Evaluating the Integration of Wireless Time-Sensitive Networking with Software-Defined Networking for Dynamic Network Configuration

Alberto Morato*, Claudio Zunino*, Manuel Cheminod*, Stefano Vitturi*,
Dave Cavalcanti† Susruth Sudhakaran†, and Federico Tramarin‡
*National Research Council of Italy, CNR-IEIIT
Email: {alberto.morato, claudio.zunino, manuel.cheminod, stefano.vitturi}@ieiit.cnr.it,
† Intel Labs, Intel Corporation, Hillsboro, Oregon, USA
Email: {dave.cavalcanti, susruth.sudhakaran}@intel.com,
‡Department of Engineering "Enzo Ferrari", University of Modena and Reggio Emilia, Modena, Italy
Email: federico.tramarin@unimore.it

***Abstract*—The introduction of Time-Sensitive Networking (TSN) is revolutionizing real-time networks and time-critical applications. Recent advancements in this field extended the TSN capabilities to wireless technologies, giving rise to the concept of wireless TSN (WTSN). This paper focuses on the integration of wireless TSN with Software-Defined Networking (SDN) to enable dynamic network configuration and improve the performance of time-sensitive applications. We present a practical test environment that uses a hybrid network configuration consisting of wireless and wired TSN links. The primary objective is to evaluate the effectiveness of combining a TSN-capable network with an SDN controller. This setup enables dynamic configuration and routing within the system, allowing for prompt actions to address network issues, such as seamlessly re-routing data paths between the two links due to an increase in latency or packet loss. A measurement setup using OpenVSwitch in the wireless TSN domain is presented, along with the evaluation of time synchronization and dynamic route selection capabilities.**

***Index Terms*—Time-Sensitive Networking (TSN), Wireless TSN, Software-Defined Networking (SDN), Seamless routing, Traffic shaping and scheduling, Hybrid TSN**

I. Introduction

The introduction of the IEEE 802.1 Time-Sensitive Networking (TSN) [1] is revolutionizing the design and implementation of industrial networks and applications [2].

TSN comprises more than 20 standards that focus on various functionalities, including synchronization (802.1AS-2020) [3], traffic shaping and scheduling (802.1Qbv) [4], and network management (802.1Qcc) [5].

Although TSN was originally conceived for wired networks (Ethernet), recently, both the industry and academia have started to actively explore a new research direction that involves the development and adoption of wireless technologies with TSN supporting capabilities, known as wireless TSN. Currently, 5G and 802.11 are the primary candidates for enabling wireless TSN. The ultimate goal of TSN research and industry efforts is to support TSN functionalities across a wide range of wireless protocols. Interestingly, both 5G and 802.11 may be integrated with Ethernet TSN, leading to the emergence of Hybrid TSN architectures. Hybrid TSN offers significant advantages, including increased flexibility, cost-effective deployment, and seamless device interoperability regardless of the type of network connection (wired or wireless) they use.

TSN–based networks may be used in conjunction with SDN (Software-Defined Networking) [6] [7], a paradigm originated from the field of Information Technology that, provides a solution for programmable network architectures by introducing centralized control entities. The idea of combining these two technologies has been explored in [8], where an architecture based on TSN synchronization and Time Aware Shaping, and SDN has been evaluated.

By merging the features of TSN with SDN, it is possible to extend and enhance the capabilities of TSN, as well as to ensure greater dynamism in the (re–)configuration of networks. This will allow to achieve better network performance, synchronization and integration of time-sensitive applications.

Recent studies have explored the integration of Software-Defined Networking (SDN) and Time-Sensitive Networking (TSN) to address challenges in industrial automation [9], mixed-criticality control applications [10], dynamic path reconfiguration [11], and failure handling [12]. However, these studies primarily focus on legacy TSN applications, i.e. those based on Ethernet networks. In this paper, we extend this scenario by introducing TSN in the wireless domain. Specifically, we present a proof of concept that accomplishes the following

- Introduction of TSN and time synchronization over the wireless link.
- Evaluation of the end-to-end latency to detect possible delays in the communication path that may cause performance bottlenecks.
- Dynamic routing of time-sensitive traffic through hybrid (wired/wireless) networks based on latency measurements, to achieve ensure reliable and low-latency

This work has been partially funded by the EU Horizon Europe SNS JU PREDICT-6G (GA 101095890) Project.

communication.

Clearly, the use of SDN for managing hybrid TSN networks is based on the capability of seamlessly switching traffic flows over different links and devices. In this scenario, the switching from a wireless link to a wired one and vice versa, represents a challenge. The assessment of such a capability is one of the main goals of this paper. Specifically, in this direction, we present a test environment that utilizes a hybrid network configuration managed by a SDN controller. This hybrid network comprises both a wireless TSN link and a wired one. The setup uses SDN to dynamically re–configure the network. Specifically, as will be better explained in the next Sections, we will investigate how a traffic flow can be seamlessly moved from the wired link to the wireless one, maintaining the same level of performance, when communication issues are detected on the wired link. In detail, the paper describes the practical setup and the results of the experimental sessions carried out to assess the effectiveness of the proposed solution.

II. Measurement Setup and Approach

In this section, we will discuss the setup used for dynamic route selection using OpenVSwitch in TSN networks.

The setup shown in Figure 1 provides an overview of the used configuration. It is mainly composed of two 11th generation Intel NUCs equipped with i5-1135G7 processors. Both devices are running Ubuntu 22.04 with kernel version 6.2. The device named *NUC1* serves as packet source, it generates packets within the Linux network namespace *src* and sends them to the OpenVSwitch (OVS) switch through the virtual Ethernet interface `veth0`. On *NUC1*, the OVS switch and a controller implemented using *Ryu manager* are executed in the userspace. The controller exposes a REST endpoint accessible from both the Ethernet and Wi-Fi physical NICs, enabling dynamic packet routing from the Linux network namespace through any of the two physical NICs. By default, the controller routes all packets through the Ethernet NIC.

As can be seen in the figure, the Ethernet NIC on *NUC1* is connected to *NUC2* via an Ethernet switch. This allows to insert a source of additional traffic on the link, represented by the network tool `iperf3`, that is used to stress the communication between the two NUCs over the wired link. To this regard, the Ethernet links between the NUCs are set to $100\,\mathrm{Mbps}$, while the interfering source is configured for $1\,\mathrm{Gbps}$. This deliberate configuration allows for easier control and observation of the network behavior under different traffic conditions.

In particular, this setup allows to investigate the dynamic route selection capabilities of OpenVSwitch in TSN networks as well as to analyze the effects of interfering traffic on latency and overall network performance.

Moving on to the wireless link, NUC1 has been configured as a SoftAP (AP) on the $5\,\mathrm{GHz}$ band to provide connectivity to NUC2, which is configured as a station (STA). This wireless connection forms the Wireless TSN (WTSN) domain where, with respect to the IEEE 802.1AS standard, NUC1 acts as the Grand Master (GM) and provides the clock reference for synchronization to NUC2, which acts as a Follower.

Noticeably, in the setup, we introduced a cutting-edge element by utilizing the wireless link for time synchronization, deviating from the traditional approach of using Ethernet for time synchronization in TSN networks. By employing wireless synchronization based on the IEEE 802.1AS standard, we explore the potential of wireless connectivity to enhance the capabilities and versatility of time-sensitive applications. In this way, we are testing in a real setup the feasibility and performance implications of integrating wireless communication within TSN networks, all obtained maintaining synchronization accuracy.

In addition to its role in time synchronization, the wireless interface implements also the IEEE 802.1Qbv. This TSN standard allows to enable time-aware packet scheduling and prioritization, facilitating the coexistence of real-time and best-effort traffic on the same network while ensuring guaranteed bounded latency. In the setup shown in Figure 1, the scheduler has been configured with a cycle of duration 1ms. Within this cycle, a window of $250\,\mu\mathrm{s}$ has been allocated for best-effort traffic, and another one of $700\,\mu\mathrm{s}$ for real-time traffic. Moreover, $50\,\mu\mathrm{s}$ have been introduced as guard band between the two windows to prevent mutual interference.

In the first set of experiments shown in this paper, however, the best effort traffic was not taken into consideration. In other words, a single data stream was configured and synchronized to exclusively utilize the real-time window. This is because the experiments carried out were primarily meant to i) assess the effectiveness of the synchronization implemented over the wireless link and ii) to investigate the seamless switching capability from the wired link to the wireless one.

Clearly, we are aware of the importance of exploring more complex configurations that involve also the use of the best-effort window and, more in general, of diverse traffic types. These scenarios will be addressed in forthcoming work, in which we will investigate the full capabilities and performance potential of the proposed approach.

The packet generator, placed in the source Linux network namespace, generates User Datagram Protocol (UDP) frames. These frames have a total length of 298 bytes and are sent at a constant rate of 1000 packets per second (pps), equivalent to a transmission of a packet every 1ms. The data field of each frame encapsulates the timestamp indicating the instant of time when the frame was scheduled for delivery. Assuming the clocks on the two NUCs are precisely synchronized through the WTSN domain, the end-to-end latency can be accurately evaluated by subtracting the transmission timestamp to the reception one. This calculation is carried out by a dedicated listener running on the userspace of NUC2 when a frame arrives at NUC2. Moreover, the listener continuously calculates the moving average of the latency over a predefined number of samples, permitting an on-the-fly analysis of the latency trends. If the moving average exceeds a predefined threshold, the listener automatically calls the REST endpoint on the controller. This action triggers the dynamic switching

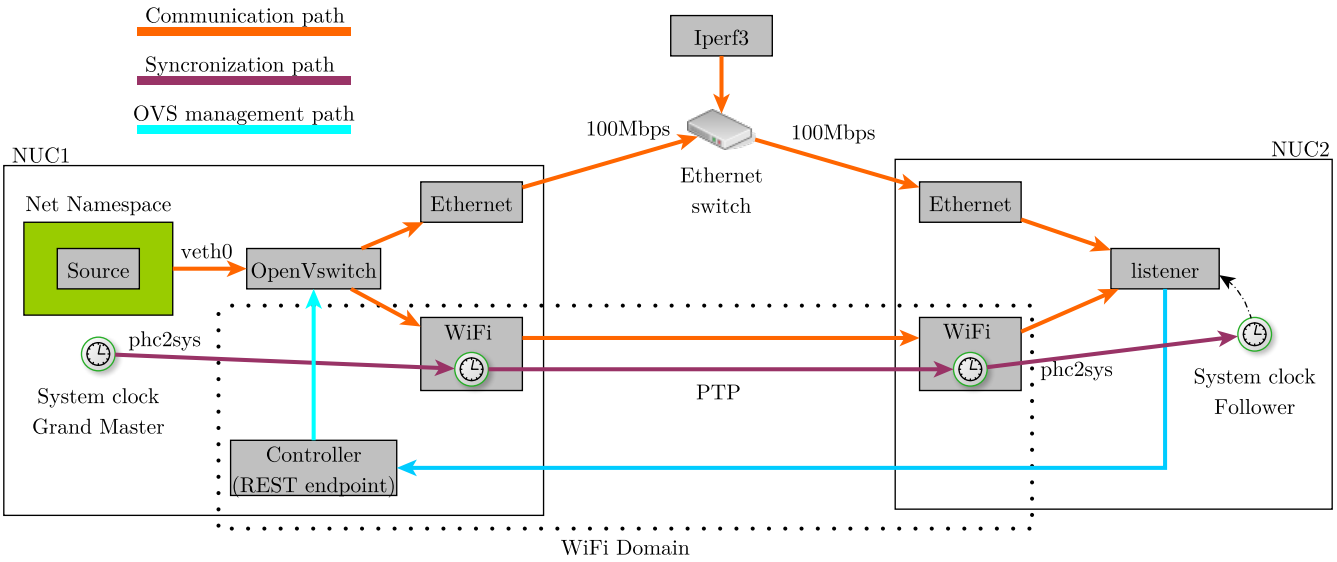

Fig. 1. Experimental setup

TABLE I
PARAMETERS USED FOR THE EXPERIMENT.

Parameter	Value
Protocol	UDP
Packet Size	298 (bytes)
Traffic type	Constant rate 1000pps
Moving avg window	200 samples
Latency threshold	45ms
Injection of interfering traffic	after about 10000pkts

of packets from the wired link to the wireless one, enabling the network to adapt and optimize its transmission path. Once the switching is successfully achieved, the packets remain routed through the WiFi connection, until a manual intervention from the user restores the original configuration.

Table I summarizes the key parameters used in the experimental setup, providing an overview of the specific values used to evaluate the system performance.

III. RESULTS

In this section, we present and discuss the results obtained from the measurement sessions carried out on the experimental setup.

A. Time Synchronization

As discussed earlier, ensuring time synchronization is essential for accurately calculating and characterizing the system latency. Figure 2 illustrates the empirical probability density function of the time synchronization offset recorded by the clock management tool, while Table II provides additional comprehensive statistics for reference.

Thus, with the achieved level of synchronization, it is possible to confidently proceed with the latency measurements and analysis, knowing that the time synchronization component

TABLE II
SYNCHRONIZATION OFFSET OF THE TWO DEVICES REPORTED BY THE CLOCK MANAGEMENT SOFTWARE

	Offset (ns)
mean	21.71
std	1703.25
min	-4934.00
max	7025.00
99th percentile	4872.77

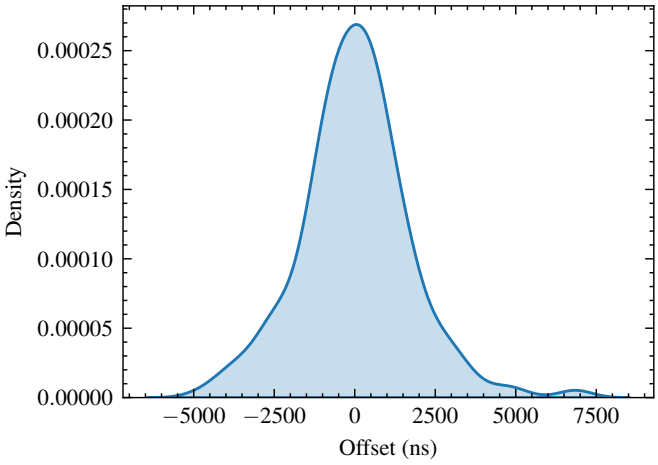

Fig. 2. Synchronization offset of the two devices reported by the clock management software

falls well within acceptable limits [13], providing assurance that data collection and analysis will be accurate and reliable.

B. Dynamic routing with OpenVSwitch

The second set of results concerns the experiments carried out with OpenVSwitch. The corresponding outcomes are

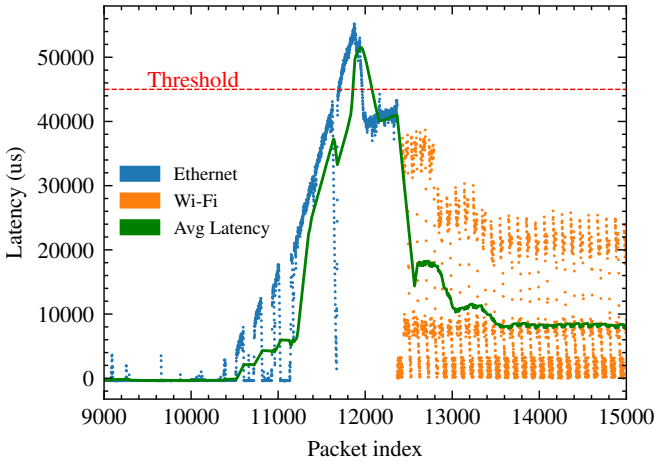

Fig. 3. End-to-End latency measured up to the userspace. The injection of interfering traffic begins at approximately packet index 10000. The green line represents the moving average of latency over 200 packets. Transients in WiFi latency are attributed to power-saving issues in the wireless NIC.

shown in Figure 3. Initially, all traffic was routed through the Ethernet NIC, resulting in a relatively low latency (i.e. the communication time from NUC1 to NUC2) of about 2 ms. However, once we introduced the source of interfering packets after transmitting approximately 10000 packets, the average latency began to increase rapidly, as clearly indicated in Figure 3. Upon reaching the latency threshold of 45 ms, the SDN controller was triggered, and the traffic flow was redirected through the wireless link (which was not interested by the interfering traffic). This resulted in a reduction in latency. However, it may be noticed that the WiFi channel exhibits sharp transients in latency, fluctuating between 2 ms and 20 ms. We have assessed that such fluctuations are mainly caused by the power-saving mechanism of the WiFi NIC. Indeed, it has been observed that the NIC periodically enters the power-saving mode, where incoming (from the higher layer) packets are necessarily enqueued for later transmission, with the consequent increase of both the latency and its randomness. Addressing this issue will be one of the focuses of future works with the aim, as discussed in [14], of achieving an end–to–end latency bounded to few milliseconds. By resolving the power-saving-related latency fluctuations, it will be possible to establish more stable, reliable, and timely latency on the wireless link. However, the experiments demonstrated the feasibility of the proposed technique, since the traffic flow was switched from one link to another without loss of packets and with significant benefits in terms of latency.

In such a direction, as an interesting experimental practice, we found that to ensure seamless link switching without packet loss, it is necessary to first create a link on the WiFi NIC with higher priority before deleting the link between the Ethernet NICs.

Finally, it has to be remarked that throughout the entire experimental process, the WTSN domain remained active, and synchronization between the two devices was maintained. This outcome is significant as it demonstrates the feasibility of utilizing the WiFi channel for time synchronization while simultaneously supporting dynamic routing with OpenVSwitch.

IV. Conclusions

In conclusion, our research focused on evaluating a hybrid network configuration that combines a TSN wireless link with a wired link. Our objective was to assess the effectiveness of integrating a TSN-capable network with an SDN controller and a probe. The preliminary setup implemented allowed for dynamic configuration adjustments and seamless transition between links for issue resolution. Further investigations will optimize the performance of this hybrid network configuration.

References

[1] "Time-Sensitive Networking (TSN) Task Group,." https://1.ieee802.org/tsn/. Accessed: 2023-05-19.

[2] S. Vitturi, C. Zunino, and T. Sauter, "Industrial communication systems and their future challenges: Next-generation ethernet, iiot, and 5g," *Proceedings of the IEEE*, vol. 107, no. 6, pp. 944–961, 2019.

[3] "Ieee standard for local and metropolitan area networks–timing and synchronization for time-sensitive applications," *IEEE Std 802.1AS-2020 (Revision of IEEE Std 802.1AS-2011)*, pp. 1–421, 2020.

[4] "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.

[5] "Ieee standard for local and metropolitan area networks–bridges and bridged networks – amendment 31: Stream reservation protocol (srp) enhancements and performance improvements," *IEEE Std 802.1Qcc-2018 (Amendment to IEEE Std 802.1Q-2018 as amended by IEEE Std 802.1Qcp-2018)*, pp. 1–208, 2018.

[6] W. Zhuang, Q. Ye, F. Lyu, N. Cheng, and J. Ren, "Sdn/nfv-empowered future iov with enhanced communication, computing, and caching," *Proceedings of the IEEE*, vol. 108, no. 2, pp. 274–291, 2020.

[7] T. Das, V. Sridharan, and M. Gurusamy, "A survey on controller placement in sdn," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 472–503, 2020.

[8] T. Kobzan, I. Blöcher, M. Hendel, S. Althoff, A. Gerhard, S. Schriegel, and J. Jasperneite, "Configuration solution for tsn-based industrial networks utilizing sdn and opc ua," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, pp. 1629–1636, 2020.

[9] M. Seliem and D. Pesch, "Software-Defined Time Sensitive Networks (SD-TSN) for Industrial Automation," in *2022 14th International Conference on Computational Intelligence and Communication Networks (CICN)*, pp. 1–7, Dec. 2022.

[10] F. Kurtz, G. Stomberg, and et. al, "Software-Defined Networking driven Time-Sensitive Networking for Mixed-Criticality Control Applications," in *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 99–104, Oct. 2022.

[11] N. S. Bülbül, D. Ergenç, and M. Fischer, "Towards SDN-based Dynamic Path Reconfiguration for Time Sensitive Networking," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–9, Apr. 2022.

[12] G. N. Kumar, K. Katsalis, P. Papadimitriou, P. Pop, and G. Carle, "Failure Handling for Time-Sensitive Networks using SDN and Source Routing," in *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*, pp. 226–234, June 2021.

[13] S. Sudhakaran, C. Hall, D. Cavalcanti, A. Morato, C. Zunino, and F. Tramarin, "Measurement method for end–to–end Time synchronization of wired and wireless TSN," in *2023 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, (Kuala Lumpur, Malaysia), IEEE, May 2023.

[14] S. Sudhakaran, K. Montgomery, M. Kashef, D. Cavalcanti, and R. Candell, "Wireless Time Sensitive Networking Impact on an Industrial Collaborative Robotic Workcell," *IEEE Transactions on Industrial Informatics*, vol. 18, pp. 7351–7360, Oct. 2022.