Software Defined Wireless Networks Considering QoS

Redes inalámbricas definidas por software que consideran la calidad del servicio (QoS)

Dainier González Romero , Rodrigo Santos
Departmento de Ingeniería Eléctrica y Computadoras, Universidad Nacional del Sur (UNS), San Andres 800, Bahia Blanca, 8000, Argentina. Instituto de Ciencias e Ingeniería de Computación, UNS-CONICET, San Andres 800, Bahia Blanca, 8000, Argentina.

DOI *https://doi.org/10.59471/raia2025228*
Enviado: junio 2025. **Aceptado**: octubre 2025. **Publicado:** diciembre 2025
Como citar: González Romero, D. & Santos, R. (2025). Redes inalámbricas definidas por software que consideran la calidad del servicio (QoS). Revista Abierta De Informática Aplicada, 9(1). *https://doi.org/10.59471/raia2025228*

Resumen

Varios sistemas IoT demandan comunicación en tiempo real, lo que introduce restricciones temporales en la transmisión de datos y ejerce presión sobre la propagación de los mensajes en la red. Muchos de estos sistemas deben abordar estos requerimientos de comunicación considerando el uso de redes inalámbricas, lo que aún representa un problema abierto. Para afrontar este escenario de comunicación, es fundamental emplear estrategias de configuración dinámica que puedan adaptarse rápidamente al comportamiento de la red, garantizando la estabilidad y previniendo fallas bajo determinadas condiciones operativas. Por lo tanto, resulta necesario recurrir a mecanismos para implementar redes definidas por software (SDN), considerando la comunicación inalámbrica en tiempo real, a fin de dar soporte a estas aplicaciones.

Este trabajo se basa en investigaciones previas de los autores y muestra cómo implementar redes definidas por software que se adapten dinámicamente a los requerimientos de tráfico en tiempo real en una red inalámbrica. El modelo de red fue implementado y evaluado utilizando el simulador NS-3. Los resultados experimentales demuestran que la incorporación de políticas SDN en redes inalámbricas mejora la predictibilidad de estos sistemas. Las bibliotecas implementadas en NS-3 fueron publicadas y puestas a disposición de investigadores y desarrolladores, quienes pueden utilizarlas para modelar y evaluar redes inalámbricas definidas por software específicas.

PALABRAS CLAVES: Redes inalámbricas definidas por software, calidad de servicio en sistemas IoT, soporte de comunicación, gestión del tráfico de mensajes.

Abstract

Several IoT systems demand real-time communication, introducing timing constraints on data transmission and stressing the network message propagation. Many of these systems should address these communication requirements, considering the use of wireless networks, which still represents an open issue. To address this communication scenario, it is crucial to employ dynamic configuration strategies that can swiftly adapt the network behavior, ensuring stability and preventing failures under certain operational conditions. Therefore, to rely on mechanisms to implement software-defined networks (SDN), considering wireless real-time communication, is necessary to support these applications. This paper builds on the author's previous work and shows how to implement software-defined networks that dynamically accommodate the real-time traffic requirements in a wireless network. The network model was implemented and evaluated using the NS-3 simulator. The experimental results demonstrate that incorporating SDN policies into wireless networks enhances the predictability of these systems. The implemented NS-3 libraries were made public and available for researchers and developers. They can utilize these libraries to model and evaluate specific software-defined wireless networks.

KEYWORDS: Software-Defined Wireless Networks, IoT Systems Quality of Service, Communication Support, Management of Message Traffic.

INTRODUCCIÓN

Systems that support smart environments, usually IoT systems, require processing data to obtain information and acquire knowledge of the current situation. Based on it, these applications make decisions when needed. This decision- making process requires that data be transmitted on time among the components of an IoT system, and processed under real-time constraints Santos, Santos, and Orozco (2005). Precision farming, Industry 4.0/5.0, smart cities, environmental monitoring, and smart health are examples of application domains where this type of communication is typically required.

Wireless Sensor Networks (WSN) have regained interest with the advent of IoT. They have a hierarchical structure in which the sensors or actuators deployed in the field are linked to the Internet through a gateway or access point, depending on the type of technology used. In the case of industrial applications, most of the communication standards used are wired (CAN, PROFIBUS/NET, and FIELDBUS ISO (2024); IEC (2023)). In other domains like smart cities or precision farming, wireless protocols like ZigBee, LoRaWAN, Sigfox, or IEEE 802.11 are generally used. Real-time traffic is not easily handled in the wireless domain. However, in the case of IEEE 802.11, there are some fields within the header that are not used, and in which the version IEEE 802.11e has implemented four different types of traffic associated mainly with multimedia transmissions. In this paper, the authors adapt the use of these fields to indicate four different priorities that can be used by a Software Defined Network (SDN) controller to handle traffic in a differentiated way.

IoT can also be implemented on 5G technology, and in fact, there are several applications where 5G is the preferred technology. For example, in the Internet of Vehicles (IoV). Although

5G is rapidly becoming the standard technology for mobile phones and mobile internet access, there are still large areas and many countries that have a long way to go to achieve the necessary coverage. While wireless protocols, such as the ones mentioned before, used the IMS (Industrial, Medical and Science) electromagnetic bandwidth, which is not licensed, 5G operates under government regulations in authorized spectrum. Although they have many differences, advances in 5G technology have been adopted in the wireless standards realm: modulation techniques and the possibility of simultaneous multiple-user transmissions (MIMO) are the most important. These are known as Wifi 6 and 7 and more generally as the Wifi Legacy. 5G uses SDN to optimize routing and resources by moving traffic between radio bases Xu (2025); also in Rahman, Wadud, Islam, Kundu, Bhuiyan, Muhammad, and Ali (2024), the authors proposed the use of SDN and blockchain to guarantee privacy and security in health care applications.

SDN is a paradigm that separates the network administration into two planes: control and data. In particular, in the data plane, the packets are forwarded into routers (access points) following a set of rules defined in the control plane. This communication paradigm is open-source and open-hardware, offers the possibility of defining non-proprietary access points, and is capable of handling traffic transparently. It also allows introducing QoS by analyzing the headers of the frames or packets processed by the access points.

For many years, the development of SDN was oriented to wired networks, and especially those working with IEEE 802.3 at the link layer Fraga, Micheletto, Llinás, Santos, and Zabala (2022). Examples of it are protocols such as OpenFlow McKeown, Anderson, Balakrishnan, Parulkar, Peterson, Rexford, Shenker, and Turner (2008), OpenConfig Shakir, Shaikh, Borman, Hines, Lebsack, and Morrow (2018), OVSDB Pfaff and Davie (2013), and NETCONF Watsen (2024). However, previous works have also shown that some protocols, like the Protocol Oblivious Forwarding (POF), can eventually be adapted to wireless networks Li, Hu, Fang, Ma, Chen, Huang, and Zhu (2017). However, the implementation of SDN on wireless networks is considerably behind its counterpart, i.e., on wired networks.

Recently, several efforts have been made to introduce the SDN paradigm within WSN Alnaser, Saloum, Sharadqh, and Hatamleh (2024); Al-Hamid, Karegar, and Chong (2023); Alzahrani and Fotiou (2021). Most of them are focused on supporting routing from the gateway to the Internet, without a particular treatment of the wireless links. Instead, in this paper, the proposal is focused on the wireless link and the way in which traffic can be ordered to guarantee time constraints and QoS. In order to address the dynamic communication usually required by IoT systems, and particularly when it involves real-time message delivery, this paper presents the implementation of a Software-Defined Wireless Networks (SDWN) able to deal with that. For this, the implementation of a new Access Point Controller (AP Controller) is developed to handle traffic in 802.11n and 802.11e based on the standard IEEE 802.11 access point specification IEEE (2021).

This proposal uses the POF and builds on the authors' previous work Llinas, Micheletto, Santos, and Ochoa (2023). In that paper, the authors presented the main idea of using the empty frame header bytes to indicate QoS (real-time and non-real-time) traffic associated with the source node. In this paper, instead, the different QoS considered in IEEE 802.11e were associated with priority types, named: high, medium, low, and non-real-time. Several SDWN instances (different numbers of nodes, layouts, and transmission demands) were implemented and evaluated using NS-3 NS-3 Project (2025). Consequently, a large set of experiments is reported to show how the implementation and tuning of the AP Controller improves different performance indicators like throughput, delay, packet loss ratio, and latency in comparison to a wireless network without the SDN controller. To make this proposal reusable for other researchers, new components and libraries were developed for the NS3 simulator, and made available in a public GitHub repository González Romero (2025).

Wireless Local Area Networks (WLAN) are becoming a de facto standard in many applications. Notebooks and personal computers have wireless card network adapters, but rarely have a wired network card. In order to operate WLANs, several access points (APs) should be distributed throughout the area in which the network is deployed. All these APs may be controlled by a WLAN Network Controller Systems (2018); Rojanala (2022). The SDN controller proposed in this paper can also operate as a WLAN controller, but it has more functions since it provides prioritization of nodes and messages in a dynamic context.

The next section provides a review of recent literature on software-defined wireless networks. Then, Section III presents the programmable access points that are the main components to implement the SDWN. Section IV describes the implementation and configuration of SDWNs using the NS-3 simulator. Section V describes the evaluation process of the SDWN performance and its results. Section VI discusses the experimental results. Finally, Section VII presents the conclusions and future work.

RELATED WORK

SDN has transformed traditional network management by decoupling the control and data planes, enabling centralized programmability through logically centralized controllers. While initially applied to wired networks using protocols like OpenFlow McKeown et al. (2008), extending SDN to wireless environments introduced new challenges, including channel variability, energy constraints, and Real-Time Quality of Service (RT-QoS) requirements. These challenges become more pronounced in WSNs, composed of devices with resource limitations Akyildiz, Su, Sankarasubramaniam, and Cayirci (2002), giving rise to Software-Defined Wireless Sensor Networks (SDWSN)Luo,

Tan, and Quek (2012), which aim to provide dynamic configurability, efficient resource management, and traffic prioritization. Next, we present and characterize the main proposals to implement SDWN, analyzing their capabilities to deal with real-time communication restrictions.

Overview of SDWNs

An early effort in this domain was ÆtherFlow Yan, Casey, Shome, Sprintson, and Sutton (2015), which adapts the OpenFlow protocol to wireless scenarios through the TinyNBI abstraction, introducing programmable control over parameters, like transmission power and SSID configuration. Evaluated using commercial access points, it showed improved performance in Layer 2 handoff scenarios. However, ÆtherFlow was tightly bound to a centralized architecture, lacked formal QoS abstractions, and did not support distributed control, limiting its scalability.

To address scalability challenges, WNOS (Wireless Network Operating System) Guan, Bertizzolo, Demirors, and Melodia (2021) introduces a distributed approach tailored to infrastructure-less networks, such as IoT, ad hoc, and sensor systems. By decoupling high-level policy design from low-level device execution, WNOS compiles centralized programs into distributed, cross-layer logic that runs locally on each node. Its architecture comprises a WiNAR for abstraction, a decomposition engine, and a programmable protocol stack. It has demonstrated notable performance improvements as well as flexibility and scalability, all without requiring global state dissemination.

Building on these ideas, Po-Fi Shi, Tian, Wang, Pan, and Zhang (2021) leverages POF to unify IEEE 802.3 and 802.11 under a common flow model. Po-Fi transforms access points into programmable entities capable of dynamic wireless management, introducing a centralized Time Division Multiple Access (TDMA) mechanism (hMAC) to mitigate interference issues common in 802.11 networks. Furthermore, decentralizing tasks (e.g., signal analysis) helps offload the central controller, improving scalability and adaptability. As network density increases, particularly in IoT contexts, newer architectures like DENIS-SDN Theodorou and Mamatas (2023) introduce network slicing to enhance performance. Through logical and physical slicing, it optimizes routing and interference management while incorporating tools such as CODET and a visual dashboard. Evaluations showed significant improvements in packet delivery, especially with physical slicing, demonstrating effectiveness in ultra-dense deployments.

Earlier frameworks, such as Sensor OpenFlow (SOF) Luo et al. (2012), laid the foundation by introducing OpenFlow-based programmability in WSNs. With support for compact and attribute-value addressing, and mecha- nisms such as Control-Message Quenching (CMQ) to reduce signaling overhead, SOF enhanced flexibility. However, it lacks integrated QoS prioritization and mechanisms for ultra-density or slicing, making it less suitable for modern IoT demands.

SDN-WISE Galluccio, Milardo, Morabito, and Palazzo (2015) further refined this idea with a design customized to WSNs. Using stateful forwarding and finite state machine modeling enables nodes to operate semi-autonomously, reducing controller dependency. It also introduces energy-saving features like duty cycling. Nonetheless, SDN-WISE does not formally incorporate QoS or address scenarios with high mobility or interference, challenges that newer systems like DENIS-SDN have begun to tackle. From a research-oriented perspective, IT-SDN Alves, Oliveira, Núñez Segura, and Margi (2017) offers a modular framework for evaluating SDN control algorithms on Contiki-based devices. Although flexible and suitable for experimentation, it lacks native support for QoS, automatic energy-saving mechanisms, and integrated security, limiting its applicability to real-world deployments.

The architecture of the controller remains a central issue in SDWN design. Centralized models (e.g., ÆtherFlow and Po-Fi) offer simplified management, but suffer from latency and fault tolerance issues. Conversely, distributed designs (e.g., SDN-WISE) provide resilience, but introduce synchronization complexity. Hybrid models, such as Logically Centralized but Physically Distributed (LCPD) controllers, balance these trade-offs. For example, Heller et al. Heller, Sherwood, and McKeown (2012) demonstrated that multiple controllers improve fault tolerance, though with diminishing returns. More recently, Mudvari and Tassiulas Mudvari and Tassiulas (2024) applied Deep Reinforcement Learning (DRL) to dynamically optimize controller placement, reducing synchronization overhead and improving load balancing at the cost of increased computational effort.

A conceptual advance in this area was presented by the authors in Llinas et al. (2023), where we proposed an SDWN architecture tailored for real-time IoT applications. By leveraging POF's flexibility, this work introduced a traffic classification system with three levels: strict, adaptive, and non-real-time. Moreover, that proposal introduces mechanisms to support up to 23 QoS levels via IEEE 802.11 fields. Although the model remains theoretical, it bridges the gap between conceptual frameworks and practical constraints, laying the foundations for implementing time-sensitive wireless applications.

In Coêlho, Silva, Martimiano, and Leonardo (2024), the authors discuss the use of 5G technology with virtual- ization and SDN controllers to implement IoT applications. The increasing demand for bandwidth for multimedia applications, among others, implies the optimization of resources. The SDN provides the possibility of programming the network to improve the performance of a group of network devices. However, this approach is not oriented to the scheduling or classification of the wireless local area networks.

In Xu (2025), an adaptive migration routing strategy is proposed for resource optimization in 5G core networks using SDN technologies. The proposal mainly refers to the main network and how to provide the available resources to end-users, optimizing its use. It is not oriented to the wireless link between, in this case, the mobile nodes and the base stations from which the traffic is routed. In Alnaser et al. (2024), the authors propose a scheduling method based on AI techniques for wireless sensor networks implemented through software. The software-defined network operates at the switches and routers, and not within the wireless traffic between the nodes and the gateways.

In Yang and Tsai (2024), the authors present a congestion control algorithm for 5G networks based on SDN. However, the proposal is oriented to control the switches and routers using the OpenFlow and OpenConfig protocols, that is, the congestion is controlled from the wired network and not the wireless links. In Al-Hamid et al. (2023), the authors introduce a testbed for smart IoT applications based on software-defined wireless sensor networks. However, in the paper, the authors developed the testbed, but there is no description of how the SDN approach works, nor the protocols used to implement it.

In Alzahrani and Fotiou (2021), the authors proposed the use of SDN in the context of IoT applications to ensure that what remote sensors announce is actually true. The SDN approach is not implemented in the last "mile" between the IoT devices and the gateways, but between the gateways and the rest of the network. In this regard, it is completely different from the proposal made in this paper, in which the SDN is applied to the link between the hosts and the access point/gateway. In Ellawindy and Shah Heydari (2021), the authors propose a crowdsourcing feedback mechanism to improve Quality of Experience (a measure of QoS applied mainly to multimedia traffic) in SDN networks. Like in the previous cases, the SDN controller handles the wired network and not the wireless traffic.

In contrast to these theoretical contributions, the work reported in this article operationalizes the model proposed in Llinas et al. (2023) by integrating IEEE 802.11e's native QoS mechanisms with the protocol-agnostic flexibility of POF, enabling effective traffic differentiation. Unlike Po-Fi Shi et al. (2021), which omits native 802.11e support, this proposal provides a functional and open-source implementation in NS-3, promoting reproducibility and real-world applicability. This integration marks a step forward in bridging SDN programmability with QoS enforcement for real- time IoT networks.

Despite an extensive literature review, to the best of the authors' knowledge, the development of SDN for wireless interfaces remains limited. Although the widespread use of OpenFlow in wired networks is not directly transferable to wireless networks due to differences in architecture, resource constraints, and mobility, its core principles have inspired several adaptations. Frameworks such as Aether Yan et al. (2015), Sensor OpenFlow Luo et al. (2012), and others represent efforts to tailor the SDN paradigm to the characteristics of wireless sensor and IoT networks. These adaptations enable more flexible control and allow for QoS management in smart environments.

PROGRAMMABLE ACCESS POINTS

As in Llinas et al. (2023), we propose using a POF-based software controller that allows programmable Access Points (named *Po-Fi AP*) to manage the network traffic. This controller classifies packets, assigning priority queues and dynamically modifying the Txop limit Shi et al. (2021). Figure 1 shows the architecture of a Po-Fi AP that includes the controller.

Operation scenario

In SDWN, each Po-Fi AP operates in a processing pipeline, where incoming flows pass through a series of tables, where filters and actions are applied based on the fields of the incoming frame. These actions can be performed at the link, network, or transport layer, depending on the filter used. After processing, the frame is either forwarded to an output port or discarded. If no matching action is found in the tables, the frame is sent to the Po-Fi controller via a *PacketIn* message. This controller defines the necessary instructions and updates the tables with a new entry to handle the frame.

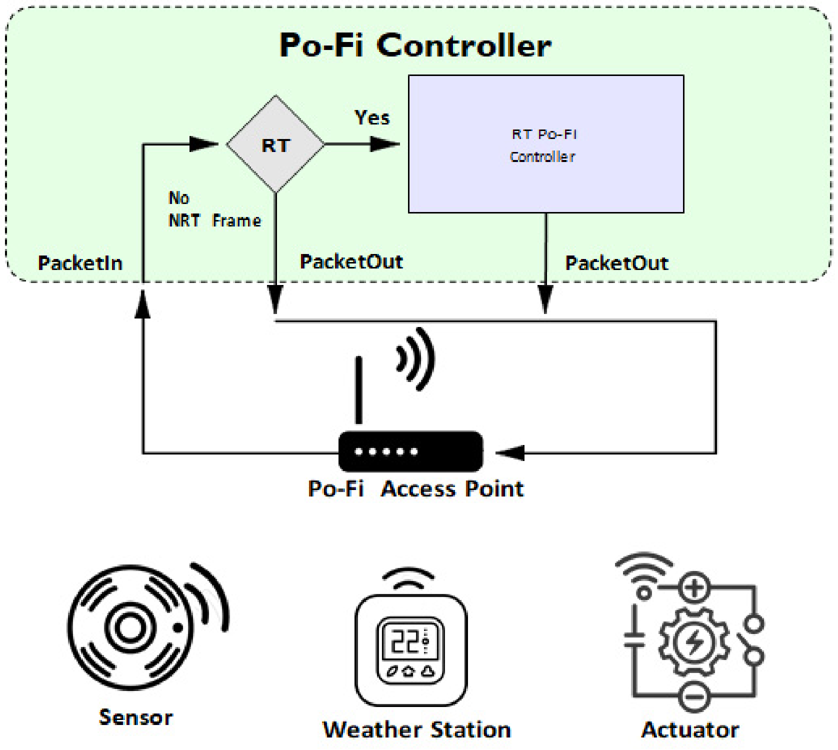
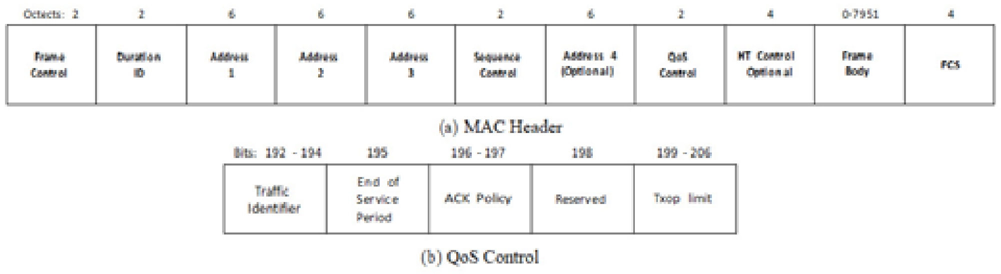


Figure 1: *Architecture of a Programmable Access Point (Po-Fi AP)*



Octects: 2	2	6	6	6	2	6	2	4	0-7951	4
Frame Control	Duration ID	Address 1	Address 2	Address 3	Sequence Control	Address 4 (Optional)	QoS Control	HT Control Optional	Frame Body	FCS

(a) MAC Header

Bits: 192 - 194	195	196 - 197	198	199 - 206
Traffic Identifier	End of Service Period	ACK Policy	Reserved	Txop limit

(b) QoS Control

Figure 2: *MAC Layer Structure: (a) MAC Header, (b) QoS Control*

In this scenario, different wireless devices (e.g., regular sensors, weather stations, and actuators in Figure 1) register with the Po-Fi AP to make them part of the network. Unlike the proposal reported in Shi et al. (2021), in this case, the Po-Fi controller has one more stage that separates the processing of message flow with and without time restrictions. Unrestricted flow is passed to the regular workflow of the controller, which accesses the AP's configuration with a lower priority. In the other case, the flow is processed by the RT Po-Fi controller, which considers the QoS according to the flow type. This operation is feasible to be implemented because POF allows packets to be processed without relying on specific protocols. Instead of using fixed IDs for header fields, it extracts information by identifying bit offsets and lengths within the packet (<offset, length >), thus allowing greater flexibility in traffic forwarding Li et al. (2017). Next, we explain the way the RT Po-Fi Controller works.

Operation of the RT Po-Fi Controller

The RT Po-Fi controller takes advantage of the POF capability to extract information, in this case, from the QoS Control field of the MAC header in IEEE 802.11, where the Traffic Identifier (TiD) and Txop Limit fields are specified (Figure 2). If the field related to Address 4 (that is, optional) is not used, then the TiD is identified at <192, 3> and the Txop Limit at <199, 8>. Otherwise, they must be shifted by 48 bits.

Packets belonging to VO (Voice) access category are assigned the highest priority, while VI (Video) packets given medium priority. BE (Best Effort) packets are considered low priority, whereas BK (Background) packets. correspond to non–real-time traffic, that is, they are transmitted whenever there is idle time or, in the same way, in the background.

The Po-Fi controller organizes these packets into three priority queues—high, medium, and low—ensuring that kets of higher importance are transmitted before those of medium priority, and so on. Within the low-priority queue, w packets have precedence over non-real-time packets. It is important to note that this management is performed at he Link layer to respect the principle of network neutrality.

The allocation of packets to the queues is done in the Po-Fi AP, according to the rules established by the Po-Fi controller. For this purpose, a message flow is sent between both, which follows a typical SDN scheme based on POF. When a Po-Fi AP receives a packet, it extracts the values of the QoS control field and checks if there is a previous rule in its POF forwarding tables. If there is no rule already defined, the Po-Fi AP sends a packet input message (i.e. Pmaecsksaegteln,) to the Po-Fi controller. This latter decides how to process the packet and responds with a flow modulation (i.e., FlowMod) that sets the rules for prioritization, queuing, and bandwidth limit modification. Finally, when is to be ansmitted, the Po-Fi AP executes a packet output message to send it to the corresponding destination. Figure 3 illustrates this process.

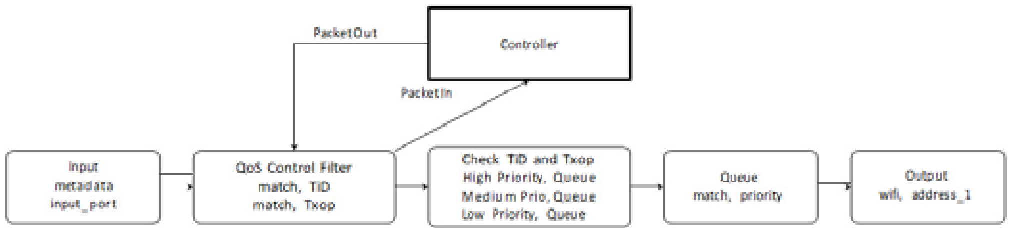


Figure 3: *Redirection tables in the Programmable Access Point (Po-Fi AP)*

The forwarding rules consider several scenarios. If two packets with the same priority arrive (e.g., high – high or low – low) Po-Fi AP will transmit one first and place the other in the respective queue until the sending process of the first one has been accomplished. If two packets correspond to the low and non–real-time priority(e.g., low -—non-real-time), the low packet will be transmitted first, while the non–real–time packet will be queued. In case that packets from different ACs and different priorities arrive (e.g., high — non-real-time, medium — high, or medium — low), the one with the highest priority will be transmitted first, ensuring that delay-sensitive traffic is not interrupted bybest-effort or background flows. When a new packet arrives while others are in the queue, its TiD is compared with those already waiting. If thenew packet has a higher TiD, it is sent before the others, regardless of whether it arrived first or later. This dynamic reordering logic in the queue allows the highest priority packets to always be processed first, regardless of when they entered the network.

The Po-Fi Controller also introduces a dynamic mechanism for adjusting the Txop Limit according to the traffic iority. Through a FlowMod message, the controller instructs the Po-Fi AP to modify the Txop Limit value (<199, 8>), allocating more or less channel usage time to high-priority packets according to the usage application. In this way, high and medium-rate packets can take advantage of transmission time to reduce latency and improve QoS in real-time applications. Besides, it provides efficient traffic management in WiFi networks using this programmable POF-based approach. Its capability to extract information from the QoS Control field, assign priority queues, and dynamically modify the Txop Limit enables bandwidth usage optimization and improved QoS for real-time applications.

DEFINING SDWNS IN THE NS-3 SIMULATOR

The current section presents the implementation and configuration of an SDWN in NS-3, in order to evaluate its formance under different simulation scenarios. The implementation was carried out using NS-3, a well-known open source network simulator that allows for simulating a wide range of networks, including wireless, mobile, and sensor networks. These simulations address the whole range from the physical layer to the application layer.

NS-3 provides an API that facilitates the creation of detailed simulations, accessible to both advanced users and hose who prefer working with a high-level language like Python. The simulator allows for capturing and analyzing network metrics, such as transmission rate, latency, and packet loss, making it a valuable tool for evaluating network performance under different conditions. The simulation process considers the stages that are described in the next subsections.

Node Modeling and Distribution

The simulation environment in NS-3 was configured according to the IEEE 802.11n standard with support for IEEE 802.11e, enabling the implementation of QoS mechanisms. The network topology comprises a Po-Fi AP, a Po-Fi Controller, and a variable number of stations (i.e., devices). One node is designated as the access point, while the others function as stations.

The WiFi network operates on channel 36 in the 5GHz band with a channel width of 20MHz. Both the AP and the stations are configured with the same SSID to ensure proper association. The physical layer parameters are set to model realistic wireless channel behavior, and the MAC layer is configured to comply with the IEEE 802.11n standard, ensuring compatibility across devices.

To implement the QoS features of IEEE 802.11e, the access point is set to operate with QoS enabled, while stations are assigned one of four access categories: non-real-time (Background (BK)), real-time lowest priority (Best Effort (BE)), real-time medium priority (Video (VI)), and real-time high priority (Voice (VO)). This configuration ensures the QoS assignment.

QoS parameters of each station are individually configured by cycling through the access categories and applying the corresponding EDCA settings. The access point is further tuned by enabling QoS support, specifying the SSID, defining the maximum size of aggregated MAC Protocol Data Units (A-MPDU) for the selected category, and disabling active probing to reduce unnecessary management traffic.

Once the device is installed, the corresponding QosTxop object is accessed, and the parameters AIFSN, CWmin, and CWmax are configured according to the EDCA profile. This approach allows for fine-grained, per-station MAC parameter configuration, providing better adaptation to different traffic types and enhancing service differentiation as defined in IEEE 802.11e.

For spatial distribution, the access point is placed in the center of the network area, and the stations are randomly located within a radius of 50 meters. Node mobility is not considered in these scenarios, and the stations remain static throughout the simulations.

The network was configured with the TCP/IP stack, assigning IP addresses within the 192.168.1.0/24 subnetwork. Each station generated UDP traffic directed to the AP, using variable packet sizes of 256, 512, and 1024 bytes. The same experiments were conducted for each of these packet sizes in order to analyze the effect that their variability produces on network performance.

Finally, QoS support was reinforced by the use of the Type of Service (ToS) field in the IP header, which was mapped to the TiD in NS-3. This approach, dynamically managed by the Po-Fi Controller, classifies the traffic into three priority categories: high, medium, low, and non-real-time traffic. This strategy allows optimizing resource utilization and ensuring differentiated treatment according to the service type.

Implementation of the device registration and packet management process

Figure 4 shows the registration and packet management process for real-time and non-real-time clients. The process operates on an SDN-based WiFi architecture with centralized flow control that uses a controller (*Po-Fi Controller*) and a controlled access point (*Po-Fi Ap*). The controller classifies flows by TiD values and determines transmission limits, while the AP manages packet reception, queuing, and prioritized forwarding with dynamic Txop configurations.

The *Po-Fi Ap* represents the access point that implements the logic for receiving, queuing, and forwarding packets with QoS control. During initialization, the UDP port is configured for communication, the socket is initialized, signals are linked to capture incoming IPv4 events, and a callback function is set to handle incoming packets.

When a client device (e.g., a sensor or actuator) sends a packet (action shown as (1) in Figure 4), it is received by the PoFiAp (2). Upon receiving data through its socket, the PoFiAp extracts the ToS (3) field from the packets—which corresponds to the TiD—and updates the TiDMap to associate IP addresses with their respective TiD values. Then, it processes each packet by querying the tidRegistry map to check if a FlowMod rule already exists for the received TiD (4). If such a rule does exist, a new query is sent to the PoFi controller via PacketIn (5).

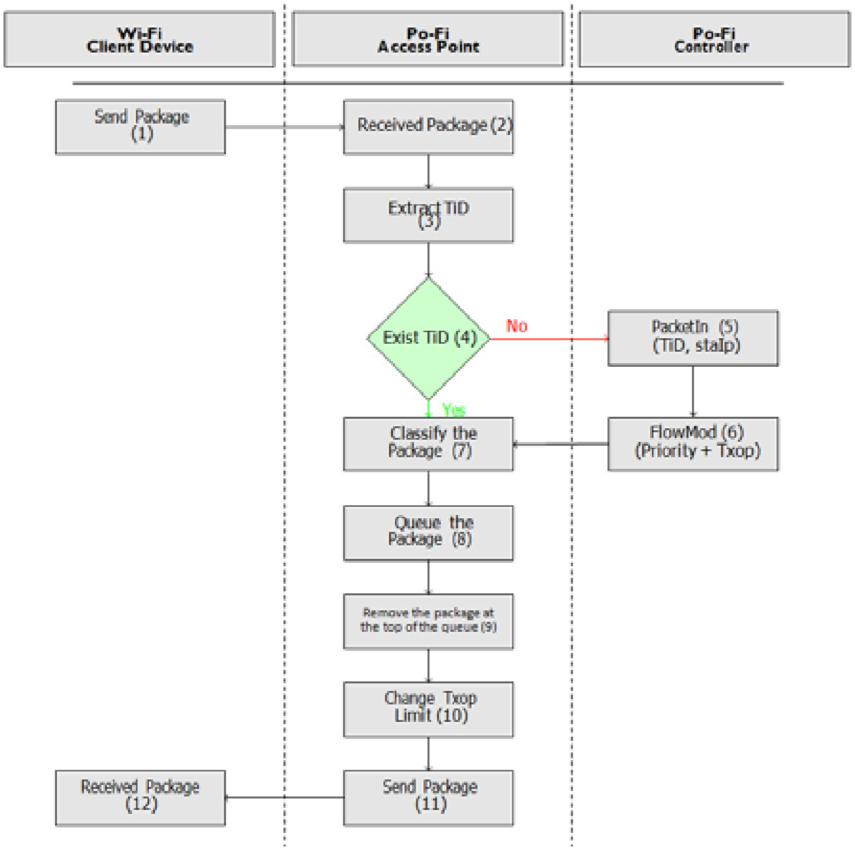


Figure 4: *Optimized registration and packet management process for real-time and non-real-time clients in a Po-Fi Access Point.*

The Po-Fi Controller acts as the logical core of the system, which is in charge of making decisions about the priority and transmission limit of the flows. When it receives a PacketIn message (5), it obtains the TiD field and the IP address of the sender node. Then, it returns a FlowMod structure containing an assigned priority category (high,.medium, low) and a Txop limit in microseconds (6). This assignment follows static rules based on the TiD value. This method is essential for flow classification and medium access control at the MAC layer.

Once the FlowMod message is received by the Po-Fi Ap, the packet is classified according to the priority and Txop assigned (7) to its TiD and placed in the corresponding priority queue (8). The packet at the front of the highest-priority queue is then selected and removed (9). Before transmission, the Po-Fi Ap adjusts the Txop limit on its Wi-Fi interface (10) by configuring the QosTxop object of the relevant priority level (high, medium, low, and non-real-time), ensuring an efficient and prioritized medium access. The packet is then transmitted to the recipient (11).

Performance metrics, such as latency and jitter, are calculated using the arrival and dispatch times and stored for later analysis. Finally, these metrics-along with received and sent packets, bytes, and throughput in Kbps—are collected and exported to a CSV file, which is essential for evaluating the effectiveness of the applied QoS management approach. In addition, we implemented other actions that enable comprehensive analysis and validation of the system. These actions included monitoring network flows that allow the collection of key metrics (e.g., throughput, latency, and packet loss), as well as flow classification.

To facilitate visualization of network behavior, animations were generated in NetAnim, using labeled nodes and real-time statistics. In addition, traces were generated in ASCII and PCAP formats, allowing detailed traffic analysis using tools like Wireshark. These complementary configurations not only enriched the simulation but also provided essential tools for debugging, optimizing, and evaluating the effectiveness of the QoS policies implemented in the SDWN.

TABLE 1. Summary of the experimental configurations

Exp.	SDWN	QoS	Interval (s)	Packet Size (bytes)	Categories	EDCA Modified
1	No	No	1s	256, 512, 1024	High+Medium+Low+Non-Real-Time	No
2	Yes	Yes	1s	256, 512, 1024	High+Medium+Low+Non-Real-Time	Yes / No

EVALUATING THE PERFORMANCE OF SDWNs

to analyze the behavior and performance of the network under different configurations and traffic patterns, a series of experiments was carried out as summarized in Table 1. The experiments were run for three different network configurations. In the first one, the wireless network was run without any kind of SDN controller. The access point implemented in the NS-3 simulator was the classical one, in which the traffic is handled according to the standard. It means that there is neither prioritization nor QoS identification of packets or hosts. Moreover, in case of conflict at the time of accessing the common channel, all of them used the traditional back-off mechanism proposed in IEEE 802.11. The second network configuration implemented the modified access point and introduced the network controller described before. However, the parameters proposed in IEEE 802.11e were kept without modification. In this approach, the contention windows that the high-priority hosts wait to re-transmit after a detected collision were shorter than the ones used for low-priority traffic. We named this configuration *Original SDWN.*

The third network configuration uses the modified access point and network controller as in the second con- figuration, but the EDCA parameters were set based on the experimental analysis performed in the simulations to optimize the behavior. AIFSN was kept as defined in IEEE 802.11e to preserve traffic differentiation, while CWmin and CWmax were set to 15 and 1024 slots, respectively. These values were chosen after several experiments to find the best combination. We named this configuration as *Modified SDWN.*

For each network configuration, the performance of the SDWN was evaluated using three different packet sizes: 256, 512, and 1024 bytes. The traffic was distributed across four priority levels: high, medium, low, and non-real-time. For each combination of packet size and network configuration, ten different networks were randomly generated with 10 to 100 hosts each, in steps of 10 hosts. The generation periods for sending high, medium, and low/background messages were set to 1 second for all of them. These experiments included 900 random networks, each one simulated for ten minutes. The next section presents the experimental results.

Experimental results

Tables 2 to 10 present the values of delay, throughput, and packet loss for the different combinations of network sizes, configuration, and packet sizes. The results are the average of ten randomly generated systems, simulated for 10 minutes each. The results show that the network performance has a breakpoint at forty hosts. When more than forty hosts are deployed, all performance criteria deteriorate significantly, whether the SDN controller is implemented and the EDCA parameters are tuned or not. In the tables, the best results are highlighted in bold, while the worst values are in red.

Analysis of the delay

The size of the packet has a significant impact on the transmission delay, regardless of the simulated network configuration. It can be seen that it is incremented by nearly 35% when the packet size is changed from 256 (Table 2) to 512 B (Table 3), and 69% when it has 1024 B (Table 4). If the comparison is made from 1024 to 256, the increment is 128%. This can be explained by considering the time required actually to transmit packets of different sizes and how it affects the backoff algorithm. As the time windows used in the process are not a function of the packet sizes, making the packet size larger increments the delay.

TABLE 2. Delay for 256 B packets under the different network configurations

Devices	256 Bytes											
	HIGH			MEDIUM			LOW			NON REAL TIME		
	No SDWN	SDWN		No SDWN	SDWN		No SDWN	SDWN		No SDWN	SDWN	
		Org.	Mod.		Org.	Mod.		Org.	Mod.		Org.	Mod.
10	0.12	0.11	**0.11**	**0.11**	0.11	0.12	0.11	0.11	0.11	0.12	**0.11**	0.12
20	0.11	0.11	0.11	0.11	0.12	**0.11**	0.11	0.11	0.11	0.11	**0.11**	0.12
30	0.12	0.11	**0.11**	0.12	0.11	**0.11**	0.12	0.11	**0.11**	0.11	0.11	**0.11**
40	0.11	0.11	**0.11**	0.11	0.11	**0.11**	0.12	0.11	**0.11**	0.11	0.11	**0.11**
50	0.30	0.34	**0.30**	0.31	**0.25**	0.36	**0.38**	0.52	0.51	0.41	0.38	**0.37**
60	0.47	**0.39**	0.42	0.46	**0.43**	0.52	**0.56**	0.70	0.73	0.59	0.54	**0.53**
70	0.57	**0.48**	0.57	**0.67**	0.68	0.71	**0.70**	0.88	0.85	0.67	0.64	**0.63**
80	0.72	0.64	**0.64**	**0.69**	0.72	0.71	**0.73**	0.88	0.85	**0.69**	0.73	0.73
90	0.80	0.85	**0.75**	0.80	**0.78**	0.86	**0.85**	1.08	1.13	**0.86**	0.93	0.89
100	**0.93**	1.02	0.95	**0.86**	0.98	0.98	**1.01**	1.21	1.24	1.02	**0.99**	1.00

TABLE 3. Delay for 512 B packets under the different network configurations

Devices	512 Bytes											
	HIGH			MEDIUM			LOW			NON REAL TIME		
	No SDWN	SDWN		No SDWN	SDWN		No SDWN	SDWN		No SDWN	SDWN	
		Org.	Mod.		Org.	Mod.		Org.	Mod.		Org.	Mod.
10	0.16	0.16	**0.15**	0.17	0.17	**0.16**	**0.15**	0.16	0.17	0.16	**0.15**	0.16
20	0.16	**0.15**	0.16	**0.15**	0.16	0.16	0.16	0.16	**0.16**	0.17	0.17	**0.15**
30	0.16	**0.15**	0.16	**0.16**	0.16	0.17	0.16	0.16	**0.16**	0.16	0.16	**0.16**
40	0.16	0.16	**0.15**	0.16	0.16	**0.16**	0.16	0.16	**0.16**	**0.15**	0.16	0.17
50	0.39	0.42	**0.39**	0.38	**0.35**	0.53	**0.49**	0.57	0.58	0.57	**0.52**	0.58
60	0.62	0.79	**0.58**	0.57	0.65	**0.52**	**0.82**	0.90	0.95	0.78	0.82	**0.74**
70	**0.71**	1.03	0.73	**0.84**	0.96	0.90	**0.83**	1.11	1.08	0.88	**0.76**	0.86
80	1.01	0.99	**0.94**	1.01	**0.95**	0.99	**0.94**	1.04	1.21	0.98	**0.87**	0.91
90	1.12	1.33	**1.09**	**1.11**	1.39	1.13	**1.20**	1.31	1.39	1.23	1.16	**1.11**
100	1.24	1.41	**1.17**	**1.20**	1.51	1.27	**1.34**	1.67	1.77	1.30	1.30	**1.28**

TABLE 4. Delay for 1024 B packets under the different network configurations

Devices	1024 Bytes											
	HIGH			MEDIUM			LOW			NON REAL TIME		
	No SDWN	SDWN		No SDWN	SDWN		No SDWN	SDWN		No SDWN	SDWN	
		Org.	Mod.		Org.	Mod.		Org.	Mod.		Org.	Mod.
10	0.26	**0.24**	0.25	0.26	**0.23**	0.24	**0.25**	0.26	0.26	0.26	0.25	**0.25**
20	0.26	0.27	**0.26**	0.25	0.25	**0.25**	**0.26**	0.26	0.27	0.26	**0.25**	0.26
30	**0.27**	0.27	0.28	0.26	0.26	**0.25**	0.26	0.26	**0.26**	**0.24**	0.25	0.26
40	0.26	0.26	**0.25**	**0.24**	0.26	0.25	**0.26**	0.27	0.27	0.25	0.25	**0.25**
50	0.75	0.81	**0.71**	0.65	0.82	**0.62**	0.88	**0.93**	1.06	**0.71**	0.86	0.75
60	1.03	**0.84**	0.99	0.93	**0.89**	0.94	**1.27**	1.29	1.32	1.16	**0.97**	1.25
70	1.16	1.40	**1.09**	**1.34**	1.73	1.47	1.43	1.51	**1.33**	**1.31**	1.42	1.45
80	1.42	1.71	**1.34**	**1.34**	1.66	1.41	**1.37**	1.65	1.60	1.54	**1.37**	1.58
90	1.81	1.97	**1.78**	**1.77**	2.02	**1.81**	1.86	1.88	2.28	1.88	2.00	**1.76**
100	2.07	2.53	**1.93**	2.04	2.53	**2.00**	2.12	**2.10**	2.37	2.13	2.17	**2.07**

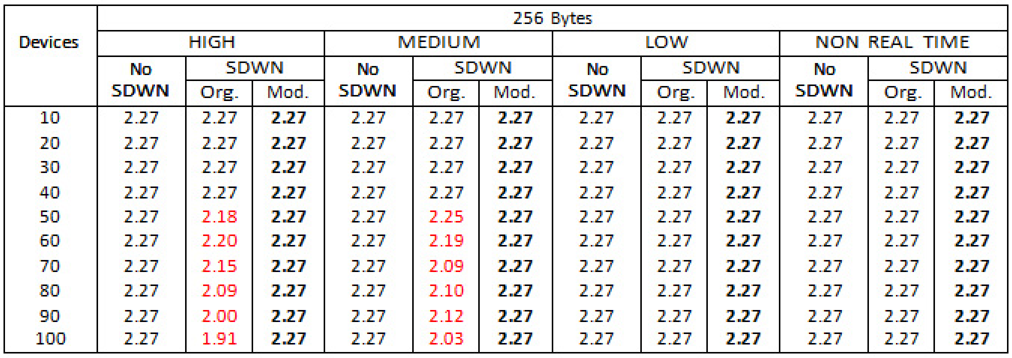
The number of hosts is also important at the moment of analyzing the delay. As can be seen, the delay is quite similar up to 40 hosts in the network, with just slight differences below 4%. However, when said number climbs to 50 or more, the delay is incremented three times from 40 to 50 nodes, and then it shows an almost linear increment of 20 to 25% with the incorporation of 10 hosts up to 100.

Finally, it is necessary to compare the behavior of the different network configurations. Using the IEEE 802.11e iginal approach is not a good choice, as the parameters that are used for the time windows (considered in the backoff algorithm) are not appropriate. In the case of high-priority traffic, the waiting time of the backoff algorithm is shorter. It might be perceived as an advantage because it will quickly retry transmission when there are many nodes trying to do the same thing. However, it becomes a disadvantage because it generates more collisions. In some way, this can be seen as a denial of service, since the network is saturated with retries in a short period, incrementing collisions. For that reason, the network configuration based on a best effort approach outperforms the use of the SDWN with the original parameter configuration. However, this is not the case when using the SDWN network controller with the time windows set up as in the best effort case. With that change, the access point serves first the high-priority traffic, resulting in an average lower delay. The behavior is not uniform across the different priorities, but while looking at the results, there is a better response with the SDWN modified controller.

Analysis of the throughput

The analysis of the throughput is based on the average data rate per node, measured in kilobits per second (kbps) according to the simulation results. The next tables present this metric as a function of packet size, traffic priority, and network configuration. A trivial result is that throughput in the simulation conditions increases with the packet size from 2.27 when the packet size is 256 B to 8.42 when it is 1024 B, almost four times. This shows the influence of the header length, which is proportionally more important for small packet sizes.

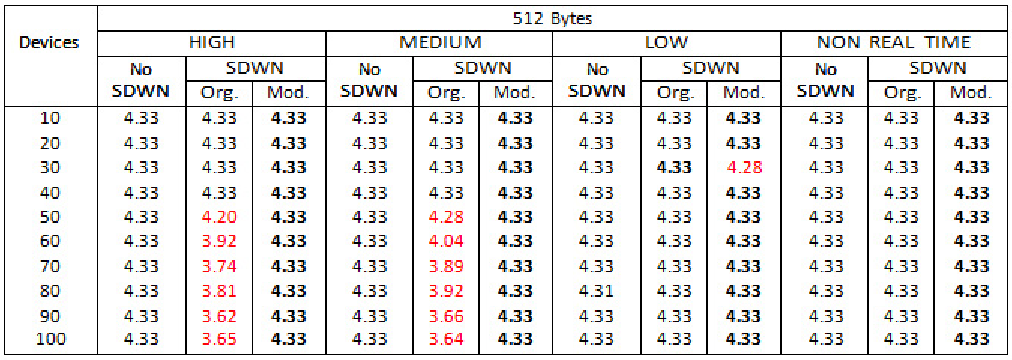
TABLE 5. Throughput for 256 B packets under the different network configurations



Devices	256 Bytes											
	HIGH			MEDIUM			LOW			NON REAL TIME		
	No SDWN	SDWN		No SDWN	SDWN		No SDWN	SDWN		No SDWN	SDWN	
		Org.	Mod.		Org.	Mod.		Org.	Mod.		Org.	Mod.
10	2.27	2.27	**2.27**	2.27	2.27	**2.27**	2.27	2.27	**2.27**	2.27	2.27	**2.27**
20	2.27	2.27	**2.27**	2.27	2.27	**2.27**	2.27	2.27	**2.27**	2.27	2.27	**2.27**
30	2.27	2.27	**2.27**	2.27	2.27	**2.27**	2.27	2.27	**2.27**	2.27	2.27	**2.27**
40	2.27	2.27	**2.27**	2.27	2.27	**2.27**	2.27	2.27	**2.27**	2.27	2.27	**2.27**
50	2.27	2.18	**2.27**	2.27	2.25	**2.27**	2.27	2.27	**2.27**	2.27	2.27	**2.27**
60	2.27	2.20	**2.27**	2.27	2.19	**2.27**	2.27	2.27	**2.27**	2.27	2.27	**2.27**
70	2.27	2.15	**2.27**	2.27	2.09	**2.27**	2.27	2.27	**2.27**	2.27	2.27	**2.27**
80	2.27	2.09	**2.27**	2.27	2.10	**2.27**	2.27	2.27	**2.27**	2.27	2.27	**2.27**
90	2.27	2.00	**2.27**	2.27	2.12	**2.27**	2.27	2.27	**2.27**	2.27	2.27	**2.27**
100	2.27	1.91	**2.27**	2.27	2.03	**2.27**	2.27	2.27	**2.27**	2.27	2.27	**2.27**

For small packets (i.e., 256 bytes, shown in Table 5), throughput remains stable (2.22 kbps per node) in low- density scenarios (<40 devices), but degrades sharply as network congestion increases, particularly affecting high- and medium-priority traffic. This effect is exacerbated when QoS is enabled with a centralized controller, due to the aggressive contention window parameters of the default EDCA (i.e., CWmin = 3, CWmax = 7). This leads to high collision rates and throughput reductions of up to 16% in dense networks (100 nodes).

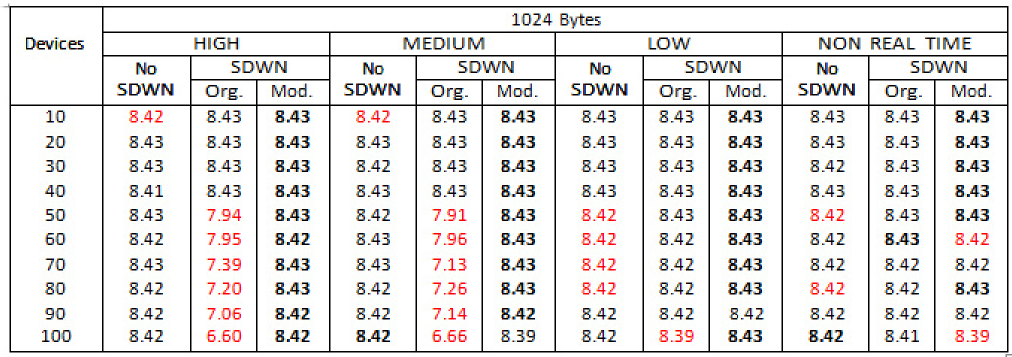
TABLE 6. Throughput for 512 B packets under the different network configurations



Devices	512 Bytes											
	HIGH			MEDIUM			LOW			NON REAL TIME		
	No SDWN	SDWN		No SDWN	SDWN		No SDWN	SDWN		No SDWN	SDWN	
		Org.	Mod.		Org.	Mod.		Org.	Mod.		Org.	Mod.
10	4.33	4.33	**4.33**	4.33	4.33	**4.33**	4.33	4.33	**4.33**	4.33	4.33	**4.33**
20	4.33	4.33	**4.33**	4.33	4.33	**4.33**	4.33	4.33	**4.33**	4.33	4.33	**4.33**
30	4.33	4.33	**4.33**	4.33	4.33	**4.33**	4.33	**4.33**	4.28	4.33	4.33	**4.33**
40	4.33	4.33	**4.33**	4.33	4.33	**4.33**	4.33	4.33	**4.33**	4.33	4.33	**4.33**
50	4.33	4.20	**4.33**	4.33	4.28	**4.33**	4.33	4.33	**4.33**	4.33	4.33	**4.33**
60	4.33	3.92	**4.33**	4.33	4.04	**4.33**	4.33	4.33	**4.33**	4.33	4.33	**4.33**
70	4.33	3.74	**4.33**	4.33	3.89	**4.33**	4.33	4.33	**4.33**	4.33	4.33	**4.33**
80	4.33	3.81	**4.33**	4.33	3.92	**4.33**	4.31	4.33	**4.33**	4.33	4.33	**4.33**
90	4.33	3.62	**4.33**	4.33	3.66	**4.33**	4.33	4.33	**4.33**	4.33	4.33	**4.33**
100	4.33	3.65	**4.33**	4.33	3.64	**4.33**	4.33	4.33	**4.33**	4.33	4.33	**4.33**

With medium-sized packets (i.e., 512 bytes, shown in Table 6), a better balance between overhead and payload is achieved, allowing throughput to reach up to 4.23 kbps per node under favorable conditions. Although performance still declines with increased load, the degradation is more gradual, with a 16% drop observed under the QoS and controller configuration.

TABLE 7. Throughput for 1024 B packets under the different network configurations



Devices	1024 Bytes											
	HIGH			MEDIUM			LOW			NON REAL TIME		
	No SDWN	SDWN		No SDWN	SDWN		No SDWN	SDWN		No SDWN	SDWN	
		Org.	Mod.		Org.	Mod.		Org.	Mod.		Org.	Mod.
10	8.42	8.43	**8.43**	8.42	8.43	**8.43**	8.43	8.43	**8.43**	8.43	8.43	**8.43**
20	8.43	8.43	**8.43**	8.43	8.43	**8.43**	8.43	8.43	**8.43**	8.43	8.43	**8.43**
30	8.43	8.43	**8.43**	8.42	8.43	**8.43**	8.43	8.43	**8.43**	8.42	8.43	**8.43**
40	8.41	8.43	**8.43**	8.43	8.43	**8.43**	8.43	8.43	**8.43**	8.43	8.43	**8.43**
50	8.43	7.94	**8.43**	8.42	7.91	**8.43**	8.42	8.43	**8.43**	8.42	8.43	**8.43**
60	8.42	7.95	**8.42**	8.43	7.96	**8.43**	8.42	8.42	**8.43**	8.42	**8.43**	8.42
70	8.43	7.39	**8.43**	8.43	7.13	**8.43**	8.42	8.42	**8.43**	8.42	8.42	8.42
80	8.42	7.20	**8.43**	8.42	7.26	**8.43**	8.42	8.42	**8.43**	8.42	8.42	**8.43**
90	8.42	7.06	**8.42**	8.42	7.14	**8.42**	8.42	8.42	8.42	8.42	8.42	8.42
100	8.42	6.60	**8.42**	**8.42**	6.66	8.39	8.42	8.39	**8.43**	**8.42**	8.41	8.39

For large packets (i.e., 1024 bytes, shown in Table 7), the network reaches maximum spectral efficiency, with throughput up to 8.23 kbps per node in non-congested conditions. However, even in this case, performance can drop by up to 21% under heavy load when default EDCA parameters are used with QoS, due to channel limitations and high packet collision probability.

Configurations without QoS or controller maintain consistent throughput across all priorities and packet sizes, reflecting equal medium access, but lacking optimization. In contrast, the configuration that combines QoS, an SDWN controller, and modified EDCA parameters (e.g., CWmin = 15, CWmax = 1023) achieves the highest throughput in all scenarios, even under high node density. Particularly, for small packets (256 bytes), the throughput was 2.23 kbps, it was 4.33 kbps for medium-sized packets (512 bytes), and 8.43 kbps for large packets (1024 bytes). This performance is attributed to the combination of broader contention windows that reduce collisions and the preservation of traffic differentiation through AIFSN (Arbitration Inter-Frame Spacing) values.

In summary, the results show that default EDCA parameters are unsuitable for dense environments with prioritized traffic. The modified EDCA configuration improves throughput by up to 27%, providing a more efficient balance between medium access control, collision avoidance, and traffic differentiation.

Analysis of packet loss

Packet loss is a critical aspect to address in wireless networks since it directly influences the network performance. A packet loss occurs when a packet is ultimately discarded by the host or access point, after multiple failed retransmission attempts. In collision conditions, the node executes an exponential backoff algorithm based on contention windows until the defined attempt limit is reached. In that context, packet size directly influences the network performance, as larger packets occupy the channel for longer, increasing the probability of collisions and, consequently, losses.

TABLE 8. Packet loss for 256 B packets under the different network configurations

Devices	256 Bytes											
	HIGH			MEDIUM			LOW			NON REAL TIME		
	No SDWN	SDWN		**No SDWN**	SDWN		**No SDWN**	SDWN		**No SDWN**	SDWN	
		Org.	Mod.		Org.	Mod.		Org.	Mod.		Org.	Mod.
10	0.00	0.08	**0.00**	0.00	0.00	0.00	0.00	0.11	**0.00**	0.00	0.00	0.00
20	0.00	0.00	0.00	0.00	0.04	**0.00**	**0.00**	0.00	0.06	0.08	0.03	**0.03**
30	0.11	0.06	**0.03**	0.10	**0.00**	**0.02**	0.04	0.06	0.02	0.08	0.02	**0.02**
40	0.10	0.02	**0.02**	0.08	0.02	**0.00**	0.12	0.03	**0.02**	0.07	0.03	**0.03**
50	0.06	4.03	**0.03**	0.03	1.07	**0.03**	0.03	0.10	**0.03**	**0.03**	0.08	0.04
60	0.04	3.44	**0.02**	0.05	3.77	**0.01**	**0.04**	0.08	0.05	0.06	0.07	**0.04**
70	0.08	5.58	**0.02**	0.06	8.01	**0.03**	0.08	0.12	**0.03**	0.07	0.11	**0.05**
80	0.07	8.25	**0.03**	0.07	7.69	**0.03**	0.08	0.13	**0.03**	0.07	0.18	**0.04**
90	0.08	12.16	**0.02**	0.05	6.89	**0.03**	0.08	0.55	**0.03**	0.09	0.56	**0.03**
100	0.07	16.11	**0.04**	0.07	11.03	**0.06**	0.06	0.17	**0.04**	0.08	0.19	**0.04**

TABLE 9. Packet loss for 512 B packets under the different network configurations

Devices	512 Bytes											
	HIGH			MEDIUM			LOW			NON REAL TIME		
	No SDWN	SDWN		**No SDWN**	SDWN		**No SDWN**	SDWN		**No SDWN**	SDWN	
		Org.	Mod.		Org.	Mod.		Org.	Mod.		Org.	Mod.
10	0.00	0.00	**0.00**	0.08	0.00	**0.00**	0.06	0.00	**0.00**	0.17	0.00	**0.00**
20	0.04	0.08	**0.00**	**0.04**	0.08	0.08	0.00	0.00	**0.00**	**0.00**	0.06	0.03
30	0.03	0.06	**0.03**	0.04	0.02	**0.00**	**0.02**	0.06	1.27	0.04	0.02	**0.02**
40	0.05	0.03	**0.02**	0.02	0.03	**0.00**	0.03	0.07	**0.03**	0.03	0.05	**0.03**
50	0.03	3.01	**0.00**	0.10	1.18	**0.03**	0.03	0.06	**0.01**	**0.04**	0.08	0.05
60	0.11	9.67	**0.01**	0.10	6.70	**0.02**	0.09	0.13	**0.04**	0.08	0.14	**0.04**
70	0.05	13.60	**0.02**	0.06	10.30	**0.04**	**0.06**	0.13	0.07	0.06		**0.03**
80	0.04	11.98	**0.04**	0.08	9.60	**0.06**	0.63	0.13	**0.04**	0.10	0.10	**0.04**
90	0.09	16.52	**0.04**	0.08	15.42	**0.05**	0.07	0.16	**0.06**	0.11	0.17	**0.06**
100	0.09	15.71	**0.05**	0.08	15.98	**0.06**	0.10	0.15	**0.05**	0.06	0.13	**0.05**

Similar to throughput behavior, in networks with low node density (fewer than 40 devices), all configurations evaluated show near-perfect performance, with loss rates below 0.1%, mainly attributable to physical channel errors (Tables 8 to 10). However, as the network becomes congested, significant differences between configurations emerge. In particular, the original IEEE 802.11e EDCA shows the worst performance in dense scenarios. In networks with 100 devices, losses reach critical levels in priority traffic: up to 21.72% for 1024-byte packets (Table 10), and 16% and 11% for 512-byte packets (Table 9) in high and medium priorities, respectively. This contradictory behavior—where supposedly priority traffic suffers more losses—is a direct consequence of short contention windows (CWmin =3, CWmax = 7), which favor the occurrence of collisions by concentrating the transmission retries.

TABLE 10. Packet loss for 1024 B packets under the different network configurations

Devices	1024 Bytes											
	HIGH			MEDIUM			LOW			NON REAL TIME		
	No SDWN	SDWN		No SDWN	SDWN		No SDWN	SDWN		No SDWN	SDWN	
		Org.	Mod.		Org.	Mod.		Org.	Mod.		Org.	Mod.
10	0.08	0.00	**0.00**	0.08	**0.00**	0.08	0.00	0.00	**0.00**	0.06	**0.00**	0.06
20	0.04	0.04	**0.04**	0.00	**0.00**	0.04	**0.00**	0.03	0.06	0.03	0.03	**0.03**
30	0.08	0.06	**0.03**	0.06	0.02	**0.02**	0.02	0.04	**0.02**	0.17	0.02	**0.02**
40	0.20	0.03	**0.03**	0.05	0.07	**0.02**	**0.02**	0.05	0.03	0.05	0.07	**0.02**
50	0.06	5.88	**0.04**	0.17	6.22	**0.04**	0.13	0.12	**0.03**	0.12	0.09	**0.04**
60	0.10	5.67	**0.08**	0.06	5.64	**0.05**	0.11	0.09	**0.05**	0.11	0.10	**0.07**
70	0.08	12.38	**0.07**	**0.07**	15.43	0.10	0.13	0.17	**0.05**	0.08	0.13	**0.10**
80	0.08	14.70	**0.07**	0.11	13.97	**0.03**	0.08	0.12	**0.08**	0.12	0.13	**0.06**
90	0.15	16.30	**0.11**	**0.13**	15.30	0.14	0.12	0.16	**0.12**	0.14	0.20	**0.12**
100	0.15	21.72	**0.10**	**0.15**	21.11	0.53	0.16	0.58	**0.06**	**0.19**	0.24	0.49

In contrast, the modified EDCA configuration—used in conjunction with SDWN and QoS—demonstrates superior control over packet loss. By expanding the contention window values (CWmin = 15, CWmax = 1023), the system increases the margin for retransmissions before a packet is discarded. Additionally, maintaining the original AIFSN values helps preserve the intended access differentiation among traffic types, ensuring that high-priority traffic retains its advantage in channel contention.

This strategy keeps the packet loss rate below 1%, even under high device density. Thus, the network not only achieves higher reliability but also shows faster recovery during severe congestion episodes, demonstrating improved operational resilience compared to standard EDCA.

DISCUSSION

The results presented in Section V reveal several key insights regarding the application of IEEE 802.11e and WN-based QoS strategies in wireless networks, especially in scenarios where traffic prioritization is required beyond aditional multimedia use cases.

First, despite its formal introduction in 2005, IEEE 802.11e was not commercialized as a standalone feature and seen limited direct adoption in commercial hardware. However, its QoS mechanisms were gradually integrated into later Wi-Fi standards, such as 802.11n and subsequent versions. This limited standalone adoption is likely not due to a lack of interest in QoS, but rather the robustness and simplicity of the default best-effort configuration of IEEE 802.11, which, as shown in the results, often performs better than the default EDCA parameters when traffic is congested. Specifically, the original EDCA configuration (CWmin = 3, CWmax = 7) leads to aggressive contention, increased collisions, and ultimately lower throughput and higher packet loss, particularly for priority traffic in dense networks. This underscores the fact that implementing QoS at the MAC layer without careful tuning of EDCA parameters can be counterproductive.

Second, the introduction of a Software-Defined Wireless Network controller offers an opportunity to dynamically manage EDCA parameters and adapt them to current network conditions. This study demonstrates that only when EDCA is modified with expanded contention windows (e.g., CWmin = 15, CWmax = 1023), the combination of QoS and SDWN provides a clear improvement over best-effort, particularly in high-density scenarios. This improvement is achieved by reducing collision rates and allowing more retransmission attempts while preserving prioritization through AIFSN differentiation. However, this enhancement is marginal compared to the simplicity and stability of best-effort in low to moderate densities of network nodes, reinforcing the idea that fine-grained QoS at the MAC layer must be carefully engineered.

Third, the number of hosts is a major factor influencing network performance. Although IEEE 802.11 standards allow up to 255 clients per access point, both simulation and practical experience suggest that network performance degrades significantly beyond 40 devices, regardless of the configuration. In the simulations, performance metrics such as throughput and packet loss deteriorated sharply after this threshold, with high-priority traffic being paradoxically penalized under default values of EDCA. This result points to the importance of proper network segmentation, particularly when implementing QoS, to ensure no more than 30–40 devices per access point.

Additionally, packet size also plays a crucial role in network performance. Smaller packets suffer from higher relative protocol overhead, resulting in lower throughput and a steeper performance drop under congestion. Conversely, larger packets are more efficient but make the network more susceptible to collisions and retransmission delays, especially if contention windows are too short. The modified EDCA with SDWN control successfully mitigates these issues, keeping packet loss under 1%, even for high-priority traffic, across different packet sizes.

Conclusions and Future Work

This paper presented the design, implementation, and evaluation of a SDWN, conceived to support applications like IoT systems, which usually involve wireless communication and real-time constraints for packet delivery. In contrast to other proposals reported in the related work section, this SDN manages QoS in wireless networks, particularly on the standard IEEE 802.11e.

The performance of the SDWN was evaluated through a large set of experiments using NS-3 for QoS management, and the results provided important insights on its behavior. In particular, the findings highlight that QoS implementation at the MAC layer must be context-aware. The default values for EDCA configuration parameters may not be suitable for non-multimedia prioritization use cases, especially in dense IoT scenarios. Leveraging an SDWN controller to apply adaptive EDCA tuning can restore the expected benefits of traffic prioritization, but careful network planning remains essential—particularly in managing node density, selecting packet sizes, and avoiding over-congestion.

The use of this kind of technology is not yet widespread. Although it seems attractive to prioritize traffic within networks, the results obtained in the experiments show that the final performance is not always improved in relation to a simple best-effort deployment. Therefore, more research efforts are required, and this proposal represents a first step towards solving the stated problem.

In order to contribute to further exploration of solutions to implement SDWN capable of managing QoS, the current implementation was made available to the research community in a public GitHub repository González Romero (2025). This will allow researchers to take advantage of this work and build on it.

As future work, the next steps consider the simulation of the proposal with different use cases that require QoS traffic differentiation (e.g., health-care environments), and the actual implementation on open-hardware access points to evaluate the performance of the SDWN under different real traffic conditions. In addition, the metrics collected during the simulations will be used to develop Machine Learning techniques aimed at dynamically determining the most appropriate values for AIFSN, CWmin, and CWmax, to improve key network performance indicators such as latency, throughput, and packet loss according to the number of nodes interacting in the network. This approach will enable more dynamic control from the SDWN controller, leading to an optimized configuration of these parameters.

REFERENCIAS

R. M. Santos, J. Santos, J. D. Orozco, A least upper bound on the fault tolerance of real-time systems, Journal of Systems and Software 78 (2005) 47–55.

ISO, 11898-1:2024; Road vehicles—Controller area network (CAN)—Part 1: Data link layer and physical coding sublayer, Standard, International Organization for Standardization, Geneva, Switzerland, 2024.

IEC, 61158-1:2023; Industrial communication networks—Fieldbus specifications—Part 1: Overview and guidance for the IEC 61158 and IEC61784 series, Standard, International Electrotechnical Commission, Geneva, Switzerland, 2023.

C. Xu, Resource optimization algorithm for 5g core network integrating nfv and sdn technologies, International Journal of Intelligent Networks (2025).

A. Rahman, A. Wadud, J. Islam, D. Kundu, T. Bhuiyan, G. Muhammad, Z. Ali, Internet of medical things and blockchain-enabled patient-centric agent through sdn for remote patient monitoring in 5g network, Scientific Reports 14 (2024).

M. Fraga, M. Micheletto, A. Llinás, R. Santos, P. Zabala, Flow scheduling in data center networks with time and energy constraints: A software- defined network approach, Future Internet 14 (2022).

N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, Openflow: enabling innovation in campus networks, SIGCOMM Comput. Commun. Rev. 38 (2008) 69–74.

R. Shakir, A. Shaikh, P. Borman, M. Hines, C. Lebsack, C. Morrow, gRPC Network Management Interface (gNMI), Internet- Draft draft-openconfig-rtgwg-gnmi-spec-01, Internet Engineering Task Force, 2018. URL: *https://datatracker.ietf.org/doc/ draft-openconfig-rtgwg-gnmi-spec/01/*, work in Progress.

B. Pfaff, B. Davie, The Open vSwitch Database Management Protocol, RFC 7047, 2013. URL: *https://www.rfc-editor.org/info/rfc7047*. doi:10.17487/RFC7047.

K. Watsen, NETCONF Client and Server Models, Internet-Draft draft-ietf-netconf-netconf-client-server-37, Internet Engineering Task Force, 2024. URL: *https://datatracker.ietf.org/doc/draft-ietf-netconf-netconf-client-server/37/*, work in Progress.

S. Li, D. Hu, W. Fang, S. Ma, C. Chen, H. Huang, Z. Zhu, Protocol oblivious forwarding (pof): Software-defined networking with enhanced programmability, IEEE Network 31 (2017) 58–66.

A. a. M. A. a. Alnaser, S. S. Saloum, A. A. Sharadqh, H. Hatamleh, Optimizing multi-tier scheduling and secure routing in edge-assisted software-defined wireless sensor network environment using moving target defense and ai techniques, Future Internet 16 (2024) 386.

D. Z. Al-Hamid, P. A. Karegar, P. H. J. Chong, A novel sdwsn-based testbed for iot smart applications, Future Internet 15 (2023) 291.

B. Alzahrani, N. Fotiou, Securing sdn-based iot group communication, Future Internet 13 (2021) 207.

IEEE, IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks–Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, Technical Report, Institute of Electrical and Electronics Engineers, 2021. URL: *https://doi.org/10.1109/IEEESTD.2021.9363693*. doi:10.1109/IEEESTD.2021.9363693.

A. Llinas, M. Micheletto, R. Santos, S. Ochoa, Software defined wireless networks with real-time constraints, in: Proceedings of the 12th Latin- American Symposium on Dependable and Secure Computing, LADC '23, Association for Computing Machinery, New York, NY, USA, 2023, p. 226–229. URL: *https://doi.org/10.1145/3615366.3625076*. doi:10.1145/3615366.3625076.

NS-3 Project, NS-3: A Discrete-Event Network Simulator, 2025. URL: *https://www.nsnam.org/*, accessed: 2025-04-04.

D. González Romero, PoFi-SDN-WiFi: Simulation of a Cognitive Access Point with SDN and QoS, 2025. URL: *https://github.com/dainiergonzalezromero/WiFi-QoS-NS3.git*, *https://github.com/dainiergonzalezromero/WiFi-QoS-NS3.git*.

L. Systems, White Paper: Wi-Fi operation models, Technical Report White Paper WLAN-Management, LANCOM Systems GmbH, Ade- nauerstr. 20/B2, 52146 Wuerselen, Germany, 2018. URL: *https://www.lancom-systems.fr/fileadmin/download/documentation/Whitepaper/WP_WLAN-Management_EN.pdf*, accessed: September 2025.

S. Rojanala, Introductory overview of Wi-Fi, WLAN Architecture, Switch, Router, Gateway, Subnet, Firewall & DMZ, and their role in the world of Enterprise Wi-Fi, Technical Report CWNP White Paper, CWNP CWNE Candidate White Paper Series, 2022. URL: *https://www.cwnp.com/uploads/introductory-overview-of-wi-fi-wlan-architecture-switch-router-gateway-subnet-firewall-&-dmz-and-their-role-in-the-world-of-enterprise-wi-fi.pdf*, accessed: September 2025.

I. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, A survey on sensor networks, IEEE Communications Magazine 40 (2002) 102–114.

T. Luo, H.-P. Tan, T. Q. S. Quek, Sensor openflow: Enabling software-defined wireless sensor networks, IEEE Communications Letters 16 (2012).

M. Yan, J. Casey, P. Shome, A. Sprintson, A. Sutton, Ætherflow: Principled wireless support in sdn, in: 2015 IEEE 23rd International Conference on Network Protocols (ICNP), 2015, pp. 432–437. URL: *https://arxiv.org/abs/1509.04745*. doi:10.1109/ICNP.2015.9. arXiv:1509.04745.

Z. Guan, L. Bertizzolo, E. Demirors, T. Melodia, Wnos: Enabling principled software-defined wireless networking, IEEE/ACM Trans. Netw. 29 (2021) 1391–1407.

Z. Shi, Y. Tian, X. Wang, J. Pan, X. Zhang, Po-fi: Facilitating innovations on wifi networks with an sdn approach, Computer Networks 187 (2021)107781.

T. Theodorou, L. Mamatas, Denis-sdn: Software-defined network slicing solution for dense and ultra-dense iot networks, 2023. URL: *https://arxiv.org/abs/2312.13662*. arXiv:2312.13662.

L. Galluccio, S. Milardo, G. Morabito, S. Palazzo, Sdn-wise: Design, prototyping and experimentation of a stateful sdn solution for wireless sensor networks, in: 2015 IEEE Conference on Computer Communications, 2015, pp. 513–521. doi:10.1109/INFOCOM.2015.7218418.

R. C. A. Alves, D. A. G. d. Oliveira, G. A. Núñez Segura, C. B. Margi, It-sdn: Improved architecture for sdwsn, in: Proceedings of the XXXV Brazilian Symposium on Computer Networks and Distributed Systems, Sociedade Brasileira de Computação, Belem, Brazil, 2017, pp. 15–19.

B. Heller, R. Sherwood, N. McKeown, The controller placement problem, in: Proceedings of the First Workshop on Hot Topics in Software Defined

Networks, HotSDN '12, Association for Computing Machinery, New York, NY, USA, 2012, p. 7–12. URL: *https://doi.org/10.1145/ 2342441.2342444*. doi:10.1145/2342441.2342444.

A. Mudvari, L. Tassiulas, Joint sdn synchronization and controller placement in wireless networks using deep reinforcement learning, in: NOMS 2024-2024 IEEE Network Operations and Management Symposium, 2024, pp. 1–9. URL: https://arxiv.org/abs/2311.05582. doi:10.1109/NOMS59830.2024.10575746.

R. W. Coêlho, R. A. Silva, L. A. F. Martimiano, E. J. Leonardo, Iot and 5g networks: A discussion of sdn, nfv and information security, Journal of the Brazilian Computer Society 30 (2024) 212–227.

D. Yang, W.-T. Tsai, Sdn-based congestion control and bandwidth allocation scheme in 5g networks, Sensors 24 (2024) 749.

I. Ellawindy, S. Shah Heydari, Crowdsourcing framework for qoe-aware sd-wan, Future Internet 13 (2021) 209.