

ISAC-Assisted DRL for Dynamic MAC Scheduler Reconfiguration in O-RAN

NECO VILLEGAS^{ID¹}, JUAN LUIS HERRERA^{ID²}, LUIS DIEZ^{ID¹}, DOMENICO SCOTECE^{ID³},
LUCA FOSCHINI^{ID³} (Senior Member, IEEE), AND RAMÓN AGÜERO^{ID¹} (Senior Member, IEEE)

¹Universidad de Cantabria, 39005 Santander, Spain

²Distributed Systems Group, 1040 Vienna, Austria

³Università di Bologna, 40136 Bologna, Italy

CORRESPONDING AUTHOR: N. VILLEGAS (neco.villegas@unican.es)

This work was supported in part by European Commission's Horizon Europe, Smart Networks and Services Joint Undertaking, Research and Innovation Program through the Project 6G-SENSES under Grant 101139282 and through the Project MULTIX under Grant 101192521; in part by the Skłodowska-Curie Actions (MSCA) Project A Framework for a Performant and Sustainable Ultra-Reconfigurable Computing Continuum (RENOS) under Grant 101205037; in part by Spanish Government through the Project MICIU/AEI/10.13039/501100011033/FEDER under Grant PDC2022-133465-I00, Grant PID2021-124054OB-C31, and Grant TED2021-130913B-I00; in part by Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, Partnership on "Telecommunications of the Future" (PE00000001-Program "RESTART"), under Grant CUP: J33C22 002880001; in part by the Regional Government of Cantabria through the Technological Network for Industrial Competitiveness (TCNIC) Program under Grant 2023/TCN/002 and through the Concepción Arenal Program under Grant UC-23-40; and in part by the Department of Economy, Science and Digital Agenda of the Government of Extremadura under Grant GR21133.

ABSTRACT This paper introduces reconfiguration mechanisms for a 5th Generation (5G) Medium Access Control (MAC) scheduler, which uses Deep Reinforcement Learning (DRL) and is deployed as an *xApp* within the RAN Intelligent Controller (RIC) framework. The objective is to optimize the allocation of radio resources in 5G networks, seeking to meet Quality of Service (QoS) requirements while minimizing resource consumption. To this end, we propose the dynamic adjustment of configurable parameters in a Lyapunov-based scheduler, which addresses the challenges posed by the highly dynamic network environment, and the need to meet multiple QoS objectives. The DRL agent dynamically reconfigures the scheduling by leveraging Integrated Sensing and Communications (ISAC)-provided sensing data alongside conventional communication metrics. Exploiting the adaptability of DRL, our solution can effectively respond to fluctuating network conditions, thereby continuously enhancing scheduling decisions in real time. The proposal is evaluated through comprehensive simulations conducted over ns-3 5G-LENA, which demonstrate notable improvements in QoS performance and resource efficiency. In the analyzed scenarios, our proposed scheduling solution achieves a 30% reduction in radio resource utilization while maintaining full compliance with QoS requirements. These results highlight the great potential of exploiting DRL and sensing data to optimize MAC scheduling in modern wireless communication systems, offering a scalable and adaptive solution for 5G and future 6G networks.

INDEX TERMS 5G, ISAC, DRL, RIC, QoS, Lyapunov, ns-3.

I. INTRODUCTION

HISTORICALLY, Radio Access Networks (RANs) have been deployed using a closed, opaque model, with proprietary hardware, software, and interfaces [1]. With the emergence of 5th Generation (5G) networks and the widespread adoption of 5G New Radio (NR), RANs have begun to adopt a more open architecture. Specifically, the Open Radio Access Network (O-RAN) Alliance fostered the development of a new RAN architecture following a non-proprietary approach [2]. The basic idea of O-RAN is to use standardized and open interfaces for all communications between hardware and software

components. This ensures interoperability across different vendors.

O-RAN introduces two different RAN management solutions: i) one closer to real-time tasks, named Near-Real Time RAN Intelligent Controller (Near-Real Time RIC); and ii) a second one with more loose time constraints, named Non-Real Time RAN Intelligent Controller (Non-Real Time RIC). The RIC is a set of software elements that interact with the RAN to gather information and send control commands [3]. A primary feature of RICs is their ability to allow developers to deploy services or applications that interact directly with the RAN. In particular, near-real time RAN applications are

the so-called *xApps*, while applications close to non-real time tasks are named *rApps*.

xApps and *rApps* can control a variety of mechanisms within O-RAN. Among these, one key mechanism at the access elements, to ensure appropriate Quality of Service (QoS) levels, is Medium Access Control (MAC) scheduling, which determines the distribution of radio resources among different traffic flows. While 5G NR provides the necessary architectural support, including *xApp*-based control, it leaves the design of the particular Medium Access Control (MAC) scheduling algorithms open. Thus, deploying effective schedulers that align with the new QoS management framework introduced by 5G is of paramount importance.

A fundamental element of the 5G QoS framework is the concept of QoS flows, each given a 5G QoS Identifier (5QI). The 5QI establishes QoS requirements for each traffic flow, encompassing parameters such as latency, reliability, priority, and allowing the definition of a Guaranteed Flow Bit Rate (GFBR). The categorization of traffic into a number of flows with distinct QoS characteristics enables more granular control over resource allocation in 5G NR. For instance, flows with a higher 5QI priority may be allocated additional resources to meet strict latency or reliability requirements, which are crucial for applications such as eXtended Reality (XR). This differentiation enables the network to adapt its configuration to appropriately tackle varying demands and service requirements.

Nevertheless, attaining this degree of QoS consistency would require a MAC scheduler, that oversees resource distribution in real time, to balance the necessities of highly heterogeneous flows. It is therefore critical to develop schedulers that align with the QoS framework, to realize the full potential of 5G and so ensure that diverse applications and services receive appropriate resource prioritization.

Very recently, Integrated Sensing and Communications (ISAC) has emerged as a key research focus for 5G and future 6G, unifying communication and sensing in a single framework [4]. The communication network itself functions as a large-scale sensor, where network elements transmit and receive wireless signals, leveraging their transmission, reflection, and scattering to sense and interpret the physical environment more effectively. This enables advanced services such as high-precision positioning, passive object detection and tracking, and motion recognition extracted from wireless signals. At the same time, sensing capabilities might be used to optimize and improve communication performance.

Based on our prior work [5], in this work we substantially extend Reinforcement-Based Intelligent Scheduling (RBIS) as a MAC scheduling solution for 5G O-RAN, based on the combination of Lyapunov optimization and Deep Reinforcement Learning (DRL) within an ISAC-enabled scenario. RBIS combines the capacity of Lyapunov optimization theory to address a dynamic optimization of the MAC scheduling process based on stability guarantees, with the potential of DRL to generalize and adapt scheduling decisions to evolving network states by learning from past actions and

exploiting real-time information obtained through user and environmental sensing.

The contributions of this paper are:

- Building upon our prior hierarchical MAC scheduling framework [5], we introduce the integration of ISAC-derived sensing data into the DRL state space, enabling the DRL agent to anticipate channel quality degradations due to obstacles and proactively adjust per-UE scheduling weights.
- We define the integration of the proposed solution within the O-RAN architecture, where the DRL agent is deployed as an *xApp* at the RIC. The DRL agent is trained *online* through a real-time REST API interface between the ns-3 simulator and the emulated *xApp*.
- The proposed MAC scheduler is implemented within the ns-3 5G-LENA [6] framework and shown to guarantee the required data rate while optimizing resource utilization under realistic conditions, including a 3GPP-compliant propagation model and dynamic UE mobility. A comprehensive performance evaluation benchmarks RBIS against state-of-the-art scheduling algorithms and a comparative analysis of different DRL approaches are provided. The implementation of the MAC scheduler in ns-3, along with an emulated *xApp*, is publicly available through a repository.¹

The rest of the paper is structured as follows. Section II discusses prior work on MAC scheduling, QoS management, and ISAC, emphasizing the gaps we address in this paper. Section III provides the necessary background on QoS management in 5G and the O-RAN architecture, serving as a foundation for the proposed approach. Section IV introduces RBIS, our proposed scheduler, which exploits DRL to refine its operation. Its performance is discussed in Section V, which also validates the correct operation of our proposed solution, by means of an extensive experiment campaign over realistic setups, based on ns3 5G-LENA. Finally, Section VII concludes the paper and provides an outlook of our future work.

II. RELATED WORK

The authors in [7] present a thorough survey of the most widespread MAC schedulers in the literature. They highlight that one of the main limitations of the widely implemented Round Robin (RR) algorithm is its rigid approach to fairness. To address this limitation, several channel-aware scheduling strategies have emerged. For example, the Maximum Rate (MR) scheduler fosters users with the highest Modulation and Coding Scheme (MCS), while the best Channel Quality Indicator (CQI) scheduler allocates resources based on channel quality. On the other hand, the Proportional Fairness (PF) algorithm was proposed to better balance resource distribution. While these schemes lead to either increased spectral efficiency or resource consumption, the latest proposals have paid attention to meeting specific QoS requirements.

¹<https://github.com/tlmat-unican/RBIS>

Furthermore, the PF metric has been extensively used as a configurable parameter in various scheduling solutions. For instance, [8] introduces a demand-based PF scheduler that prioritizes users with lower data rate requirements and favorable channel conditions to enhance user satisfaction. In [9], a generalized PF approach is proposed to balance throughput and fairness, discussing how tuning the PF metric affects both targets and emphasizing the critical role of MAC scheduling in QoS satisfaction and energy efficiency. More recently, [10] combines the PF metric with 5QI parameters to explicitly guide resource allocation, with the proposed scheduler evaluated in ns-3 against conventional schemes.

Research on applying Lyapunov optimization to MAC scheduling across various network contexts is relatively limited. In [11], a MAC and power adaptation framework based on Lyapunov stochastic optimization is introduced. The formulation distinguishes between stochastic and deterministic problems: the former is solved through convex optimization, whereas the latter relies on max-weight scheduling or non-cooperative game theory. This modular design enables adaptability across different MAC configurations with tunable parameters, achieving optimal network utility and supporting distributed beam scheduling. However, appropriate tuning of these parameters is not addressed in the paper, and are assumed to be fixed during network configuration. Lyapunov methods have also been applied to other radio resource management problems in 5G slicing scenarios. In [12], resources are dynamically allocated among multiple network slices to meet heterogeneous requirements. A different direction is pursued in [13], where a zero-latency MAC protocol for cyber-physical systems is developed. Instead of suppressing interference, the scheme exploits it to improve remote state estimation. Closed-form stability conditions derived via Lyapunov theory confirm scalability and robust system performance. Finally, [14] applies the drift-plus-penalty algorithm to 5G scheduling, addressing RB allocation and packet drop control. This approach minimizes long-term packet losses while guaranteeing queue stability, bounded delays, and GFBF compliance through admission control. These methods require network managers to manually tune the Lyapunov system to the needs of the network, which can prove complex.

There is a growing interest in applying Artificial Intelligence (AI) to 5G NR. The authors in [15] present a framework combining Lyapunov optimization and DRL for online computation offloading in mobile-edge computing networks. Similarly, other works [16], [17] introduce Lyapunov-guided DRL models to realize the trade-off between a penalty (e.g., energy consumption) and queue stabilization. In these approaches, the Lyapunov objective function transforms a long-term objective optimization problem into subproblems that are directly used as the DRL reward signal, so that the DRL agent learns a policy that minimizes the Lyapunov objective itself. Furthermore, [18] proposes a Reinforcement Learning (RL) scheduler that uses various resource allocation policies to minimize latency and

packet drop rate and so satisfy QoS requirements. The authors conclude that the application of a single scheduling rule is ineffective in meeting the objectives of dynamic network conditions and heterogeneous traffic types, requiring more complex systems.

Recent research on AI focuses on techniques for resource management at higher layers, where systems can tolerate greater latency in decision-making processes [19]. The application of these algorithms for lower-level resource management and slot-level scheduling poses significant challenges, due to timing constraints and the limited or outdated feedback. For instance, [20] introduced a random forest model to predict user modulation and MCS, effectively removing the reliance on the Channel State Indicator (CSI) feedback scheme. In another study, [21] used a Machine Learning (ML) algorithm to forecast buffer occupancy and CQI values in the MAC scheduler, based on historical data. This work proposed leveraging one-time resource allocation for a certain time interval through Prediction Based Scheduling (PBS), to save Physical Downlink Control Channel (PDCCH) resources and reduce resource allocation time. However, this approach operates under the assumption that the channel remains stable, without significant fluctuations, during the allocated periods. Additionally, traditional RL methods, such as Q-learning and Policy Gradient, are often limited to situations with small discrete action spaces. As the number of active users increases, an optimization scheme using a discrete action space becomes increasingly complex and challenging [22]. The growing interest in applying AI to RAN control naturally aligns with the principles of the O-RAN architecture. O-RAN explicitly promotes intelligent, data-driven control through standardized interfaces and the deployment of rApps and xApps within the Non-RT and Near-RT RIC, respectively, enabling the integration of advanced learning-based methods into operational networks [23], [24].

ISAC technology is recognized as a key breakthrough for optimizing resource allocation and enabling mutual benefits between radar sensing and wireless communications. Several studies have considered optimizing the integration of sensing and communications by providing sufficient resources to each [4]. However, very few works, at the time of writing, leverage sensing information to directly enhance communications. The proposed ISAC scheme in [25] exploits sensing capabilities to monitor channel dynamics, identify blockages, and track user mobility, thereby enabling fast and reliable handovers along with adaptive beam switching. Moreover, in [26], [27], and [28], the authors propose leveraging ISAC-provided sensing data, such as target position and speed estimates, to enhance communication in the Internet of Vehicles (IoV).

To the best of our knowledge, no other work has specifically addressed the online configuration of a MAC scheduler based on Lyapunov optimization in 5G/6G networks using a hierarchical, two-timescale architecture. In other works, the application of DRL is restricted to the selection of alternative scheduling rules or to address subproblems within

a predefined scheduling policy. Other approaches leverage Lyapunov-guided DRL, using the Lyapunov objective function to guide the agent learning through the reward, while RBIS uses DRL to periodically adapt the Lyapunov objective itself. RBIS leverages the potential advantages of integrating DRL with dynamic configuration schemes, which would lead to more responsive and efficient scheduling decisions. As a result, the Lyapunov objective remains the core optimization criterion of the underlying MAC scheduler, while the DRL agent operates at a higher timescale. This design turns Lyapunov optimization into a configurable mechanism whose parameters are adapted by DRL over long windows (~ 1 second), instead of collapsing Lyapunov control and DRL into a single slot-level policy (≤ 1 ms). Consequently, the DRL agent has the required time to make decisions, which are tailored to the scheduler and aligned with specific QoS requirements. Time constraint challenges are then effectively addressed by enabling the scheduler to be the responsible of handling the most demanding real-time decisions. Furthermore, while several works have addressed the joint optimization of sensing and communications, few works exploit sensing information to enhance communication performance. Advances in ISAC technology now provide high-quality information on user positions, obstacles, and other relevant context. The application of sensing data to MAC scheduling in 5G/6G networks, however, remains unexplored. By exposing such information to a DRL-based xApp, our approach introduces a novel mechanism for real-time resource allocation that jointly leverages network metrics and sensing insights to improve both QoS and resource efficiency.

III. BACKGROUND

This section provides the technical foundations to understand our approach, covering the QoS management in 5G NR and the fundamentals of the O-RAN architecture.

A. QOS MANAGEMENT WITHIN 5G NR

A major advancement in 5G QoS management is the clearer and more practical separation of responsibilities between the Core Network (CN) and the RAN [29]. The User Plan Function (UPF) enables Service Data Flows (SDFs) to be mapped onto QoS flows. As a result, the QoS flow represents the finest level of granularity for QoS differentiation within the PDU session. All user-plane traffic within a session that shares the same QoS Flow Identifier (QFI) is grouped under a single QoS flow profile, which specifies how the traffic should be handled. This profile is identified by a standardized 5QI [29], which defines its resource type, Guaranteed Bit Rate (GBR) or Non-GBR, along with other parameters like priority and delay budget. This allows the RAN to understand the flow's requirements without needing complex signaling.

For GBR flows, one key parameter is the GFBR, which specifies the minimum bit rate that the network guarantees for a given QoS flow over a defined averaging time window. This ensures that the QoS flow consistently attains the required throughput, meeting the performance needs of the corresponding application or service.

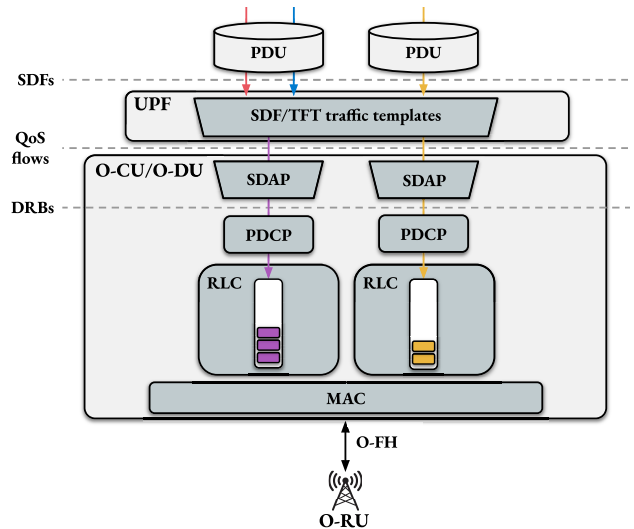


FIGURE 1. 5G RAN and CN QoS management decoupling through UPF mapping of SDF onto QoS flows.

QoS flows can be constructed from SDFs using either a one-to-one or many-to-one mapping, as illustrated in Figure 1. For example, in XR sessions, the many-to-one mapping is preferred, combining all application-generated SDFs into a single flow to maintain uniform QoS. The Service Data Adaptation Protocol (SDAP) manages these flows and supports both mapping strategies, adapting to the specific requirements of different services.

B. O-RAN ARCHITECTURE AND XAPPS

The O-RAN architecture, specified by the O-RAN Alliance, establishes an open, intelligent, and disaggregated paradigm for cellular networks [30]. As shown in Figure 2, it decomposes the traditional RAN into functional entities, such as the O-RU, O-DU, O-CU-UP, and O-CU-CP, interconnected through standardized interfaces [1]. This disaggregation facilitates fine-grained control and optimization through RIC. The Near-Real Time RIC, which operates within near-real-time control loops (10ms to 1s), communicates with the gNB via the E2 interface and hosts custom logic applications known as xApps. Conversely, the Non-Real Time RIC supports non-real-time optimization (greater than 1s) via the O1 interface [30].

The primary role of the Near-Real-Time RIC (highlighted in purplish in Figure 2) is to execute control and optimization tasks at a near-real-time scale. To achieve this, the Near-RT RIC hosts a set of xApps — lightweight microservices designed to perform specific, tightly scoped control functions [31]. These xApps interact directly with the RAN through standardized interfaces, enabling functionalities such as network slicing, mobility management, interference mitigation, and retransmission monitoring.

IV. SYSTEM MODEL

This section introduces the proposed architecture and scheduling solution. First, it outlines the overall O-RAN-based architecture, and how it is enhanced with ISAC

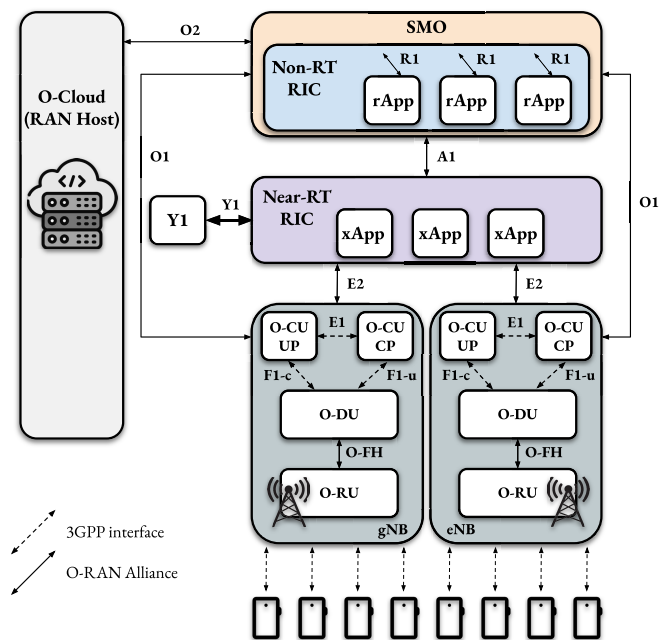


FIGURE 2. O-RAN Architecture with RIC components and standardized interfaces.

capabilities, discussing how sensing data is made available to the network. Then, it describes the RBIS MAC scheduler, outlining its configurable parameters and their role to keep traffic queues stability, meeting QoS requirements, and minimizing resource usage. Finally, it introduces the DRL agent, deployed as an xApp, which dynamically reconfigures the scheduler by tuning its parameters, leveraging network conditions and sensing data.

A. PROPOSED ARCHITECTURE

Figure 3 provides the overview of the proposed architecture. The scheduler is responsible of the RB allocation among UEs, considering the RLC queues and the GFBR, by maintaining the stability of virtual queues (scale of millisecond), with the additional goal of using resources efficiently. In this context, RBIS is deployed as an O-RAN xApp, enabling the reconfiguration of the scheduler in real time (scale of second).

Several metrics are continuously collected from the RAN, such as the throughput achieved by each UE, the occupancy of their RLC buffers, and the resource utilization resulting from previous MAC scheduling decisions. These metrics provide a real-time view of the network conditions and the users' traffic dynamics. The MAC scheduler leverages this information to make slot-level RB allocation decisions, aiming to satisfy the QoS requirements defined by the GFBR, while maintaining fairness among users and an efficient utilization of radio resources.

Moreover, ISAC features are exploited, enabling dedicated sensing devices [32] or integrated sensing functions within the base stations [4] to estimate the position and mobility patterns of users, as well as detect the presence of obstacles and environmental blockages that may impact signal propagation. This additional layer of context-awareness enhances

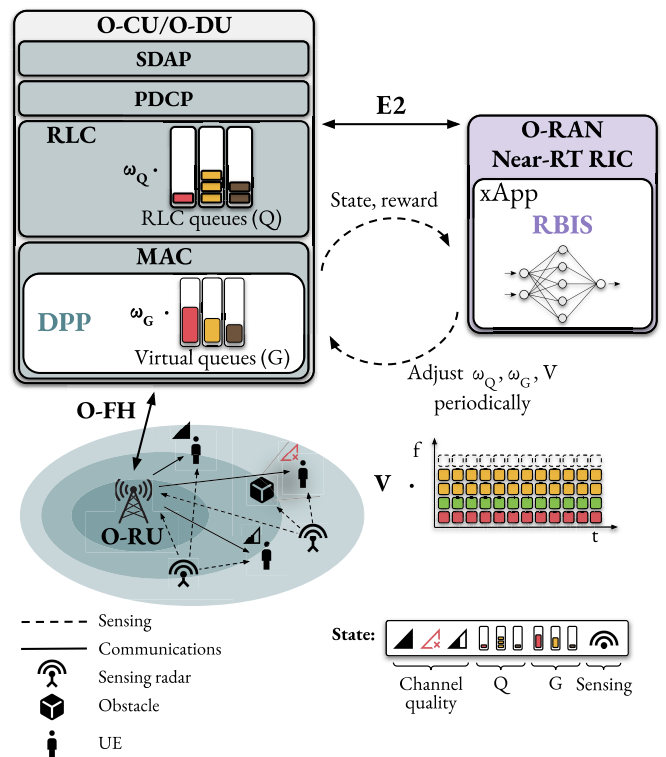


FIGURE 3. Overview of the proposed 5G O-RAN MAC scheduling architecture.

the scheduler's situational understanding, allowing it to anticipate variations in channel quality and to proactively adapt the resource allocation.

The aforementioned RAN information, together with the sensing data, is transmitted from the RAN to the near-real-time RIC through the E2 interface. There, it becomes available to the xApp, where the DRL agent exploits this multi-source information to dynamically optimize the MAC scheduler configuration. By continuously learning from both communication and sensing feedback, the xApp enables the network to respond to rapid variations in traffic demand, user mobility, and radio conditions, ultimately improving throughput, stability, and overall spectral efficiency.

B. DRIFT-PLUS-PENALTY SCHEDULING

In the following we will focus on downlink scheduling, but a similar approach can be straightforwardly applied to the uplink. For the remainder of this section, Table 1 provides a notation summary, where elements are listed in order of appearance within the paper. Let $\mathcal{N}$ represent the set of Radio Link Control (RLC) queues. A QoS flow is mapped to a corresponding RLC entity (or queue), which ensures the QoS parameters are respected. We define Γ_n as the GFBR value of the n^{th} flow. The Drift-Plus-Penalty (DPP) scheduler originally proposed in [33] aims to maintain the stability of the traffic queues, while ensuring the GFBR and an efficient resource utilization. To achieve this, a decision regarding the Resource Block (RB) allocation is made by DPP within a given slot time. The duration of this slot, t , and

TABLE 1. List of formulation notations.

Notation	Meaning
$\mathcal{N}$	Set of RLC queues
$\Gamma_n(t)$	GFBR value for the n^{th} flow at time slot t
t	Time slot in the Lyapunov formulation
$\mathcal{A}$	Available resources
$Q_n(t)$	Backlog of the n^{th} queue at time slot t
$\alpha_n(t)$	Number of RBs assigned to the n^{th} flow at time slot t
$a_n(t)$	Incoming traffic at the n^{th} queue at time slot t
$b_n(t)$	Outgoing traffic at the n^{th} queue at time slot t
$\omega(t)$	General random parameters (e.g., Signal to Interference and Noise Ratio (SINR)) at time slot t
$G_n(t)$	Virtual throughput queue for the n^{th} flow at time slot t
$D_n(t)$	Throughput deficit for the n^{th} flow at time slot t
$\rho_n(t)$	Throughput obtained by the n^{th} flow at time slot t
V	Parameter that adjusts the importance of resource efficiency
ω_{Q_n}	Parameter that adjusts the importance of stabilizing the RLC queue for the n^{th} flow
ω_{G_n}	Parameter that adjusts the importance of stabilizing virtual throughput queues for the n^{th} flow
τ	Time step for the DRL agent
$\mathcal{T}$	Maximum number of time steps
$\mathcal{S}$	Space state
$\mathcal{A}$	Action space
$S(\tau)$	State of the environment at time step τ
$A(\tau)$	Action taken by the agent at time step τ
$R(S(\tau), A(\tau))$	Reward given to the agent for taking action $A(\tau)$ in state $S(\tau)$
$\pi(A(\tau) S(\tau))$	Policy of the agent: probability distribution of choosing action $A(\tau)$ given state $S(\tau)$
γ	Reward discount rate
$G(\tau)$	Discounted return at time step τ
$V(\pi, s)$	State-value function for policy π and state s
$Q(\pi, s, a)$	Action-value function for policy π , state s , and action a
$\mathcal{O}$	Observation space
M_n	MCS used for the n^{th} flow
Λ	Available Physical Resource Blocks (PRBs)
$\bar{Q}_n$	Average buffer size for the n^{th} flow
$\bar{G}_n$	Average virtual queue size for the n^{th} flow
P_n	Channel degradation prediction for the n^{th} flow
T	Weighting factor for throughput in the reward function
B	Reward bonus for satisfying the throughput requirement
$\mathbf{1}_{\{R_{thput}=1\}}$	Indicator function for assessing the fulfillment of the throughput requirement

the amount of available resources, denoted as $\mathcal{A}$, depend on the configuration of the system, particularly the numerology and the bandwidth.

Let $Q_n(t)$ denote the backlog of the n^{th} queue at time slot t , and $\alpha_n(t)$ the scheduling decision, i.e. the number of RBs assigned to the n^{th} flow. The traffic arrivals at the n^{th} queue are represented by $a_n(t)$ and, in general, the outgoing traffic can be defined as $b_n(t) = \hat{b}_n(\alpha(t), \omega(t))$, where $\omega(t)$ holds for general random parameters, such as the SINR. Consequently, the dynamics of the traffic queues are described by Eq. 1.

$$Q_n(t+1) = \max\{Q_n(t) - b_n(t), 0\} + a_n(t) \quad (1)$$

The average window time, defined by the 5QI, indicates the time over which the GFBR shall be calculated. The default value for GBR and Delay-Critical GBR resource types

is set at 2 seconds for each standardized 5QI value. The scheduling decisions are made in much shorter time units, order of milliseconds, and the throughput requirement is thus defined as a time average objective in DPP. For this purpose, a virtual throughput queue, $G_n(t)$, is defined as follows: $G_n(t+1) = \max\{G_n(t) + D_n(t), 0\}$. It depends on the throughput deficit $D_n(t) = \Gamma_n - \rho_n(t)$, where $\rho_n(t)$ corresponds to the throughput obtained at the current slot, considering the scheduling decision and the SINR. As described in [33], the resulting problem can be tackled using the *min drift-plus-penalty* algorithm, which boils down to solving Problem 1 at every slot.

Problem 1:

$$\min_{\alpha(t)} V \sum_{n \in \mathcal{N}} \alpha_n(t) + \sum_{n \in \mathcal{N}} \omega_{Q_n} Q_n(t) [a_n(t) - b_n(t)] + \omega_G \sum_{n \in \mathcal{N}} G_n(t) D_n(t) \quad (2)$$

$$\text{s.t.} \sum_{n \in \mathcal{N}} \alpha_n(t) \leq \Lambda \quad \forall t \quad (3)$$

$$b_n(t) \leq Q_n(t) \quad \forall n \in \mathcal{N}, \forall t \quad (4)$$

The first term of the objective function in Problem 1, the *penalty*, implies the minimization of the allocated resources. The second and third terms correspond to the so-called *drift*. The objective is to keep this *drift* as small as possible, thereby reducing the delay and ensuring that throughput requirements are met, while minimizing the resource usage via the *penalty* term. Furthermore, some configurable parameters are introduced. These are: V , which adjusts the importance given to resource efficiency; and ω_Q and ω_G , which do the same for the goal of stabilizing RLC and virtual throughput queues, respectively.

The optimal values of these weights depend on the flow characteristics and the scenario. For instance, in the event of high traffic rates, the algorithm may prioritize the stabilization of RLC queues over GFBR compliance. Consequently, increasing ω_G in the objective function could be seen as a strategy to enhance the performance. On the other hand, if the system capacity is enough to meet the needs of all traffic flows, it may be a better policy to adjust the value of V according to the other weights. This fosters minimum resource usage, while ensuring GFBR and traffic queue stability.

Figure 4 illustrates the average throughput achieved by some UEs sending XR traffic. In particular, the figure shows the throughput observed by Virtual Reality (VR) and Cloud Gaming (CG) UEs for various illustrative and static configurations using the DPP scheduler. The specific configuration in question, i.e. a combination of V , ω_Q and ω_G , implies a distinct set of allocations and adjustments that influence resource distribution, throughput, and latency management. Therefore, every configuration parameter is designed to prioritize the optimization of a particular Key Performance Indicator (KPI). Dashed lines on Figure 4 correspond to throughput targets for each traffic type, i.e. their GFBR. Besides, channel conditions

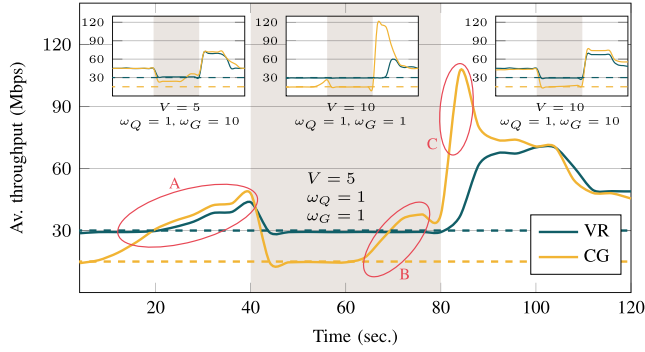


FIGURE 4. Example of throughput performance upon different configurations.

were worsened within the time interval highlighted by the shaded area (between seconds 40 and 80). Furthermore, red circles, A, B and C, correspond to some regions where the behavior of the scheduler is most noticeably affected by the particular scheduler configuration. We argue that an appropriate reconfiguration in these regions might help to mitigate abrupt changes in the performance, caused by unexpected and random variations in the environment, such as channel degradation, or changes in user demands, in terms of either traffic volume or QoS requirements. As an example, increasing the ω_G parameter leads to a throughput flattening in region C, thereby enhancing the overall performance. Given the considerable number of potential configurations and the intricacy of their real-time scenario-specific adjustment, the deployment of a DRL agent to smartly configure the scheduling policy, enabling a more efficient selection process, would allow the system to autonomously adapt, and so achieve an optimal performance.

Let $\mathcal{K} = \{G, Q\}$ denote the set of queue types for each user, and $\Theta(t)$ the global queue state matrix, where $\Theta_n^k(t)$ represents the queue backlog of type $k \in \mathcal{K}$ for the n -th UE at time slot t (i.e., $\Theta_n^G(t) \equiv G_n(t)$ and $\Theta_n^Q(t) \equiv Q_n(t)$). The parameters (V , ω_n^G , and ω_n^Q) are positive weights that satisfy the following bounds. For notational simplicity, we use ω to represent both ω^G and ω^Q .

$$0 \leq V_{\min} \leq V(t) \leq V_{\max} < \infty \quad (5)$$

$$0 < \omega_{\min} \leq \omega_i(t) \leq \omega_{\max} < \infty, \forall i, t \quad (6)$$

Consider the weighted quadratic Lyapunov function $L(\Theta(t)) \triangleq \frac{1}{2} \sum_{n=1}^N \sum_{k \in \mathcal{K}} \omega_n^k(t) \Theta_n^k(t)^2$. Let $\Delta L(t) \triangleq \mathbb{E}[L(t+1) - L(t) | \Theta(t)]$ denote the conditional Lyapunov drift.

Since the weights (V , $\omega_n^k(t)$) are piecewise constant, within each update interval the one-slot drift satisfies the following bound for some constant $B_0 < \infty$:

$$\begin{aligned} \Delta L(t) \leq B_0 + \mathbb{E} \left[\sum_{n=1}^N \omega_n^G(t) G_n(t) (r_n^* - r_n(t)) \right. \\ \left. + \sum_{n=1}^N \omega_n^Q(t) Q_n(t) (a_n(t) - b_n(t)) \mid \Theta(t) \right] \quad (7) \end{aligned}$$

In each time slot t , the scheduler observes the current state and parameters, and selects an action $\alpha(t)$, which determines the rates $r_n(t)$ and $b_n(t)$, to minimize the following expression:

$$\begin{aligned} \sum_{n=1}^N \omega_n^G(t) G_n(t) (r_n^* - r_n(t)) \\ + \sum_{n=1}^N \omega_n^Q(t) Q_n(t) (a_n(t) - b_n(t)) + V(t) \sum_{n=1}^N \alpha_n(t) \quad (8) \end{aligned}$$

Assume the average arrival rates lie strictly inside the capacity region. Comparing the scheduler's action to that of a stationary optimal policy (which exists by the capacity assumption), and leveraging the lower bound on weights ($\omega_n^k(t) \geq \omega_{\min}$), we obtain the drift inequality valid for all t within an update interval:

$$\Delta L(t) + V(t) \mathbb{E} \left[\sum_{n=1}^N \alpha_n(t) \mid \Theta(t) \right] \leq B - \epsilon \sum_{n=1}^N \sum_{k \in \mathcal{K}} \Theta_n^k(t) \quad (9)$$

for constants $B, \epsilon > 0$ that depend on $\omega_{\min}, \omega_{\max}$ and the capacity slack. The constant B is sufficiently large to absorb the penalty term $V(t) \sum_{n=1}^N \alpha_n(t)$, noting that $V(t) \leq V_{\max} < \infty$. Even when $V_{\min} = 0$ is allowed, the algorithm reduces to the max-weight policy, preserving the negative drift component $-\epsilon \sum_{n,k} \Theta_n^k(t)$ necessary for stability.

At the boundaries of these update intervals, the change in weights $\omega_n^k(t)$ introduces a discontinuity in $L(\Theta(t))$. However, since $0 < \omega_{\min} \leq \omega_n^k(t) \leq \omega_{\max} < \infty$, the value of the Lyapunov function is simply scaled by a finite factor. These finite jumps occur only when the weights are updated.

Summing over $t = 0, \dots, T-1$, dividing by T , and letting $T \rightarrow \infty$, the contribution of these finite boundary jumps vanishes (scales as $1/K$, where K is the update interval length, e.g., $K = 1000$). Consequently, the queues remain strongly stable and the time-average total queue size satisfies:

$$\limsup_{T \rightarrow \infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{n=1}^N \sum_{k \in \mathcal{K}} \mathbb{E}[\Theta_n^k(t)] \leq C V_{\max} \quad (10)$$

for some constant $C < \infty$ (i.e., $O(V_{\max})$ backlog). Moreover, the time-average throughput satisfies $\bar{r}_n \geq r_n^*$, and the penalty is within $O(1/V_{\min})$ of the optimum (or the system yields $O(1)$ backlog if $V = 0$). These bounds are insensitive to the initial condition $\Theta(0)$.

C. REINFORCEMENT-BASED INTELLIGENT SCHEDULING

This section introduces RBIS, as an efficient solution to the problem depicted in Section IV-B. First, we briefly introduce the theoretical framework behind DRL and some of its basic concepts. We then discuss how it fits within the system model. Finally, we describe the specifics and technical details on how RBIS integrates DRL.

The learning process in DRL can be described as a Markov Decision Process (MDP), a mathematical model that defines the interactions of a DRL system with a given environment and how it can be steered towards interactions deemed

better [34]. An MDP can be structured as a decision-making algorithm, known as the *agent*, and an *environment* that the agent must interact with. The agent interacts with the environment by making decisions at discrete time steps: $\tau \in 0, 1, \dots, T$. The environment comprises a *state space* S , that defines every possible representation of its status (i.e., all possible values of all metrics deemed relevant to the environment), and an *action space* A , entailing all the possible decisions the agent can make.

At each time step τ , the agent is given, as input, a certain *state* $S(\tau) \in S$, i.e., a representation of the specific situation of the environment. Based on this state, the agent takes an *action* $A(\tau) \in A$. This action entails making a decision, moving the environment to a new state $S(\tau+1) \in S$ and, as a result, receiving a *reward* $R(S(\tau), A(\tau)) \in \mathbb{R}$. This reward is the result of a *reward function*, which evaluates the fitness of the action for the given state, i.e., how good or bad the action is, considering the current state: $\mathcal{R} : S \times A \rightarrow \mathbb{R}$. The reward allows the agent to learn, steering its decisions to maximize the reward obtained over time.

Formally, the goal of the agent is to define a policy $\pi(A(\tau)|S(\tau))$ that describes the probability distribution of choosing the action $A(\tau)$ given the state $S(\tau)$, maximizing the total reward. Nonetheless, it is common to use the *discounted reward function* for this purpose, which uses a discount rate parameter ($\gamma \in [0, 1]$) to benefit agents that took good decisions in the past. Formally, the discounted return for a time step $G(\tau)$ is defined as $G(\tau) = \sum_{k=0}^T (\gamma^k \mathcal{R}(S(\tau+k), A(\tau+k)))$. Based on these definitions, DRL also leverages the *state-value function*, which defines the expected discounted return the agent gets by applying the policy π from state s onwards. Formally, the state-value function for state s and policy π is defined as $V(\pi, s) = \mathbb{E}_\pi[G(\tau)|S(\tau) = s]$. Finally, it is also necessary to define the *action-value function*, i.e., the expected discounted return from applying action a in state s , and following policy π onwards, or formally: $Q(\pi, s, a) = \mathbb{E}_\pi[G(\tau)|S(\tau) = s, A(\tau) = a]$. The objective of DRL is to learn a policy π that is as close to the optimal one as possible. DRL can thus be used for two different goals: learning a policy π , in which case the objective is to approximate the optimal policy, or learning the function $Q(\pi, s, a)$, thus trying to approximate the optimal policy by choosing the actions that would maximize the learned Q -value.

To adapt the aforementioned problem for DRL, we need to define how its elements and the underlying system model map into an MDP. The first aspect to model is the *observation space* O , which is a subspace of the state space that defines the parts of the environment the agent is able to observe. Formally, $O \subseteq S$, and hence, $O(\tau) \subseteq S(\tau) \forall t \in [0, T]$. In this specific problem, the agent is allowed to observe the complete state space, and hence, $O = S$. The state comprises the number of RLC queues $N = |\mathcal{N}| \in \mathbb{N}$ that exist in the scenario, the MCS used for each of the flows M_n , the GFBR for each flow $\Gamma_n(t)$, the total number of available PRBs Λ , the average buffer size for each flow $\bar{Q}_n$, the average size of the virtual queues defined by the DPP scheduler $\bar{G}_n$, and a

channel degradation prediction based on sensing information of UE and obstacle positions P_n (as detailed in Section IV-D). S can thus be formally defined as:

$$S = \{M_i, \Gamma_i, \Lambda_i, \bar{Q}_i, \bar{G}_i, P_i \mid i = 1, \dots, N\} \quad (11)$$

The objective of the DRL agent is to adjust the values of the three weights of the *min drift-plus-penalty* algorithm, namely, V , ω_Q , and ω_G . Hence, an action taken by the agent represents a set of values given to each of these three weights, respectively. Formally, the action space A is defined as:

$$A = \{V, \omega_{Q_i}, \omega_{G_i} \mid i = 1, \dots, N\} \quad (12)$$

The final element to map from the problem domain to the MDP is the reward function $\mathcal{R}(S(\tau), A(\tau))$. The DRL agent receives a reward based on the average throughput and resource efficiency achieved after the *min drift-plus-penalty* algorithm is executed with the weights previously chosen by the DRL agent. The specific reward function leveraged by the DRL agent combines the average value of the throughput across all flows with the fraction of free PRBs. Both rewards have been normalized between 0 and 0.5, leading to an overall reward between 0 and 1. Furthermore, to lead the agent towards desired states and ensure it does not take defective decisions (e.g., not assigning resources to maximize free PRBs), the resource efficiency part of the reward is not considered unless the GFBR is achieved. Formally, $\mathcal{R}(S(\tau), A(\tau))$ is defined as follows:

$$\begin{aligned} \mathcal{R}(S(\tau), A(\tau)) &= T \cdot \frac{1}{N} \sum_{n \in \mathcal{N}} \min\left(1, \frac{\bar{\rho}_n}{\Gamma_n}\right) \\ &+ \left[B + (1 - T - B) \left(1 - \frac{\sum_{n \in \mathcal{N}} \bar{\alpha}_n}{\Lambda}\right)\right] \cdot \mathbf{1}_{\{R_{\text{thput}}=1\}} \end{aligned} \quad (13)$$

where P is a weighting factor controlling the contribution of the throughput term, and B is a reward bonus that provides an additional incentive when the throughput requirement is satisfied. The indicator function $\mathbf{1}_{\{R_{\text{thput}}=1\}}$ is defined as:

$$\mathbf{1}_{\{R_{\text{thput}}=1\}} = \begin{cases} 1, & \text{if } \frac{\sum_{n \in \mathcal{N}} \min(1, \frac{\bar{\rho}_n}{\Gamma_n})}{N} = 1, \\ 0, & \text{otherwise} \end{cases} \quad (14)$$

Hence, we propose a two-timescale architecture combining the DPP scheduler with a high-level DRL agent. The DPP scheduler operates every slot using the current parameters $(V(t), \omega(t))$. The high-level (slow timescale) DRL agent updates these parameters every K slots (e.g., $K = 1000$).

The DRL action space is restricted to the compact set $\mathcal{P} \triangleq [V_{\min}, V_{\max}] \times [\omega_{\min}, \omega_{\max}]^{2N}$. Since every $(V, \omega) \in \mathcal{P}$ induces a stabilizing DPP policy with a uniform drift bound ($\epsilon \geq \epsilon_{\min} > 0$), the overall policy guarantees global stability and $O(V_{\max})$ backlog scaling regardless of the DRL agent's exploration or specific decisions.

Furthermore, the formulation acts as a fundamental fail-safe. Although the DRL agent typically allows physical

queue weights to settle at the lower bound $\omega_{\min}$ (as neither $Q_n(t)$ nor delay is explicitly in the reward), strong stability is inherently preserved. The underlying drift term $\sum_{n=1}^N \omega_n^Q(t) Q_n(t) \geq \sum_{n=1}^N \omega_{\min} Q_n(t)$ ensures that if physical backlogs grow significantly, this term dominates the minimization objective. This forces the scheduler to service the queues and prevents unbounded growth, fulfilling the $O(V_{\max})$ guarantee regardless of the high-level throughput optimization goals.

To instantiate this high-level DRL agent, we carefully considered the characteristics of the underlying hierarchical scheduling problem and its discrete action space $\mathcal{P}$. Specifically, we selected two distinct algorithms to drive the weight-adaptation process: Deep Q-Network (DQN) and Proximal Policy Optimization (PPO).

DQN was chosen as a highly sample-efficient, Off-Policy baseline. Because the current environment features sporadic and rare critical events, such as sudden channel fading caused by users' mobility, an algorithm utilizing an Experience Replay Buffer is theoretically well-suited to store and reuse these rare transitions, preventing the agent from discarding crucial states. Conversely, PPO is a well-known On-Policy actor-critic method. A major concern when applying DRL to network scheduling is the risk of policy collapse during training. PPO mitigates this through its clipped surrogate objective, which theoretically prevents destructively large policy updates. We selected PPO to test whether this mathematically guaranteed training stability translates into a more consistent weight-tuning performance within the compact set $\mathcal{P}$. The specific implementation details for both agents, including the corresponding neural network architectures, hyperparameter selection methodology, and all their configuration parameters are detailed in Section V, which also discusses their performance.

D. ISAC-ASSISTED SCHEDULING

The proposed architecture integrates ISAC capabilities directly into the MAC layer scheduling process. Traditional reactive schedulers struggle to maintain QoS requirements in dynamic 5G environments, where user mobility and physical obstacles frequently cause severe and sudden Line of Sight (LoS) blockages. To mitigate this, the RBIS framework leverages sensing data, and specifically, sensing-based location information, to provide the scheduler with predictive environmental awareness.

Within our system model, the ISAC functionality provides a continuous stream of sensing information regarding the physical environment. We assume that the exact positions of static physical obstacles are known a priori, which is a realistic premise in controlled or well-mapped deployments such as industrial environments, smart factories, or indoor warehouses. To abstract this sensing capability into a predictive variable P_n for each UE n , the system leverages a sliding window of the UE's historical positions to estimate its current movement trajectory. By projecting this trajectory forward, the system evaluates whether the UE is heading towards a

“high-probability blockage zone”: a region where the LoS signal from the gNB is completely obstructed by a known obstacle. If the projected path intersects with this shadowed area, the boolean indicator P_n is triggered to true.

The integration of this predictive sensing data fundamentally shifts the weight-adaptation behavior from a reactive to a proactive paradigm. Without sensing data, the scheduler increases the priority of a UE (i.e., by increasing ω_Q and/or ω_G) and relaxes the global V parameter (to reduce the penalty on RB consumption) only after observing a backlog buildup or a delayed MCS update caused by a sudden channel degradation. By that time, attempting to stabilize the rapid growth of both physical and virtual queues requires a significant amount of RBs due to the poor channel state. Conversely, by incorporating P_n into the DRL state space, the DRL agent learns to anticipate impending capacity drops and adjust the Lyapunov parameters (V , ω_{Q_n} , and ω_{G_n}) beforehand.

V. EVALUATION

This section discusses simulation results that validates the proposed scheduling solution, whose performance is compared with that of alternative solutions from the literature.

A. EVALUATION SETUP

The simulations have been performed on a server with an Intel® Xeon® E3-1241 v3 processor (4 cores, 8 threads) running at 3.50 GHz, with 32 GB of DDR3 RAM (4 × 8 GB, 1600 MHz), under Ubuntu 20.04.6 LTS.

The evaluation focuses on a single-cell scenario, consistent with the intra-cell scope of the proposed MAC scheduling framework. Specifically, it consists of a single gNB and four UEs, all within the same beam. Simulations of the RAN are carried out using ns-3 5G-LENA v4.1, a comprehensive framework that offers accurate and detailed modeling of the 5G NR PHY and MAC layers. The DPP MAC scheduler was previously integrated within 5G-LENA in [33]. All simulated UEs generate mixed 3GPP XR traffic, including two UEs configured with VR services and two with CG traffic. XR traffic is selected due to its stringent QoS requirements, and the two distinct traffic types capture variability in network behavior. The VR traffic comprises high-bandwidth and low-latency video streams for immersive virtual environments, characterized by high resolution and stringent delay requirements. Meanwhile, the CG traffic models workloads involving computation offloading to a remote server, where low-latency and reliable connectivity between the UE and cloud are critical. Both traffic types exhibit different statistical characteristics, including packet sizes, inter-arrival times, and burstiness, creating challenging demands on the RAN. Table 2 summarizes the main characteristics of the configured XR traffic. Moreover, Table 3 summarizes the configuration parameters of the considered scenario. Extending the evaluation to larger scenarios is left as future work, as discussed in Section VI.

The objective is to satisfy the specific QoS requirements of each XR UE, while minimizing the consumption of

TABLE 2. Configuration of the XR traffic.

Virtual Reality traffic		Cloud Gaming traffic	
#Applications (UEs)	2	#Applications (UEs)	2
Number of flows	1	Number of flows	1
5QI value	87	5QI value	3
Average rate offered	10 Mbps	Average rate offered	10 Mbps
FPS	60	FPS	60
Guaranteed Bit Rate	2 Mbps	Guaranteed Bit Rate	4 Mbps

TABLE 3. Configuration of the evaluation setup.

Parameter	Value
Numerology (μ)	0
Carrier frequency	6 GHz
Bandwidth	20 MHz
gNB Tx power	41 dBm
UE Tx power	23 dBm
gNB antenna height	25 m
UE antenna height	1.5 m
gNB antenna array	1 TXRU: 4×8 (3GPP elements)
gNB antenna element gain	8 dBi
UE antenna array	1 TXRU: 1×1 (isotropic element)
UE antenna element gain	0 dBi
gNB noise figure	5 dB
UE noise figure	7 dB
Building width	2 m
Building length	2 m
Building height	10 m
Noise power spectral density	-174 dBm/Hz
Propagation model	3GPP
Duplexing mode	TDD
TDD pattern	DDDDDDU
PDCCP discarding timer	5 ms for 5QI = 87, 50 ms for 5QI = 3
Mobility pattern	RandomWalk2dMobilityModel

radio resources to promote an efficient use of the available spectrum. To this end, the MAC scheduler handles the RB allocation, while an xApp is deployed to further enhance the scheduling process. Specifically, a DRL agent is trained online at the xApp to optimize the MAC scheduler performance by tuning its configurable parameters. These parameters control the relative weighting of three key objectives: stabilizing traffic queues, meeting QoS requirements and minimizing RB usage. The agent periodically receives configuration requests for the scheduler, while continuously learning and refining its policy to meet the stringent QoS demands of XR services and maximize spectral efficiency. In particular, a constant period of 1 second of simulated time is considered for the weights updates.

Communication between the emulated RAN (5G-LENA) and the xApp is performed via a REST API that emulates a real RIC interface. This design not only enables online training of the DRL agent with real-time feedback from the RAN, but also decouples the xApp from the network infrastructure, allowing seamless experimentation in simulators as well as deployment in operational testbeds.

TABLE 4. Key hyperparameters for DRL algorithms.

Hyperparameter	PPO	DQN
Network Architecture		
Policy Network (pi)	[64, 64]	[64, 64]
Value Network (vf)	[64, 64]	N/A
Activation Function	Tanh	ReLU
Common Parameters		
Learning Rate	3×10^{-4}	1×10^{-4}
Discount Factor (γ)	0.99	0.99
Batch Size	128	64
PPO Specific		
Number of Steps (n_{steps})	1280	N/A
Number of Epochs	20	N/A
Clip Range	0.2	N/A
GAE Parameter (λ)	0.95	N/A
Entropy Coefficient	0.01	N/A
Value Function Coef.	0.5	N/A
Max Gradient Norm	0.5	N/A
DQN Specific		
Replay Buffer Size	N/A	50,000
Learning Starts	N/A	1,000
Target Update Interval	N/A	1,000 steps
Train Frequency	N/A	4 steps
Exploration (ϵ)	N/A	$1.0 \rightarrow 0.02$
Exploration Fraction	N/A	0.1 (10%)

The DRL agents were implemented using the *Stable-Baselines3* library² operating within a custom Gymnasium environment. For the DQN agent, we utilized the default MlpPolicy, which consists of two fully connected hidden layers with 64 neurons each and ReLU activation functions. For the PPO agent, the neural network architecture was explicitly defined with decoupled actor (policy) and critic (value) networks. Both the policy (pi) and value (vf) branches consist of two hidden layers with 64 neurons each, utilizing the Tanh activation function.

The hyperparameter selection methodology is based on empirical tuning. We initialized the parameters using the established robust default values provided by the *Stable-Baselines3* framework, which offers reliable implementations of widely used DRL algorithms [35]. Subsequently, we conducted preliminary experiments to manually fine-tune critical parameters, such as the learning rate, batch size, and exploration fractions, adapting them to the specific time scales and state-space dynamics of our hierarchical scheduling environment. The final values, which ensured stable convergence without excessive policy oscillations, are summarized in Table 4.

To evaluate the ISAC-assisted scheduling framework, the simulation environment incorporates dynamic UE mobility, using the *RandomWalk2dMobilityModel* and fixed physical

²<https://github.com/DLR-RM/stable-baselines3>

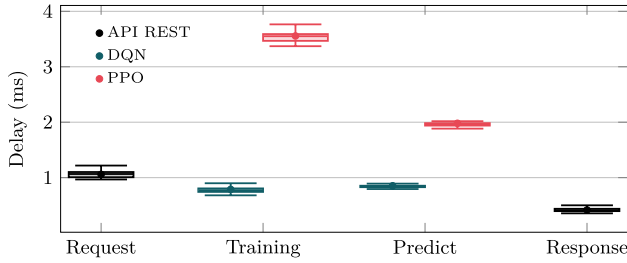


FIGURE 5. Latency overhead in the scheduler reconfiguration process.

obstacles, exploiting the ns-3 buildings module, which are capable of causing LoS blockages that severely degrade channel quality. While the simulator inherently tracks the exact instantaneous coordinates of all nodes, our framework abstracts this precise positioning information into a predictive boolean indicator, P_n , which flags whether a UE is projected to enter a high-probability blockage zone, based on its recent trajectory. Because the UEs follow stochastic mobility patterns with random changes in direction, these trajectory-based predictions are naturally imperfect. They inherently contain estimation errors, including false positives and missed detections.

Consequently, the DRL agent does not train on perfect environmental knowledge, but rather on noisy, imperfect forecasts. This forces the agent to learn a robust policy capable of extracting meaningful scheduling gains despite unreliable sensing cues. While this binary representation is a simplified abstraction, it effectively captures the stochastic uncertainty necessary to validate the ISAC integration at the MAC layer. Future evaluations in operational testbeds will further expose the system to raw hardware measurement noise.

B. OPERATIONAL VALIDATION

Figure 5 depicts the time required by RBIS to take a decision, i.e., for the DRL agent to take a step. It is critical to be as quick as possible, as this allows RBIS to react to the status of the RAN and adjust the MAC scheduling within a sensible time interval. The analysis considers the decision-making times of RBIS both during and after training, as well as the request and response times between the emulated xApp and the RAN, where the former provides the state metrics and receives the scheduler configuration in return. The delay in each case is represented using boxplots, where the lower and upper edges of each box correspond to the 25th and 75th percentiles, respectively. The whiskers extend up to 1.5 times the Inter Quartile Range (IQR), approximately covering the range between the 2nd and 98th percentiles. Filled circles within each boxplot denote the average time. As can be seen, the average latency to obtain the requested action stays in all tests below 5 ms, even in the worst case (PPO in the training phase), which is a reasonable behavior, considering that reconfigurations occur on a time scale of seconds.

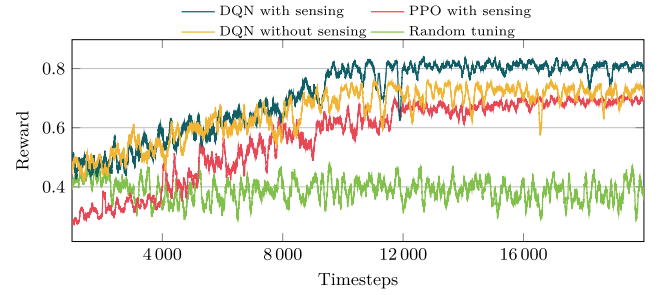


FIGURE 6. Comparison of the reward obtained over time.

TABLE 5. Reward statistics.

Agent	Trained	Sensing	Mean	Median	STD	Min - Max
Rand	—	—	0.38	0.20	0.24	0.10 - 0.92
PPO	—	✓	0.69	0.71	0.07	0.18 - 0.73
PPO	✓	✓	0.70	0.71	0.03	0.56 - 0.73
DQN	—	—	0.71	0.75	0.13	0.14 - 0.91
DQN	✓	—	0.70	0.75	0.12	0.19 - 0.91
DQN	—	✓	0.81	0.78	0.12	0.14 - 0.91
DQN	✓	✓	0.82	0.90	0.09	0.70 - 0.91

Figure 6 compares the reward over time obtained with three agents: DQN, PPO, and a random tuning baseline, where the values of V , ω_q , and ω_g are randomly selected. For DQN, both state configurations, with and without sensing data are included. In contrast, for PPO, only the configuration using sensing information is shown, as the differences were negligible. For clarity, the plotted data correspond to a moving average computed every 100 steps.

As can be seen, all agents converge after approximately 10,000 steps. During the initial training phase, RBIS is not yet ready for decision-making, making early comparisons with other methods, such as reward evaluation, unfair. However, once fully trained, RBIS can be effectively compared to random-based tuning. The random-based agent achieves rewards with an average of 0.37, a median of 0.20, and a standard deviation of 0.24. Note that the average reward remains below 0.5, with $T = 0.2$ and $B = 0.3$, indicating that the GFBR requirement is not consistently satisfied, as described in Section IV-C. It is worth noting that a reward below 0.5 ($T + B$) evinces that the GFBR is not satisfied. In contrast, RBIS achieves a narrower reward range, with an average reward of 0.81 for DQN with sensing, and 0.71 and 0.69 for DQN without sensing and PPO with sensing, respectively. RBIS using DQN with sensing also shows a median reward of 0.78 and strong stability, with a standard deviation of 0.12. All these statistics are reported in Table 5, which also includes results for the trained models, representing the agents' best learned policies. For the training agents, the statistics are computed after convergence. As observed, higher rewards are achieved with the pretrained models, which skip the exploration phase and therefore exhibit more stable and consistent performance. However,

lower minimum rewards, 0.19, are obtained by the trained DQN without sensing data, which can be attributed to the lack of environmental awareness, which prevents the agent from anticipating throughput drops when a UE temporarily moves behind an obstacle.

The results highlight a significant performance disparity between DQN and PPO, and particularly between Off-Policy and On-Policy learning paradigms. This difference is attributable to their fundamentally distinct approaches to leveraging past experiences to learn, which is a critical factor in an environment with a vast and diverse state space, where brief events, such as channel degradations caused by environmental obstacles, may occur too infrequently to be effectively learned by On-Policy methods.

The PPO agent, representative of On-Policy learning, shows a premature convergence to a safe, suboptimal policy. This behavior is characteristic of algorithms that learn exclusively from data generated by the current policy. In an environment with a wide array of possible state configurations, many of which represent common or simple scenarios, the agent quickly learns a competent baseline strategy. However, rare but critical experiences, such as discovering a high-reward opportunity may exist. Once the policy is updated, this valuable data is discarded. The agent's policy, now dominated by the more frequently encountered "safe" states, is unlikely to deliberately recreate the conditions for these low-frequency, high-reward scenarios. Consequently, the agent struggles to assign sufficient importance to these critical events and fails to escape the suboptimal strategy. Indeed, even after extensive hyperparameter tuning aimed at enforcing exploration (via the entropy coefficient) and amplifying the learning signal (by increasing the training epochs per data batch), the agent's ability to escape the suboptimal strategy remained significantly limited.

In contrast, the Off-Policy DQN agent demonstrates markedly superior sample efficiency, enabling it to discover the globally optimal policy. Its advantage stems from the use of an Experience Replay Buffer, which stores a rich history of transitions from all regions of the state space it had previously explored. When a rare event occurs, this critical transition is not discarded, but is stored at the buffer. By repeatedly sampling this experience during subsequent training steps, the DQN agent effectively amplifies the learning signal of these infrequent but crucial events. This allows it to learn their true value, even if its current policy would not visit them often. This decoupling of experience collection from the learning update makes Off-Policy methods inherently more robust and efficient for this particular scenario.

C. EVALUATION RESULTS

Initial results correspond to a synthetic scenario where channel conditions degrade in a controlled manner whenever a UE becomes obstructed by an obstacle. In this case, the sensing information that anticipates channel degradation is assumed to be perfectly reliable.

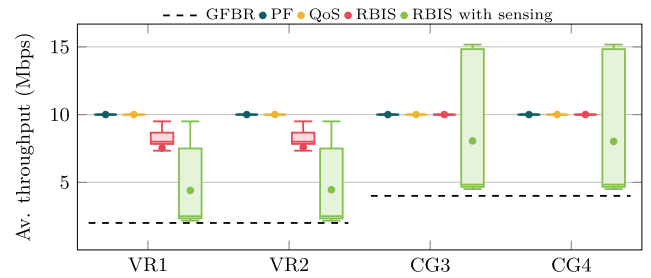


FIGURE 7. Comparison of the average throughput achieved by each UE.

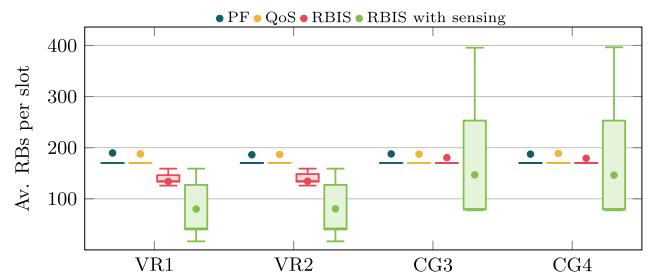


FIGURE 8. Comparison of the average number of RBs allocated to each UE.

Figures 7 and 8 illustrate the achieved throughput and resource usage, respectively. Both metrics are represented using boxplots, with circles indicating the mean values. Each boxplot corresponds to a single UE, and each color represents a different scheduler: PF, QoS-aware [10], or the DPP scheduler enhanced with 1-second dynamic reconfiguration via the xApp. The figures compare results obtained for the four UEs, showing the average values over the temporal window in which a new configuration is applied, i.e., every 1 second. For DPP, the figures depict results for its two variants: one that utilizes sensing data and one that operates without it.

As shown in Figure 7, all schedulers meet the minimum throughput requirement (GFBR), which is 2 Mbps for VR UEs and 4 Mbps for CG UEs. The difference lies in the DPP scheduler, which achieves lower throughput while carefully ensuring that the GFBR is satisfied. This controlled reduction in throughput allows a significant reduction in resource usage, as illustrated in Figure 8.

The reduction in RB usage is more clearly illustrated in Figure 9, which depicts the average number of RBs allocated per slot by each scheduler. The RBIS requires up to 36.61% fewer RBs compared to the other solutions in the current scenario, while simultaneously ensuring that the throughput requirements specified by the QoS profiles of each traffic flow are fully met. Notably, this significant RB reduction is observed even when the scheduler operates without sensing information to predict channel degradations.

To further evaluate the schedulers under more realistic conditions, a second set of results is presented. This scenario employs the 3GPP propagation model together with the ns-3 buildings module to simulate obstacle-induced signal attenuation and blockage effects. Furthermore, the *RandomWalk2dMobilityModel* is used to mimic UEs' mobility,

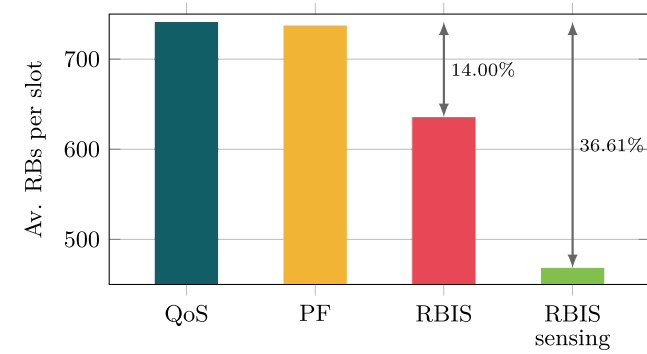


FIGURE 9. Average number of RBs allocated per slot.

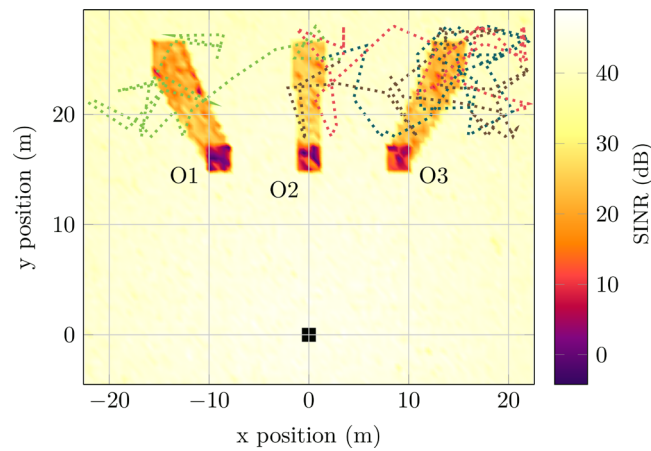


FIGURE 10. Spatial SINR distribution including obstacles and the tracking of UEs' positions during the simulation.

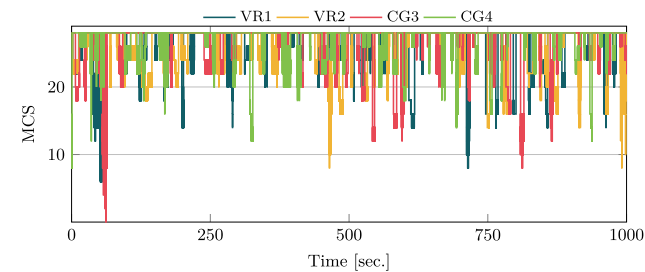


FIGURE 11. MCS per UE under 3GPP channel and random mobility models over time.

which, combined with the environmental obstacles, creates a highly dynamic and challenging radio environment, as illustrated by the SINR map in Figure 10. In the figure, the effect of the three obstacles is evident, and the random paths of the 4 UEs are overlaid to show their movement through these varying channel conditions. To further emphasize the temporal impact of the channel, Figure 11 shows the evolution of the MCS for each UE throughout the simulation.

In this setup, the evolution of the channel is fully stochastic and beyond control. The sensing data is used to predict the future UE trajectory based on its historical positions, which, as expected, introduces inaccuracies due to the random nature

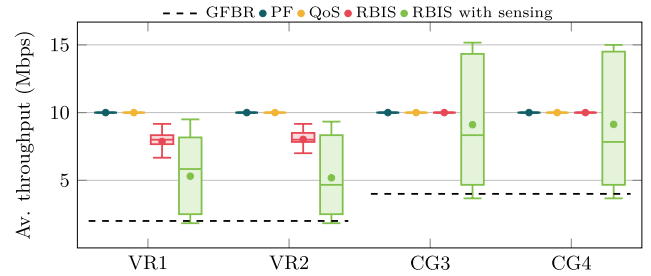


FIGURE 12. Average throughput per UE under 3GPP channel and random mobility models.

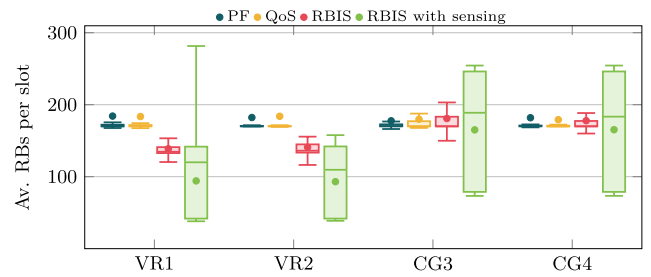


FIGURE 13. Average number of RBs allocated to each UE under 3GPP channel and random mobility models.

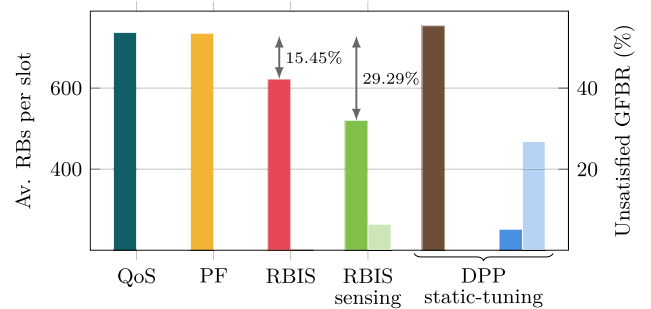


FIGURE 14. Average number of RBs allocated per slot and unsatisfied GFBR rate under 3GPP channel and random mobility models.

of the mobility model. This configuration allows assessing the scheduler's performance in a realistic and highly dynamic environment.

Similarly to the previous results, Figures 12 and 13 show the average throughput and RBs allocation, respectively. As expected, RBIS is the only scheduler that attempts to increase resource efficiency. Unfortunately, inaccuracies in the predicted trajectories of the UEs occasionally lead to temporary violations of the GFBR.

Finally, Figure 14 illustrates the achieved resource efficiency, which in this case is influenced by errors in UE mobility prediction and the inherent variability of the channel model. While the previous, more controlled scenario (Figure 9) showed an RB-per-slot reduction of nearly 37%, a significant 29.29% reduction is observed under these more realistic conditions. In contrast, the xApp operating without sensing data, unaffected by the random mobility model

(as it does not perform mobility predictions), maintains a similar level of RB reduction, as expected. Additionally, a static-tuning DPP baseline is included in this scenario to highlight the impact of DRL-based dynamic reconfigurations. Specifically, two static configurations are evaluated: one that prioritizes queue stability while relaxing resource efficiency, and another that maximizes resource reduction at the expense of stability. Furthermore, Figure 14 includes a second set of lightly colored bars depicting the unsatisfied GFBR (i.e., the percentage of times the 1-second average throughput falls below the GFBR). As expected, the more “aggressive” DPP configuration achieves the highest reduction in RB usage; however, this comes at the expense of a 26.93% of unsatisfied GFBR. The remaining approaches satisfy the GFBR in almost all circumstances, with the exception of RBIS with sensing, which exhibits an unsatisfied GFBR rate of 6.49%. This occurs because the sensing-assisted DRL agent learns to risk more aggressive resource reductions, occasionally making suboptimal decisions when sensing mismatches mask true channel degradations.

VI. FUTURE EXPERIMENTAL VALIDATION

To bridge the gap between simulation and practical deployment, our immediate future work focuses on evaluating the RBIS framework within a real-world O-RAN testbed. To this end, we plan to integrate the DPP MAC scheduler implemented in OAI and the DRL xApp deployed via FlexRIC with a mmWave ISAC hardware platform [32].

This hardware platform is built upon a mmWave Software-Defined Radio (SDR) architecture, featuring an AMD ZCU111 evaluation board and dual phased-array antennas for precise beam steering. It is capable of bidirectional data transmission at rates up to 5.8 Gbps, alongside monostatic radar detection that provides a range resolution of 6.7 cm. Operating at a sensing rate of 100 Hz, the subsystem captures channel estimations across 63 angular beams and 1024 subcarriers. Through an inverse Fast Fourier Transform, these raw measurements are continuously processed into high-fidelity environmental heatmaps, represented as 63×128 matrices mapping angle and range.

In our upcoming experimental phase, this raw, high-dimensional sensing data will enhance the results obtained from our current simulation setup. Processing these real-world environmental heatmaps directly at the Near-RT RIC will definitively validate the framework’s capability to learn and execute robust scheduling policies despite the inherent imperfections and hardware noise characteristic of physical ISAC systems.

VII. CONCLUSION

In this paper we have presented RBIS, which promotes the use of DRL for real-time 5G MAC scheduler reconfiguration. The proposed approach leverages the strengths of DRL and ISAC-provided sensing data to adaptively adjust the behavior of a MAC scheduler to optimize resource allocation between users with a wide variety of QoS requirements.

To assess the feasibility of our proposed solution, we have conducted experiments in a 5G cell populated with XR UEs, using a multi-objective Lyapunov-based MAC scheduler to manage available resources. The simulation results evince the robustness of our proposal, which consistently finds optimal configurations for the MAC scheduler. Furthermore, RBIS mitigates the risk of scheduler malfunction due to suboptimal configurations, which can arise from channel variability or the distinct user demands. Potential errors in the predictions derived from the sensing information, resulting from more realistic channel propagation and mobility models, were also considered, showing that the scheduler remains robust even under these situations. In addition, we have also studied the overhead imposed by the DRL agent, and we have seen that it is kept at reasonable levels, thus ensuring the applicability of the scheduling policy.

In our upcoming work, we plan to extend the evaluation using more complex network scenarios. Furthermore, we intend to further explore enhancements in the reward function to balance additional network objectives, such as delay, by primarily tuning the weight ω_Q to keep queuing time below a specific target. Furthermore, we also aim to deploy RBIS in real-world testbeds, integrating the xApp within operational O-RAN environments using SDRs. A primary objective of this transition will be to evaluate the robustness and adaptability of the DRL agent when exposed to raw measurement noise, radar clutter, and hardware-specific estimation errors inherent to physical sensing equipment. Additionally, we plan to thoroughly analyze scalability, as both the observation and action spaces grow linearly with the number of active data flows ($\mathcal{O}(N)$). This experimental deployment will enable us to fully assess the performance of the DRL-based MAC scheduler under realistic conditions, including actual network dynamics, hardware constraints, and end-to-end latency considerations, thereby bridging the gap between simulation and practical deployment.

REFERENCES

- [1] M. Polese, L. Bonati, S. D’Oro, S. Basagni, and T. Melodia, “Understanding O-RAN: Architecture, interfaces, algorithms, security, and research challenges,” *IEEE Commun. Surveys Tuts.*, vol. 25, no. 2, pp. 1376–1411, 2nd Quart., 2023.
- [2] O-RAN Alliance. (2018). *O-RAN: Towards an Open and Smart RAN*. [Online]. Available: <https://mediastorage.o-ran.org/white-papers/O-RAN.White-Paper-2018-10.pdf>
- [3] L. Bonati, M. Polese, S. D’Oro, S. Basagni, and T. Melodia, “Open, programmable, and virtualized 5G networks: State-of-the-art and the road ahead,” *Comput. Netw.*, vol. 182, Dec. 2020, Art. no. 107516.
- [4] Z. Wei et al., “Integrated sensing and communication signals toward 5G-A and 6G: A survey,” *IEEE Internet Things J.*, vol. 10, no. 13, pp. 11068–11092, Jul. 2023.
- [5] N. Villegas, J. L. Herrera, L. Diez, D. Scotece, L. Foschini, and R. Agüero, “DRL-based dynamic MAC scheduler reconfiguration in O-RAN,” in *Proc. IEEE Int. Conf. Commun.*, Jun. 2025, pp. 5023–5028.
- [6] *5G-LENA Website*. Accessed: Oct. 15, 2025. [Online]. Available: <https://5g-lena.cttc.es/>
- [7] A. Mamane, M. Fattah, M. E. Ghazi, M. E. Bekkali, Y. Balboul, and S. Mazer, “Scheduling algorithms for 5G networks and beyond: Classification and survey,” *IEEE Access*, vol. 10, pp. 51643–51661, 2022.
- [8] A. M. Nor et al., “Demand based proportional fairness scheduling for 5G eMBB services,” in *Proc. IEEE Int. Black Sea Conf. Commun. Netw. (BlackSeaCom)*, Jun. 2022, pp. 263–268.

- [9] S. O. Aramide, B. Barakat, Y. Wang, S. Keates, and K. Arshad, "Generalized proportional fair (GPF) scheduler for LTE—A," in *Proc. 9th Comput. Sci. Electron. Eng. (CEECE)*, Sep. 2017, pp. 128–132.
- [10] K. Koutlia, B. Bojovic, S. Lagén, X. Zhang, P. Wang, and J. Liu, "System analysis of QoS schedulers for XR traffic in 5G NR," *Simul. Model. Pract. Theory*, vol. 125, May 2023, Art. no. 102745. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1569190X23000230>
- [11] X. Zhang, S. Sarkar, A. Bhuyan, S. K. Kasera, and M. Ji, "A stochastic optimization framework for distributed beam scheduling in 5G mm-wave networks over non-cooperative operators," in *Proc. 54th Asilomar Conf. Signals, Syst., Comput.*, Nov. 2020, pp. 539–543.
- [12] A. T. Z. Kargari and W. Saad, "Stochastic optimization and control framework for 5G network slicing with effective isolation," in *Proc. 52nd Annu. Conf. Inf. Sci. Syst. (CISS)*, Mar. 2018, pp. 1–6.
- [13] S. Cai and V. K. N. Lau, "Zero MAC latency sensor networking for cyber-physical systems," *IEEE Trans. Signal Process.*, vol. 66, no. 14, pp. 3814–3823, Jul. 2018.
- [14] R. Prasad and A. Sunny, "QoS-aware scheduling in 5G wireless base stations," *IEEE/ACM Trans. Netw.*, vol. 32, no. 3, pp. 1999–2011, Jun. 2024.
- [15] S. Bi, L. Huang, H. Wang, and Y.-J.-A. Zhang, "Lyapunov-guided deep reinforcement learning for stable online computation offloading in mobile-edge computing networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 11, pp. 7519–7537, Nov. 2021.
- [16] L. Dai, J. Mei, Z. Yang, Z. Tong, C. Zeng, and K. Li, "Lyapunov-guided deep reinforcement learning for delay-aware online task offloading in MEC systems," *J. Syst. Archit.*, vol. 153, Aug. 2024, Art. no. 103194. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1383762124001310>
- [17] Y. Chen, Z. Huang, Y. Zhang, W. Fang, and N. N. Xiong, "LyDRL: Lyapunov-guided deep reinforcement learning for stable task offloading in connected autonomous vehicles," *ACM Trans. Auto. Adapt. Syst.*, vol. 20, no. 3, pp. 1–29, Sep. 2025, doi: [10.1145/3715333](https://doi.org/10.1145/3715333).
- [18] I.-S. Comsa et al., "Towards 5G: A reinforcement learning-based scheduling solution for data traffic management," *IEEE Trans. Netw. Service Manage.*, vol. 15, no. 4, pp. 1661–1675, Dec. 2018.
- [19] R. Li et al., "Deep reinforcement learning for resource management in network slicing," *IEEE Access*, vol. 6, pp. 74429–74441, 2018.
- [20] S. Imtiaz, H. Ghauch, G. P. Koudouridis, and J. Gross, "Random forests resource allocation for 5G systems: Performance and robustness study," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops (WCNCW)*, Apr. 2018, pp. 326–331.
- [21] A. Chilmulwar and V. Sinha, "A novel machine learning based MAC scheduler algorithm using ARIMA," in *Proc. 16th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2019, pp. 1–8.
- [22] X. Guo et al., "A novel user selection massive MIMO scheduling algorithm via real time DDPG," in *Proc. GLOBECOM IEEE Global Commun. Conf.*, Dec. 2020, pp. 1–6.
- [23] N. M. Yungaicela-Naula, V. Sharma, and S. Scott-Hayward, "SLAQ: An SLA-driven 6G O-RAN QoS framework using deep reinforcement learning," in *Proc. Int. Symp. Netw., Comput. Commun. (ISNCC)*, 2025, pp. 1–6.
- [24] M. Tsampazi et al., "PandORA: Automated design and comprehensive evaluation of deep reinforcement learning agents for open RAN," *IEEE Trans. Mobile Comput.*, vol. 24, no. 4, pp. 3223–3240, Apr. 2024.
- [25] Y. Huang, G. Chen, M. A. Badiu, and J. Coon, "ISAC-enabled channel-aware beam management in THz networks with sensing errors," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Jun. 2025, pp. 1389–1394.
- [26] C. Y. Liu, M. Xia, J. Zhao, H. Li, and Y. Gong, "Optimal resource allocation for integrated sensing and communications in Internet of Vehicles: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 74, no. 2, pp. 3028–3038, Feb. 2024.
- [27] W. Xu, Y. Zhang, J. Wang, L. Zhang, and Q. Li, "Dynamic resource allocation for ISAC enabled Internet of Vehicles," in *Proc. 30th Annu. Int. Conf. Mobile Comput. Netw.*, Dec. 2024, pp. 2216–2221, doi: [10.1145/3636534.3698232](https://doi.org/10.1145/3636534.3698232).
- [28] K. Meng, Q. Wu, W. Chen, and D. Li, "Intelligent surface enabled sensing-assisted communication," in *Proc. IEEE Int. Conf. Commun.*, May 2023, pp. 4236–4241.
- [29] *System Architecture for the 5G System (5GS)*, document 23.501, 3GPP, 2024.
- [30] *O-RAN Architecture Description*, document TS 13.00, 3GPP, 2024. [Online]. Available: <https://www.specifications.o-ran.org/download?id=789>
- [31] J. L. Herrera, S. Montebugnoli, D. Scotece, L. Foschini, and P. Bellavista, "A tutorial on O-RAN deployment solutions for 5G: From simulation to emulated and real testbeds," *IEEE Commun. Surveys Tuts.*, vol. 28, pp. 1709–1748, 2025.
- [32] M. Petri and N. Maletic, "A mmWave JCAS system for real-time high-data rate communication and RADAR sensing," *IEEE Access*, vol. 13, pp. 117700–117715, 2025.
- [33] N. Villegas, A. Larrañaga, L. Diez, K. Koutlia, S. Lagén, and R. Agüero, "Extending QoS-aware scheduling in ns-3 5G-LENA: A Lyapunov based solution," in *Proc. Workshop Ns-3*, Jun. 2024, pp. 54–59.
- [34] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *Found. Trends Mach. Learn.*, vol. 11, nos. 3–4, pp. 219–354, 2018.
- [35] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *J. Mach. Learn. Res.*, vol. 22, no. 268, pp. 1–8, Jan. 2021.