On Integrated Wired and Wireless Time-Sensitive Networking using SDN

Mehdi Nobakht
School of Systems and Computing
University of New South Wales, Canberra, Australia
mehdi.nobakht@unsw.edu.au
https://orcid.org/0000-0003-4676-536X

Frank den Hartog
School of Systems and Computing
University of New South Wales, Canberra, Australia
frank.den.hartog@unsw.edu.au
https://orcid.org/0000-0001-5293-6140

***Abstract*—IEEE Time-Sensitive Networking (TSN) Task Group has standardized enhancements for Ethernet networks to ensure high reliability, deterministic latency and minimal jitter for time-critical traffic. However, TSN capabilities have not yet been extended to wireless standards. We discuss the inherent characteristics of wireless communications that currently limit TSN performance. We argue for the development of a framework for an integrated wired and wireless TSN using Software-Defined Wireless Networking (SDWN) to address this gap. Our approach focuses on addressing the mobility of wireless stations by dynamically adjusting the wireless channel bandwidth through SDWN. This adaptive approach takes into account the proximity of wireless stations to the access point and configures the wireless channel bandwidth to meet the stringent latency requirements of time-sensitive wireless applications. In this paper, we sketch a general architecture for the framework. Furthermore, we present the development of a TSN-enabled network emulator testbed, leveraging a suite of open-source tools including Mininet, Mininet-WiFi, a modified version of Open Virtual Switch (OVS), Linux TAPRIO, RYU SDN controller, and SDWN controller. Our experimental evaluation demonstrates an average one-way latency of approximately 14 µs, achieved through the utilization of a modified version of the OVS software switch.**

I. Introduction

Time-sensitive systems, such as vehicular networks, require precise time synchronization, deterministic end-to-end latency and jitter, and extremely low packet loss. Time-critical flows, dedicated by several real-time constraints, often have stringent timing and performance requirements. In addition to time-critical traffic, there is also non-critical (best-effort) traffic that can negatively impact the performance of critical messages. Ethernet networks were not designed to support the transport and routing of time-critical traffic alongside best-effort traffic. To address this, Time-Sensitive Networking (TSN) emerged as a networking technology that can provide precise time synchronization, timeliness, and preemption in Ethernet networks. The IEEE TSN Task Group has developed a set of standards to provide bounded latency, low jitter, and guaranteed high reliability, while avoiding interference from best-effort traffic on Ethernet-based communications.

Ethernet-based (wired) Local Area Networks (LANs) may not be suitable for certain applications as they do not scale well with large numbers of hosts, and do not support mobility requirements. Additionally, the installation and maintenance of wired networks can be challenging and often impose significant burdens. Thus, there is a strong motivation to use wireless communications such as IEEE 802.11 Wi-Fi, Cellular technologies (4G, 5G, and beyond), and wireless technologies for IoT applications (ZigBee, LoRaWAN, and NB-IoT). Modern time-sensitive applications, such as industrial automation and IoT, are expected to leverage both wired and wireless communications to enable greater flexibility, mobility, and efficiency.

Most TSN standards are limited to Ethernet-based networks and do not support wireless access networks. While a few TSN capabilities have been extended over Wi-Fi 802.11, including accurate time synchronization, traffic identification, and stream reservations, mechanisms for controlling congestion and achieving deterministic latency and low packet loss in wireless networks have not yet been explored. Extending TSN capabilities to wireless environments, such as Wi-Fi (IEEE 802.11) and cellular, is an open research problem.

Extending TSN capabilities over the wireless medium is a significant research challenge due to the unique physical properties of wireless channels, such as variable capacity and higher packet loss caused by radio-frequency interference and the stochastic nature of the channel. To improve wireless link reliability, various techniques can be employed, such as adjusting transmit power, using redundant paths, and adapting appropriate modulation and coding schemes based on channel conditions. However, these measures may increase interference or latency. Additionally, channel access procedures are a major source of delay in most wireless systems, which allows multiple users to share the medium.

We advocate for the establishment of a framework that seamlessly integrates wired and wireless Time-Sensitive Networking (TSN), aligning with the IEEE TSN standards. Our central focus is to ensure bounded latency and mitigate jitter, particularly in the context of station mobility within wireless environments. We assume that the influence of other wireless characteristics on the evaluated performance metrics is minimal. To address these objectives, we propose a dynamic adjustment of the wireless channel bandwidth, by considering the proximity of wireless stations to the access point.

The proposed framework leverages a centralized Software-Defined Networking (SDN) controller to implement a control mechanism for TSN capabilities through three steps: topology

978-3-903176-59-1 ©2023 IFIP

Table I: IEEE 802.1 TSN Standards

IEEE Standard	Area of definition
1588, 802.1AS	Timing and synchronization
802.1Qca	Path control and reservation
802.1Qbv	Time-aware scheduling
802.1Qbu & 802.3br	Frame preemption
802.1Qcc	Central configuration method
802.1Qci	Time-based ingress policing
802.1CB	Frame replication and elimination

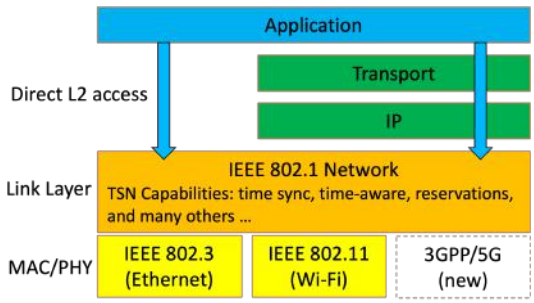


Figure 1: TSN Reference Protocol Stack [1]

discovery, scheduling based on TSN flow requirements, and schedule distribution. Furthermore, we employ a Software-Defined Wireless Networking (SDWN) controller to enable programmability of wireless parameters, facilitating the dynamic configuration of the wireless network, including adjustments to channel bandwidth.

Deployment and evaluation of TSN frameworks often requires the use of dedicated and costly hardware solutions. In contrast, this paper describes a flexible and straightforward testbed to evaluate the proposed framework. To this end, we have developed a testbed prototype, employing Mininet and Mininet-WiFi network emulators, a modified version of Open vSwitch (OVS), and Linux Time-Aware Shaper (TAS) toolset. The testbed achieves an average one-way latency of approximately 14 µs for a TSN UDP stream with 1518-byte payload and a 10 ms inter-packet gap. This work represents a step towards developing a testbed using existing open-source tools as a platform to study the performance of TSN networks, both in general and within the context of wireless TSNs.

II. Background on Time Sensitive Networking

The IEEE 802.1 TSN standards aim to deliver data flows with very low packet loss rates and bounded end-to-end latency and jitter over IP networks. The IEEE TSN working group is developing Ethernet-based TSN standards that provide deterministic communications, including time synchronization, traffic scheduling, and bandwidth reservation. A summary of a subset of the TSN standards and definitions can be found in Table I. Below, we will provide a brief overview of the key aspects relevant to this work.

IEEE 802.1Qcc is a centralized paradigm that manages and controls the network globally. It enhances the existing decentralized mechanism for flow reservation and stream management by having flow requirements sent from a Centralized User Configuration (CUC) to a Centralized Network Configuration (CNC). In this fully centralized model, the CNC is responsible for the configuration and control of the TSN switches, while the CUC is responsible for the endpoints (Talkers/Listeners). The CUC communicates with the CNC using the User Network Interface (UNI) for routing and reservation.

While the majority of TSN standards are designed for Ethernet as the transport medium, TSN can be extended to support other types of transport media as well. Figure 1 illustrates a TSN protocol stack with three physical layer media including Ethernet, Wi-Fi, and cellular.

Some TSN functionalities have been extended to IEEE 802.11, such as time synchronization (timing measurement capability in 802.11-2012), stream reservation (SRP over 802.11 for Audio and Video defined in 802.11aa), and 802.11 links in an 802.1Q network (802.11ak). However, other QoS features to control congestion by reducing latency and jitter have not been explored. Therefore, further research is needed to address the worst-latency and reliability requirements. It is worth noting that it is not practical to meet TSN-grade performance in wireless TSN due to the characteristics of wireless environments.

III. Related Work

Numerous studies have explored diverse facets of TSN standards, encompassing topics like time synchronization, scheduling, routing, and protocols for congestion control.

Several studies have examined techniques and challenges related to scheduling and routing in time-sensitive SDN. Nayak et al. [2] proposed routing and scheduling algorithms for a Time-Sensitive SDN (TSSDN) that leverages the global view of the control plane on the data plane to schedule and route time-critical flows. Another approach by Said et al. [3] implements the fully centralized IEEE 802.1Qcc model using SDN, and explores how SDN capabilities can speed up the process of updating new flows or configuring a TSN network.

Several surveys have explored the impact of wireless protocols, such as 5G and unlicensed spectrum, on latency in TSN [4]. Studies have also looked into the problem of low latency resource allocation and scheduling for wireless control systems in 5G and Wi-Fi networks [5]. In addition, [6] presents a work that integrates IEEE 802.1AS synchronization in IEEE 802.11 for wireless TSN.

Cavalcanti et al. [7] provide an overview of the potential applications, requirements, and unique research challenges of extending TSN capabilities over wireless networks, particularly for industrial automation systems. Susruth et al. [8] utilize wireless TSN features to enable low-latency wireless communications in an industrial collaborative robotics scenario. They implement a TSN control plane GCL on a Wi-Fi network stack using Linux kernel tools on the nodes. In our work, we do not limit ourselves to a specific application, and instead aim to

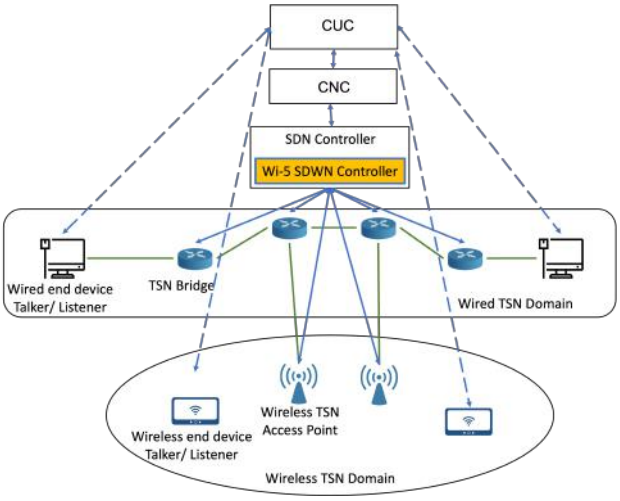


Figure 2: System Architecture

develop a wireless TSN paradigm that can be applied to a broader range of fields.

IV. An Integrated Approach for Wired and Wireless TSN

Figure 2 illustrates an overview of our proposed architecture for extending wired TSN domain standards to wireless TSN domain. Our integrated wired and wireless TSN model is based on the management model for wired TSN as defined in IEEE 802.1Qcc. We assume an integrated wired and wireless SDN composed of multiple TSN-enabled bridges and several wired and wireless talker/listener pairs, generating and receiving time-critical traffic. The green lines in Figure 2 represent data plane communication. Control plane communication is represented by solid blue lines for switches and dashed blue lines for managing traffic flows on end hosts, which include talkers and listeners.

In our approach, we adopt a fully centralized TSN control plane architecture as presented in IEEE 802.1Qcc to support QoS capabilities such as congestion control by reducing latency and jitter, and increasing reliability. CUC and CNC are the two main control plane components. In our architecture, CUC and CNC act as SDN applications. CUC is responsible for discovering end devices, retrieving their capabilities and user requirements, and configuring TSN flows on the end devices. CNC performs the appropriate TSN routing and scheduling updates. CNC uses remote management to discover the physical topology, retrieve bridge capabilities, and configure TSN features/resources on each bridge. The functionality of CUC can also be integrated into CNC. The SDN controller is responsible for traditional network configuration and monitoring, thus configuring the bridges in normal network conditions.

To extend TSN features to wireless networks, we utilize SDWN controller to configure radio parameters such as wireless channel bandwidth to meet time-critical flow requirements. We propose using an intelligent application running on the SDWN controller to provide deterministic access to wireless channels and reduce latency by increasing wireless channel bandwidth for time-critical flows and limiting best-effort traffic bandwidth.

As there is a correlation between the distance of wireless stations from the access point and the required wireless channel bandwidth for time-critical flows, we have developed an algorithm that monitors the positions of wireless stations and calculates their distances from the Wi-Fi access point. Based on these measurements, the algorithm generates the necessary channel bandwidth to meet the requirements of TSN. By dynamically adapting the channel bandwidth based on wireless station positions, we aim to achieve bounded latency for time-critical applications in wireless networks.

Before talkers and listeners can start exchanging TSN flows, several steps are required to provision a TSN path. We describe a workflow of these steps to show how the entities communicate with each other.

Topology discovery. First, the CUC requests the CNC to discover the physical network topology. Learning the physical topology is a necessary step in preparing for schedule computation. The SDN paradigm, with its logically centralized controller, is aware of the entire network and has a complete view of the network and its topology. Upon discovering the topology, the CNC knows how the end devices are connected and will inform the CUC. Any changes in the topology due to link failure, node addition, or removal is handled by the SDN controller, which updates the CNC accordingly to take the necessary action.

Provisioning the required network resources. The CUC determines which end device (talker) intends to communicate with other end devices (listener/s). It then defines the latency requirements for the communication, the maximum size of the Ethernet packet that will be sent, and other dependencies.

Computing schedule. In the next step, the CUC initiates a request to the CNC to compute the schedule with consideration of the discovered topology and TSN flow requests. The CNC returns a response to the computation request indicating success or failure of the scheduling logic. The computational requirements can be complex and may require a detailed knowledge of the application within each end device.

Confirming the computation result. After the schedule has been computed, the CNC sends the details to the CUC. This information includes the details for each end device and bridge involved in TSN flows, as well as the necessary information for configuring the flows. This includes information for each end device and bridge involved in the TSN flows, such as unique flow identifiers, transmission and reception window times at each hop, and end-to-end latency. It is important to note that when transmitting a TSN flow, the talker is given a window in which to transmit.

Distributing the schedule. Once the CUC confirms that the schedule will work and meets the requirements, it issues a request to the CNC to distribute the computed schedule to the TSN bridges. The CUC will also inform the talkers and

listeners for the TSN flows. The talkers are then responsible for transmitting every TSN flow according to the agreed schedule.

V. Implementation

We implemented the proposed scheme for an integrated wired and wireless TSN environment using Mininet and Mininet-WiFi network emulators and a software TSN switch with Linux support for TAS. Mininet-WiFi [9] is a fork of Mininet which allows the using of both WiFi Stations and Access Points. In the virtualized environment, we used OVS instances for traffic forwarding between the talker and listener hosts and the TSN switches. The RYU SDN controller was used to program the OVS switches.

Note that when we use network emulators, the clocks of all virtualized TSN switches are synchronized, so all switches refer to the same cycle base time with their schedules. As a result, we do not need to implement PTP to synchronize clocks in network devices.

To support TSN scheduling functionality, we used the Time-Aware Priority Shaper (TAPRIO) Qdisc on the egress ports of the OVS in the Mininet environment. TAPRIO is the Linux queuing discipline (Qdisc) that implements TAS as a simplified version of IEEE 802.1Qbv. In more detail, TAS uses a gate behind each queue. When the gate of a queue is open, the first packet in the queue is allowed to be transmitted. The time schedule in the GCL determines whether a gate is open or closed. Each entry in the GCL has a timestamp that defines the time when the state of the gates should change to a given state. TAPRIO allows for the configuration of GCL entries.

TAPRIO requires a multi-queued network interface. To enable multiple transmission queues on Virtual Ethernet pairs, we have modified Linux kernel 5.2.16. This allows us to use multi-queued Virtual Ethernet devices inside Mininet.

The configuration of Linux TAPRIO Qdisc is done using the Linux *traffic control* tool (TC). In order to define how to assign packets to traffic classes, TAPRIO uses the priority field of the Linux socket buffer structure. Since the socket buffer structure is an internal kernel data structure for managing packets, it cannot be directly set from user space. We leverage the Priority Code Point (PCP) field of the VLAN tag to classify traffic into TSN and non-TSN traffic classes. By utilizing this field, we assign appropriate priorities to different types of traffic. Additionally, we have made modifications to OVS to enable it to read the VLAN tag and set the socket buffer priority right before the packet reaches the QDISC.

To facilitate SDWN control, we utilize an extended SDN controller designed to enable the programming of radio parameters through an open Southbound API for wireless access points, as well as an open Northbound API for the Centralized Network Controller (CNC). By employing such SDWN, we gain the ability to program Wi-Fi access points and configure bandwidth settings as outlined in IV. Furthermore, we have implemented an agent based on [10] that provides programmability for a range of Wi-Fi parameters at the physical, MAC, and Link-Layer levels. This agent operates in parallel with OpenFlow and utilizes a specifically developed protocol to accomplish its tasks effectively. With this setup, we achieve enhanced control and programmability over various aspects of the Wi-Fi network, enabling us to optimize its performance and meet TSN requirements.

VI. Preliminary Evaluation

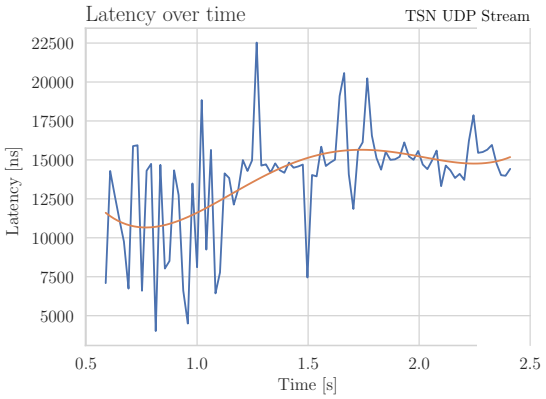
Our ultimate goal is to evaluate the performance of converged wired and wireless TSNs. With that in mind, the purpose of this preliminary evaluation is to assess the performance of the testbed we developed utilizing a suite of readily available open-source tools and off-the-shelf solutions. The outcomes of this evaluation will provide valuable insights into overcoming the challenges associated with the integration of wired and wireless TSNs.

We implemented a prototype of the proposed architecture in Mininet and Mininet-WiFi network emulators as explained in Section V. As previously discussed, the Linux Kernel was modified to support multi-queued Virtual Ethernet. The prototype system included the RYU SDN controller version 4.34 and Mininet version 2.3.0 and Mininet-WiFi.

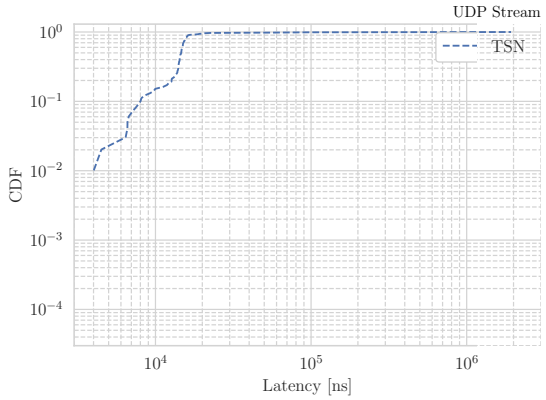
We built a testbed topology consisting of four OVS TSN-enabled software bridges between the talker and listener pairs. The capacity of all links was set to 1 Gbps. The experiment used two sets of flows: ***i***) time-critical traffic involves the transmission of UDP packets within a 1518-byte Ethernet frame at a rate of 100 packets per second and ***ii***) best-effort traffic of UDP packets with 1518 bytes Ethernet frame. In our findings, we detail the effects observed when utilizing a customized version of OVS and TSN TAPRIO scheduling on latency and jitter. These evaluations were conducted within a 10 ms cycle time while systematically varying the ratio between time-critical and best-effort traffic.

It is important to note that, in general, the average latency of a virtual TSN switch is expected to be higher than for a physical TSN switch due to the overhead incurred by the software implementation of the switch functionality. We evaluated the latency of a GCL entry with a time-sensitive traffic duration of 7.5 ms, followed by 2.5 ms of best effort traffic within a 10 ms cycle. In our experiments, we emulated the network using both Mininet and Mininet-WiFi. To ensure reliability and accuracy, we conducted each experiment 10 times, and present the average outcomes derived from these multiple iterations. Figure 3 illustrates the one-way latency over time, alongside its Cumulative Distribution Function (CDF), for the mentioned UDP stream within the Mininet emulator, as observed in one experimental run. Furthermore, Table II provides statistical insights into one-way latency, including payloads of diverse sizes ranging from 64 bytes to 1518 bytes. Note that these reported results represent the averages derived from 10 distinct experiment runs.

Furthermore, we examined the impact of a GCL entry with 5 ms allocated to both time-sensitive and best-effort traffic. As a result, we observed an increase in the average delay for time-critical traffic, reaching 15.5 µs in Mininet. This experiment shows that *even when the background (best-*



(a) Latency over time



(b) CDF graph

Figure 3: Latency for a one-way TSN UDP stream 1518B.

Table II: Comparison of latency measured for TSN UDP stream with data rate of 100 packets per second.

Frame Size	Latency [µs]		
	low	mean	peak
1518B	4.013	14.636	22.580
1024B	4.179	17.968	23.872
512B	4.653	15.271	22.586
256B	4.373	12.831	21.724
64B	3.182	17.025	19.364

effort) traffic saturates the link, time-critical traffic continues to guarantee deterministic latency.

VII. Conclusion

We have argued that wireless TSN is an important and rich area for future research, highlighting the challenges in extending TSN capabilities to wireless networks. We proposed an architecture for integrating wired and wireless TSN using SDN and SDWN. To extend TSN features to wireless medium during mobility of stations, we utilized an algorithm that dynamically adapts the channel bandwidth based on wireless station positions in order to guarantee bounded latency of time-critical applications in wireless networks.

Furthermore, we have developed a prototype of testbed for implementing TSN in an integrated wired and wireless network and reported the implementation results using Mininet, a modified version of OVS, and the Linux traffic shaper TAPRIO. Our observations reveal an average one-way latency of approximately 14 µs for a TSN UDP stream featuring a 1518 bytes frame size, operating at a data rate of 100 packets per second. These results were achieved through the implementation of a customized version of the OVS software switch. Building upon the insights gained from our preliminary evaluation of the developed testbed, we plan to delve into a detailed examination of the algorithm's performance, with a specific focus on guaranteeing bounded latency within the dynamic context of station mobility. Furthermore, we intend to expand our research scope and consider additional wireless characteristics, including the impact of radio-frequency interference and the stochastic nature of the channel.

Acknowledgment

This work is partly funded by the Australian National Next Generation Technology Fund under agreement number ID 10385.

References

[1] D. Cavalcanti, M. Rashid, G. Venkatesan, L. Cariou, and R. Stacey. (2018) Time-Aware Traffic Shaping Over 802.11. [Online]. Available: https://imat.ieee.org/802.11/dcn/18/11-18-1542-00-0000-time-awaretraffic-shaping-over-802-11.pptx

[2] N. G. Nayak, F. Dürr, and K. Rothermel, "Incremental Flow Scheduling and Routing in Time-Sensitive Software-Defined Networks," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 5, pp. 2066–2075, 2018.

[3] S. B. H. Said, Q. H. Truong, and M. Boc, "SDN-Based Configuration Solution for IEEE 802.1 Time Sensitive Networking (TSN)," *SIGBED Reveiw*, vol. 16, no. 1, pp. 27–32, February 2019.

[4] 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, 2019.

[5] J. Li, S. K. Bose, and G. Shen, "Cooperative Resource Scheduling for Time-Sensitive Services in an Integrated XGS-PON and Wi-Fi 6 Network," *IEEE Communications Letters*, vol. 26, no. 6, pp. 1338–1342, 2022.

[6] A. M. Romanov, F. Gringoli, and A. Sikora, "A Precise Synchronization Method for Future Wireless TSN Networks," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 5, pp. 3682–3692, 2021.

[7] D. Cavalcanti, J. Perez-Ramirez, M. M. Rashid, J. Fang, M. Galeev, and K. B. Stanton, "Extending Accurate Time Distribution and Timeliness Capabilities Over the Air to Enable Future Wireless Industrial Automation Systems," *Proceedings of the IEEE*, vol. 107, no. 6, pp. 1132–1152, 2019.

[8] 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, no. 10, pp. 7351–7360, 2022.

[9] R. R. Fontes, S. Afzal, S. H. B. Brito, M. A. S. Santos, and C. E. Rothenberg, "Mininet-WiFi: Emulating software-defined wireless networks," in *2015 11th International Conference on Network and Service Management (CNSM)*, 2015, pp. 384–389.

[10] J. Schulz-Zander, L. Suresh, N. Sarrar, A. Feldmann, T. Hühn, and R. Merz, "Programmatic Orchestration of WiFi Networks," in *2014 USENIX Annual Technical Conference (USENIX ATC 14)*. Philadelphia, PA: USENIX Association, Jun. 2014, pp. 347–358.