of satisfying the strict requirements of delay-sensitive flows and ensuring that their data is delivered within certain deadlines. More specifically, on the network layer, the IETF DETerministic NETwork (DETNET) [5] working group investigates new techniques for deterministic QoS, spanning from explicit routes, packet replication and elimination to congestion protection with end-to-end synchronization. On the other hand, similar efforts over Ethernet are concentrated on enabling asynchronous and deterministic low-latency services. In this scope, IEEE 802.1 Time-Sensitive Networking (TSN) has introduced various standards, such as IEEE 802.1 Qbv, also known as Time-Aware Shaper (TAS), which essentially provides support for fine-grained scheduling across multiple queues on switch egress ports [6].

Despite the various recent advancements, TSN still entails significant challenges, such as the synchronization among talkers and TSN switches for efficient operation, the translation of high-level flow requirements or intents into low-level Gate Control List (GCL) configurations. Such problems can be thoroughly investigated only using a TSN platform that encompasses both data and control plane elements. With respect to the latter, TSN is associated with Centralized Network Control (CNC), which has full knowledge of the network topology and flow requirements [7]. In the fully centralized TSN control plane model, the flow requirements are conveyed to the CNC via a logical entity, namely Centralized User Configurator (CUC), based on IEEE 802.1Qdj. Despite the emergence of TSN control-plane specifications, there is a shortage of TSN prototype implementations that integrate TSN schedulers with a CNC. In particular, we are mainly aware of [3] that jointly performs flow re-routing and TSN schedule configuration using segment routing over IPv6.

In this respect, we present a TSN platform that facilitates experimentation with TSN mechanisms (*e.g.,* Time-Aware Priority Shaper - TAPRIO) in a testbed environment. The TSN platform encompasses (i) a TAPRIO-enabled switch datapath that can be configured to assign traffic flows to certain queues (on egress ports), which are associated with configurable GCLs, and (ii) a prototype implementation of a CNC that generates TSN schedules for installation into TSN datapath. To this end, we rely on the NETCONF [8] communication protocol, whereas YANG [9] is utilized for the representation of TSN schedule configurations. We further provide the means for experimentation with TSN in emulation environments, such as Mininet, by porting TAPRIO into Mininet based on our previous work [1].

We take advantage of our TSN platform for assessing various data and control plane aspects of TSN within Mininet. More specifically, we assess the impact of diverse TSN schedules on latency and jitter using TAPRIO. Furthermore, we quantify and decompose the TAPRIO configuration overhead by conveying TSN schedule configurations from CNC into TAPRIO using NETCONF. Our experimental results corroborate the proper operation of TAPRIO scheduler and also indicate an efficient interaction between CNC and the underlying TAPRIO.

The remainder of the paper is organized as follows. Section II provides background on TSN and further discusses TSN mechanisms pertaining to IEEE 802.1Qbv. In Section III, we elaborate on the design and implementation of the TSN platform. Section IV discusses our evaluation results using the proposed TSN platform, whereas Section V provides an overview of related work. Finally, Section VI highlights our conclusions and provides directions for future work.

ISBN 978-3-903176-57-7

II. Time-Sensitive Networking

TSN encompasses a set of standards developed by the Time-Sensitive Networking task group within the IEEE 802.1 working group. These TSN standards specify methods for transmitting data with high time-sensitivity over Ethernet networks that are deterministic. The majority of TSN projects extend the IEEE 802.1Q – Bridges and Bridged Networks standards, which deal with Virtual Local Area Networks (VLAN) and network switches. These extensions aim at data transmission with bounded latency and high reliability [4]. TSN mechanisms are particularly relevant to areas, such as automotive and industrial control, where real-time Audio/Video Streaming and control streams are used in converged networks.

Numerous IEEE 802.1 specifications are available, including 802.1Qbv – Time Aware Shaper [6], IEEE 802.1 Qbu Preemption [10], and IEEE 802.1AS Timing and Synchronization [11]. These specifications provide support for various features and functionalities for network communication. For instance, 802.1Qbv is associated with the Time-Aware Shaper (TAS) mechanism for controlling latency, whereas 802.1Qbu offers the Preemption feature for interrupting and resuming frame transmission. In addition, 802.1AS focuses on timing and synchronization within the network. These IEEE 802.1 standards are integral components of modern networking, providing essential tools for the transmission of data over networks with different performance requirements.

In the following, we elaborate further on IEEE 802.1Qbv, which is currently supported by our TSN platform.

IEEE 802.1Qbv. In the context of IEEE 802.1 standards, 802.1Qbv introduces a transmission gate operation concept for each traffic class queue, as depicted in Fig. 1. At the egress port of a TSN switch, outgoing frames go through a Traffic Classification block that categorizes different streams to their respective traffic classes. This classification process is vital in preventing traffic overload, which could otherwise affect the switches. At *T1*, the traffic class 0, which is assigned to traffic priority 0, is open for transmission. The packets are then queued into various traffic classes based on the state of the transmission gates. These gates are either open or closed, and their status is controlled by a GCL. A GCL contains multiple schedule entries for each output port, allowing selected traffic to pass through open gates to the transmission selection block, which provides access to the medium. For a TSN switch with IEEE 802.1Qbv schedules to function as expected, the clocks of all switches should be synchronized. This would ensure that all TSN switches reference the same cycle base time in their schedules. Such synchronization can be attained via the Precision Time Protocol (PTP). In essence, IEEE 802.1Qbv coupled with PTP provides the means for the implementation of appropriate GCL schedules and time synchronization, which can guarantee the transmission of high-priority critical traffic without interference from other best-effort traffic. This eventually can ensure bounded delay and low jitter of scheduled traffic, as well as protection of high-priority flows.

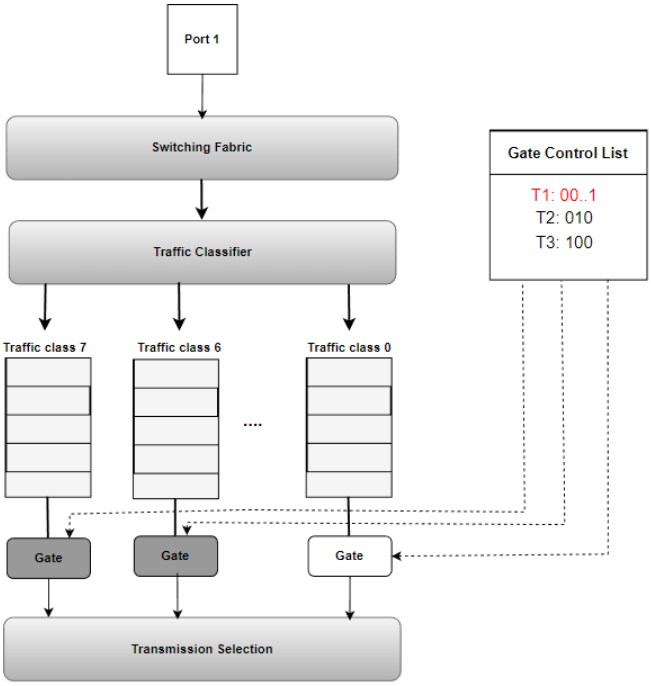

Fig. 1. Example of IEEE 802.1Qbv.

III. TSN Platform

In this section, we present our TSN platfrom design and implementation. The platform couples a TAPRIO-based TSN datapath with a TSN controller (*i.e.,* CNC), empowering experimentation with a fully-fledged TSN bridge that encompasses both data and control plane elements (Fig. 2). In the following, we discuss in detail the functionality of the TSN platform.

A. TSN Data Plane

In order to utilize TSN, we activate it by employing TAPRIO. TAPRIO is a queuing discipline for Linux and is included in the traffic control *tc* tool of Linux, which allows us to implement the 802.1Qbv Time-Aware Shaper. TAPRIO empowers the configuration of a series of gate states, each one enabling outgoing traffic for a subset of traffic classes based on the concept of a time slice. TAPRIO implements packet classification to a specific traffic class by utilizing the priority field of the socket buffer employed by the network stack of the Linux kernel (skb *to* priority).

As we utilize IPV6 addressing, the mapping of the traffic classes to queues is carried out by modifying the DSCP field of the IPV6 packet header. To modify the *skb* priority field of the socket buffer based on the DSCP field in the packet header, we employ the *iptables* method in order to establish the priority field of SKB, before packets are directed to the *Qdisc*. *iptables* is a packet filter tool at the IP layer that is used to configure, sustain, and inspect the IP packet filter rule tables in the Linux kernel. We incorporate the relevant classifier rules into *iptables* to modify the *skb* priority field appropriately.

The mangle table at the IP layer is employed to modify packets. The classifier rule is added to the POSTROUTING

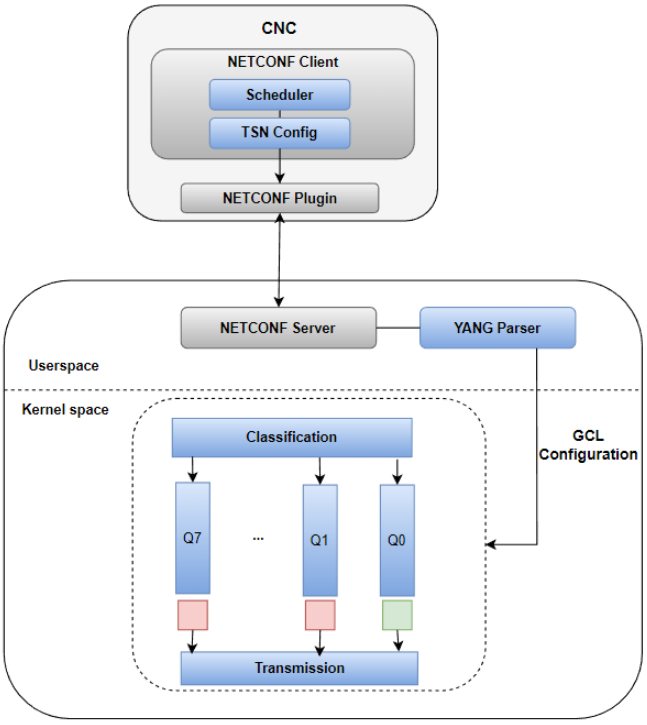

Fig. 2. TSN platform overview.

chain, which is triggered after the forwarding decision, just before the packet reaches the *Qdisc*. This empowers TAPRIO Qdisc to recognize the priority field designated by the *iptables* rule (classifier).

B. TSN Control Plane

Regarding the control plane, we utilize a hybrid TSN implementation, as described in the IEEE 802.1Qcc standard [12], in order to automate the TAPRIO configuration process. More specifically, we have implemented a CNC that has the capability to compute TSN 802.1Qbv schedules and populate these schedules into TSN-enabled switch datapaths. The CNC communicates with the switches using the NETCONF [8] remote communication protocol to establish communication and IETF YANG data models.

A YANG parser module parses the YANG-TSN models and translate them into a set of *tc* commands that affect the queuing disc layer of the Linux kernel. As such, the CNC is enabled to remotely access and configure the switches to support TAPRIO and other TSN features, thus simplifying the network management process. The interaction between the CNC and the TAPRIO is aligned with [3], which also relies on NETCONF and YANG for the installation of TSN schedules into the datapath.

IV. Experimental Study

In this section, we utilize our TSN platform and conduct a set of experiments to assess various aspects of TSN. More specifically, we assess the impact of TAPRIO scheduling on high-priority and best-effort traffic, and we also quantify the

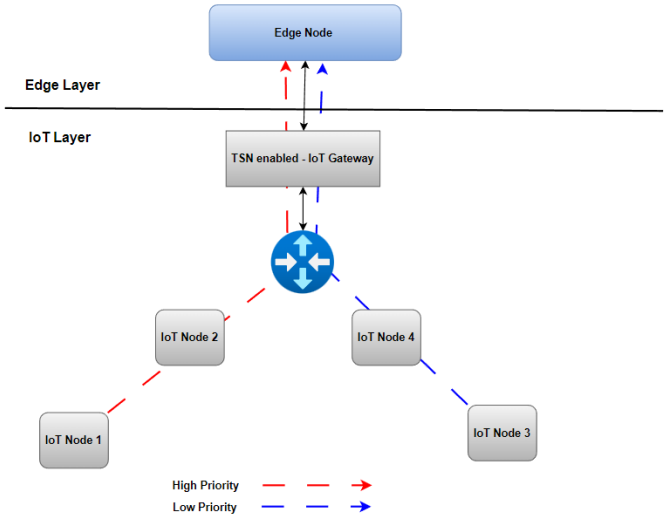

Fig. 3. Experimental topology in Mininet.

control communication overhead in relation to the CNC. In order to assess the benefits of TSN, we activate TAPRIO on the egress port of an IoT Gateway using the topology depicted in Fig. 3, which is created through a modified version of Mininet 2.3.1 [13]. In order to use TAPRIO in this topology, based on [1] we modify Mininet in order to support multi-queued NIC interfaces, since by default Mininet only supports single-queue interfaces. More specifically, the modified Mininet version can support up to 8 TX/RX queues. We also use IPMininet [14] in order to support IPV6 addressing. All experiments are carried out on an Ubuntu 20.04.1 LTS Virtual Machine with 8 virtual CPUs and 8 GB of RAM, running kernel version 5.4.0-139.

A. Impact of Diverse TAPRIO Schedules

The goal of this experiment is to demonstrate the impact of TAPRIO scheduling on an IoT Gateway. Based on the Mininet topology depicted in Fig. 3, we set up the TAPRIO Qdisc with two traffic classes: (i) High Priority and (ii) Best Effort, where the former is configured to match traffic with DSCP field value of 0x40, wheras the latter matches best-effort traffic with DSCP field value of 0x00. To generate traffic, we rely on Iperf [15]. Specifically, we inject packets of size 1,440 bytes at a rate of 2,000 packets/s for High-Priority traffic and CBR traffic of 1,440 bytes packet size for Best-Effort traffic.

The impact of 802.1Qbv scheduling on latency and jitter for a 1 ms cycle time is shown in Figs. 4 and 5, by varying the allocated time ratio between High-Priority and Best-Effort traffic. For the sake of simplicity, only average values are presented as the network cycle is not synchronized with the application cycle (such synchronization is extremely difficult to attain within Mininet).

The average latency and jitter for High-Priority traffic yield a linear increase as the percentage of allocated cycle time decreases. Conversely, the average latency and jitter for Best-Effort traffic decreases as the percentage of allocated time for this traffic class increases. It is important to note that these results are consistent with the expected behavior of TAPRIO's

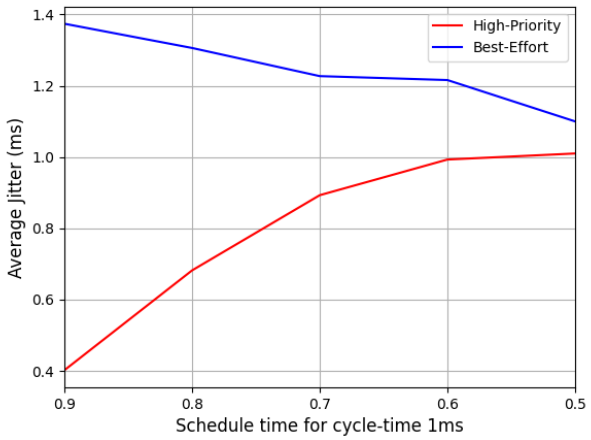

Fig. 4. Impact of 802.1Qbv on jitter for High-Priority and Best-Effort traffic.

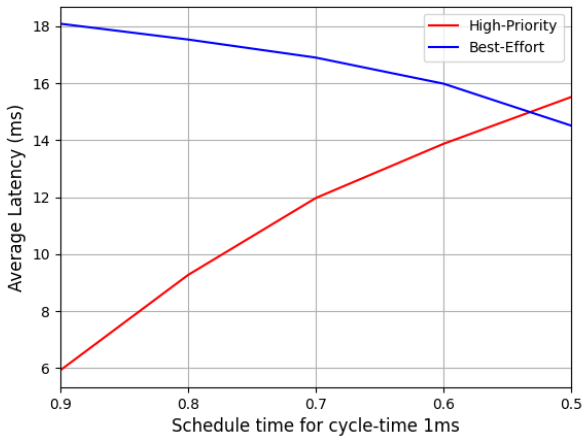

Fig. 5. Impact of 802.1Qbv on latency for High-Priority and Best-Effort traffic.

scheduling mechanism, where High-Priority traffic is prioritized over Best-Effort traffic. In addition, we observe that the maximum latency and jitter are highly dependent on the current policy in effect, which is determined by the allocated time ratio between the two traffic classes. Overall, these results highlight the effectiveness of TAPRIO in managing traffic and prioritizing delay-sensitive traffic in a network environment.

B. Impact of TAPRIO 800:200

We conduct another experiment, at which we utilize again the Mininet topology of the previous experiment and send 1440-byte packets at a rate of 2000 packets per second both for High-Priority and Best-Effort traffic. The traffic is injected using Iperf, and the primary goal of this experiment is to measure the effect of different allocation times in the GCL. More specifically, we allocate 80% of the time to high-priority traffic and 20% to low-priority traffic on a cycle time of 1 ms. We measure the impact of this allocation on jitter and latency, by plotting the CDF for each metric (Figs. 6 and 7, respectively).

Fig. 6 illustrates the distribution of jitter for High-Priority and Best-Effort traffic. This plot confirms that High-Priority is associated with lower jitter compared to Best-Effort traffic. Specifically, the measurements of High-Priority traffic indicate a jitter of less than 1 ms for 80 percent of the observations, whereas Best-Effort traffic experiences jitter of less than 1.6 ms for the same percentage of observations. These results imply that allocating a higher percentage of the GCL time to High-Priority traffic can lead to reduced (and potentially bounded) jitter in the network, which is beneficial for real-time applications that require both low latency and jitter.

The plot in Fig. 7 depicts the distribution of latency for High-Priority and the Best-Effort traffic. We reach a similar observation, as High-Priority traffic exhibits lower latency. In particular, on the 80 percent of the observations, the latency for High-Priority traffic does not exceed 13 ms, whereas Best-Effort traffic can experience latency up to 23 ms. Furthermore, we observe a long tail in the latency distribution of Best-Effort traffic (as opposed to High-Priority). This implies that TAPRIO can practically lead to bounded latency for traffic scheduled with high priority.

It is important to note that the high latency measurements in our experiments are an artifact of Mininet. Achieving stable and accurate measurements in a software-based emulation environment is difficult, and as such, high-precision measurements mandate hardware solutions, such as programmable NICs or NetFPGAs, in conjunction with specialized capture cards. Also, recall the difficulty in enabling synchronization among the TSN switches and the talkers within an emulation environment. In the future, we will seek to gain access to such specialized equipment and utilize our platform for measurements over TSN with higher precision.

C. Control Communication Overhead

In addition to the previous experiments on the effect of TAPRIO on scheduled traffic, we extend our experimentation to the interaction between the CNC and the TSN datapath. In particular, our aim is to measure and understand the communication overhead of the CNC when TSN schedules are populated into the switch. To this end, we perform tests on a topology consisting of a talker-listener pair, 1–10 switches, and the CNC (Fig. 8).

Initially, we measure the control communication overhead when inserting the TSN schedule configuration into only one switch. Subsequently, we increase the number of switches and measure the delay, as we populate configurations into switches ranging from two to ten. To better reason about the time spent for TAPRIO configuration, we measure (i) the time required to generate the TAPRIO configuration at the CNC, (ii) the communication delay between the NETCONF server and the NETCONF client, and (iii) the delay incurred by the server to parse the configuration from the NETCONF client and convert it to the appropriate format. These essentially comprise three subsequent steps for the configuration of TAPRIO by the CNC.

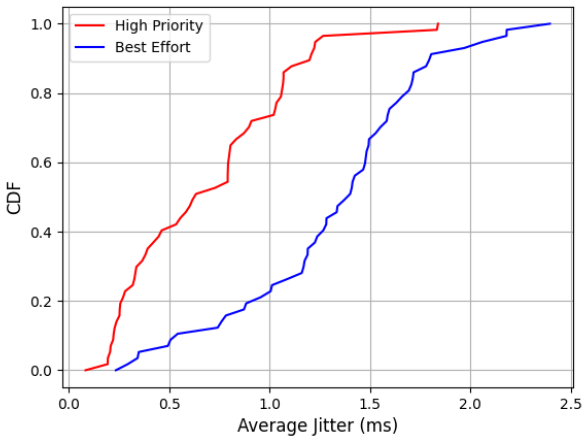

Fig. 6. CDF of jitter with TAPRIO 800:200 (80% of the time allocated to High-Priority traffic and 20% to Best-Effort traffic on a cycle time of 1 ms).

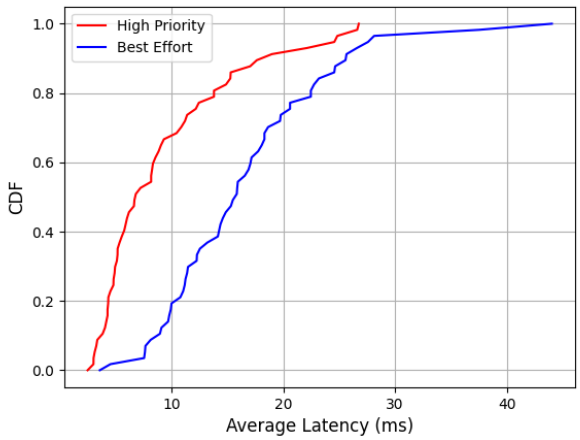

Fig. 7. CDF of latency with TAPRIO 800:200 (80% of the time allocated to High-Priority traffic and 20% to Best-Effort traffic on a cycle time of 1 ms).

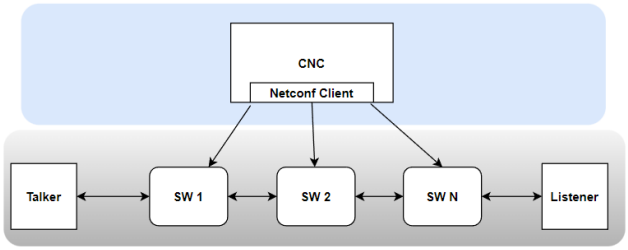

Fig. 8. Experimental setup for the measurement of control communication overhead (N = 1...10).

The corresponding results are shown in Table I. The time required by the CNC to generate the TAPRIO configuration represents a significant portion of the control communication overhead, in comparison to the communication delay and the XML parsing overhead. More specifically, we observe that the time required to complete the first step (*i.e.,* TAPRIO schedule generation at the CNC) increases linearly with the number of switches, as illustrated more clearly in Fig 9. The delays incurred for the subsequent steps (2 and 3) are minimal and mostly unaffected by the increase in the number switches. This stems from the parallelization of each process across the switches. Overall, the low control communication overhead measured in our experiments corroborates the efficacy of our TSN platform, especially in terms of CNC and TAPRIO interaction.

V. Related Work

In this section, we provide an overview of related work on (i) TSN data plane and (ii) TSN control plane. Both have attracted significant attention and, especially with respect to IEEE 802.1Qbv.

Data Plane. Authors in [5] provide an overview of various existing strategies, including TSN, in order to achieve ultra-low latency. Furthermore, [16] performs a comparison between IEEE 802.1Qbv and 802.1Qbu based on simulations, highlighting the drawbacks of TAS in terms of configuration complexity and guard bands inefficiencies. The TSN 802.1Qbv scheduling problem has been addressed using various techniques. For example, the problem has been tackled using Satisfiability Modulo Theory (SMT) and Optimization Modulo Theory (OMT) [17], [18]. Authors in [19] implement a simulation TSN environment by leveraging on OMNet++ and examine the impact of TSN scheduling on different classes of traffic. Finally, the work in [20] describes the development of a TSN system that is reliable and scalable. The system has successfully implemented the IEEE 802.1ASrev standard for clock synchronization and the IEEE 802.1Qbv standard for ordered packet sending on Linux end-devices.

Control Plane. As far as the control plane is concerned, authors in [2] present an approach for ultra-fast recovery of TSN using Software-Defined Networking (SDN). More specifically, they employ Source Routing for a nearly-stateless data plane in order to achieve fast failure recovery. Another approach

TABLE I
Control Communication Overhead

Switches	TAPRIO Configuration Generation (sec)	NETCONF Communication (sec)	YANG Parsing (sec)
1	0,128	0,026	0,019
2	0,257	0,024	0,019
3	0,482	0,025	0,020
4	0,625	0,025	0,020
5	0,803	0,022	0,021
6	1,017	0,024	0,021
7	1,198	0,025	0,020
8	1,379	0,024	0,020
9	1,599	0,025	0,022
10	1,781	0,027	0,021

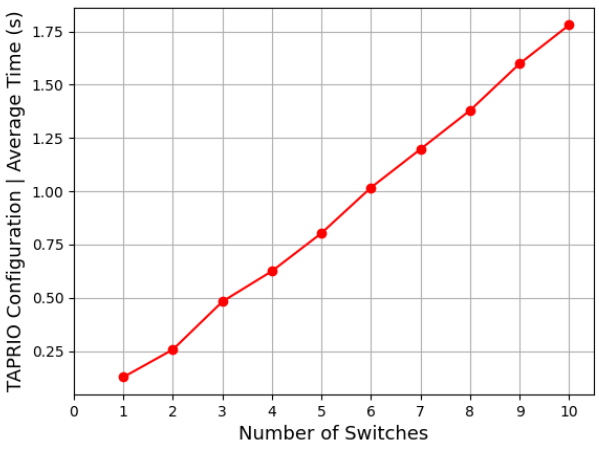

Fig. 9. Control communication overhead vs. number of switches.

introduces a mechanism for managing reliability in SDN-controlled TSN networks, by utilizing NETCONF and Finite State Machines (FSMs) [21]. By proactively instructing nodes to perform specific actions in response to certain events such as performance degradation or failures, the proposed approach reduces recovery time and can effectively bypass the SDN controller. Alvarez et al. [7] have implemented a Dockerized CNC, which supports various TSN implementations. Authors in [22] argue that P4 is not sufficiently abstract to program a TSN-enabled switch on its own, but can be used instead in combination with other tools or platforms (*e.g.,* tc, DPDK, OpenvSwitch) in order to achieve the necessary modularity.

VI. Conclusions

In this paper, we presented a TSN platform that enables experimentation with TSN mechanisms in both testbed and network emulations environments. Our TSN platform encompasses (i) a TAPRIO-enabled switch datapath that can be configured to handle traffic under pre-defined TSN schedules and (ii) a prototype implementation of a CNC that generates TSN schedules and installs them into TAPRIO using NETCONF.

Our evaluation results show that TSN schedules have a significant impact on latency and jitter experienced by scheduled traffic. In terms of CNC and TAPRIO interaction, we investigated and decomposed the process of generation of TSN schedules and conveying them into TAPRIO. In this respect, we measured a low control communication overhead, which corroborates the feasibility of our TSN platform design.

Future work will be focused on the utilization of this TSN platform in a testbed environment for the investigation of challenges entailed by TSN, such as the synchronization among talkers and TSN switches for the prioritization of time-critical traffic under strict delay bounds.

VII. Acknowledgments

This work was funded by the European Union's Horizon Europe research and innovation program under grant agreement No. 101070487 (NEPHELE).

References

[1] G. N. Kumar, K. Katsalis, and P. Papadimitriou, "Coupling source routing with time-sensitive networking," in *IFIP Networking Conference (Networking)*, 2020, pp. 797–802.

[2] G. N. Kumar, K. Katsalis, P. Papadimitriou, P. Pop, and G. Carle, "Failure handling for time-sensitive networks using sdn and source routing," in *7th IEEE International Conference on Network Softwarization (NetSoft)*, 2021, pp. 226–234.

[3] ——, "Srv6-based time-sensitive networks (tsn) with low overhead rerouting," *International Journal of Network Management, Wiley*, 2022.

[4] S. Fu, H. Zhang, and J. Chen, "Time sensitive networking technology overview and performance analysis," *ZTE Communications*, vol. 16, no. 4, pp. 57–64, 2018.

[5] A. Nasrallah, A. S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. ElBakoury, "Ultra-low latency (ull) networks: The ieee tsn and ietf detnet standards and related 5g ull research," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 88–145, 2018.

[6] IEEE, *IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 25: Enhancements for Scheduled Traffic*, ser. IEEE Std 802.1Qbv-2015. IEEE, Mar. 2016.

[7] I. Álvarez, A. Servera, J. Proenza, M. Ashjaei, and S. Mubeen, "Implementing a first cnc for scheduling and configuring tsn networks," in *27th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2022, pp. 1–4.

[8] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman, "Network Configuration Protocol (NETCONF)," in *RFC 6241*, 2011.

[9] M. Bjorklund, "Yang-a data modeling language for the network configuration protocol (netconf)," Tech. Rep., 2010.

[10] IEEE, *IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 26: Frame Preemption*, ser. IEEE Std 802.1Qbu-2016. IEEE, Aug. 2016.

[11] IEEE, *Standard for Local and Metropolitan Area Networks—Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks, IEEE Standard 802.1AS-2011, pp. 1–292, Mar. 2011.*

[12] *IEEE, Std 802.1Qcc-2019: Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks–Amendment 9: Stream Reservation Protocol (SRP) Enhancements and Performance Improvements*, 2019.

[13] "Mininet," http://mininet.org/, accessed: 2023-02-28.

[14] "IPMininet," https://ipmininet.readthedocs.io/en/latest/, accessed: 2023-02-28.

[15] V. GUEANT, "iperf-iperf3 and iperf2 user documentation," *Iperf. fr. Np*, 2017.

[16] A. Arestova, K.-S. J. Hielscher, and R. German, "Simulative evaluation of the tsn mechanisms time-aware shaper and frame preemption and their suitability for industrial use cases," in *IFIP Networking Conference (IFIP Networking)*, 2021, pp. 1–6.

[17] W. Steiner, S. S. Craciunas, and R. S. Oliver, "Traffic planning for time-sensitive communication," *IEEE Communications Standards Magazine*, vol. 2, no. 2, pp. 42–47, 2018.

[18] S. S. Craciunas, R. S. Oliver, M. Chmelík, and W. Steiner, "Scheduling real-time communication in ieee 802.1 qbv time sensitive networks," in *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, 2016, pp. 183–192.

[19] C. Park, J. Lee, T. Tan, and S. Park, "Simulation of scheduled traffic for the ieee 802.1 time sensitive networking," in *Information Science and Applications (ICISA) 2016*. Springer, 2016, pp. 75–83.

[20] J. Lázaro, J. Cabrejas, A. Zuloaga, L. Muguira, and J. Jiménez, "Time sensitive networking protocol implementation for linux end equipment," *Technologies*, vol. 10, no. 3, p. 55, 2022.

[21] N. Sambo, S. Fichera, A. Sgambelluri, G. Fioccola, P. Castoldi, and K. Katsalis, "Enabling delegation of control plane functionalities for time sensitive networks," *IEEE Access*, vol. 9, pp. 136 151–136 163, 2021.

[22] S. T. Borda and J. Ermont, "An evaluation of software-based tsn traffic shapers using linux tc," in *2022 IEEE 18th International Conference on Factory Communication Systems (WFCS)*, 2022, pp. 1–4.