

ns-O-RAN: Simulating O-RAN 5G Systems in ns-3

Andrea Lacava*
Northeastern University
Boston, Massachusetts, USA
lacava.a@northeastern.edu

Matteo Bordin
Northeastern University
Boston, Massachusetts, USA
bordin.m@northeastern.edu

Michele Polese
Northeastern University
Boston, Massachusetts, USA
m.polese@northeastern.edu

Rajarajan Sivaraj
Mavenir, Inc.
Richardson, Texas, USA
rajarajan.sivaraj@mavenir.com

Tommaso Zugno
University of Padova
Padova, Italy
tommasozugno@gmail.com

Francesca Cuomo
University of Rome
Rome, Italy
francesca.cuomo@uniroma1.it

Tommaso Melodia
Northeastern University
Boston, Massachusetts, USA
melodia@northeastern.edu

ABSTRACT

O-RAN is radically shifting how cellular networks are designed, deployed and optimized through network programmability, disaggregation, and virtualization. Specifically, RAN Intelligent Controllers (RICs) can orchestrate and optimize the Radio Access Network (RAN) operations, allowing fine-grained control over the network. RICs provide new approaches and solutions for classical use cases such as on-demand traffic steering, anomaly detection, and Quality of Service (QoS) management, with an optimization that can target single User Equipments (UEs), slices, cells, or entire base stations. Such control can leverage data-driven approaches, which rely on the O-RAN open interfaces to combine large-scale collection of RAN Key Performance Measurements (KPMs) and state-of-the-art Machine Learning (ML) routines executed in the RICs. While this comes with the potential to enable intelligent, programmable RANs, there are still significant challenges to be faced, primarily related to data collection at scale, development and testing of custom control logic for the RICs, and availability of Open RAN simulation and experimental tools for the research and development communities. To address this, we introduce ns-O-RAN, a software integration between a real-world near-real-time RIC and an ns-3 simulated RAN which provides a platform for researchers and telco operators to build, test and integrate xApps. ns-O-RAN extends a popular Open RAN experimental framework (OpenRAN Gym) with simulation capabilities that enable the generation of realistic datasets without the need for experimental infrastructure. We implement it as a new open-source ns-3 module that uses the E2 interface to connect different simulated 5G base stations with the RIC, enabling

the exchange of E2 messages and RAN KPMs to be consumed by standard xApps. Furthermore, we test ns-O-RAN with the O-RAN Software Community (OSC) and OpenRAN Gym RICs, simplifying the onboarding from a test environment to production with real telecom hardware controlled without major reconfigurations required. ns-O-RAN is open source and publicly available, together with quick-start tutorials and documentation.

CCS CONCEPTS

• **Networks** → Network simulations; Programming interfaces.

KEYWORDS

O-RAN, Open RAN, RIC, ns-3, 5G, simulation

ACM Reference Format:

Andrea Lacava, Matteo Bordin, Michele Polese, Rajarajan Sivaraj, Tommaso Zugno, Francesca Cuomo, and Tommaso Melodia. 2023. ns-O-RAN: Simulating O-RAN 5G Systems in ns-3. In *2023 Workshop on ns-3 (WNS3 2023)*, June 28–29, 2023, Arlington, VA, USA. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3592149.3592161>

1 INTRODUCTION

Emerging wireless networking use cases (e.g., support for diverse traffic, private networks, and ultra-dense networks) are increasing the complexity of network deployments. Traditional and monolithic Radio Access Networks (RANs) architectures cannot support these demands, due to their inflexible, hardware-based designs, the need for on-site setup and labor, and the lack of support for customizable algorithmic optimization [7]. The Open RAN paradigm is drastically changing the approach to the management and optimization of cellular networks, moving from static designs to more advanced and flexible network architectures, such as cloud-native and virtualized RANs that can better support the demands of 5th generation (5G) network use cases and provide improved network performance.

Specifically, the O-RAN Alliance is implementing the Open RAN vision through technical specifications that extend the capabilities 3rd Generation Partnership Project (3GPP) networks. O-RAN networks are based on disaggregation, with network functions split across multiple software and white-box hardware components,

*Also with University of Rome, Rome, Italy.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

WNS3 2023, June 28–29, 2023, Arlington, VA, USA

© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-0747-6/23/06...\$15.00

<https://doi.org/10.1145/3592149.3592161>

virtualization, and programmability of the network. Algorithmic control is enabled by the RAN Intelligent Controller (RIC), a centralized abstraction of the network that has access to all the analytics collected by the RAN functions and can apply control actions. The RIC extends classical Radio Resource Management (RRM) with data-driven approaches based on live telemetry from the RAN processed by Machine Learning (ML) and Artificial Intelligence (AI) pipelines [4]. These models are on-boarded in the software containers called xApps, which operate on a time scale between 10 ms and 1 s on the near-real-time RIC, and rApps, for control loops above 1 s in the non-real-time RIC [9].

However, several challenges must be addressed to fully enable intelligent control in O-RAN, primarily in the area of datasets—that need to be representative and generalize well—and testing of the proposed solutions—which should not impact production network performance. Indeed, O-RAN control needs to be effective at a large scale, and in generic scenarios, leveraging interactions and patterns that emerge when hundreds of User Equipments (UEs) interact with the network. Such patterns are hard to reproduce because they require the use of expensive hardware, such as Software-defined Radios (SDRs) or commercial equipment, or large-scale production deployments to make sure that the data acquired can help the AI model generalize well. The SDR-based framework OpenRAN Gym [6] has been designed to enable the end-to-end design of AI/ML pipelines for O-RAN and has shown very promising results for a variety of control use cases (e.g., slicing, neutral host applications), but still require dedicated hardware and experimental platforms. This practical limitation is usually addressed by the use of simulators, however, their integration with the O-RAN framework requires different development pipelines, meaning that moving a trained policy from the simulated environment to its real deployment as xApp requires additional efforts and thus costs [8].

To address such challenges, in this paper, we present ns-O-RAN, the first open-source simulation platform that combines a functional 4G/5G protocol stack in ns-3 with an O-RAN-compliant E2 interface to enable the communication between the simulated environment and a near-real-time RIC. ns-O-RAN extends the OpenRAN Gym framework with simulation capabilities in ns-3, enabling dataset generation for large-scale scenarios with hundreds of UEs. ns-O-RAN is open source and publicly available as part of the O-RAN Software Community (OSC) projects, together with tutorials and the additional Open RAN Gym components¹.

Through ns-O-RAN, we provide the capability to define simulated closed-loop control routines, where ns-O-RAN provides the cell Key Performance Measurements (KPMs) to the near-real-time RIC, and xApps in this platform define control actions which are sent back to the simulated environment. The latter then applies the control actions to the RAN model, adapting the behavior of the simulation according to the strategy defined by the O-RAN infrastructure. This enables the study of use cases that usually are related to large deployments such as Traffic Steering (TS) (e.g., to load balance users across cells), Quality of Service (QoS) (e.g., to control bearer parameters), and more. Specifically, we integrated ns-O-RAN with the base station implementation of the 5G module for ns-3 introduced in [13].

ns-O-RAN also increases the versatility of the OpenRAN Gym approach, where the development of a xApp and AI-based control policies can be first done on simulation, leveraging its capability of generating large-scale 3GPP-compliant datasets, and then transitioned to experimental platforms such as Colosseum for emulation with hardware in the loop [5], and to over-the-air testbeds for further testing [6]. This can be done by exploiting a key characteristic of the ns-O-RAN design: we connect the emulated environment to any O-RAN-compliant near-real-time RIC with real xApps, which do not need to be re-implemented from scratch to work in the simulated and/or in the experimental environments. This is a deliberate design choice that simplifies the development life-cycle of end-to-end, intelligent control solutions for Open RAN.

ns-O-RAN has been used in [11] to demonstrate intelligent control of handovers in large-scale 5G scenarios. This paper provides a technical overview of how ns-O-RAN has been implemented, together with a tutorial on how it can be deployed with the OpenRAN Gym RIC and a profiling of the performance of the E2 interface. The remainder of this paper is organized as follows. In Section 2, we present the O-RAN architecture and its main principles, describing what is the current state of the art for experimenting with this architecture. In Section 3, we introduce ns-O-RAN, discussing its open source components and the technical choices adopted to enable the end-to-end communication with the near-RT RIC. In Sections 4 and 5, we present a practical example of an ns-O-RAN use case with a simple ns-3 scenario that is able to communicate with the near-real-time RIC, discussing also its preliminary results and showing the flow of the KPMs from the simulated environment to the OpenRAN Gym near-real-time RIC [17]. We conclude the paper and discuss what are the next steps in further extending ns-O-RAN in Section 6.

2 TOWARD OPEN RAN NETWORKS

The O-RAN architecture is a new approach to building mobile networks with greater innovation, competition, and interoperability in the telecommunications industry. The O-RAN architecture disrupts the classical approach by adopting the principles of *disaggregation*, *openness*, *virtualization*, and *programmability*, making it possible to expose data and analytics and enabling data-driven optimization, closed-loop control, and automation. In this section, we provide a brief introduction to the O-RAN architecture and then review the state of the art on tools for the design and development of O-RAN solutions.

2.1 O-RAN: A Primer

The RAN disaggregation splits base stations into different functional units: the Radio Unit (RU), the Distributed Unit (DU) and the Centralized Unit (CU). The RU manages Radio Frequency (RF) components and part of the physical layer, the DU provides support for the higher part of the physical layer, Medium Access Control (MAC), and Radio Link Control (RLC) layers, and the CU features the higher layers of the protocol stack such as Service Data Adaptation Protocol (SDAP), Packet Data Convergence Protocol (PDCP) and Radio Resource Control (RRC). This separation allows for the

¹openrangym.com

CU, DU and RU to be developed, procured and operated independently, enabling a more flexible and cost-effective network deployment [18]. The interfaces between the different nodes are *open* and standardized, to expose RAN telemetry and control to the external world, to achieve multi-vendor interoperability and the integration of different vendors' equipment and solutions into the network.

An additional core innovation is the RIC, a new architectural component that provides a centralized abstraction of the network, allowing operators to implement and deploy custom control plane functions for the integration of new technologies, such as 5G and AI, into the network. O-RAN envisions different loops operating at timescales that perform management and control of the network at near-real-time from 10 ms to 1 s, through third-party applications called xApps, and non-real-time, i.e., for more than 1 s through applications called rApps.

The near-RT RIC is the core of the control and optimization of the RAN and its main components are the xApps. A xApp is a plug-and-play element that implements custom logic, for example for RAN data analysis and RAN control. xApps can receive data and telemetry from the RAN and send back control using the E2 interface. The E2 interface is an open interface between two endpoints, i.e., the near-RT RIC and the so-called E2 nodes, i.e., DUs, CUs, and O-RAN-compliant LTE eNBs. The E2 interface has been logically structured into two distinct protocols: the E2 Application Protocol (E2AP) and E2 Service Model (E2SM). The E2AP is a fundamental procedural protocol that facilitates coordination between the near-RT RIC and the E2 nodes, dictating how they communicate with one another and providing a basic set of services. E2AP messages can embed different E2SMs, which implement specific functionalities (i.e., the reporting of RAN metrics or the control of RAN parameters).

2.2 Experiments and Simulations for O-RAN

O-RAN development is still in its early stage, thus there is a limited number of solutions for developing and testing on the O-RAN architecture.

Two frameworks were recently developed to overcome several problems related to dataset availability, developing, designing, prototyping, and testing O-RAN-ready solutions. OpenRAN Gym [6], an open-source toolbox to develop AI/ML O-RAN-compliant inference and control algorithms, to deploy them as xApps on the near-RT RIC, and to test them on a large-scale software-defined RAN controlled by the RIC. OpenRAN Gym is platform-independent, and it allows users to perform data collection campaigns to build datasets, develop, design, prototype, and test O-RAN-ready solutions at scale. This framework provides a lightweight implementation of the OSC near-RT RIC which has been adapted to run on the Colosseum system as a set of standalone Docker containers—as well as automated pipelines for the deployment of the various services of the RIC. OpenRAN Gym leverages srsRAN and OpenAirInterface to implement the RAN through software-defined radios. The Open AI Cellular (OAIC) initiative has introduced an O-RAN framework to manage cellular networks—based on srsRAN—through AI-enabled controllers, and to interact with systems that locate implementation, system-level, and security flaws in the network itself [19].

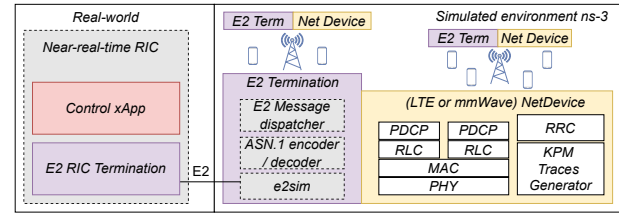


Figure 1: ns-O-RAN Architecture

In this paper, we extend the OpenRAN Gym with an ns-3 integration, to provide an environment that does not require experimental infrastructure and software-defined radios. Network Simulator 3 (ns-3) is a discrete-event network simulator which is highly flexible and customizable: users can configure various simulation parameters, such as network topology, node mobility, and traffic patterns, to match the specifics of their network scenario. Specifically, we leverage the availability of very accurate 3GPP stochastic models [21] and the ns-3 5G module from [13]. This extends the ns-3 LTE module with additional features such as 5G-compliant physical and MAC layers, antenna patterns, beamforming algorithms, and mobility procedures for multi-connected devices.

The ns-3 simulator is also an ideal platform for enhancing AI solutions for networks. In recent years, various studies have extended the standard capabilities of ns-3 by integrating its potential with popular machine learning development software. This has allowed for the creation of more advanced AI solutions for networks. In [10], authors propose ns-3-gym, a framework that integrates OpenAI Gym and ns-3, two popular tools in the fields of reinforcement learning and network simulation, respectively. ns3-gym enables the use of reinforcement learning techniques for network optimization and management problems, by combining the simulation capabilities of ns-3 with the reinforcement learning algorithms of OpenAI Gym. ns-3-ai [20] provides a high-efficiency solution for data interaction between ns-3 and other python-based AI frameworks, following the principles of ns-3-gym. However, neither ns-3-gym nor ns-3-ai can be used as a framework for developing O-RAN xApps that are suitable for immediate use in a production environment, unlike the proposed ns-O-RAN framework in this paper, which we describe in details next.

3 NS-O-RAN

ns-O-RAN extends the OpenRAN Gym framework, embedding the first open-source simulation platform that combines a functional 4G/5G protocol stack in ns-3 and an O-RAN-compliant E2 interface for simulated base stations. Such a platform was designed to enhance data collection and xApp testing capabilities, a critical step toward enabling efficient and generic AI and ML solutions for Open RAN and the next generation of cellular network systems. To this end, ns-O-RAN enables the integration of O-RAN software such as the OpenRAN Gym and OSC near-RT RICs with large-scale 5G simulations using 3GPP channel models and detailed modeling of the full 3GPP RAN protocol stack. This allows data collection of RAN KPMs at scale, in different simulated scenarios, and with different applications (e.g., multimedia streaming, web browsing, wireless virtual reality, etc).

ns-O-RAN supports an O-RAN-compliant E2 interface and implements two different E2SMs, i.e., E2SM KPM, for reporting, and E2SM RAN Control (RC), to push control actions to the RAN (for example, of traffic steering and mobility management). At its core, ns-O-RAN is an ns-3 external module that provides an Stream Control Transmission Protocol (SCTP) connection between the simulator and the near-RT RIC for the support of the E2AP and E2SM. The E2 termination of the RIC can connect to a set of E2 terminations running in the ns-3 simulation, responsible for handling all the E2 messages from and to the simulated environment. This connection was developed by extending the OSC E2 simulator (i.e., `e2sim`) and wrapping it into an ad hoc module for ns-3.

This framework can generate realistic datasets based on stochastic 3GPP-defined wireless channel, that can be fed straight into the O-RAN RIC with no need for infrastructure. Moreover, ns-O-RAN inherits the ease of customization from ns-3, i.e., it can be easily configured to support countless possible scenarios and use cases that can be studied over the same platform. Such scenarios can then be the foundation to build O-RAN-compliant xApps that can be tested and tuned on ns-3 and then deployed on a real RAN with no software changes.

As shown in Figure 1, ns-O-RAN features three different software applications: (i) the `e2sim` [14] software, which was originally developed by the OSC community and extended as part of this effort to process E2SM RC handover management actions and handle multiple E2 terminations; (ii) the `ns3-mmWave` module [13]; and (iii) the ns-O-RAN module, introduced in [11], which is an ns-3 external module that uses the `e2sim` to create a SCTP connection with the RIC and provides E2 terminations to the simulated base stations.

3.1 `e2sim`

The E2 simulator, namely `e2sim` [14], is an OSC software designed to simulate the RAN E2 termination to allow the development of the hosts on near-RT RIC and xApps. It is an SCTP client that implements the E2AP basic specifications, allowing for end-to-end E2 flow testing by creating a connection to the E2 Termination in a near-RT RIC platform. It is capable of decoding incoming messages from the RIC and providing feedback, as well as streaming RAN telemetry to the RIC. The system also behaves as an external interface library, exposing a main class called `E2Sim` that provides basic APIs for managing connection and message reception. By referencing this class in their codebase, external applications can establish a direct connection with the RIC.

More precisely, at startup, the `e2sim` generates one E2 Setup Request for each RAN node, including the RAN function definition and identifiers (IDs) that specify the RAN node capabilities (e.g., control actions that it can ingest, performance metrics it can report). The RIC replies with an E2 Setup Response for each request, to acknowledge the presence of the nodes within the network. `e2sim` uses an event-driven system made of callbacks to forward the messages from the RIC to the external calling applications.

When a new message from the RIC is received, `e2sim` decodes it and identifies the RAN Function ID to trigger an event to all the registered callbacks sharing the incoming message. If no callback is registered, no action is taken. The calling application can subscribe

to new callbacks using the E2SM RAN Function ID in `e2sim` to notify its capability to support a specific feature. The xApp can register to the E2SMs by sending a RIC Subscription Request indicating which RAN function is going to use during its working. It will be then the responsibility of the calling application to generate and send the appropriate responses to the xApp and this can be done by using the `E2Sim SendMessage` function.

Extension of the `e2sim` library – The original `e2sim` project was designed to simulate the E2 RAN Termination of a single Next Generation Node Base (gNB) that is connected to the near-RT RIC. Therefore its software base relies on this design choice and uses global variables that are meant to describe the features of the Base Station (BS), such as the gNB identifier, the SCTP socket file descriptor, and the client port that determine the connection in the Linux system. On the contrary, in ns-3 all the BSs are managed by a single process that simulates the interactions in form of time-based scheduled events. For this reason, we extended the `e2sim` library² to support multiple gNBs simultaneously over the same machine. To do so, we removed the global variables and integrated them as configurable parameters of the `E2Sim` class. Moreover, compared to a standard RAN deployment, where the BSs are expected to have different IP addresses that distinguish the connections to the RIC, ns-3 runs on a single process/host, making it more difficult to expose different IPs. To address this issue and improve the performance of the simulation, we developed a novel approach that uses multiple threads in ns-3 to establish independent SCTP connections for each E2 link. Specifically, we created new parameters to support the inclusion of a local port number and extended ns-3 to create separate threads guaranteeing the independence of each E2 data flow.

By doing so, we can successfully establish connectivity between multiple RAN nodes and the near-RT RIC even if a single IP is associated with the simulation process.

Finally, in the original `e2sim` codebase the only RAN function ID supported was the number 200, which identifies the possibility for the BS to collect KPM reports and send them to the RIC through E2 Indication Messages. In our version, we extended the E2SM capabilities including the RC service model, which allows the xApp to send to the RAN control actions. To support these extensions, we have integrated the callback system in our application to handle the RC messages from the xApp, thus enabling the RAN control.

3.2 The ns-O-RAN Module

The integration between ns-3 and `e2sim` has been implemented through a dedicated module³. It uses `e2sim` as a library and implement its callbacks, besides providing streamlined support for multiple E2 termination instances configuration. These operations are handled by the `E2Termination` class, which wraps all the major `e2sim` functionalities exposing them to ns-3. Figure 2 shows the simplified Unified Modeling Language (UML) diagram, illustrating how the ns-O-RAN classes interact with each other and how they work in connection with the ns-3 `mmWave` module. To enable the exchange of the RAN functions and capabilities through E2AP, we create the O-RAN RAN Function descriptors in the simulation

²<https://github.com/wineslab/ns-o-ran-e2-sim>

³<https://github.com/o-ran-sc/sim-ns3-o-ran-e2>

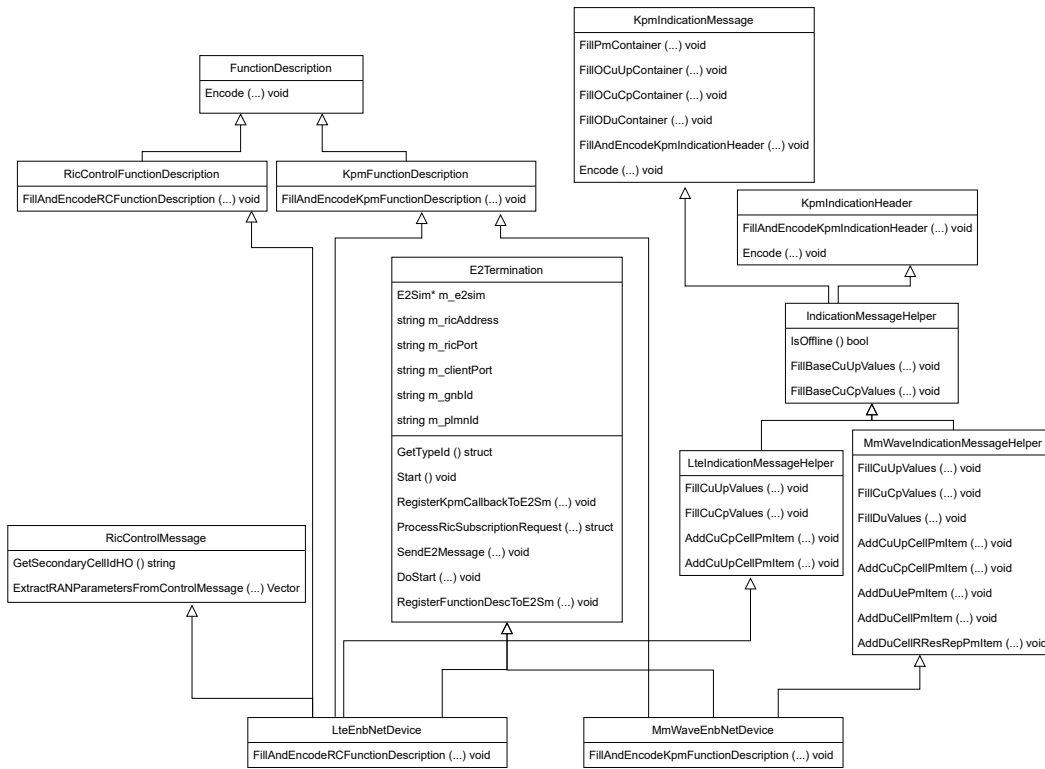


Figure 2: Simplified UML Diagram of ns-O-RAN and its Main Connections with the ns-3 mmWave Module

with the auxiliary class `FunctionDescription` and its extensions `RicControlFunctionDescription` and `KpmFunctionDescription`.

The ns-O-RAN module implements the creation and encoding of periodic reports from the RAN using the RIC Indication Messages according to the O-RAN KPM specification [15]. This is implemented in the `KpmIndicationHeader` and in the `KpmIndicationMessage` classes that are created and encoded using the `IndicationMessageHelper`. Moreover, we developed the RC capability for the O-RAN TS use-case [16] through the Handover Management message, so that the xApp can send a RC Control Message with RAN Function ID equal to 300. In this message, the xApp specifies the identity of an UE and the target cell to which the handover should be performed. The procedure is initiated by a source gNB, which transfers the UE’s context to the target cell. The RIC Control Messages are decoded in the ns-O-RAN module using the class `RicControlMessage`. Finally, the `asn1c`-types files contain wrapper classes for the Abstract Syntax Notation One (ASN.1) C structures that have re-designed to improve their allocation and deallocation and thus the general memory management of the module.

The ns-O-RAN module follows the classical installation steps of an external contribution module of ns-3 and requires the `e2sim` library to be installed to build the whole ns-3 project.

3.3 Integration with the 5G Cellular Model

The modules described in the previous sections are responsible for the setup of the connection between the simulated RAN and the RIC through the E2 interface. Such modules are designed to

be agnostic with respect to the ns-3 module in use and they can be integrated with any cellular network implementation done for ns-3. For this work, we extended the ns-3 mmWave module⁴ for the simulation of 5G cellular networks, providing its simulated base stations the E2 Termination support.

The ns-3 mmWave module [13] is designed to perform end-to-end simulations of 3GPP-style cellular networks. Built upon the ns-3 Long Term Evolution (LTE) module (LENA) [3], this 5G module enables the support of a wide range of channel models for frequencies between 0.5 and 100 GHz, thus including 3GPP NR Frequency Range 1 (FR1) and Frequency Range 2 (FR2). The Physical (PHY) and MAC classes in this system have been specifically designed to support the 3GPP NR frame structure and numerologies, ensuring compatibility with the latest cellular technologies. At the MAC layer, it supports carrier aggregation [22] and multiple scheduling policies, to provide additional capacity. Finally, the module also enables dual connectivity with the LTE base stations, featuring fast secondary cell handover and channel tracking. For a full list of the features implemented in this module, we refer the reader to the original repository of the project⁵.

The goal of this integration is to enable an end-to-end RAN simulation for the O-RAN RICs. This requires the RAN side to enable E2SM exchanges with the RIC, i.e., the delivery of the KPM reports and the digestion of the control actions. The architecture

⁴<https://github.com/wineslab/ns-o-ran-ns3-mmwave>

⁵<https://github.com/nyuwireless-unipd/ns3-mmwave>

of the ns-3 mmWave module along with the structural changes we performed to integrate ns-O-RAN with it are shown in Figure 3.

In the original module, the RAN functions are mainly managed by two central NetDevice classes called `LteEnbNetDevice` and `MmWaveEnbNetDevice`, which are responsible to generate and trigger the functions for the management of the full radio protocol stack and of the RLC, RRC and PDCP data flows in the simulation. The first major design modification was to adapt these classes to support the data collection according to the O-RAN specifications. We created three new functions also shown in the simplified UML of Figure 2, which are responsible for the generation of the KPMs from the simulation and their dispatch through the E2 Termination. Each function manages the data relative to one of the O-RAN specified disaggregated units, namely Centralized Unit - Control Plane (CU-CP), Centralized Unit - User Plane (CU-UP), and DU. Each disaggregated unit has its own statistics calculator that is able to generate and send to the RIC a different Indication Message. At the current moment, we do not support the LTE DU reports. Both the NetDevices are registered to the E2 updates through the callback system described in the previous sections for the KPM RAN Function description.

With respect to the control, the mmWave module implements a Non Stand Alone (NSA) architecture having a Primary LTE cell, also called evolved Node Base (eNB), that is responsible for the PDCP operations of the RAN and set of secondary gNB 5G cells. This led us to implement the processing of the control actions only in the `LteEnbNetDevice` class.

Following the O-RAN specifications, we adapted the code of the base station NetDevices to include an object of type `E2Termination` as the `m_e2term` private attribute to provides E2 connectivity through ns-O-RAN and `e2sim`. In this way, when the ns-3 simulation starts, every `LteEnbNetDevice` and every `MmWaveEnbNetDevice` instantiated creates its own instance of an `E2Termination C++` object to represent different RAN-side E2 terminations that implement the message exchange described in Section 3.1. This operation is done by the `MmWaveHelper` class, which is also responsible for instantiating the objects used to trace and compute the PDCP, RLC and the MAC and physical layer measurement that the E2 terminations will send to the near-RT RIC. This process is transparent to the RIC, which indeed does not need to be aware that the RAN it is serving is a simulated one. Each RAN function is bound to just one

E2 interface, as depicted in Figure 1, and has its own socket pair address.

One of the challenges of integrating simulating systems with software running in the real world is time synchronization among the different hosts. While the real-world near-RT RIC is expected to work with control loops between 10 ms and 1 s, ns-3 is a discrete-event framework that can execute faster or slower than the wall clock time, thus generating a possible time gap between the two systems. To synchronize the two systems, at the beginning of the simulation ns-3 stores the current Unix time in milliseconds and uses it as the baseline timestamp. Whenever an E2 message is sent to the RIC, ns-3 will sum the simulation time elapsed and the baseline timestamp. In this way, the RIC is always able to correctly reorder the messages preventing any inconsistency and invalidation of the received data. The xApps onboarded on the RIC shall use the timestamp provided by these messages to follow the simulation and eventually generate control actions based on it.

We refer to the interactions between the 5G module and the near-RT RIC described in this section as *online mode* of ns-O-RAN, since the absence of an SCTP connection with the RIC would prevent the correct behavior and collection of the simulation data. Nevertheless, the ns-O-RAN and ns-3 mmWave modules can also work in a *stand-alone* or *offline mode*, where at the beginning of the simulation no connection is established and the KPMs collected by the three main functions are stored as traces on files in the local machine. This mode is especially useful for large data collection campaigns (e.g., orchestrated with the Simulation Execution Manager [12]), where the main goal is not to control the simulation flow, but to collect data to later study the RAN behavior.

4 END-TO-END DEPLOYMENT WITH OPENRAN GYM RIC

In this section, we show how to set up ns-O-RAN on a virtualized environment to create a connection between the RIC and ns-3 and to allow the exchange of the E2 messages, thus creating a *simulated* closed control loops between ns-3 and a near-RT RIC. For this tutorial, we use the near-RT RIC⁶ from the OpenRAN Gym framework [17], which can be installed on a local workstation or loaded into experimental platforms such as Colosseum [5]. In this latter case, we already provide a publicly available Colosseum LXC image.

From the RAN side to the RIC, we can collect the implemented KPMs from the simulation, wrap them up into RIC Indication Messages and send them through the E2 interface. The RIC is then able to receive such messages, read the KPMs and Key Performance Indicators (KPIs) to define a data-driven policy (or to simply apply an adaptive policy), create a Control Action and then send the action inside a RIC Control Message to implement the policy. Finally, the RAN receives the Control Message and applies the changes request by the RIC. We define these exchanges as a *simulated control loops* since the control actions sent to the RIC can tune the simulation and change its action in a controlled and reproducible environment.

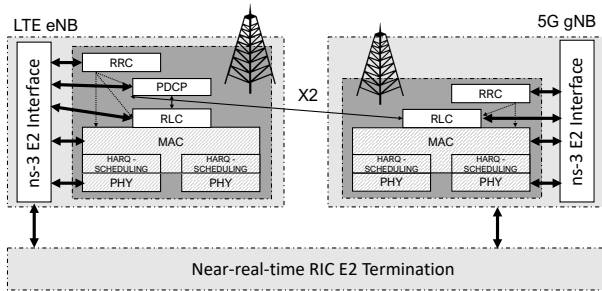


Figure 3: Integration Between the near-RT RIC and the ns-3 mmWave Models Through ns-O-RAN

⁶<https://github.com/wineslab/colosseum-near-rt-ric/tree/ns-o-ran>

4.1 Local Setup of the near-RT RIC

A near-RT RIC (e.g., the OpenRAN Gym RIC) is required to work with ns-O-RAN in its online mode. First, the RIC repository shall be cloned and set up by configuring and executing the main components that are going to interact with the ns-3 environment:

- the near-RT RIC, which includes multiple Docker containers for the E2 termination, an internal message routing manager, a database, and a manager for the E2 connections; and
- the sample xApp that will receive and process the indication messages from ns-O-RAN.

These entities can be instantiated by running the commands provided in Listing 1.

```
1 #!/bin/bash
2 git clone -b ns-o-ran
  https://github.com/wineslab/colosseum-near-rt-ric
3 cd colosseum-near-rt-ric/setup-scripts
4 ./import-wines-images.sh # import and tag base images
  from Docker hub
5 ./setup-ric-bronze.sh # setup and launch
```

Listing 1: Commands to Start the near-RT RIC

Once this is done the RIC components should be up and running in different Docker containers (which can be listed using the `docker ps` command). As a next step, we advise opening a terminal window for logging the values of the E2 Termination and check the E2AP messages exchange, with the command shown in Listings 2. This helps understanding if E2AP and E2SM messages are received correctly.

```
1 #!/bin/bash
2 docker logs e2term -f --since=1s 2>&1 | grep gnb:
```

Listing 2: Commands to Show RIC E2 Termination Logs. The Last `grep` Command Will Filter the Output to Show only when a Base Station is Interacting

4.2 Installation and Connection with ns-O-RAN

Several options are available to install ns-3 and the ns-O-RAN extension, including the use of a *Dockerfile* provided in the root of the `ns-o-ran` branch of the OpenRAN Gym near-RT RIC repository discussed above. The three components required by ns-O-RAN need to be installed in sequence to properly configure the system, as shown in Listing 3.

At first, the enhanced version of the `e2sim` should be cloned, built and installed with all the due prerequisites. In this step, it is possible to set the verbosity of the `e2sim` by changing the relative argument passed to the build script. This software is a strict dependency for the `ns3-mmWave` version adapted for O-RAN, and thus it should be set up before installing the main toolchain. After this step, it is possible to set up the ns-3 mmWave main project and ns-O-RAN module, which is basically an external module that can be plugged in ns-3 and uses the `e2sim` to create a SCTP connection with the RIC. Finally, we can clone the ns-O-RAN module and add it in the `ns3-mmWave` project in the `contrib` directory and we can build ns-3.

```
1 #!/bin/bash
2 cd oran-e2sim/e2sim/
3 mkdir build
4 ./build_e2sim.sh <verbosity> # verbosity = [0,3]
5 cd ../../
6 # clone mmwave project in a folder called
  ns-3-mmwave-oran
7 git clone
  https://github.com/wineslab/ns-o-ran-ns3-mmwave
  ns-3-mmwave-oran
8 cd ns-3-mmwave-oran/contrib
9 # clone ns-O-RAN from the OSC repository in a folder
  called oran-interface
10 git clone -b master
  https://github.com/o-ran-sc/sim-ns3-o-ran-e2
  oran-interface
11 cd .. # go back to the ns-3-mmwave-oran folder
12
13 ./waf configure --enable-examples --enable-tests
14 ./waf build
```

Listing 3: Commands to Setup ns-O-RAN Components and Build the ns-3 Toolchain

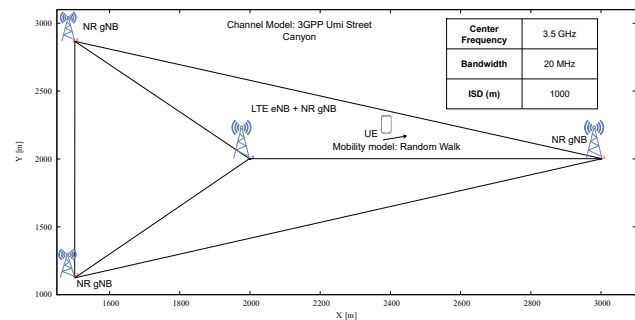


Figure 4: Scenario Zero Implemented to Test the Simulated Control Loop on ns-O-RAN

4.3 Simulating the RAN – Sample Scenario

To instantiate the RAN and test connectivity to the RIC, we provide a sample ns-3 scenario in the file `scenario-zero.cc`. As shown in Figure 4, this scenario features a NSA 5G setup with one LTE eNB in the center of the scenario, one gNB co-located with the LTE base station and three gNBs around it with an inter-site distance of 1000 meters with the center of the scenario.

By default, this scenario is deployed on the FR1 frequency range, using as center frequency 3.5 GHz with a bandwidth of 20 MHz. One antenna is assigned to each UE and BS. This scenario is a simplification of the Dense Urban Scenario described in the 3GPP Technical Release 38.913 [1]. The channel model is the 3GPP UMi-Street Canyon from [2].

The number of UEs in this scenario is 12 and they are allocated in their initial positions uniformly, i.e., with constant density, randomly within a disc having as center the co-located eNB-gNB stations and a radius equal to the inter-site distance. The mobility model of the UEs is a random bi-dimensional walk. Each UE moves with a uniform speed between 2 m/s and 4 m/s. The source traffic model simulates Enhanced Mobile Broadband (eMBB) UEs with a full buffer constant transmission of 20.48 Mbps.

The simulation time of this scenario is 2 seconds and the periodicity of the generation and delivery to the RIC of the E2SM messages is 100 ms.

We can run the scenario by using the commands in Listing 4. ns-O-RAN is designed to be compliant with a generic O-RAN-compliant RIC, thus the simulation scenario only needs the RIC E2 termination IP address, which can be configured using the e2TermIp attribute.

```
1 #!/bin/bash
2 ./waf --run scratch/scenario-zero.cc
   --e2TermIp="10.0.2.10"
```

Listing 4: Commands to Run the Scenario Zero

4.4 xApp Instantiation

After the RAN is connected to the RIC, it is possible to start the xApp and log its container to verify that it receives E2 messages. The procedure to start the sample xApp we provide as a Docker container is shown in Listing 5.

```
1 #!/bin/bash
2 cd colosseum-near-rt-ric/setup-scripts
3 ./start-xapp-ns-o-ran.sh
4 # In Docker
5 cd /home/sample-xapp
6 ./run_xapp.sh
```

Listing 5: Commands to Create the xApp Container and Run the xApp Logic

5 EXAMPLE RESULTS

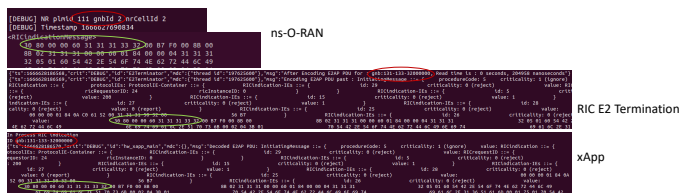


Figure 5: RIC Indication Message Read on ns-O-RAN, the E2 Termination and the xApp

By running the sample scenario, it is possible to observe the message exchange between the simulated RAN and the real-world RIC:

- E2 Setup Request (from ns-O-RAN to E2 Termination on the RIC);
- E2 Setup Response (from the E2 Termination on the RIC to ns-O-RAN);
- E2 Subscription Request (from the xApp to ns-O-RAN through the E2 Termination on RIC, after the xApp is instantiated);
- E2 Subscription Response (from ns-O-RAN to the xApp through the E2 Termination on RIC);
- E2SM RIC Indication Message (from ns-O-RAN to the xApp through the E2 Termination on RIC).

There are several ways to analyze and study these messages. If the verbosity parameter has been set to 3 during the e2sim installation, ns-3 will show the encoded E2AP messages that are generated and sent to the RIC. The same Indication messages are also logged in the E2 Termination of the RIC and in the xApp, which decodes them, extracts the KPMs, and digest them according to xApp program logic. Figure 5 shows the logs of three different components involved in the communication. In each of these logs, it is possible to inspect the Indication Messages, thus confirming the correct behavior of the end-to-end RAN-to-xApp communication. We show in red the gNB ID and in green the initial header of the message. The gNB ID format for ns-O-RAN on the near-RT RIC is gnb:131-133-3<gnbld><padding>

Moreover, it is possible to observe and capture the traffic on the host’s interfaces with common packet dissectors such as *Wireshark* and *tcpdump*. This displays all the interactions between the near-RT RIC and the simulated nodes, which can be identified on Wireshark by their IP address and port pair and by their gNB-ID in the E2AP protocol. An example of this is shown in Figure 6, whose left part shows the Indication Message sent by ns-3 and whose right part reports the same packet captured and dissected in Wireshark. By further exploiting the packet dissector, at the end of the simulation, we can compute the statistics of the overall E2 traffic between the hosts to provide some metrics that can be useful to better understand the bandwidth usage of the E2AP exchanges. Table 1 summarizes the E2AP packets collected during the execution of Scenario Zero and gives some general statistics about its consumption from a network usage point of view. It is important to specify that the time span is the real time duration of the simulation since we are analyzing real exchanged packets with Wireshark but instantiated by ns-3. From the number of bytes exchanged we notice that the traffic, which includes bytes exchanged, average data rate expressed in Bytes per second (Bps) and bit per second (bps), generated by the eNB is small if compared to the gNB. This is mostly because the 5G BS also provides the E2 DU Indication reports.

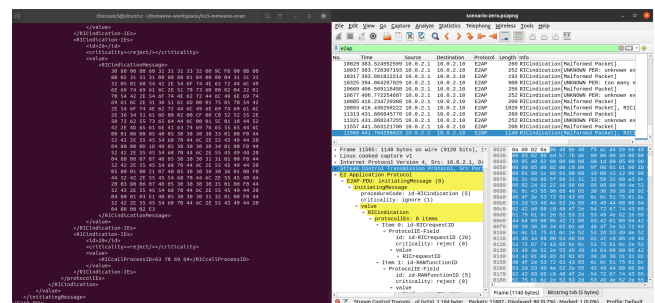


Figure 6: Dissected Packets from the Sample Scenario

Example use case. We have used ns-O-RAN and the E2SM implemented to study the O-RAN TS use case [16] in our work [11], where we leverage ns-O-RAN and the Mavenir near-RT RIC to deploy an AI-driven solution based on Deep Q-Network (DQN) for the control of handover in an NSA 5G simulated RAN with seven NR gNBs and one LTE base station. This is an example of how ns-O-RAN can be used to study the O-RAN architecture and to enable

Table 1: E2AP Traffic Exchange Between ns-O-RAN and the xApp During the Execution of a Simulation with the Sample ns-O-RAN Scenario. No Control Message Was Sent by the RIC During this Experiment

Measurement	All	Single eNB	Single gNB
Wireshark Filter	e2ap	e2ap and sctp.port == 38471	e2ap and sctp.port == 38472
Number of Packets	157	40	40
Time span (s)	439.220	429.324	439.211
Average pps	0.4	0.1	0.1
Average. size (B)	396	259	661
Bytes exchanged	62148	10352	26456
Average Data Rate (Bps)	141	24	60
Average Data Rate (bps)	1.131	192	481

the development of practical and effective AI solutions on open cellular networks. In the same work, we prove how ns-O-RAN can scale in complex scenarios with eight cellular BS and more than one hundred UEs connected and generating data which is subsequently reported to the near-RT RIC.

6 CONCLUSIONS AND FUTURE WORKS

In this paper, we presented ns-O-RAN, a framework for the integration of 5G simulations in ns-3 and near-real-time RICs. It enables the development of xApps on real-world platforms and the testing of their capabilities on a simulated RAN, as well as extensive data collection campaigns to generate datasets for the training of AI/ML components in the xApps themselves. We discussed the main components of ns-O-RAN, and reviewed the instructions on how it can be configured and instantiated. We also introduced sample results on E2 traffic and described different methods to interact with message exchanges in the E2 termination.

As part of our future work, we will continue to improve and extend ns-O-RAN, which is a project part of the O-RAN Alliance OSC group. Currently, ns-O-RAN only ingests handover control messages. We plan to extend the E2SM RC implementation support to more control actions related, for example, to PDCP split control and slicing. In addition, we will upgrade the ns-3 module to the latest release, which supports the new Cmake-based build system, and merge the extension of the 4G and 5G NetDevices in the upstream ns-3 mmWave module. Finally, we will keep updating the ASN.1 definitions of the E2AP and E2SM to the latest O-RAN Alliance specifications, and consider introducing a research-oriented E2SM implemented through JSON schemas or protobufs to simplify the development of new use cases without the need to extend ASN.1 definitions.

ACKNOWLEDGMENTS

This work was partially supported by Mavenir, by Sapienza, University of Rome under Grant "Progetti per Avvio alla Ricerca - Tipo 1", year 2022 (AR1221816B3DC365), and by the U.S. National Science Foundation under Grants CNS-1923789 and CNS-2112471. At the time the research was conducted, Tommaso Zugno was affiliated with Northeastern University and University of Padova.

REFERENCES

- [1] 3GPP. 2015. *Study on Scenarios and Requirements for Next Generation Access Technologies*. Technical Release (TR) 38.913. 3GPP. Version 14.3.0.

- [2] 3GPP. 2020. *Study on Channel Model for Frequencies from 0.5 to 100 GHz*. Technical Specification (TS) 38.901. 3GPP. Version 16.1.0.
- [3] Nicola Baldo, Marco Miozzo, Manuel Requena-Esteso, and Jaume Nin-Guerrero. 2011. An Open Source Product-oriented LTE Network Simulator Based on ns-3. In *Proceedings of the 14th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. ACM, Miami, Florida, USA, 293–298.
- [4] Leonardo Bonati, Salvatore D’Oro, Michele Polese, Stefano Basagni, and Tommaso Melodia. 2021. Intelligence and Learning in O-RAN for Data-driven NextG Cellular Networks. *IEEE Communications Magazine* 59, 10 (October 2021), 21–27.
- [5] Leonardo Bonati, Pedram Johari, Michele Polese, Salvatore D’Oro, Subhramoy Mohanti, Micael Tehrani-Moayyed, Davide Villa, Shweta Shrivastava, Chinenye Tassie, Kurt Yoder, Ajeet Bagga, Paresch Patel, Ventz Petkov, Michael Seltser, Francesco Restuccia, Abhimanyu Gosain, Kaushik R. Chowdhury, Stefano Basagni, and Tommaso Melodia. 2021. Colosseum: Large-Scale Wireless Experimentation Through Hardware-in-the-Loop Network Emulation. In *Proceedings of IEEE DySPAN*. IEEE, Virtual Conference, 105–113.
- [6] Leonardo Bonati, Michele Polese, Salvatore D’Oro, Stefano Basagni, and Tommaso Melodia. 2022. OpenRAN 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*. IEEE, Austin, TX, USA, 518–523.
- [7] Massimo Condoluci and Toktam Mahmoodi. 2018. Softwarization and Virtualization in 5G Mobile Networks: Benefits, Trends and Challenges. *Computer Networks* 146 (December 2018), 65–84.
- [8] Marcin Dryjański, Lukasz Kulacz, and Adrian Kliks. 2021. Toward Modular and Flexible Open RAN Implementations in 6G Networks: Traffic Steering Use Case and O-RAN xApps. *Sensors* 21, 24 (2021).
- [9] Andres Garcia-Saavedra and Xavier Costa-Perez. 2021. O-RAN: Disrupting the Virtualized RAN Ecosystem. *IEEE Communications Standards Magazine* 5, 4 (2021), 96–103.
- [10] Piotr Gawłowicz and Anatolij Zubow. 2019. ns-3 Meets OpenAI Gym: The Playground for Machine Learning in Networking Research. In *Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. ACM, Miami Beach, FL, USA, 113–120.
- [11] Andrea Lacava, Michele Polese, Rajarajan Sivaraj, Rahul Soundrarajan, Bhawani Shanker Bhati, Tarunjeet Singh, Tommaso Zugno, Francesca Cuomo, and Tommaso Melodia. 2023. Programmable and Customized Intelligence for Traffic Steering in 5G Networks Using Open RAN Architectures. *IEEE Transactions on Mobile Computing* (2023), 1–16. <https://doi.org/10.1109/TMC.2023.3266642>
- [12] Davide Magrin, Dizhi Zhou, and Michele Zorzi. 2019. 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* (Miami Beach, FL, USA) (MSWIM ’19). Association for Computing Machinery, New York, NY, USA, 121–125. <https://doi.org/10.1145/3345768.3355942>
- [13] Marco Mezzavilla, Menglei Zhang, Michele Polese, Russel Ford, Sourjya Dutta, Sundeep Rangan, and Michele Zorzi. 2018. End-to-End Simulation of 5G mmWave Networks. *IEEE Communications Surveys Tutorials* 20, 3 (April 2018), 2237–2263.
- [14] O-RAN Software Community. 2022. sim-e2-interface repository. <https://github.com/o-ran-sc/sim-e2-interface>.
- [15] O-RAN Working Group 1. 2021. O-RAN Use Cases Analysis Report 6.0. O-RAN.WG1.Use-Cases-Analysis-Report-v06.00 Technical Specification.
- [16] O-RAN Working Group 3. 2021. O-RAN Use Cases and Requirements 1.00. O-RAN.WG3.UCR-v01.00.
- [17] Michele Polese, Leonardo Bonati, Salvatore D’Oro, Stefano Basagni, and Tommaso Melodia. 2022. CoO-RAN: Developing Machine Learning-based xApps for Open RAN Closed-loop Control on Programmable Experimental Platforms. *IEEE*

- Transactions on Mobile Computing* (July 2022), 1–14. <https://doi.org/10.1109/TMC.2022.3188013>
- [18] Michele Polese, Leonardo Bonati, Salvatore D’Oro, Stefano Basagni, and Tommaso Melodia. 2023. Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges. *IEEE Communications Surveys & Tutorials* (2023), 1–1. <https://doi.org/10.1109/COMST.2023.3239220>
- [19] Pratheek S Upadhyaya, Aly S Abdalla, Vuk Marojevic, Jeffrey H Reed, and Vijay K Shah. 2022. Prototyping Next-Generation O-RAN Research Testbeds with SDRs. *arXiv preprint arXiv:2205.13178* (2022).
- [20] Hao Yin, Pengyu Liu, Keshu Liu, Liu Cao, Lytianyang Zhang, Yayu Gao, and Xiaojun Hei. 2020. ns3-ai: Fostering Artificial Intelligence Algorithms for Networking Research. In *Proceedings of the 2020 Workshop on ns-3* (Gaithersburg, MD, USA) (WNS3 2020). Association for Computing Machinery, New York, NY, USA, 57–64. <https://doi.org/10.1145/3389400.3389404>
- [21] Tommaso Zugno, Michele Polese, Natale Patriciello, Biljana Bojović, Sandra Lagen, and Michele Zorzi. 2020. Implementation of a Spatial Channel Model for ns-3. In *Proceedings of the 2020 Workshop on ns-3* (Gaithersburg, MD, USA) (WNS3 2020). Association for Computing Machinery, New York, NY, USA, 49–56.
- [22] Tommaso Zugno, Michele Polese, and Michele Zorzi. 2018. Integration of Carrier Aggregation and Dual Connectivity for the ns-3 mmWave Module. In *Proceedings of the 2018 Workshop on ns-3*. Association for Computing Machinery, New York, NY, USA, 45–52. <https://doi.org/10.1145/3199902.3199909>