Soft-W-TSN : extending Time-Sensitive Networking capabilities to Wi-Fi using virtualized elements

Soft-W-TSN: Extending Time-Sensitive Networking Capabilities to Wi-Fi using Virtualized Elements

Gilson Miranda Jr.*†, Jetmir Haxhibeqiri‡, Daniel F. Macedo†, Johann M. Marquez-Barja*
*IDLab - imec, University of Antwerp, Belgium
‡IDLab - imec, Ghent University, Belgium
†Universidade Federal de Minas Gerais - Computer Science Department, Brazil
{gilson.miranda,johann.marquez-barja}@uantwerpen.be, jetmir.haxhibeqiri@ugent.be, damacedo@dcc.ufmg.br

***Abstract*—Industrial processes are becoming increasingly complex, demanding accurate monitoring and timely communication for process correction and optimization actions. Time-Sensitive Networking (TSN) comprises a set of standards being developed by IEEE to enable reliable and bounded low-latency communication over Ethernet networks. These standards allow an operator to have more control over traffic prioritization and latency, and offer features to enhance network communication reliability. Extending such features to wireless networks is a natural step in achieving more flexible deployments and supporting use cases involving mobility. In this work, we present an architecture and proof-of-concept to support extending TSN capabilities to Wi-Fi deployments and to enable flexible deployments of Wi-Fi access points tailored for different use cases. We show the experimental results with a testbed highlighting the main functionalities implemented to support an example use case.**

***Index Terms*—TSN, Network Programmability, SDN**

I. Introduction

The fast-evolving and ever-increasing complexity of industrial processes nowadays requires communication systems for process monitoring, tracing, and control to be more reliable and flexible [1]. To support use cases with critical requirements, the IEEE Time-Sensitive Networking (TSN) Work Group has been developing a wide range of standards to improve the reliability and predictability of Ethernet networks [2]. TSN standards provide a broad set of features such as precise time synchronization, frame scheduling, frame replication and elimination, and management capabilities for TSN networks [3]. To leverage the advantages of wireless networks, many studies have evaluated the extension of TSN features to wireless domains, including Wi-Fi [4].

Wireless TSNs (W-TSNs) are a promising solution for supporting flexibility in industrial use cases that still require reliable low-latency communication. Despite operating on Industrial, Scientific, and Medical (ISM) bands, thus susceptible to interference from other devices, industrial environments are commonly subject to rigorous access control and have policies regarding Access Points (APs) installation, usage of wireless devices in the premises, and user circulation. For example, networks used by machinery can be deployed on different bands than networks provided for personnel and visitors. Unlike industrial wireless technologies such as WirelessHART and ISA100.11a, Wi-Fi (IEEE 802.11) devices supporting TSN can allow the coexistence of Information Technology (IT) and Operational Technology (OT) flows, increasing flexibility and reducing costs.

With a common technology for IT and OT, the infrastructure can be dynamically adapted to support a wider set of use cases. However, many hardware and software limitations hinder the effective use of Wi-Fi for TSN applications. Hardware limitations, such as the absence of a hardware clock or a limited number of hardware packet queues, can impose constraints on time synchronization accuracy, and scheduling capabilities and present challenges in assigning distinct priorities to various data flows. In some cases, the limitations can be overcome in software (e.g., using software queues) at the cost of higher processing times. However, even software components can also have limitations. For instance, the drivers and firmware available for Wi-Fi 6 (IEEE 802.11ax) devices still offer limited access for managing some features [5].

These limitations delay full support and configurability, particularly with open-source software, even many years after the standard and initial commercial devices are released. Virtualized environments provide a flexible evolution path for communication infrastructure, allowing the adoption of mature software without the need for hardware changes. In this paper, we present a proof-of-concept for providing reliable communication over Wi-Fi with TSN-like features, leveraging virtualization and softwarized components. The proposed concept enables flexible deployments of Wi-Fi infrastructures for private industrial use cases using current technology while keeping an open path for supporting features under development by the community.

The paper is organized as follows: Section II provides a brief background on TSN, as well as software and hardware requirements. Section III describes the use case considered in this work. Section IV describes the proposed solution. Section V describes the experimental setup while section VI present achieved results. Section VII concludes the paper and gives some directions for future work.

II. Background

TSN emerged from the multimedia domain through the Audio Video Bridging (AVB) Task Group by IEEE 802.1 in 2007 [6]. While AVB improved IEEE 802.1 networks, further mechanisms were necessary to meet the demands of automotive and industrial applications [3]. The four main

pillars of TSN, as categorized by Lo Bello et al. [3] are i) timing and synchronization; ii) bounded low latency; iii) reliability; and iv) resource management.

The IEEE 802.1AS standard defines a method for time synchronization using the Precision Time Protocol (PTP), such that all nodes have a common time reference [7]. For optimal operation, this requires the nodes to have a nanosecond-scale precision hardware clock to achieve accurate synchronization. The IEEE 802.1Qbv defines a method for scheduling traffic using the Time-Aware Shaper (TAS) to ensure that time-sensitive traffic is scheduled to arrive at its destination at a specific time [8]. The TAS can be used to allocate different priorities to flows and leverage hardware features such as hardware queues and schedule offloading capabilities [9]

Other relevant standards include IEEE 802.1Qcc [10] addressing configuration models and resource management, and IEEE 802.1CB [11] describing a method for frame replication and elimination for redundancy and reliability improvement. For an overview of these and other standards, we refer the reader to the work of Lo Bello et al. [3].

Many software and hardware components need proper configuration and coordination to achieve a reliable TSN network. As standardization is ongoing and new features are expected in the future, quickly integrating these advancements into existing TSNs can provide competitive advantages. In previous work, we introduced an architecture to manage wired/Wi-Fi TSN networks, building on top of an open-source Wi-Fi chip design with added TSN capabilities [12], [13]. In this work, we extend our controller architecture to support Commercial-Off-The-Shelf (COTS) Wi-Fi devices. We leverage virtualization and softwarized components to implement TSN features in a flexible and upgradable manner, aiming to simplify the adoption of functionalities of Wi-Fi 6 devices yet to be implemented, as well as keep up with evolving TSN standards and technologies.

III. Use case example

A Wi-Fi AP is deployed in an industrial plant to support monitoring and control applications (OT flows) but also serve IT flows in a best-effort manner. Many similar APs provide redundant and continuous coverage over the area. During maintenance services, an AP can be redefined, receiving an optimized configuration to support Augmented Reality (AR) applications used by workers (e.g., to get real-time manuals and assembly assistance). To support this wide set of applications, devices using the latest Wi-Fi standards are employed.

These latest technologies often take months or even years to have proper support, especially in open-source software, due to a lack of documentation, openness, or even interest of the manufacturers in providing full-featured drivers. As the technology matures, these functions start to become available due to diligent work of the open-source community, documentation releases from manufacturers, or more interest in improving the support for the product. At this point, integrating such technologies might require extensive updates on already deployed infrastructure, such as kernel version update/customization, and updating related software.

IV. Proposed Solution

The solution builds on our Software Defined Networking (SDN) architecture for TSN networks from our previous work [12]. The architecture follows a Controller/Agent model, with the entities TSN Controller (TSNC) and TSN Agent (TSNA), that was extended to support AP virtualization. With APs deployed as Virtual Machines (VMs), driver, firmware, and other software updates can be extensively tested and tuned on a testing environment, and then deployed in production by replacing the AP VM. The solution leverages *PCI Passthrough* – a feature that allows VMs to directly access a physical PCI device and its full capabilities.

Figure 1 shows the internal components of the TSNC. For conciseness, we left out of this paper the details of each component, and focus on the added elements for supporting the AP deployment using VMs. For a detailed description of the components, we refer the reader to our previous work in reference [12]. The figure highlights the *Image Service*, which manages and stores VMs images of APs. The images are transmitted to the APs during their initialization process. To avoid re-sending the image every time the AP is restarted, the TSNAs cache images locally.

Figure 2 shows the internal components of a TSNA with the additional components for the VM-based AP deployment. The *VM Manager* manages the life-cycle of VMs and the cache of the most recently used ones. The AP can be set up using different Operating Systems (OSs). The diagram in Figure 2 shows our Soft-W-TSN framework, implemented using Hostapd[1], Ubuntu Linux, and the Click Modular Router [14]. Other OSs such as OpenWRT[2] can be used instead. The VM has no control plane connection to the TSNC, thus we define APIs to perform this bridging. The *AP Management API* is part of the TSNA and translates the commands received from the controller into API calls for modules inside the VM (e.g., to manage Hostapd/Click Router modules), through the *AP VM API* (or OpenWRT APIs if that OS is used).

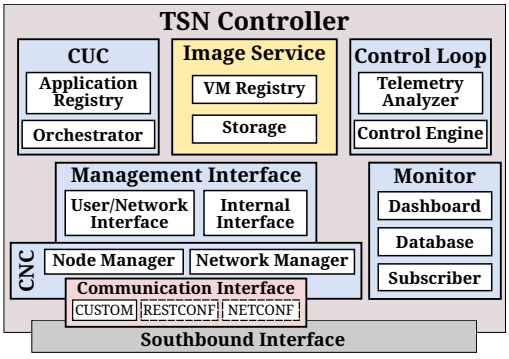

Figure 1: TSN Controller components

[1]https://w1.fi/

[2]https://openwrt.org/

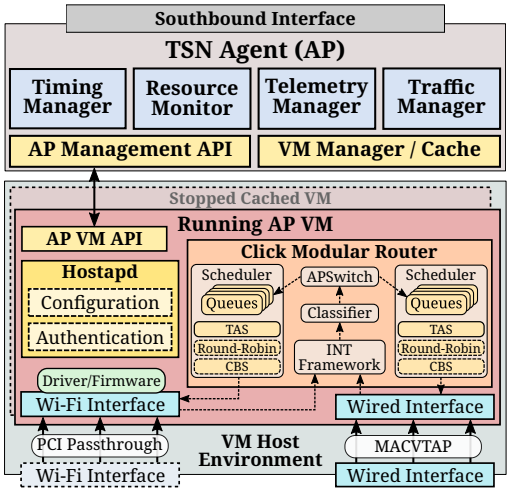

Figure 2: TSN Agent with VM and Soft-W-TSN AP

The Wi-Fi interface is allocated to the VM using *PCI Passthrough*, while the wired interface of the AP is bridged using the MACVTAP driver. We use Click for the bridging between wired and Wi-Fi interfaces in the VM, as well as to implement the traffic shaping modules and In-band Network Telemetry (INT). The dashed lines and arrows detail the flow of packets through Click elements. The packets enter an interface and go to the INT framework, where telemetry data is appended to data packets using extended headers. Later, a classifying element uses the DSCP field to determine in which queue the packet must be placed. The *APSwitch* defines to which interface packets must go and sends them to the *Scheduler* element. The *Scheduler* has internal queues to buffer the packets and has different scheduling algorithms that can be selected. The module supports *Round Robin*, the Credit-Based Shaper (CBS) algorithm (based on IEEE 802.1Qav), and the TAS algorithm based on IEEE 802.1Qbv, which is be the focus of our experiments.

V. Experimental Setup

The topology used for this Proof-of-Concept (PoC) is shown in Figure 3. SW1 and SW2 are TSN switches based on industrial mini PCs, the other nodes are Intel NUCs. The AP and CLI are equipped with Sparklan WNFT-238AX(BT) Wi-Fi 6 boards. All nodes, including the AP VM, run Ubuntu Server 22.04, and Linux Kernel 6.4 with real-time patches. The AP runs Hostapd v2.11-devel. The switches and AP run the TSNA and are connected to the TSNC, allowing us to apply TAS schedules to node interfaces.

The *PC* generates 8 simultaneous UDP flows to *CLI*, which is a Wi-Fi client, passing through two TSN switches (*SW1* and *SW2*) and the *AP*. Each flow is allocated to a different queue of the scheduler. The packets are captured at the egress interface of the PC, and at the ingress Wi-Fi interface of CLI. To correctly verify the delivery of the packets according to the schedule applied to the AP, the CLI is synchronized via PTP to SW2 through Ethernet, however, all data packets flow

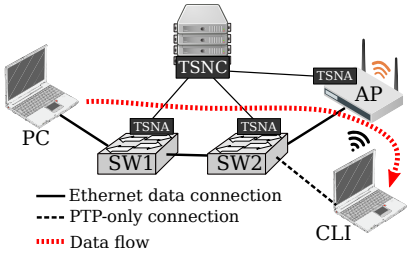

Figure 3: Topology for the experiments

through the Wi-Fi connection. We deploy 3 versions of the AP VM, the first operating as Wi-Fi 4, the second as Wi-Fi 5, and the last one as Wi-Fi 6.

The experiments show the traffic differentiation capability and performance of the scheduling module across different Wi-Fi generations. We used an 8-slot schedule (with fixed durations, although varying durations are supported), allocating one slot for each virtual queue. Slot durations ranged from 10 ms to 100 μs. We generated UDP traffic in both unsaturated (at 1000 packets per second (pps) per flow) and saturated conditions (around 4500 pps per flow, the maximum generation capacity of our node), with a 36-byte data payload containing sequence numbers and timestamps. We collected one million packets at the receiver for statistical analysis.

VI. Results

We assessed the capacity packet delivery of the system according to the TAS configuration. By comparing packet timestamps at the Wi-Fi client to their scheduled times, we counted the number of slot mismatches. Some mismatch is expected due to factors such as hardware buffering and processing at lower layers, causing packets to arrive slightly after the slot ends.

Slot (ms)	Slot mismatch ratio (%)					
	Unsaturated			Saturated		
	n	ac	ax	n	ac	ax
10.0	0.54	0.51	**0.43**	0.39	0.56	**0.29**
5.0	1.25	1.52	**0.88**	0.75	1.87	**0.55**
2.5	4.41	1.89	**1.66**	3.44	3.99	**1.11**
1.0	12.72	**4.03**	10.43	12.22	13.89	**3.78**
0.5	19.14	13.15	**11.61**	48.85	**38.64**	54.81
0.25	69.67	**55.08**	63.95	90.86	**79.28**	83.07
0.1	95.24	96.12	**94.63**	93.34	95.15	**92.44**

Table I: Slot delivery mismatch ratio for Wi-Fi running on modes *n*, *ac*, *ax*, for unsaturated and saturated traffic.

Table I shows the mismatch percentage according to slot sizes, the Wi-Fi standard enabled, and the traffic type. The table clearly shows the advantage of using *IEEE 802.11ac* and *ax* over operating in *n* mode. As the recent standards allow faster PHY layer transmissions, the packets are transmitted faster after the software-based queuing/scheduling done by Click. In general, lower mismatch ratios were obtained by using the *ax* standard, however, we still observed significantly lower mismatches by using *ac* with saturated traffic, for slots of 500 μs and 250 μs. During preliminary tests, we

achieved higher throughput between the nodes when the AP was configured in *ac* mode, and this might have been reflected in the observed performance for saturated traffic.

We observe a steep mismatch ratio increase for slots shorter than 1 ms. With shorter slots, packets from different flows are sent to lower layers in shorter intervals. These packets are again queued and subject to the CSMA mechanism. The higher throughput allowed by the AP in modes *ac* and *ax* reduces the packet buffering in lower network layers and results in more accurate scheduling.

Figure 4 shows a boxplot of the arrival times of packets within the schedule cycle, for the *saturated* tests with slots of 2.5 ms. Each flow (indicated by *Flow ID*) was assigned to a different slot (i.e., Flow ID 0 is assigned to the first slot of 2.5 ms). Thus, the figure shows a cycle of 20 ms and each box indicates the distribution of packets of a Flow ID through the cycle. The distributions show that most packets are delivered within the assigned slot. Table I shows that for this test 1.11 % of the packets were delivered out of their slots when using *ax* mode. The figure allows us to visualize how the delivery times of packets from the different flows are spread across the schedule cycle, highlighting the gains achieved by coupling our TAS-based scheduler with the AP in *ax* mode.

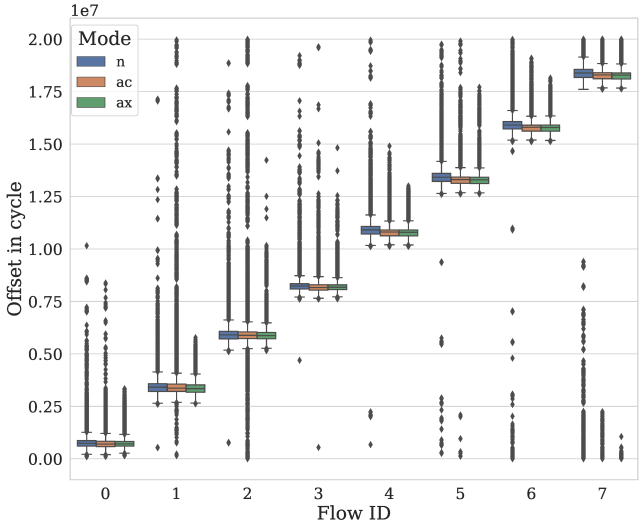

Figure 4: Packet delivery times for the different modes on a schedule with 8 slots of 2.5ms

VII. Conclusion

In this work, we presented an architecture and test results from a PoC using virtualization and softwarized components to flexibly implement TSN capabilities on COTS Wi-Fi devices. We evaluated the benefits of deploying upgraded versions of the AP, enabling more functionalities of our Wi-Fi interface, as an example of how the architecture can leverage software upgrades for cutting-edge technologies. In future work. we will study extending the VMs to run edge applications and provide functions other than operating as an AP.

Acknowledgment

This research is partially funded by the imec ICON project VELOCe - VErifiable, LOw-latency audio Communication (Agentschap Innoveren en Ondernemen project nr. HBC.2021.0657). This research is also supported by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, and São Paulo Research Foundation (FAPESP) with Brazilian Internet Steering Committee (CGI.br), grants 2018/23097-3 and 2020/05182-3.

References

[1] P. A. M. Devan, F. A. Hussin, R. Ibrahim, K. Bingi, and F. A. Khanday, "A survey on the application of wirelesshart for industrial process monitoring and control," *Sensors*, vol. 21, no. 15, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/15/4951

[2] J. L. Messenger, "Time-Sensitive Networking: An Introduction," *IEEE Communications Standards Magazine*, vol. 2, no. 2, pp. 29–33, jun 2018. [Online]. Available: doi.org/10.1109/mcomstd.2018.1700047

[3] L. Lo Bello and W. Steiner, "A Perspective on IEEE Time-Sensitive Networking for Industrial Communication and Automation Systems," *Proceedings of the IEEE*, vol. 107, no. 6, pp. 1094–1120, 2019. [Online]. Available: doi.org/10.1109/JPROC.2019.2905334

[4] J. Haxhibeqiri, X. Jiao, M. Aslam, I. Moerman, and J. Hoebeke, "Enabling TSN over IEEE 802.11: Low-overhead time synchronization for Wi-Fi clients," in *ICIT2021, the 22nd International Conference on Industrial Technology*, 2021, pp. 1068–1073. [Online]. Available: doi.org/10.1109/ICIT46573.2021.9453686

[5] E. Mozaffariahrar, F. Theoleyre, and M. Menth, "A survey of wi-fi 6: Technologies, advances, and challenges," *Future Internet*, vol. 14, no. 10, 2022. [Online]. Available: https://www.mdpi.com/1999-5903/14/10/293

[6] N. Finn, "Introduction to Time-Sensitive Networking," *IEEE Communications Standards Magazine*, vol. 2, no. 2, pp. 22–28, 2018. [Online]. Available: doi.org/10.1109/MCOMSTD.2018.1700076

[7] "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. [Online]. Available: doi.org/10.1109/IEEESTD.2020.9121845

[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. [Online]. Available: doi.org/10.1109/IEEESTD.2016.8613095

[9] T. L. D. Project, "Linux iproute2 tc-taprio man page," Man page, 2018, accessed on July 20, 2023. [Online]. Available: https://man7.org/linux/man-pages/man8/tc-taprio.8.html

[10] "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. [Online]. Available: doi.org/10.1109/IEEESTD.2018.8514112

[11] "IEEE Standard for Local and metropolitan area networks–Frame Replication and Elimination for Reliability," *IEEE Std 802.1CB-2017*, pp. 1–102, 2017. [Online]. Available: doi.org/10.1109/IEEESTD.2017.8091139

[12] G. Miranda, E. Municio, J. Haxhibeqiri, J. Hoebeke, I. Moerman, and J. M. Marquez-Barja, "Enabling time-sensitive network management over multi-domain wired/wi-fi networks," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2023. [Online]. Available: https://doi.org/10.1109/TNSM.2023.3274590

[13] X. Jiao, W. Liu, M. Mehari, M. Aslam, and I. Moerman, "openwifi: a free and open-source IEEE802.11 SDR implementation on SoC," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, 2020, pp. 1–2. [Online]. Available: https://doi.org/10.1109/VTC2020-Spring48590.2020.9128614

[14] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The Click modular router," *ACM Transactions on Computer Systems (TOCS)*, vol. 18, no. 3, pp. 263–297, 2000. [Online]. Available: https://doi.org/10.1145/354871.354874