This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3266642



# Programmable and Customized Intelligence for Traffic Steering in 5G Networks Using Open RAN Architectures

Andrea Lacava, *Student Member, IEEE,* Michele Polese, *Member, IEEE,* Rajarajan Sivaraj, *Senior Member, IEEE*, Rahul Soundrarajan, Bhawani Shanker Bhati, Tarunjeet Singh, Tommaso Zugno, Francesca Cuomo, *Senior Member, IEEE* and Tommaso Melodia *Fellow, IEEE*

**Abstract**—5G and beyond mobile networks will support heterogeneous use cases at an unprecedented scale, thus demanding automated control and optimization of network functionalities customized to the needs of individual users. Such fine-grained control of the Radio Access Network (RAN) is not possible with the current cellular architecture. To fill this gap, the Open RAN paradigm and its specification introduce an "open" architecture with abstractions that enable closed-loop control and provide data-driven, and intelligent optimization of the RAN at the user-level. This is obtained through custom RAN control applications (i.e., xApps) deployed on near-real-time RAN Intelligent Controller (near-RT RIC) at the edge of the network. Despite these premises, as of today the research community lacks a sandbox to build data-driven xApps, and create large-scale datasets for effective Artificial Intelligence (AI) training. In this paper, we address this by introducing *ns-O-RAN*, a software framework that integrates a real-world, production-grade near-RT RIC with a 3GPP-based simulated environment on ns-3, enabling at the same time the development of xApps and automated large-scale data collection and testing of Deep Reinforcement Learning (DRL)-driven control policies for the optimization at the user-level. In addition, we propose the first user-specific O-RAN Traffic Steering (TS) intelligent handover framework. It uses Random Ensemble Mixture (REM), a Conservative $Q$-learning (CQL) algorithm, combined with a state-of-the-art Convolutional Neural Network (CNN) architecture, to optimally assign a serving base station to each user in the network. Our TS xApp, trained with more than 40 million data points collected by ns-O-RAN, runs on the near-RT RIC and controls the ns-O-RAN base stations. We evaluate the performance on a large-scale deployment with up to 126 users with 8 base stations, showing that the xApp-based handover improves throughput and spectral efficiency by an average of 50% over traditional handover heuristics, with less mobility overhead.

**Index Terms**—O-RAN, ns-3, Deep Reinforcement Learning, Traffic Steering, Network Intelligence

## 1 Introduction

Fifth generation (5G) cellular networks and beyond shall provide improved wireless communications and networking capabilities, enabling heterogeneous use cases such as Ultra Reliable and Low Latency Communications (URLLC), Enhanced Mobile Broadband (eMBB), and massive machine-type communications, ranging from industrial Internet of Things (IoT) to metaverse, telepresence and remote telesurgery. The use-case requirements and deployment scenarios keep changing with evolving radio access technologies. As a consequence, 5G and beyond Radio Access Networks (RANs) are expected to be complex systems, deployed at a scale that is unforeseen in commercial networks so far [1].

This complexity and the evolving use-case requirements have prompted research, development, and standardization efforts in novel RAN architectures, and, notably, in the O-RAN paradigm. Nowadays, classic RANs are deployed with monolithic network functions (e.g., base stations) on black-box hardware. Such architecture is considered static and hard to reconfigure on-demand without any manual on-site intervention. The O-RAN architecture disrupts the classical approach by adopting the principles of *Disaggregation*, *Openness*, *Virtualization*, and *Programmability*. In O-RAN, the classic base station is *disaggregated*, i.e divided across multiple RAN nodes. The interfaces between the different nodes are *open* and standardized, to achieve multi-vendor interoperability. Network functions that implement the classic RAN operations are virtualized and software-based and deployed on white box hardware [2]. Software enables algorithmic and programmatic control based on the current network status, enabling the dynamic configuration of the infrastructure.

The combination of these principles introduces complex, virtualized architectures with RAN Intelligent Controllers (RICs) that (i) have a centralized abstraction of the network; and (ii) host applications performing closed-loop control of the RAN. This custom logic leverages the centralized aggregation of analytics on multiple network functions to run advanced data-driven Artificial Intelligence (AI) and Machine Learning (ML) techniques. For example, the near-real-time (near-RT) RIC hosts third-party applications called xApps [3] that interact with the RAN through the E2 interface and take Radio Resource Management (RRM) decisions at a time scale between 10 ms and 1 second [4]. Such architecture can efficiently learn complex cross-layer interactions across nodes, going beyond traditional control heuristics

*A. Lacava, M. Polese, and T. Melodia are with the Institute for the Wireless Internet of Things at Northeastern University, Boston, MA, 02115 USA. Tommaso Zugno was affiliated with Northeastern University and University of Padova at the time the research was conducted.*
*R. Sivaraj, R. Soundrarajan, B. S. Bhati and T. Singh are with Mavenir, Richardson, TX, USA.*
*A. Lacava and F. Cuomo are with Sapienza, University of Rome, IT 00185.*
*The corresponding author is A. Lacava, email: lacava.a@northeastern.edu.*
*M. Polese and R. Sivaraj equally contributed to the paper.*
*This work was partially supported by Mavenir and by the U.S. National Science Foundation under Grants CNS-1923789 and CNS-2112471.*

toward optimal RRM [5, 6].

Unlocking the intelligence in the networks is a crucial aspect of O-RAN. Specifically, the xApps integrate custom logic and AI/ML algorithms for the RAN [7, 8], paving the way to an enhanced network control with an User Equipment (UE)-level granularity that would not be possible with the classical RAN architectures. Indeed, the availability of data and analytics on the network in a centralized location (i.e., the RIC) enables new approaches to traditional network management problems. One of this use cases is the Traffic Steering (TS), i.e., the management of the mobility management of individual UEs served by the RAN [9]. TS involves key RAN procedures, such as handover management, dual connectivity, and carrier aggregation, among others. While handover management is a classic problem, the requirements and deployment scenarios for optimizing handovers keep changing with evolving radio access technologies and use-cases, posing newer challenges and requiring newer optimization strategies [10]. As an example, the handover optimization requirements for broadband access, e.g., eMBB UEs streaming high-quality video, are different from those of an autonomous car, i.e., an URLLC UEs.

In this context, traditional RRM solutions, largely based on heuristics only involving channel quality and load thresholds, are not primed to handle UE-centric handover decisions for new use-cases, and are often based on local, and thus limited, information. Data-driven solutions at the RIC can leverage a centralized point of view to learn complex inter-dependencies between RAN parameters and target the optimization to the Quality of Service (QoS) requirements of each UE.

Despite the promising architectural enablers, how to design and test effective and intelligent RAN control solutions to embed into xApps is still a challenge [11]. First, any ML solution needs to be properly trained. This requires data collection on large-scale setups, with a massive amount of data points to collect to properly represent the state of the system and allow the agent to learn an accurate representation of the system. Then, when it comes to closed-loop control through Deep Reinforcement Learning (DRL), the ML infrastructure requires an isolated environment for testing and online exploration, to avoid impacting the performance of production RANs. The TS mentioned before, for instance, includes the optimization of handover across multiple base stations through data-driven xApps. The data collection would need to cover large scale deployments, with different network configurations, operating frequencies, combinations of source traffic, and user mobility. At the same time, testing poorly trained solutions or performing online exploration on a large scale, commercial deployment may cause users to unexpectedly lose connectivity or experience a degraded service due to sub-optimal handover decisions [12].

**Contributions —** In this paper, we adopt a system-level approach and introduce a novel framework for handover management for TS, based on conservative Q-learning and the capabilities exposed by the O-RAN infrastructure. We first build an O-RAN-compliant near-RT RIC platform. Then, we propose the first TS xApp with DRL to optimally control mobility procedures at a UE level, using the centralized viewpoint of the RIC, RAN Key Performance Measurements (KPMs), and advanced Reinforcement Learning (RL) techniques to select the optimal target cells for handover of individual UEs. We also propose *ns-O-RAN*, a software module to connect the near-RT RIC to Network Simulator 3 (ns-3) to collect the data for training the TS xApp and to evaluate the end-to-end performance of our system, which improves relevant KPMs by up to 50%. Specifically, the contributions of this paper are as follows.

- *System Design*: We design and build a standard-compliant near-RT RIC platform with O-RAN-defined open interfaces and service models (i.e., standardized mechanisms to interact with RAN nodes). The relevant system design details are discussed in Section 3.
- *Integration*: We build *ns-O-RAN*, a virtualized and simulated environment for O-RAN, which bridges large scale 5G simulations in the open-source ns-3 with a real-world near-RT RIC. ns-O-RAN combines the scale and flexibility of a simulated RAN with any real-world, E2-compliant RIC. In this context, simulation based on realistic channel and protocol stack models contributes to the collection of data for the ML-based xApps without the need of large scale deployments. ns-O-RAN extends the ns-3 5G RAN module [13] by adding an O-RAN compliant E2 implementation, including the protocol capabilities and advanced service models. ns-O-RAN enables the RAN to stream events and data to the near-RT RIC, and the RIC to send control actions to the RAN over the E2 interface. These control actions are reflected in the call processing of the RAN functions and the updated data are streamed to the RIC. Thus, ns-O-RAN enables xApps development without relying on RAN baseband and radio units; the same xApps can subsequently be tested on a real RAN, without additional development effort. The relevant details are discussed in Section 3. We pledge to release ns-O-RAN as open-source in the O-RAN Software Community (OSC)[1] and the OpenRAN Gym platform [14].
- *TS Optimization*: We build a data-driven AI-powered TS xApp in the near-RT RIC to maximize the UE throughput utility, specifically, through handover control, as defined in the O-RAN technical specifications [15]. We use ns-O-RAN to collect data for, design, and test the TS xApp. We formulate the problem as a Markov Decision Process (MDP) and solve it using RL techniques. In particular, we use novel variants of the Deep-Q Network algorithm, namely Conservative Q-Learning (CQL) and Random Ensemble Mixture (REM) to model the Q-function and the loss function, along with a custom Convolutional Neural Network (CNN) design to maximize the expected reward. Our design enables multi-UE control with a multi-dimensional state space using a single RL agent. The problem formulation and optimization details are discussed in Section 4.
- *Performance Evaluation*: We extensively evaluate the xApp using different Key Performance Indicators (KPIs), such as UE throughput, spectral efficiency, and mobility overhead on a large-scale of RAN network created by ns-O-RAN. Leveraging the fine-grained UE-level intelligence and optimization at the near-RT RIC, we demonstrate significant performance improvements ranging from 30% to 50% for the above KPIs in Section 5.

1. The code is available at https://gerrit.o-ran-sc.org/r/gitweb?p=sim/ns3-o-ran-e2.git.

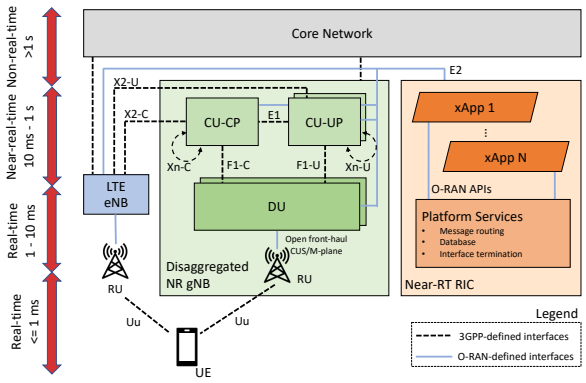



Fig. 1: O-RAN architecture with the near-RT RIC functions, aside packet core.

## 2 Background

In this section, we review the state of the art on O-RAN, TS and ns-3.

### 2.1 O-RAN Cellular Architecture

**RAN Protocol Stack —** Figure 1 provides a high-level overview of the O-RAN architecture. In this, the baseband unit of the NR base station, also called the Next Generation Node Base (gNB), is logically split into RAN functional nodes - Centralized Unit - Control Plane (CU-CP), Centralized Unit - User Plane (CU-UP), Distributed Unit (DU), and Radio Units (RUs). These functions are deployed as logical nodes in the RAN and connected through standardized O-RAN and 3rd Generation Partnership Project (3GPP)-defined interfaces. In particular, the CU-CP features the Radio Resource Control (RRC) and Packet Data Convergence Protocol - Control Plane (PDCP-C) layers, and manages the connectivity and mobility for the UEs. The CU-UP handles the Service Data Adaptation Protocol (SDAP) and Packet Data Convergence Protocol - User Plane (PDCP-U) layers, dealing with Data Radio Bearers (DRBs) that carry user traffic. The DU features the Radio Link Control (RLC), Medium Access Control (MAC) and Upper Physical (PHY-U) layers, for buffer management, radio resource allocation, and physical layer functionalities, such as operating the NR cells. For what it regards the Long Term Evolution (LTE), all the layers are managed in a single function called evolved Node Base (eNB). Finally, the RU is responsible for Lower Physical (PHY-L) layer, dealing with transmission and beamforming.

**RAN Intelligent Controllers —** The near-RT RIC is typically deployed as a network function in a virtualized cloud platform at the edge of the RAN. It onboards extensible applications (xApps) [3], apart from O-RAN standardized platform framework functions, to optimize RRM decisions for dedicated RAN functionalities using low-latency control loops at near-RT granularity (from 10 ms to 1 second). The near-RT RIC connects through the E2 interface to the CU-CP, CU-UP, DU and eNB, collectively referred to as the E2 nodes.

**E2 interface —** E2 is a bi-directional interface that splits the RRM between the E2 nodes and the near-RT RIC. With this architecture, the call processing and signaling procedures are implemented in the E2 nodes, but the RRM decisions for these procedures are controlled by the RIC through xApps, i.e., micro services onboarded on the RIC [3]. For example, the handover procedures for a UE are executed by the E2 node, but the UE's target cell for handover is decided and controlled by the RIC.

The procedures and messages exchanged over the E2 interface are standardized by E2 Application Protocol (E2AP) [2]. Using E2AP, the E2 nodes can send reports to the near-RT RIC with RAN data or UE context information. In addition, the near-RT RIC can send control actions containing RRM decisions and policies to the E2 node. The xApps in the near-RT RIC encode and decode the payload of the E2AP messages containing RRM-specific information, as defined by the E2 Service Models (E2SMs) [16]. The xApps have access to the RAN reports and controllers, and they can also embed data-driven and AI-based policies to control and optimize the RAN using the values provided by the disaggregated units. The service models define the information model and semantics of RRM operations over E2. Two E2SMs of interest in this paper are *E2SM-KPM* [17], which allows E2 nodes to send RAN performance data to the RIC, with granularity down to the UE-level, and *E2SM-RAN Control (RC)* [18], which allows the RIC to send back control based on RRM decisions from xApps [2].

### 2.2 Intelligence in the RIC

As already discussed, the disaggregation of the RAN functions enables the generation of large datasets that can be leveraged to study data driven approaches to the classical RRM problems. To support this trend, the O-RAN alliance has defined specifications for life cycle management of ML-driven RAN control. During the training, a model usually tries to explore all the possible states and, thus can decide to apply actions that can lead to the disruption of RAN functionalities, with network outages and degradation of the quality of service of the final users. For this reason, in O-RAN any ML model shall be trained offline [19] and deployed as xApps for online inference and RRM control in the RIC.

One of the most promising approaches is the RL, which teaches an *agent* how to choose an *action* from its action space, within a particular environment, to maximize *rewards* over time. The goal of the RL agent is then to compute a policy, which is a mapping between the environment *states* and actions so as to maximize a long term reward. RL problems are particularly of interest to RIC, because of their natural *closed-loop* form.

The RL model of interest to this paper is Deep Q-Network (DQN), which is a model-free, off-policy, value-based RL. Our RL algorithm uses a $Q$-value that measures the expected reward for taking a particular action at a given state. DQNs can be trained offline with an online refinement of the learned policy, thus they can subsequently keep getting deployed in the inference host towards generating optimal actions, as the agent receives live data streams from the environment [20].

### 2.3 Dual connectivity and traffic steering

Dual connectivity is a mode of 5G RAN deployment, where the UE is jointly connected to more than one base station. One of them is designated as the master node, which is responsible for control plane procedures of a UE, and the other is considered as a secondary node and it is responsible for data transfer for the UE along with the master node. A prevalent 5G deployment in North America and globally is E-UTRAN-NR Dual Connectivity (EN-DC) Non Stand Alone (NSA) mode 3X, where the LTE eNB is the master node, and NR gNB is the secondary node.

Traffic Steering is a RAN functionality of the RRC layer for managing connectivity and mobility decisions of UEs in the RAN. More specifically, TS handles (on a UE basis): (i) Primary cell selection and handover, (ii) selection and change of master and secondary nodes for dual connectivity, (iii) selection and handover of the secondary cell.

As previously discussed, the handover problem has been widely studied and optimized in the literature. Without O-RAN, this mechanism can be implemented by using different approaches [10]. Generally, it is common practice to perform handover based on channel quality hysteresis, and/or to advance handovers from overloaded to less loaded ones for load balancing. More recent approaches exploit RL to select the target node for the handover.

In the literature, there are several examples of AI-based handover procedures. One of the possible approaches is represented by the use of a centralized RL agent with handover control using Q-learning [21] and subtractive clustering techniques [22] to optimize the UEs mobility. Other work considers distributed Q-Learning approaches or cooperative multi-agents [23, 24] to optimize the handover process on Self-Organizing Networks (SONs). Another area of interest is represented by the use of the Deep Neural Network (DNN) in both the online training mode on the UEs [25] or via offline schemes [6, 26, 27]. [25] uses DNN with supervised learning to transfer knowledge based on traditional handover mechanisms and avoid negative effects of random exploration for an untrained agent. Other examples of similar works are represented by [25, 28]. [29] proposes a unified self-management mechanism based on fuzzy logic and RL to tune handover parameters of the adjacent cells. More examples can be found in [30], which discusses the state of the art and the challenges of intelligent RRM.

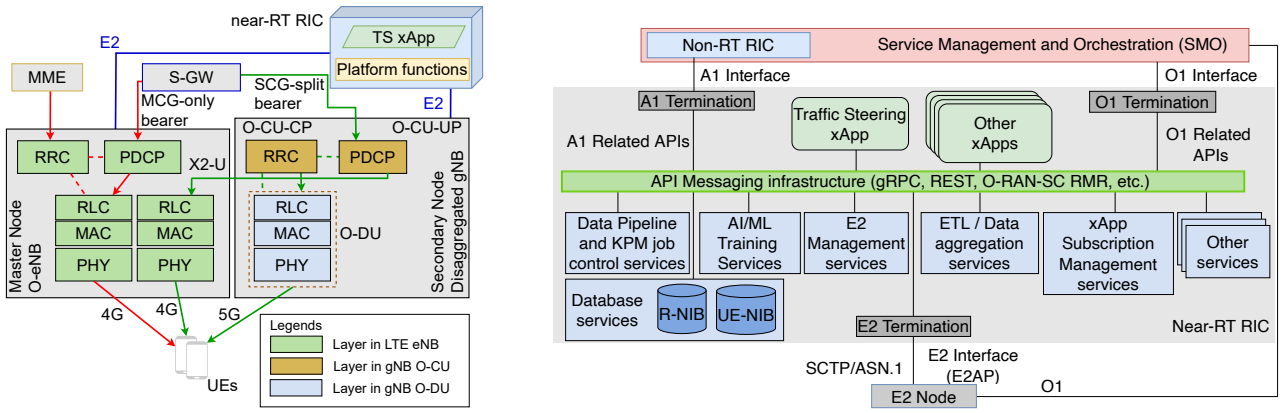
These works, however, generally do not optimize the performance of individual UEs and do not fully satisfy the need for per-UE control and optimization. Indeed, existing cellular networks implement procedures which are mostly cell-centric, even if there are usually high variations across the performance, requirements, and channel state of different UEs in the same cell [9]. Improved performance thus can be achieved with the UE-based approach we propose, enabled by the O-RAN architecture. User-centric handover schemes have been proposed for non-conventional radio access architectures, such as non-terrestrial networks [31, 32], ultra-dense networks with cooperative transmissions [33–35], cell-free massive MIMO [35, 36], and vehicular networks [37]. These solutions, however, on non-standard 3GPP information and/or parameters that are inaccessible at the RAN side (e.g., the user position and speed). This limits their potential implementation on a real network.

Recent literature considers the traffic steering use case and handover management in Open RAN networks [38–40]. In [38], the authors present a general overview of O-RAN and its potentialities by exploring the TS use case, showing the flexibility of this novel architecture and its orchestration capabilities. The authors also show how the dynamic setup of three different xApps dedicated to the TS management can affect and change the performances of the network. However, their work is more tailored to the O-RAN capabilities rather than their xApp performances. In [39], the authors consider a contextual multi-armed bandit problem to model handover across 5G cells, and considers per-UE Reference Signal Received Power (RSRP) metrics as input. In this paper, we consider a more complete set of RAN KPMs as input, thanks to the support of the O-RAN E2 interface between the RAN and the near-RT RIC. This makes it possible to improve the overall RAN performance, as discussed in Section 5. The growing interest in the O-RAN architecture has also led the scientific community to focus its attention on the implementation of xApps and rApps. Several papers in the literature implement xApps [14, 41, 42] and rApps [43] or both working cooperatively [44, 45] that can be onboarded and tested on the RICs for studying different use cases such as network slicing, orchestration and RAN management, and security. Compared to these works, in this paper, we focus on the TS use case which is a problem usually for large-scale scenarios with hundreds of UEs. In [42], Kouchaki et al. illustrate the step-by-step design, development, and testing of an AI-based resource allocation xApp for the near-RT RIC of the O-RAN architecture without focusing on the actual results of the AI-scheme proposed, but it is not clear if their online training can be performed on a real RAN. In contrast, our work focuses more on the xApp performances and the creation of a novel framework to ease the study of RL applied to O-RAN. Additionally, we support natively the large scale scenario generation without the need for hardware infrastructure, allowing the data to feed the xApp agent with an offline training process that does not require RIC deployment. O-RAN-based closed-loop control is also discussed in [40], where the authors study the power adjustment of the transmitters in a 5G cellular orientation using two different xApps that act as a simulation of the network in the RIC. This simulator, however, does not rely on 3GPP stochastic channels and thus the results may not be plausible once the model is deployed on a real network. In this paper, we implement an O-RAN compliant near-RT RIC and use xApps with standard-compliant service models that can be deployed on a real network. In addition, we test the performance of the xApp combining the real-world RIC with a large scale RAN deployment based on end-to-end, full-stack, 3GPP-based simulations in ns-3.

### 2.4 5G and AI in ns-3

ns-3 [46] is a discrete-event time network simulator targeted for research and educational use. It is considered the de facto standard for network simulators because of the variety of protocols used, its wide deployment and the widespread support of the scientific community. The discrete event

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3266642




(a) System model.

(b) Near-RT RIC Software Architecture.

Fig. 2: System Model and Software Architecture

approach allows researchers to simulate the interactions in form of time-based events, allowing the modeling of each aspect of the communication, from the application layer to the physical layer. Such layers are the fundamental building blocks of a wireless network simulation and they can be combined in ns-3 to study particular use cases that otherwise would be very hard with a real deployment. Indeed, one of the major reasons for using ns-3 in this work is because of the very accurate 3GPP stochastic models [47] available within the simulator combined with the possibility of creating large scale deployments with no telecommunication hardware required. These peculiarities are also enhanced by the possibility to integrate real world features, such as buildings and obstacles and mobility models for the wireless nodes, to create realistic scenarios. In this work, we use the 5G and mmWave ns-3 module [13], which extends the ns-3 LTE module with new detailed modeling of the mmWave channel that can capture spatial clusters, path dynamics, antenna patterns and beamforming algorithms.

The ns-3 simulator is also an optimal tool for implementing better AI solutions for the networks. In recent years, different works have extended the normal capabilities of ns-3 to combine its potentialities with some well-known ML development software. In [48], the authors propose ns3-gym, a framework that integrates both OpenAI Gym and ns-3 in order to encourage usage of RL in networking research. Following the same principles of ns3-gym, ns3-ai [49] provides a high-efficiency solution to enable the data interaction between ns-3 and other python based AI frameworks. However, both of these tools cannot be used as a framework for the development of O-RAN xApps that can be used directly in a production environment, unlike the ns-O-RAN framework proposed in this paper.

## 3 System Design and Architecture

Here, we discuss the system model assumption, the near-RT RIC software architecture and the ns-O-RAN design.

### 3.1 System Model

The system architecture is shown in Figure 2a. We consider a network with $M$ LTE cells, and E2 nodes of $N$ NR cells, and a set $U$ of 5G UEs. The infrastructure is deployed as a 5G NSA network with EN-DC RAN and option 3X for dual connectivity [50]. With this, a 5G UE is jointly connected to an LTE eNB (master node) and the E2 nodes of a 5G gNB (secondary node). Each UE is jointly served by the primary cell of its master node and the secondary cell of its secondary node in EN-DC. The UEs subscribe to heterogeneous types of data traffic (as detailed in Section 5). In the RAN, each UE-subscribed data traffic flow is split at the PDCP-U layer of the gNB CU-UP. Each packet is sent to the lower RLC layer at either the gNB DU (over the F1 interface) or the LTE eNB over the X2-U interface for subsequent transmission to the UE via the NR or LTE radio, respectively. In addition, we consider a near-RT RIC connected to each LTE and NR cell through the E2 interface. The near-RT RIC is deployed at the edge of the RAN and features the TS xApp to optimize UE handover. The delivery of the KPM data between the E2 nodes and the RIC allows the exchange of network data and handover control actions at near-RT periodicity and is enabled through the use of the E2SM-KPM service model. We use the E2SM-RC service model to generate control actions from the RIC to the E2 node for handover of specific UEs from their current serving cells to the target cells identified by the TS xApp. Additionally, E2SM-RC is used to report UE-specific L3 RRC measurements (such as RSRP, or Signal to Interference plus Noise Ratio (SINR) with respect to its serving and neighbor cells) from the E2 node to the RIC periodically and during mobility events. We assume that the RAN configures the UE measurement reporting so that it can provide periodic estimates of the channel quality toward current and neighboring cells. Overall, the combination of the capabilities provided by the E2 interface, and the granularity required for optimization and control of the handover process in highly dynamic wireless environments makes xApps on the near-RT RIC the ideal hosts for traffic steering control.

### 3.2 Near-RT RIC Software Architecture

We implement a near-RT RIC platform [51] with the components shown in Figure 2b. In general, the near-RT RIC has two sets of applications, namely the xApps (for the control of the RRM of dedicated RAN functionalities) and O-RAN-standardized platform services [2]. The latter manage integration of xApps, interfacing with E2 nodes, and the

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3266642

overall functioning of the RIC. In particular, they include the *E2 Termination* service, which routes the E2AP messages between the platform services and the E2 nodes over the E2 interface based on SCTP transport protocol. The service also performs ASN.1 encoding/decoding and manages data exposed by E2 nodes. The *xApp Subscription Management* service maintains, manages, validates, and sends/receives xApp subscriptions toward E2 nodes. The data collection and aggregation for the xApps is managed by two additional platform services. The *Data Pipeline and KPM job control* makes sure that xApps do not duplicate KPM requests to the RAN by interacting with the Subscription Management service and filtering duplicated subscription requests on behalf of the xApps. The KPM data received by the RAN is aggregated, processed, and presented to the xApps by the *Extract, Transform and Load (ETL), data aggregation and ingestion* service. In our implementation, the TS xApp leverages the services of the RIC platform to (i) collect KPMs on the status of the network; (ii) process them and perform online inference to decide if one or more UEs should perform a handover to a different cell; and, eventually, (iii) send the handover control action to the RIC Routing Manager, which will decide whether deliver or not the message to the RAN. The TS xApp triggers an E2 node KPM subscription specifying the parameters for the data collection, i.e., the list of KPMs and serving-cell and neighbor-cell L3 RRC measurements, and the periodicity at which these values need to be reported by the E2 nodes. The TS xApp and the simulated RAN implemented with ns-O-RAN (described in Section 3.3) collectively support streaming 40 UE-level, cell-level, and node-level KPMs from E2 nodes.

The E2 nodes accepts the subscription and starts streaming KPMs and L3 RRC measurements. The raw streamed KPM data is stored by Data Pipeline and KPM job control service. The ETL, data aggregation and ingestion service retrieves relevant measurements stored in this data repository, and correlates and aggregates in time series the UE level KPM information and L3 RRC measurements. The TS xApp can then fetch and process the data to perform inference with the algorithm described in Section 4. If a handover needs to be performed, the TS xApp communicates with the *E2 termination* service to send the control action to the RAN.

### 3.3 Connecting O-RAN with ns-3: ns-O-RAN

One key contribution of this paper is represented by ns-O-RAN, the first O-RAN integration for ns-3. ns-O-RAN is an ns-3 module that connects a real-world near-RT RIC with ns-3, enabling large scale (i) collection of RAN KPMs and (ii) testing of closed-loop control of simulated cellular networks. We use the term "real-world" to indicate that the RIC used in this framework is a standard compliant O-RAN near-RT RIC that is also capable of communicating with real hardware equipment. This aspect allows ns-O-RAN to be a powerful tool for the development of the xApp that can be then activated on real world RANs. Indeed, thanks to the flexibility of ns-3, such integration eases the design, development, and testing of xApps across different RAN setups with no infrastructure deployment cost. As already introduced in Section 2, ns-3 provides realistic modeling capabilities for large-scale wireless scenarios. It features

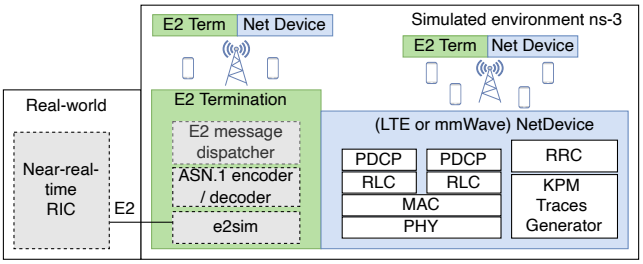



Fig. 3: ns-O-RAN Architecture.

a channel model with propagation and fading compliant with 3GPP specifications [52], and a full-stack 5G model for EN-DC RAN [13], besides the TCP/IP stack, multiple applications, and mobility models.

ns-O-RAN bridges ns-3 to our real-world, O-RAN-compliant RIC to enable production code to be developed and tested against simulated RANs. These capabilities also help reducing the cost associated with AI training. Indeed, ns-O-RAN can be used to identify an effective set of weights for the xApp neural networks over a simulated generalized scenario, which makes it possible to quickly refine the model with online training on a real deployment (as also discussed in [11, 14] for emulation to over-the-air transitions). The main features of ns-O-RAN are discussed in the next paragraphs.

• **e2sim —** We connect the E2 termination of the RIC to a set of E2 endpoints in ns-3, which are responsible for handling all the E2 messages from and to the simulated environment. This connection was developed by extending the E2 simulator, namely `e2sim` [53], and wrapping it into an ad hoc module for ns-3. `e2sim` is a project in the O-RAN Software Community that provides basic E2 functionalities to perform integration testing of the near-RT RIC. ns-O-RAN leverages and extends `e2sim` as the E2 termination on the ns-3 side. It can decode, digest, and provide feedback for all the messages coming from the RIC, and streams RAN telemetry based on simulation data to the RIC.

• **Message dispatching —** The design of ns-O-RAN addresses several challenges that would otherwise prevent communications between the simulated and real-world environments. Firstly, as discussed in Section 2, the near-RT RIC expects to interface with a number of disaggregated and distinct endpoints, i.e., multiple DUs and CU-CPs/CU-UPs, which are usually identified by different IP addresses and/or ports. Instead, all the ns-3 simulated RAN functions are handled by a single process. `e2sim` itself was not designed to handle multiple hosts at once, while the E2 protocol specifications, which rely on the SCTP protocol for E2AP, do not pose any limitation in this sense. To address this, we extended the `e2sim` library to support multiple endpoints at the same time and created independent entities (i.e., C++ objects) in the simulated environment to represent different RAN-side E2 terminations. Each RAN function is bound to just one E2 interface, as depicted in Figure 3, and has its own socket address. ns-O-RAN can successfully establish connectivity between multiple RAN nodes and the near-RT RIC even if a single IP is associated to the simulation process, as it can filter E2AP messages through unique ports, ensuring the independence of data flow with

the near-RT RIC from the others. In this way, it is possible to assign the same IP to every base station created in the simulation, but each with a different and unique port and is possible to enable the data flow from multiple independent e2 terminations in the simulation and the RIC. Moreover, we extended ns-3 to instantiate independent threads for each E2 termination and use callbacks that can be triggered when data is received or transmitted over E2.

• **Time Synchronization —** Finally, there is also a gap in timing between the real-world RIC and the ns-3 simulator, which is a discrete-event framework that can execute faster or slower than the wall clock time. This may potentially lead to inconsistencies between the simulated environment and the near-RT RIC expecting the real-world timing. To synchronize the two systems, at the beginning of the simulation ns-3 stores the current Unix time in milliseconds and uses it as baseline timestamp. Whenever an E2 message is sent to the RIC, the simulator will sum the simulation time elapsed and the baseline timestamp. In this way, the RIC can correctly reorder the messages according to the same timestamp of the simulation without any loss of generality and thus ensuring consistency on both sides of the happened-before relationship.

## 4 Traffic Steering Optimization

In this section, we formulate the optimization problem for the traffic steering xApp and discuss the algorithm design to determine the optimal target cells for handover of UEs.

To the best of our knowledge, this is the first paper to develop a data-driven UE-based traffic steering/handover optimization technique based on Conservative Q-learning.

Firstly, we formulate the general optimization problem in Eqs. 1 and 2 of Section 4.1, showing that there is no closed forms for the optimization objective as function of the 5G KPIs considered in production networks. Secondly, in Section 4.2, we embed the aforementioned equations in a data-driven RL framework to study the UE-based optimization and apply the of Conservative Q-Learning algorithm which lead to a performance gain of up to 50% in throughput and spectral efficiency that will be discussed in Section 5.

### 4.1 Problem Formulation

We consider as objective function the weighted cumulative sum of the logarithmic throughput of all the UEs across time, as a function of their instantaneous target Primary cell of the Secondary Node (PSCell). The optimization goal is to maximize the objective function by optimizing the choice of the target PSCells for all UEs. At the same time, we want to avoid frequent handovers for individual UEs, since they increase network overhead and deteriorate their performance. Thus, we associate a cost function for every UE-specific handover and model it as an exponential decay function of the linear difference in time since the previous handover for that UE. This means that smaller the difference in time, higher is the cost, and vice-versa. We add this cost function as a constraint to make sure that the cost does not exceed a predefined cost threshold.

Let $\beta_u$ be a weight associated with any UE $u \in U$. $R_{u,t}$ is the throughput at any discrete window of time $t$, which depends on $c_{u,t}$, i.e., the PSCell assigned to $u$ during $t$, and on $B$ RAN performance parameters $b_1, b_2, \dots b_B$.

These metrics are available during the time window $t$ at the near-RT RIC, where the optimization is solved, thanks to the KPM reports from the E2 nodes $C^{\text{NR}}$ is the universe of all the $N$ NR cells. The reporting periodicity of the E2 nodes is set to 100 ms, which also represents the time window available to the AI agent to trigger the handover actions of one or more UEs. The cost associated with handover for UE $u$ at time $t$ is given by $K_{u,t}$, the initial cost is $K_0$ (where $K_0 > 0$), the decay constant is $\delta$ (where $0 < \delta < 1$), $t'_u$ is the time when the previous handover was executed for $u$, $X_{u,t}$ is a $0/1$ decision variable which yields a value 1, if $u$ was subject to handover at time $t$, and 0, otherwise. $W$ is a predefined cost threshold, which represents a maximum value that cannot be exceeded by the cost function. We consider any time window $t$ for an infinite time horizon ranging from $t_0$ to $\infty$. The constrained optimization problem is formulated as follows:

$$\begin{aligned} \underset{c_{u,t} \in C^{\text{NR}}}{Maximize} \quad & \sum_{t=t_0}^{\infty} \sum_{u \in U} \beta_u \log R_{u,t}(c_{u,t}, b_1, b_2, \dots b_B) \\ \text{subject to} \quad & K_{u,t} \cdot X_{u,t} \leq W, \\ & X_{u,t} \in [0,1], \end{aligned} \tag{1}$$

where $K_{u,t} = K_0 e^{-\delta \cdot (t - t'_u)}$, $K_0 > 0$ and $0 < \delta < 1$. Applying Lagrangian multiplier $\lambda$ to the constrained optimization problem in Eq. 1, the constrained optimization problem becomes as follows:

$$\begin{aligned} \underset{c_{u,t} \in C^{\text{NR}}}{Maximize} \quad & \sum_{t=t_0}^{\infty} \sum_{u \in U} \beta_u \log R_u(c_{u,t}, b_1, b_2, \dots b_B) \\ & - K' e^{-\delta \cdot (t - t'_u)} X_{u,t} + W' \\ \text{subject to} \quad & X_{u,t} \in [0,1] \text{ and } \lambda \geq 0 \end{aligned} \tag{2}$$

where $K' = \lambda K_0$ and $W' = \lambda W$.

### 4.2 Algorithm Design

**MDP and RL —** We use a data-driven approach (specifically, RL) to model and learn $R_{u,t}$ as a function of $\{c_{u,t}, b_1, b_2, \dots b_B\}$, due to the lack of a deterministic closed-form equation for $R_{u,t}$ as a function of the parameters, and its relationship with cost $K_{u,t}$ and the handover decision variable $X_{u,t}$. We consider the infinite time horizon MDP to model the system, where the environment is represented by ns-O-RAN, and a single RL agent is deployed in the near-RT RIC containing the TS xApp. The system is modeled as an MDP because the TS xApp in the RIC controls the target PSCell for the UEs handover, while the resulting state (including the RAN performance parameters and the user throughput) is stochastic. The MDP is defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \bigcirc, \mathcal{I} \rangle$, where:

• $\mathcal{S}$ is the state space, comprising of per-UE E2SM-KPM periodic data and per-UE E2SM-RC periodic/event-driven data. Let $C'_{u,t} \subseteq C^{\text{NR}}$ be the set of serving PSCell and neighboring cells for any UE $u$ at time $t$. The state vector for $u$ at time $t$ from the environment ($\vec{s}_{u,t}$) includes the UE identifier for $u$ and the set of parameters $b_1, b_2, \dots b_B$. The latter includes (i) the UE-specific L3 RRC measurements (obtained

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3266642

from the E2 node CU-CP) such as $\text{sinr}_{u,c,t}$ for any cell $c \in C'_{u,t}$ for the UE $u$; (ii) $\text{PRB}_{c,t}$, the cell-specific Physical Resource Block (PRB) utilization for $c$ at time $t$ obtained from the E2 node DU; (iii) $Z_{c,t}$, the cell-specific number of active UEs in the cell $c$ with active Transmission Time Interval (TTI) transmission at $t$ obtained from DU; (iv) $P_{c,t}$, the total number of MAC-layer transport blocks transmitted by cell $c$ across all UEs served by $c$ at time $t$ (obtained from the E2 node DU); (v) $p_{c,t}^{\text{QPSK}}$, $p_{c,t}^{\text{16QAM}}$, $p_{c,t}^{\text{64QAM}}$, the cell-specific number of successfully-transmitted transport blocks with QPSK, 16QAM and 64QAM modulation rates from the cell $c$ to all UEs served by the $c$ at time $t$ normalized by $P_{c,t}$. (vi) Finally, the set of parameters includes the cost the UE $u$ would incur, if handed over to $c_{u,t}$ at $t$ (i.e., where $c_{u,t} \neq c_{u,t-1}$), given by:

$$k(c_{u,t}) = K_0 e^{-\delta \cdot (t - t'_u)} x(c_{u,t});$$
$$\text{where } x(c_{u,t}) = \begin{cases} 1 & \text{if } c_{u,t} \neq c_{u,t-1} \\ 0 & \text{otherwise} \end{cases}$$

Note that the cost $k(c_{u,t})$ is zero if there is no handover, i.e., $c_{u,t} = c_{u,t-1}$. The above state information are aggregated across all the serving and neighbor cells of $u$, i.e., $\forall c \in C'_{u,t} \subseteq C^{\text{NR}}$, along with the cell identifier for $c$, during the reporting window $t$ to generate a consolidated record for $u$ for $t$. This aggregated state information for $u$ is fed as input feature to the RL agent on the TS xApp. This is done for all UEs in $U$, whose aggregated state information is fed to the same RL agent. If any of the parameters in the state information from the environment for any UE $u$ is missing, the RIC ETL service uses a configurable small window $\epsilon$ to look back into recent history (tens to few hundred of ms) and fetch those historical parameters for the missing ones.

• $\mathcal{A}$ is the action space, given by:

$$\mathcal{A} = \{\text{HO}(c_1), \text{HO}(c_2), \ldots \text{HO}(c_N), \overline{\text{HO}}\}$$

where, $c_1, c_2, \ldots c_N \in C^{\text{NR}}$. Here, $a_{u,t} = \text{HO}(c)$, where $a_{u,t} \in \mathcal{A}$, indicates that the RL agent is recommending a handover action for $u$ to any cell $c$ at $t$, and $a_{u,t} = \overline{\text{HO}}$ indicates no handover action for $u$ at $t$, meaning that the UE shall continue being served by its current primary serving cell.

• $\mathcal{P}(\vec{s}_{u,t+1} | \vec{s}_{u,t}, a_{u,t})$ is the state transition probability of UE $u$ from state $\vec{s}_{u,t}$ at $t$ to $\vec{s}_{u,t+1}$ at $t+1$ caused by action $a_{u,t} \in \mathcal{A}$.

• $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function for UE $u$ at $t+1$, as a result of action $a_{u,t}$, given by the following:

$$\mathcal{R}_{u,t+1} = \beta_u \cdot (\log R_{u,t+1}(c_{u,t+1}) - \log R_{u,t}(c_{u,t})) - k(c_{u,t+1}) \quad (3)$$

The reward for UE $u$ is the improvement in the logarithmic throughput $R_{u,t}$ due to the transition from $\vec{s}_{u,t}$ to $\vec{s}_{u,t+1}$ caused by action $a_{u,t}$ taken at $t$, minus the cost factor. The reward is positive, if the improvement in $\log$ throughput is higher than the cost, and negative, otherwise. $R_{u,t}$ is obtained from CU-UP using E2SM-KPM.

• $\gamma \in [0, 1]$ is the discount factor for future rewards. The value function $V^\pi(s)$ is the net return given by the expected cumulative discounted sum reward from step $t$ onwards due to policy $\pi$, provided as follows:

$$V^\pi(s) = \mathbb{E}\left[\sum_{u \in U} \sum_{i=0}^{\infty} \gamma^i \mathcal{R}_{u,t+i} | \vec{s}_{u,t} = s, \pi(a|s)\right] \quad (4)$$

• $\mathcal{I}$ is the initial distribution of the UE states.

• We consider two policies: (i) a target policy $\pi(a|s)$, to learn the optimal handover action $a$ for any state $s = \vec{s}_{u,t}$; and (ii) a behavior policy $\mu(a|s)$, to generate the handover actions which result in state transition and a new state data from the environment.

**Q-function and Deep-Q Network —** We use *Q-learning*, a model-free, off-policy, value-based RL approach. We compute the $Q$ function, an action-value function which measures the expected discounted reward upon taking any action $a$ on any given state $s$ based on any policy $\pi$. The value returned by the $Q$-function is referred to as the $Q$-value, i.e.,

$$\begin{aligned} Q^\pi(s,a) &= \mathbb{E}\left[\sum_{u \in U} \sum_{i=0}^{\infty} \gamma^i \mathcal{R}_{u,t+i} | \vec{s}_{u,t} = s, a_{u,t} = a, \pi(a|s)\right] \\ &= r(s,a) + \gamma \mathbb{E}_{\mathcal{P}(s'|s,a)}\left[Q^\pi(s', a') | s, a, \pi\right] \end{aligned} \quad (5)$$

Here, $r(s,a) = \mathbb{E}\left[\sum_{u \in U} \mathcal{R}_u | \vec{s}_{u,t} = s, a_{u,t} = a, \pi(a|s)\right]$. From (5) and (4), we have

$$V^\pi(s) = \sum_a \pi(a|s) Q^\pi(s,a). \quad (6)$$

The optimal policy $\pi^\star$ is the one that maximizes the expected discounted return, and the optimal $Q$ function $Q^\star(s,a)$ is the action-value function for $\pi^\star$ given by the Bellman equation as follows:

$$\begin{aligned} \pi^\star(a|s) &= \arg\max_\pi Q^\pi(s,a) \\ Q^\star(s,a) &= r(s,a) + \\ &\quad + \gamma \mathbb{E}_{\mathcal{P}(s'|s,a)}\left[\max_{a'} Q^\star(s', a') | \vec{s}_{u,t} = s, a_{u,t} = a, \pi^\star\right] \end{aligned} \quad (7)$$

We use the $Q$-learning algorithm to iteratively update the $Q$-values for each state-action pair using the Bellman equation, as seen in 8, until the $Q$ function converges to $Q^\star$. The value iteration by the RL agent leverages the *exploration-exploitation* trade-off to update the target policy $\pi$. It *explores* the state space of the environment by taking random handover control actions and learning the $Q$-function for the resulting state-action pair, and *exploits* its learning to choose the optimal control action maximizing the $Q$-value, i.e.,

$$Q^\pi_{i+1}(s,a) = r(s,a) + \gamma \mathbb{E}\left[\max_{a'} Q^\pi_i(s', a' | s, a, \pi)\right]. \quad (8)$$

Such value iteration algorithms converge to the optimal action-value function, i.e., $Q^\star := \lim_{i \to \infty} Q^\pi_i$. The Bellman error $\Delta$, as in (9), is the update to the expected return of state $s$, when we observe the next state $s'$. $Q$-learning repeatedly

adjusts the $Q$-function to minimize the Bellman error, as shown in (9) as in

$$\begin{aligned}\Delta_{i+1} &= \left[r(s,a) + \gamma \max_{a'} Q_i^{\pi}(s',a')\right] - Q_{i+1}^{\pi}(s,a) \\ Q_{i+1}^{\pi}(s,a) &\leftarrow (1-\omega) Q_{i+1}^{\pi}(s,a) + \\ &+ \omega \left[r(s,a) + \gamma \max_{a'} Q_i^{\pi}(s',a')\right].\end{aligned} \tag{9}$$

This approach of $\lim_{i\to\infty} Q_i^{\pi} \to Q^{\star}$ has practical constraints, as discussed in [54]. To address this, we use a CNN approximator with weights $\theta$ to estimate the $Q$ function $Q(s,a;\theta)$, and refer to it as the $Q$-network. Our CNN architecture design is shown in Fig. 4. Deep $Q$-learning comes from parameterizing $Q$-values using CNNs. Therefore, instead of learning a table of $Q$-values, we learn the weights of the CNN $\theta$ that outputs the $Q$-value for every given state-action pair. The $Q$-network is trained by minimising a sequence of loss functions $MSE_i(\theta_i,\pi)$ for each iteration $i$. The optimal $Q$-value, as a result of CNN approximator, is given by $\overline{Q}^{\star}$ as follows:

$$\begin{aligned}&MSE_i(\theta_i,\pi) = \\ &= \mathbb{E}\left[\left(r(s,a) + \gamma \max_{a'} Q^{\pi}(s',a';\theta_{i-1}) - Q^{\pi}(s,a;\theta_i)\right)^2\right] \\ &\overline{Q}_i^{\pi} = \arg\min_{Q^{\pi}} \left\{E\left[Q^{\pi}(s,a,\theta_i)|s,a,\pi(a|s)\right] + \omega MSE(\theta_i,\pi)\right\} \\ &\overline{Q}^{\star} := \lim_{i\to\infty} \overline{Q}_i^{\pi}\end{aligned} \tag{10}$$

Here, $\mathbb{E}_{\mathcal{P}(s'|s,a)}\left[r(s,a) + \gamma \max_{a'} Q^{\pi}(s',a';\theta_{i-1})|s,a,\pi\right]$ is the target for iteration $i$. The parameters from the previous iteration $\theta_{i-1}$ are fixed for optimizing the loss function $MSE(\theta_i)$. The gradient of the loss function is obtained by differentiating the loss function in Eq. 10 with respect to $\theta$ and the loss can be minimized by computing its stochastic gradient descent.

Thanks to the CNN, we are able to train and use the same weights for different cells, i.e., one single model for all, and we can approximate a non-linear dependency between input values and the reward function with a reduced dimensionality.

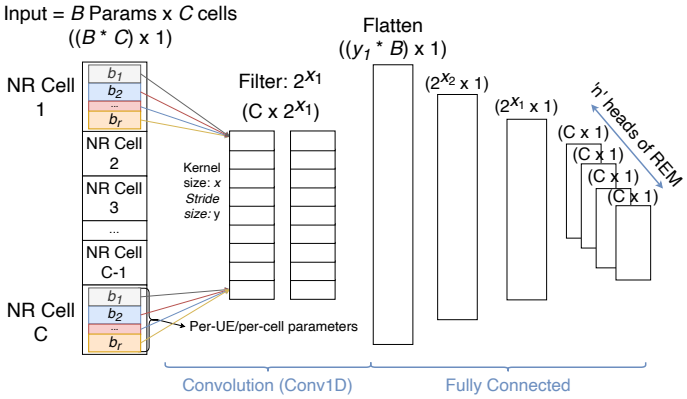



Fig. 4: Our CNN architecture design

We use an *off-policy* $Q$-learning algorithm, called DQN [54] for this purpose. The DQN algorithm leverages an experience replay buffer, where the RL agent's experiences at each step $e_t = (s_t, a_t, r_t, s_{t+1})$ are collected using the behavior policy $\mu$ and stored in a replay buffer $\mathcal{D} = \{e_1, e_2, \ldots e_{t-1}\}$ for the policy iterate $\pi_i$. $\mathcal{D}$ is pooled over many episodes, composed of samples from policy iterates $\pi_0, \pi_1, \ldots \pi_i$, so as to train the new policy iterate $\pi_{i+1}$ (as $Q^{\star} = \lim_{i\to\infty} Q_i^{\pi}$). At each time step of data collection, the transitions are added to a circular replay buffer. To compute the loss $MSE(\theta_i)$ and the gradient, we use a mini-batch of transitions sampled from the replay buffer, instead of using the latest transition to compute the loss and its gradient. Using an experience replay has advantages in terms of an off-policy approach, better data efficiency from re-using transitions and better stability from uncorrelated transitions [54].

**REM and CQL —** To leverage the full potential of the integrated ns-3 simulation environment in ns-O-RAN and harness large datasets generated from the simulator via offline data collection for data-driven RL, we use offline $Q$-learning. This enables us to learn the CNN weights by training the $Q$-network using the DQN model from dataset $\mathcal{D}$ collected offline based on any behavior policy (potentially unknown, using any handover algorithm) $\pi$ without online interactions with the environment and hence, no additional exploration by the agent beyond the experiences $e_t$ available in $\mathcal{D}$ via $\mu$. The trained model is then tested against a fresh batch of simulations and the $Q$-function is iteratively updated online according to the values generated by the simulations. We use a robust offline $Q$-learning variant of the DQN algorithm, called *REM*, which enforces optimal Bellman consistency on $J$ random convex combinations of multiple $Q$-value estimates to approximate the optimal $Q$-function [55]. This approximator is defined by mixing probabilities on a $(J-1)$ simplex and is trained against its corresponding target to minimize the Bellman error [55].

$$\begin{aligned}&\hat{MSE}_i(\theta_i,\pi) = \\ &= \mathbb{E}\left[\left(r(s,a) + \gamma \max_{a'} \hat{Q}^{\pi}(s',a';\theta_{i-1}) - \hat{Q}^{\pi}(s,a;\theta_i)\right)^2\right] \\ &= \mathbb{E}[(r(s,a) + \gamma \max_{a'} \sum_j \alpha_j Q_j^{\pi}(s',a';\theta_{i-1}) - \\ &- \sum_j \alpha_j Q_j^{\pi}(s,a;\theta_i))^2] \\ &\tilde{Q}_i^{\pi} = \arg\min_{Q^{\pi}} \hat{MSE}_i(\theta_i,\pi)\end{aligned} \tag{11}$$

Here, $\alpha_j \in \mathbb{R}^J$, such that $\sum_{j=1}^{J} \alpha_j = 1$ and $\alpha_j \geq 0, \forall j \in [1,J]$. $\alpha_j$ represents the probability distribution over the standard $(J-1)$-simplex. While REM prevents the effect of outliers and can effectively address imbalances in the offline dataset $\mathcal{D}$, offline-$Q$ learning algorithms suffer from action distribution shift caused by a bias towards out-of-distribution actions with over-estimated $Q$ values [56]. This is because the $Q$-value iteration in the Bellman equation uses actions from target policy $\pi$ being learned, while the $Q$-function is trained on action-value pair generated from $\mathcal{D}$ generated

**Algorithm 1** Offline $Q$-learning training

1: Store offline data (generated from ns-3) using any handover algorithm and behavior policy $\mu$ into replay buffer $\mathcal{D}$ consisting of UE-specific records ($\forall u \in U$)
2: **while** $\mathcal{D}$ not empty and value iteration $i$
3: Begin training step:
4: Select a batch of $2^{x_1}$ samples for input to the CNN
5: Use the $Q$-function and loss function $\hat{MSE}$ from Eq. 12 to train the CNN weights $\theta_i$ based on CQL and REM for value iteration $i$ of target policy $\pi$ for $\breve{Q}_i^{\pi}$
6: Set $i \leftarrow i+1$
7: **end while**

**Algorithm 2** Online value iteration and inference

1: **while** Incoming experience data $e_t$ for any UE $u$ from RAN environment to near-RT RIC for $t \in [t_0, \infty]$
2: Append $e_t$ to replay buffer $\mathcal{D}' \subseteq \mathcal{D}$ in AI/ML training services with length $D' \leq D$
3: Begin inference step:
4: Repeat steps 4 and 5 from Algorithm 1
5: Generate HO control action for $u$ from the TS xApp over E2 to RAN environment based on $\breve{Q}_i^{\pi}$
6: Set $i \leftarrow i+1$
7: **end while**

using behavior policy $\mu$. To avoid this problem of overestimation of $Q$-values for out-of-distribution actions, we use a conservative variant of offline DQN, called CQL, which learns a conservative, lower-bound $Q$-function by (i) minimizing $Q$-values computed using REM under the target policy distribution $\pi$ and (ii) introducing a $Q$-value maximization term under the behavior policy distribution $\mu$ [56]. From Eq. 10, the iterative update for training the $Q$-function using CQL and REM is given by:

$$\begin{aligned}\breve{Q}_i^{\pi} \leftarrow \arg\min_{\hat{Q}^{\pi}} \{ & \underbrace{\mathbb{E}\left[\hat{Q}^{\pi}(s, a_{\pi}; \theta_i) | s, a_{\pi}, \pi(a_{\pi}|s)\right]}_{\text{minimize REM Q-value under } \pi} \\ & - \underbrace{\mathbb{E}\left[\hat{Q}(s, a_{\mu}; \theta_i) | s, a_{\mu}, \mu(a_{\mu}|s)\right]}_{\text{maximize REM Q-value under } \mu} \\ & + \omega \hat{MSE}_i(\theta_i, \pi)\} \\ \breve{Q}^{\star} := \lim_{i \to \infty} & \breve{Q}_i^{\pi}\end{aligned} \tag{12}$$

Here, $\hat{MSE}_i(\theta_i, \pi)$ and $\hat{Q}^{\pi}(s, a; \theta_i)$ are as defined in Eq. 11.

To summarize, the sequence of steps is outlined below in Algorithms 1 and 2. The Q-learning algorithm is trained offline with Algorithm 1 and deployed in the TS xApp for online inference and control following Algorithm 2.

## 5 Performance Evaluation

In this section, we first describe the simulation scenario, the baseline handover modes considered for the comparison, and the metrics of interest. We then discuss the results based on a large scale evaluation in different deployment scenarios.

**Dense urban scenario —** We model a dense urban deployment, based on the 3GPP TR 38.913 [57], with $M = 1$ eNB and $N = 7$ gNBs, as shown in Figure 5. One of the gNBs is co-located with the eNB at the center of the scenario, the others provide coverage in an hexagonal grid. Each node has an independent E2 termination, with reporting periodicity set to 100 ms. We study two different configurations:

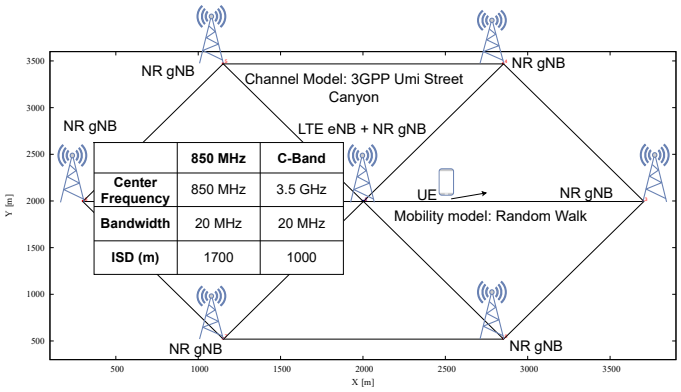



Fig. 5: Simulation scenario.

(i) low band with center frequency 850 MHz and inter-site distance between the gNBs of 1700 m; and (ii) C-band, with center frequency 3.5 GHz and inter-site distance of 1000 m. In each configuration, the bandwidth is 10 MHz for the eNB and 20 MHz for the gNBs. The channel is modeled as a 3GPP Urban Microcell (UMi) street canyon channel [52]. The 3GPP NR gNBs use numerology 2. $N_{\mathrm{UE}} = |U|$ dual-connected UEs are randomly dropped in each simulation run with a uniform distribution, and move according to a random walk process with minimum speed $S_{\min} = 2.0$ m/s and maximum speed $S_{\max} = 4.0$ m/s. This setup represents the average condition for typical 3GPP scenarios (pedestrian to slow vehicle mobility). We focus on the subset of UEs that are more interested by handovers, rather than, for example, static users, with a random walk model to generalize the mobility through the simulations.

**Traffic model —** The users request downlink traffic from a remote server with a mixture of four traffic models, each assigned to 25% of the UEs. The traffic models include (i) full buffer Maximum Bit Rate (MBR) traffic, which saturates at $R_{\mathrm{fb,max}} = 20$ Mbit/s, to simulate file transfer or synchronization with cloud services; (ii) bursty traffic with an average data rate of $R_{\mathrm{b,max}} = 3$ Mbit/s, to model video streaming applications; and (iii) two bursty traffic models with an average data rate of 750 Kbit/s and 150 Kbit/s, for web browsing, instant messaging applications, and Guaranteed Bit Rate (GBR) traffic (e.g., phone calls). The bursty traffic models feature on and off phases with a random exponential duration.

**Baseline Handover Strategies —** We consider three baseline handover models [58] for training the AI agent from in Section 4 and to evaluate its effectiveness. They represent different strategies generally used for handovers in cellular networks [10]. We consider a RAN RRM heuristic, which decides to perform a handover if a target cell has a channel quality metric (in this case, the SINR) above a threshold (specifically, 3 dB) with respect to the current cell. The other algorithms use more advanced heuristics, based on a combination of a threshold and a Time-to-Trigger (TTT). The first (called SON1 in the rest of the paper) assumes a fixed TTT, i.e., the handover is triggered only if the target cell SINR is above a threshold (3 dB) for a fixed amount of time (110 ms). The second (called SON2) uses a dynamic TTT, which is decreased proportionally to the difference between the target and current cell SINR [58].

| Hyperparameters | Value |
|---|---|
| **DQN Agent (Offline)** | |
| Target update period | 8000 |
| Batch size | 32 |
| Number of heads ($n$ heads in Fig. 4) | 200 |
| Number of actions ($N$ in Fig. 4)) | 7 |
| Minimum replay history | 20000 |
| Terminal (Episode) length | 1 |
| Gamma | 0.99 |
| Replay capacity | 1000000 |
| Number of iterations | 400 |
| Training steps | 100000 |
| **Optimizer** | |
| Optimizer | AdamOptimizer |
| Learning rate | 0.00005 |
| **Neural Network (Fig. 4)** | |
| Conv1D Layer | filters=32<br>kernel size=$B$=8<br>strides=$B$=8<br>activation=ReLu |
| Flatten Layer | 225 neurons |
| Dense Layer 1 | 128 neurons |
| Dense Layer 2 | 32 neurons |
| Dense Layer 3 | 1400 neurons |

TABLE 1: RL hyperparameters and their values.

**Performance metrics —** For the performance evaluation of the TS xApp we consider the metrics related to throughput, channel quality, spectral efficiency, and mobility overhead. For the first, we report the average UE throughput at the Packet Data Convergence Protocol (PDCP) layer, i.e., including both LTE and NR split bearers, as well as the $10^{\text{th}}$ and $95^{\text{th}}$ percentiles of all the users in a simulation, averaged over multiple independent runs. The channel quality is represented by the SINR. For the spectral efficiency, we analyze the average value for each UEs and cell, as well as the $10^{\text{th}}$ percentile, and the percentage of PRBs used for downlink traffic. Finally, we evaluate the UE mobility overhead $H_u$ as the number of handovers per unit time weighted by a throughput factor $\hat{R}_u = \mathbb{E}(R_u)/\sum_{u' \in U} \mathbb{E}(R'_u)$, where $\mathbb{E}(R_u)$ is the average throughput for the user over the same unit time.

**Data collection and agent training —** The data collection is based on a total of more than 2000 simulations for the different configurations, including multiple independent simulation runs for each scenario. We used the Simulation Execution Manager (SEM) for ns-3 [59] to create multiple independent permutations of the available parameters for ns-3 scenarios. Each permutation generate an execution of the scenario which is independent from the others and completely reproducible. While collecting the full permutation of all the possible states is impractical, the offline dataset we generated features the combination of all possible parameters that can be configured in the ns-3 scenario, and iterations over different random seeds for each combination. This makes it possible to correctly represent the generalized context of the problem with a reduced bias probability.

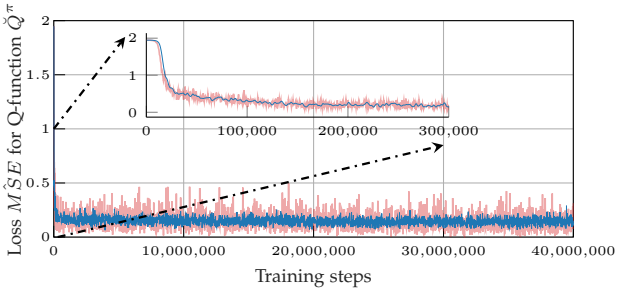



Fig. 6: Loss $\hat{MSE}$ for the Q-function $\breve{Q}^{\pi}$ for the offline training.

Table 1 provides the list of RL hyperparameters and their values considered in this paper. In the offline training, the frequency with which the target network gets updated is set to 8000 training steps. We perform 400 iterations during the offline training, and each iteration has 100K training steps, for a total of 40 million training steps. In a training step a batch of 32 samples (or data points) are selected randomly for input to Neural Network. The first layer of the network is the Conv1D layer with $2^{x_1} = 32$ filters (see Fig. 4). The kernel size and strides are set to $B = 8$, as each cell has $B = 8$ input parameters (Section 4), and the activation function is ReLU. This is followed by a flattening layer which flattens the output of the Conv1D layer (with $y_1 B = 225$) concatenated with the $t - t_{u'}$ parameter. Third, fourth and fifth layers are fully-connected layers with $2^{x_2} = 128$, $2^{x_3} = 32$ and 1400 units/neurons, respectively. The number of units in the last layer is given by the product of $n = 200$, the number of heads of the REM, and the number of actions $N = 7$. We use the Adam optimizer with a learning rate of 0.00005. Figure 6 shows the trend of the loss $\hat{MSE}$ for the Q-function $\breve{Q}^{\pi}$ (as discussed in Section 4) during the training of the RL agent, including a focus on the first $3 \cdot 10^5$ iterations. The initial cost $K_0$ from Eq. 1 is 1, and $\delta$ (decay constant) is 0.1.

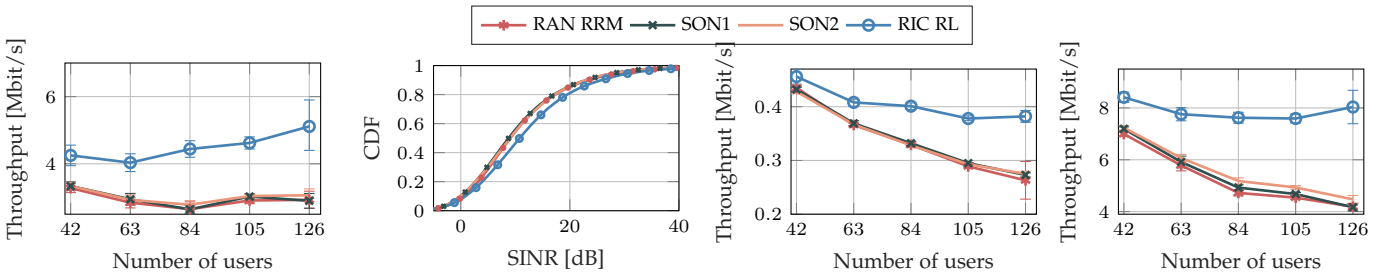
The likelihood of the loss curve $\hat{MSE}$ is regular and its trend approaches values close to zero, showing that the weights of the CNN are actually improving after each iteration and the information learned is a good approximation of the non-linear dependency between the actions and the reward.

### 5.1 Results

In this section, we analyze the results we obtained after the training and the online testing of the xApp described in Section 4. The RL agent was tested in simulations with the baselines Handovers (HOs) disabled. The experiments were repeated with different numbers of UEs, and averaged around 600,000 records for FR1 850 MHz and around 300,000 records for FR1 C-band in online evaluation.

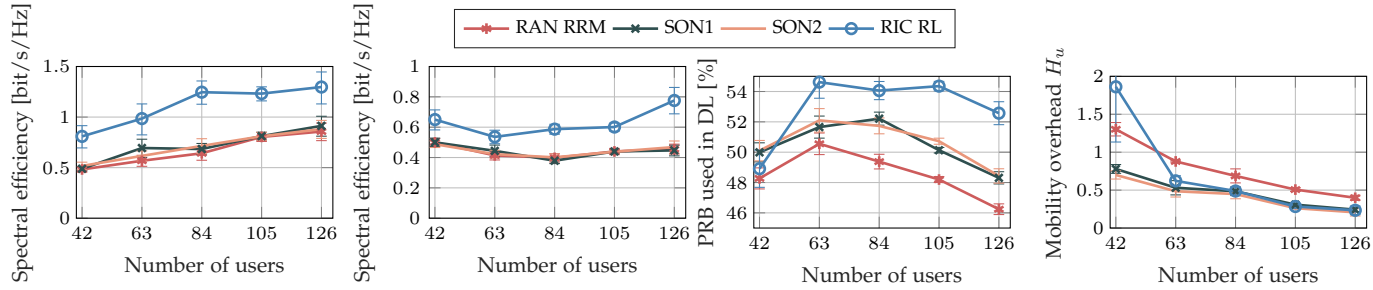
Figure 7a shows the average UE throughput for the 850 MHz deployment, while Fig. 7b reports the CDF of the SINR with 126 UEs. The RIC RL introduces an improvement of the average throughput (averaging around 50%) and SINR with respect to the baselines, meaning that the RL agent is able to customize the HO-control for each single UE. This trend is also confirmed as the number of UEs increases, proving the scalability of this approach over baselines. Indeed, while the total amount of resources available to the network does not change, the RIC RL algorithm leverages handovers to perform load balancing across cells (i.e., moving users from loaded cells to base stations with lower resource utilization) and to improve the channel quality that the user is experiencing. Overall, this leads to a network configuration where

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3266642




(a) Average UE throughput (LTE+NR). (b) SINR CDF for 126 users. (c) $10^{\text{th}}$ percentile user throughput. (d) $95^{\text{th}}$ percentile user throughput.

Fig. 7: Throughput and SINR Cumulative Distribution Function (CDF) for the 850 MHz deployment, for the different baselines and the xApp-driven handover control. The average throughput accounts for the traffic on the LTE and NR split bearer.




(a) Average spectral efficiency, per UE. (b) Average spectral efficiency, per cell. (c) Average number of PRB used in downlink, per cell, percentage. (d) UE mobility overhead $H_u$.

Fig. 8: Spectral efficiency and mobility overhead metrics for the 850 MHz deployment, as a function of the number of users, for the different baselines and the xApp-driven handover control.

resources are better matched to users' demands. In addition, as the number of users increases, the RIC RL algorithm can exploit additional degrees of freedom to perform its optimization, e.g., in terms of where and when to offload users across cells. This is enabled by the RIC and xApps ecosystem, which allows for the customization of the mobility of individual UEs through a centralized and abstracted view of the network status.

To further analyze the throughput trend of Fig. 7a, we report in Fig. 9 the average UE throughput for the RIC RL and SON2 algorithms and for different kinds of source user traffic, specifically, full buffer and video streaming. It can be seen that for 42 users, the video streaming performance is saturated (considering the combination of the channel conditions and source traffic requests), thus the full buffer users request additional traffic. As the number of users increases, the network becomes more loaded, thus the full buffer users back off (also considering that the scheduler is primarily a round robin scheduler), while the traffic demand from the other users keeps increasing, up to the point where the average throughput for the two classes is similar (126 users). The configuration with 63 users represents the inflection point, as the video streaming/bursty traffic has not increased but the full buffer traffic request decreases by about 1.2 Mbps. Overall, this leads to the non-monotonic behavior of the throughput in Fig. 7a, which decreases from 42 to 63 users and then increases again.

Additional insights on the performance improvement are provided by the percentiles of the user throughput (Fig. 7). It can be seen that our RL agent brings consistent improvement not only on the average UEs, but also between the worst (10-th percentile, Fig. 7c) users, showing 30% improvements and best (95-th percentile, Fig. 7d) users, showing around 60% improvement. The 126 UEs result is particularly relevant, as also testified by the improvement in SINR shown in Fig. 7b (the median point with RIC RL is 1.99 dB higher than the median for SON2). Note that the variance in performance across users increases in the simulations for 126 users compared to scenarios with fewer users, leading to larger confidence intervals in the throughput and spectral efficiency metrics. Contrary to heuristic-based HOs, the RL algorithm leverages UE-level and cell-level KPMs to take the decision to centrally handover/steer the user to an optimal NR neighbor, in terms of load and SINR. This results in an improved spectral efficiency (and thus throughput), as shown in Figs. 8a and 8b, demonstrating 52% and 35% improvements, respectively. The same holds for the PRB utilization (Fig. 8c). Indeed, since RIC RL utilizes cell-level KPMs at 100 ms granularity, it is able to handover UEs to a target cell with higher residual PRBs. The non monotonic trend of the PRB utilization can be explained by the interaction between the source traffic models, the RAN scheduling process, and the handover policies, with the full-buffer users requesting fewer resources as the network becomes more congested. Nonetheless, the trend observed in Fig. 8c differs for the baseline approaches and for the proposed RIC RL algorithm, showing that the latter is more efficient in using the available resources and in allowing the mobile terminals to request more traffic.

However, these improvements in the throughput could eventually come with a major cost in terms of HO management, and thus energy. The mobility overhead $H_u$ of Fig. 8d

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3266642

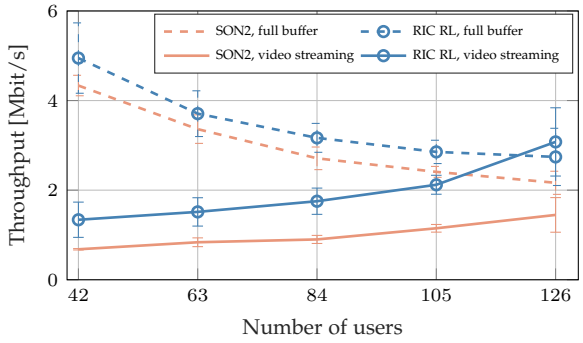



Fig. 9: Comparison between the average UE throughput for RIC RL and SON2 for different kinds of source user traffic (full buffer, video streaming).

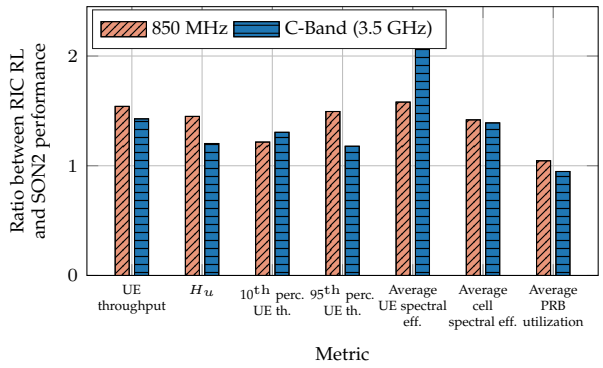



Fig. 10: Comparison between the performance gain in the 850 MHz band and in the 3.5 GHz band (or C-Band). Each bar represents the ratio between the performance with the RIC RL and SON2 for the corresponding metric.

clearly shows that our RL agent is not causing more HOs, but instead follows the trend of the baselines, while at the same time delivering better throughput. The only exception is for 42 UEs, where the RL agent triggers more HOs than all baselines. One of the possible reasons can be identified in the cost function described in Eq. (2) (Section 4), where the reward (logarithmic throughput gain, which is higher with fewer users) compensates for the cost of handover thereby resulting in an increase in mobility overhead $H_u$.

Furthermore, Fig. 10 compares the already discussed results for 850 MHz with the C-Band deployment. In this figure, we show the relative gains of the performances of the RL agent in the two bands. The gain of each KPM shown in the x-axis is defined as the ratio between the performance with the RIC RL and SON2 for the corresponding metric. Given this definition, the RL agent is performing better than the baseline when the ratio is greater than 1. The analysis of the relative gains shows that while the average PRB utilization of the RIC falls below the baseline, the other KPMs improves consistently, showing the adaptability of RIC RL through different bands.

We also compare the performance of the proposed RIC-enabled RL agent against the contextual multi-armed bandit RL agent proposed in [39]. To do so, we implemented the agent from [39] in the xApp, and trained it on the same dataset used to train our agent. Fig. 11 compares the performance between the two agents in terms of 10$^{\text{th}}$ percentile throughput, for different numbers of users and

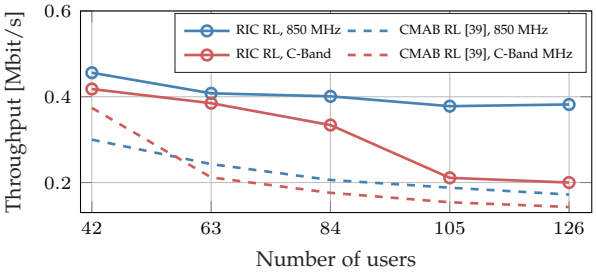



Fig. 11: Comparison between the 10$^{\text{th}}$ percentile user throughput with the proposed xApp (RIC RL) and an xApp implementing the handover control logic from [39].

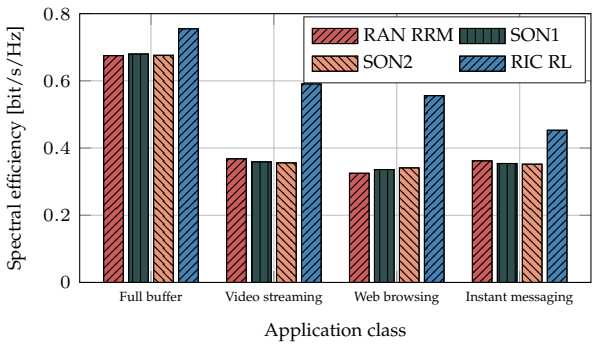



Fig. 12: Average cell spectral efficiency for the different traffic types, for 105 users and the 850 MHz deployment.

for deployments in the 850 MHz and C-Band frequencies. The method proposed in [39] aims at improving the RSRP for individual UEs through handover to different cells. However, this does not improve the cell-edge throughput as much as the RIC-enabled RL optimization proposed in this paper, which provides consistently higher cell-edge user throughput in the two frequency bands and with different numbers of users. Compared to [39], the O-RAN-driven solution we introduce in this paper exploits a richer input feature set, which makes it possible to characterize the user status with higher precision, and thus to select control actions that go beyond the RSRP improvement optimizing the user throughput itself.

Finally, one key aspect enabled by the per-UE control made possible by our xApps design and the O-RAN architecture is the possibility of improving the performances of a heterogeneous UEs, with different traffic models. Fig. 12 indeed shows the average cell spectral efficiency for the different traffic types, for 105 users and the 850 MHz deployment. Thanks to the optimized handover management, the RIC RL policy is able to improve the conditions of the all the UEs with significant gains for traffic models such as the video streaming, the web browsing, and the instant messages, whose performance fails to be optimized by the baselines policies.

## 6 Conclusions

This paper introduced a complete, system-level, O-RAN-compliant framework for the optimization of TS in 3GPP networks. Specifically, we focused on throughput maximization through the selection of the NR serving cell in an EN-DC setup. We implemented a cloud-native near-RT RIC,

which we connect through open, O-RAN interfaces to a simulated RAN environment in ns-3. We developed a custom xApp for the near-RT RIC, with a data-driven handover control based on REM and CQL. Finally, we profiled the performance of the agent on a large scale deployment in multiple frequency bands, evaluating its gain over traditional handover heuristics. The results show that, thanks to the UE-level control at the near-RT RIC, our solution achieves significant performance improvements ranging from 30% to 50% for the average throughput and spectral efficiency, demonstrating its effectiveness over different combinations of UEs.

## REFERENCES


[1] K. B. Letaief, W. Chen, Y. Shi, J. Zhang, and Y. A. Zhang, "The Roadmap to 6G: AI Empowered Wireless Networks," *IEEE Communications Magazine*, vol. 57, no. 8, pp. 84–90, August 2019.

[2] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges," *arXiv:2202.01032 [cs.NI]*, February 2022. [Online]. Available: https://arxiv.org/abs/2202.01032

[3] O-RAN Working Group 3, "O-RAN Near-Real-time RAN Intelligent Controller Architecture & E2 General Aspects and Principles 2.00," O-RAN.WG3.E2GAP-v02.01 Technical Specification, July 2021.

[4] A. S. Abdalla, P. S. Upadhyaya, V. K. Shah, and V. Marojevic, "Toward Next Generation Open Radio Access Network–What O-RAN Can and Cannot Do!" *arXiv preprint arXiv:2111.13754 [cs.NI]*, November 2021.

[5] U. Challita, H. Ryden, and H. Tullberg, "When machine learning meets wireless cellular networks: Deployment, challenges, and applications," *IEEE Communications Magazine*, vol. 58, no. 6, pp. 12–18, June 2020.

[6] S. Chinchali *et al.*, "Cellular network traffic scheduling with deep reinforcement learning," in *Proc. of Thirty-Second AAAI Conf. on Artificial Intelligence*, New Orleans, LA, 2018, pp. 766–774.

[7] L. Bonati, S. D'Oro, M. Polese, S. Basagni, and T. Melodia, "Intelligence and Learning in O-RAN for Data-driven NextG Cellular Networks," *IEEE Communications Magazine*, vol. 59, no. 10, pp. 21–27, October 2021.

[8] S. Mollahasani, M. Erol-Kantarci, and R. Wilson, "Dynamic CU-DU Selection for Resource Allocation in O-RAN Using Actor-Critic Learning," *arXiv preprint arXiv:2110.00492 [cs.NI]*, October 2021.

[9] O-RAN Working Group 1, "O-RAN Use Cases Detailed Specification 6.0," O-RAN.WG1.Use-Cases-Detailed-Specification-v06.00 Technical Specification, July 2021.

[10] M. Tayyab, X. Gelabert, and R. Jantti, "A survey on handover management: From lte to nr," *IEEE Access*, vol. 7, pp. 118 907–118 930, 2019.

[11] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "ColO-RAN: Developing Machine Learning-based xApps for Open RAN Closed-loop Control on Programmable Experimental Platforms," *IEEE Transactions on Mobile Computing*, pp. 1–14, 2022.

[12] J. Tanveer, A. Haider, R. Ali, and A. Kim, "An Overview of Reinforcement Learning Algorithms for Handover Management in 5G Ultra-Dense Small Cell Networks," *Applied Sciences*, vol. 12, no. 1, 2022.

[13] M. Mezzavilla *et al.*, "End-to-end simulation of 5G mmWave networks," *IEEE Communications Surveys Tutorials*, vol. 20, no. 3, pp. 2237–2263, April 2018.

[14] L. Bonati, M. Polese, S. D'Oro, S. Basagni, and T. Melodia, "Open-RAN Gym: An Open Toolbox for Data Collection and Experimentation with AI in O-RAN," in *Proc. of IEEE WCNC Workshop on Open RAN Architecture for 5G Evolution and 6G*, Austin, TX, USA, April 2022.

[15] O-RAN Working Group 3, "O-RAN use cases and requirements 2.00," O-RAN.WG3.UCR-v02.00, July 2022.

[16] ——, "O-RAN near-real-time RAN intelligent controller E2 service model 2.00," ORAN-WG3.E2SM-v02.00 Technical Specification, July 2021.

[17] ——, "O-RAN near-real-time RAN intelligent controller E2 service model (E2SM) KPM 2.0," ORAN-WG3.E2SM-KPM-v02.00 Technical Specification, July 2021.

[18] ——, "O-RAN near-real-time RAN intelligent controller E2 service model, ran control 1.0," ORAN-WG3.E2SM-RC-v01.00 Technical Specification, July 2021.

[19] O-RAN Working Group 2, "O-RAN AI/ML workflow description and requirements 1.03," O-RAN.WG2.AIML-v01.03 Technical Specification, July 2021.

[20] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv:2005.01643 [cs.LG]*, 2020. [Online]. Available: https://arxiv.org/abs/2005.01643

[21] Y. Koda, K. Yamamoto, T. Nishio, and M. Morikura, "Reinforcement learning based predictive handover for pedestrian-aware mmwave networks," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2018, pp. 692–697.

[22] Q. Liu, C. F. Kwong, S. Wei, L. Li, and S. Zhang, "Intelligent handover triggering mechanism in 5g ultra-dense networks via clustering-based reinforcement learning," *Mobile Networks and Applications*, vol. 26, no. 1, pp. 27–39, 2021.

[23] S. Mwanje and A. Mitschele-Thiel, "Minimizing handover performance degradation due to lte self organized mobility load balancing," in *IEEE 77th Vehicular Technology Conference (VTC Spring)*, 2013.

[24] D. Guo, L. Tang, X. Zhang, and Y. Liang, "Joint optimization of handover control and power allocation based on multi-agent deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 13 124–13 138, 2020.

[25] Z. Wang, L. Li, Y. Xu, H. Tian, and S. Cui, "Handover control in wireless systems via asynchronous multiuser deep reinforcement learning," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4296–4307, December 2018.

[26] M. S. Mollel *et al.*, "Intelligent handover decision scheme using double deep reinforcement learning," *Physical Communication*, vol. 42, p. 101133, 2020.

[27] S. Wang, H. Liu, P. Gomes, and B. Krishnamachari, "Deep reinforcement learning for dynamic multichannel access in wireless networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 4, no. 2, pp. 257–265, June 2018.

[28] M. Sana, A. De Domenico, E. Strinati, and A. Clemente, "Multi-agent deep reinforcement learning for distributed handover management in dense mmwave networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 8976–8980.

[29] P. Muñoz, R. Barco, and I. de la Bandera, "Load balancing and handover joint optimization in lte networks using fuzzy logic and reinforcement learning," *Computer Networks*, vol. 76, pp. 112–125, 2015.

[30] F. D. Calabrese *et al.*, "Learning radio resource management in rans: Framework, opportunities, and challenges," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 138–145, 2018.

[31] J. Li, K. Xue, J. Liu, and Y. Zhang, "A user-centric handover scheme for ultra-dense leo satellite networks," *IEEE Wireless Communications Letters*, vol. 9, no. 11, pp. 1904–1908, 2020.

[32] H. Xu *et al.*, "Qoe-driven intelligent handover for user-centric mobile satellite networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 9, pp. 10 127–10 139, 2020.

[33] N. M. Kibinda and X. Ge, "User-centric cooperative transmissions-enabled handover for ultra-dense networks," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 4, pp. 4184–4197, 2022.

[34] E. Demarchou, C. Psomas, and I. Krikidis, "Intelligent user-centric handover scheme in ultra-dense cellular networks," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, 2017, pp. 1–6.

[35] N. Kibinda and X. Ge, "Advanced group-cell handover skipping for user-centric cooperative communications in dense cellular networks," in *2022 7th International Conference on Computer and Communication Systems (ICCCS)*, 2022, pp. 430–435.

[36] C. D'Andrea, G. Interdonato, and S. Buzzi, "User-centric handover in mmwave cell-free massive mimo with user mobility," in *2021 29th European Signal Processing Conference (EUSIPCO)*, 2021, pp. 1–5.

[37] Y. Lin *et al.*, "Heterogeneous user-centric cluster migration improves the connectivity-handover trade-off in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 16 027–16 043, 2020.

[38] M. Dryjański, L. Kułacz, and A. Kliks, "Toward modular and flexible open ran implementations in 6g networks: Traffic steering use case and o-ran xapps," *Sensors*, vol. 21, no. 24, 2021.
[39] V. Yajnanarayana, H. Rydén, and L. Hévizi, "5G Handover using Reinforcement Learning," in *IEEE 3rd 5G World Forum (5GWF)*, 2020, pp. 349–354.
[40] T. Karamplias *et al.*, "Towards closed-loop automation in 5g open ran: Coupling an open-source simulator with xapps," in *2022 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*, 2022, pp. 232–237.
[41] R. Bitton *et al.*, "Adversarial machine learning threat analysis in open radio access networks," *arXiv preprint arXiv:2201.06093*, 2022.
[42] M. Kouchaki and V. Marojevic, "Actor-critic network for o-ran resource allocation: xapp design, deployment, and analysis," in *2022 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2022, pp. 968–973.
[43] K. Ramezanpour and J. Jagannath, "Intelligent zero trust architecture for 5g/6g networks: Principles, challenges, and the role of machine learning in the context of o-ran," *Computer Networks*, vol. 217, p. 109358, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128622003929
[44] C. Puligheddu, J. Ashdown, C. F. Chiasserini, and F. Restuccia, "Sem-o-ran: Semantic and flexible o-ran slicing for nextg edge-assisted mobile systems," *arXiv preprint arXiv:2212.11853*, 2022.
[45] J. Thaliath *et al.*, "Predictive closed-loop service automation in o-ran based network slicing," *IEEE Communications Standards Magazine*, vol. 6, no. 3, pp. 8–14, 2022.
[46] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," in *Modeling and tools for network simulation*. Springer, 2010, pp. 15–34.
[47] T. Zugno *et al.*, "Implementation of a spatial channel model for ns-3," in *Proceedings of the 2020 Workshop on Ns-3*, ser. WNS3 2020. New York, NY, USA: Association for Computing Machinery, 2020, p. 49–56.
[48] P. Gawłowicz and A. Zubow, "ns-3 meets OpenAI Gym: The Playground for Machine Learning in Networking Research," in *ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, November 2019. [Online]. Available: http://www.tkn.tu-berlin.de/fileadmin/fg112/Papers/2019/gawlowicz19_mswim.pdf
[49] H. Yin *et al.*, "Ns3-ai: Fostering artificial intelligence algorithms for networking research," in *Proceedings of the 2020 Workshop on Ns-3*, ser. WNS3 2020. New York, NY, USA: Association for Computing Machinery, 2020, p. 57–64. [Online]. Available: https://doi.org/10.1145/3389400.3389404
[50] 3GPP, "NR; multi-connectivity; overall description; stage-2," 3GPP, Technical Specification (TS) 37.340, 12 2021, version 16.8.0.
[51] B. Balasubramanian *et al.*, "Ric: A ran intelligent controller platform for ai-enabled cellular networks," *IEEE Internet Computing*, vol. 25, no. 2, pp. 7–17, 2021.
[52] 3GPP, "Study on channel model for frequencies from 0.5 to 100 ghz," 3GPP, Technical Specification (TS) 38.901, 1 2020, version 16.1.0.
[53] O-RAN Software Community. (2022) sim-e2-interface repository. https://github.com/o-ran-sc/sim-e2-interface. Accessed March 2022.
[54] V. Mnih *et al.*, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.
[55] R. Agarwal, D. Schuurmans, and M. Norouzi, "An optimistic perspective on offline reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 104–114.
[56] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative Q-Learning for Offline Reinforcement Learning," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 1179–1191.
[57] 3GPP, "Study on scenarios and requirements for next generation access technologies," 3GPP, Technical Release (TR) 38.913, 12 2015, version 14.3.0.
[58] M. Polese, M. Giordani, M. Mezzavilla, S. Rangan, and M. Zorzi, "Improved handover through dual connectivity in 5g mmwave mobile networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 9, pp. 2069–2084, 2017.
[59] D. Magrin, D. Zhou, and M. Zorzi, "A simulation execution manager for ns-3: Encouraging reproducibility and simplifying statistical analysis of ns-3 simulations," in *Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2019, pp. 121–125.

**Andrea Lacava** received his B.S. in Computer Engineering and his M.S. in Cybersecurity from Sapienza, University of Rome, Italy in 2018 and 2021, respectively. He is currently pursuing a double Ph.D. degree in Computer Engineering at the Institute for the Wireless Internet of Things at Northeastern University, MA, USA and in Information and Communication Technology (ICT) at Sapienza, University of Rome, Italy. His research interests focus on the O-RAN architecture, 5G and beyond cellular networks. He is IEEE graduate student member.

**Michele Polese** is a Principal Research Scientist at the Institute for the Wireless Internet of Things, Northeastern University, Boston, since March 2020. He received his Ph.D. at the Department of Information Engineering of the University of Padova in 2020. He also was an adjunct professor and postdoctoral researcher in 2019/2020 at the University of Padova, and a part-time lecturer in Fall 2020 and 2021 at Northeastern University. During his Ph.D., he visited New York University (NYU), AT&T Labs in Bedminster, NJ, and Northeastern University. His research interests are in the analysis and development of protocols and architectures for future generations of cellular networks (5G and beyond), in particular for millimeter-wave and terahertz networks, spectrum sharing and passive/active user coexistence, open RAN development, and the performance evaluation of end-to-end, complex networks. He has contributed to O-RAN technical specifications and submitted responses to multiple FCC and NTIA notice of inquiry and requests for comments, and is a member of the Committee on Radio Frequency Allocations of the American Meteorological Society (2022-2024). He collaborates and has collaborated with several academic and industrial research partners, including AT&T, Mavenir, NVIDIA, InterDigital, NYU, University of Aalborg, King's College, and NIST. He was awarded with several best paper awards, is serving as TPC co-chair for WNS3 2021-2022, as an Associate Technical Editor for the IEEE Communications Magazine, and has organized the Open 5G Forum in Fall 2021. He is a Member of the IEEE.

**Rajarajan Sivaraj** is director for RIC architecture and standards at Mavenir, where he is responsible for the standardization and productization of the Near-RT RIC, Non-RT RIC, SMO, xApps, rApps and interfaces. He holds a PhD in Computer Science from UC Davis, and has a career spanning over 12 years in cellular telecommunications with prior positions at AT&T labs, Microsoft Research, Intel labs, NEC labs, Broadcom, Uhana (VMWare), etc. He has had numerous publications and granted patents, and his scholarly works have been highly cited.

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3266642

**Rahul Soundrarajan** is Sr. Director for RAN Analytics at Mavenir. As of his role, he is responsible for design, development and evaluation of Machine Learning Algorithms for Near-RT RIC and Non-RT RIC. His career spanning 21 years is a rich mix of Systems Architecture and Engineering of 2/3/4G RAN and applying ML algorithms for network optimization, for which he holds several patents. His prior experiences include positions at Lucent, Alcatel-Lucent, Nokia and HCL Technologies.

**Bhawani Shanker Bhati** received the Ph.D. degree in engineering from the Indian Institute of Science (IISc), Bengaluru, India, in 2018. He is currently a Senior Member of Technical Staff at Mavenir, India. His research interests include ad-hoc networks, Near-RT RIC, communication protocols, ubiquitous computing, security, and privacy in wireless networks.

**Tarunjeet Singh** is Director Engineering at Mavenir Systems. He has wide array of experience in data analytics, cloud native technologies, telecom networks, telecom VAS and customer experience and billing. He received his Bachelor's degree in Computer Science in 2002, from the University of Delhi, India. In his current role, he is leading R&D for Near RT RIC platform and xApps at Mavenir Systems. He has held positions with erstwhile Alcatel-Lucent, Nokia and Bharti Airtel providing him exposure to the engineering methods of some of the key leaders and innovators in the telecom space.

**Tommaso Zugno** received the Ph.D. degree from the Department of Information Engineering, University of Padua, in 2022. From May 2018 to October 2018, he was a Postgraduate Researcher with the Department of Information Engineering, University of Padua. He is with Huawei Technologies, Munich Research Center, Germany. His research interests include the design and evaluation of algorithms and architectures for next-generation cellular networks. He was awarded the Best Paper Awards at WNS3 2020 and IEEE MedComNet 2020.

**Francesca Cuomo** received the Ph.D. in Information and Communications Engineering in 1998 from Sapienza University of Rome. From 2005 to October 2020 she was Associate Professor and from November 2020 she joined "Sapienza" as Full Professor teaching courses in Telecommunication and Networks. Prof. Cuomo has advised numerous master students in computer engineering, and has been the advisor of 13 PhD students in Networking. Her current research interests focus on: Vehicular networks and Sensor networks, Low Power Wide Area Networks and IoT, 5G Networks, Multimedia Networking, Energy saving in the Internet and in the wireless system. Francesca Cuomo has authored over 156 peer-reviewed papers published in prominent international journals and conferences. Her Google Scholar h-index is 30 with over 3850 citations. Relevant scientific international recognitions: Two Best Paper Awards. She has been in the editorial board of Computer Networks (Elsevier) and now is member of the editorial board of the Ad-Hoc Networks (Elsevier), IEEE Transactions on Mobile Computing, Sensors (MDPI), Frontiers in Communications and Networks Journal. She has been the TPC co-chair of several editions of the ACM PE-WASUN workshop, TPC Co-Chair of ICCCN 2016, TPC Symposium Chair of IEEE WiMob 2017, General Co-Chair of the First Workshop on Sustainable Networking through Machine Learning and Internet of Things (SMILING), in conjunction with IEEE INFOCOM 2019; Workshop Co-Chair of AmI 2019: European Conference on Ambient Intelligence 2019. She is IEEE senior member.

**Tommaso Melodia** is the William Lincoln Smith Chair Professor with the Department of Electrical and Computer Engineering at Northeastern University in Boston. He is also the Founding Director of the Institute for the Wireless Internet of Things and the Director of Research for the PAWR Project Office. He received his Ph.D. in Electrical and Computer Engineering from the Georgia Institute of Technology in 2007. He is a recipient of the National Science Foundation CAREER award. Prof. Melodia has served as Associate Editor of IEEE Transactions on Wireless Communications, IEEE Transactions on Mobile Computing, Elsevier Computer Networks, among others. He has served as Technical Program Committee Chair for IEEE Infocom 2018, General Chair for IEEE SECON 2019, ACM Nanocom 2019, and ACM WUWnet 2014. Prof. Melodia is the Director of Research for the Platforms for Advanced Wireless Research (PAWR) Project Office, a $100M public-private partnership to establish 4 city-scale platforms for wireless research to advance the US wireless ecosystem in years to come. Prof. Melodia's research on modeling, optimization, and experimental evaluation of Internet-of-Things and wireless networked systems has been funded by the National Science Foundation, the Air Force Research Laboratory the Office of Naval Research, DARPA, and the Army Research Laboratory. Prof. Melodia is a Fellow of the IEEE and a Senior Member of the ACM.