O-RAN with Machine Learning in ns-3

Wesley Garey
National Institute of Standards and Technology
Gaithersburg, Maryland, USA
wesley.garey@nist.gov

Tanguy Ropitault
Associate, National Institute of Standards and Technology
Prometheus Computing LLC
Bethesda, Maryland, USA
tanguy.ropitault@nist.gov

Richard Rouil
National Institute of Standards and Technology
Gaithersburg, Maryland, USA
richard.rouil@nist.gov

Evan Black
National Institute of Standards and Technology
Gaithersburg, Maryland, USA
evan.black@nist.gov

Weichao Gao
Associate, National Institute of Standards and Technology
Dakota Consulting, Inc.
Silver Spring, Maryland, USA
weichao.gao@nist.gov

ABSTRACT

The Open Radio Access Network (O-RAN) Alliance is an industry-led standardization effort, with the main objective of evolving the Radio Access Network (RAN) to be open, intelligent, interoperable, and autonomous to support the ever growing need of improved performance and flexibility in mobile networks. This paper introduces an extension to Network Simulator 3 (ns-3) which mimics the behavior and components of the O-RAN Alliance's O-RAN architecture. In this paper, we will describe the O-RAN architecture, our model in ns-3, and a Long Term Evolution (LTE) case study that utilizes Machine Learning (ML) and its integration with ns-3. At the end of this paper, the reader will have a general understanding of O-RAN and the capabilities of our fully simulated contribution so it can be leveraged to design and evaluate O-RAN-based solutions.

CCS CONCEPTS

• **Networks** → **Mobile networks**; • **Computing methodologies** → **Modeling and simulation**; *Machine learning*.

KEYWORDS

O-RAN, ns-3, LTE, ONNX, Mobile Networks, Modeling and Simulation, Machine Learning

ACM Reference Format:
Wesley Garey, Tanguy Ropitault, Richard Rouil, Evan Black, and Weichao Gao. 2023. O-RAN with Machine Learning in ns-3. In *2023 Workshop on ns-3 (WNS3 2023), June 28–29, 2023, Arlington, VA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3592149.3592157

Disclaimer: Certain equipment, instruments, software, or materials, commercial or non-commercial, are identified in this paper in order to specify the experimental procedure adequately. Such identification is not intended to imply recommendation or endorsement of any product or service by the National Institute of Standards and Technology (NIST), nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.

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

ACM ISBN 979-8-4007-0747-6/23/06…$15.00
https://doi.org/10.1145/3592149.3592157

1 INTRODUCTION

The Third Generation Partnership Project (3GPP) is the standards body responsible for creating and maintaining specifications that define how a User Equipment (UE) communicates with a Base Station (BS) over the RAN, so a UE can make use of the various audio and data services provided by Long Term Evolution (LTE) networks and beyond. However, as the traffic of mobiles within a cellular network increases, the need for a more intelligent, agile, and virtualized Radio Access Network (RAN) is required to ensure that the network can adapt to meet the demand [16]. In 2018, five mobile network operators formed the O-RAN Alliance to help meet these requirements [15]. As of February 2023, this global alliance consists of over 30 mobile network operators and 320 companies, all sharing the same goal of enhancing the RAN.

Building upon the effort of 3GPP, the O-RAN Alliance is determined to extend the RAN so it has an open interface, smart RAN controllers, and greater flexibility. To further elaborate, as networks become more complex, the use of Machine Learning (ML) can be leveraged to correlate network parameters with network state to increase network performance [17]. Artificial Intelligence (AI) may be used in combination with real-time data and statistics for autonomous control over network decisions so human intervention is not required to ensure that the network can better adapt to changing environments [17].

Due to the work of the O-RAN Alliance, several standards exist that define potential use cases and the architecture of O-RAN. Now the O-RAN Alliance is focused on accelerating the deployment and integration of O-RAN with commercial networks [18]. This provides an opportunity for both affiliated and non-affiliated O-RAN Alliance members to create, implement, and test O-RAN solutions. To support this effort, we would like to present a fully simulated research tool, ns3-oran[1], that can be used to design and evaluate O-RAN-based solutions utilizing Network Simulator 3 (ns-3) [34].

[1]https://github.com/usnistgov/ns3-oran.git

The outline of this paper is as follows: In Section 2, we provide an overview of the O-RAN architecture for background information on the technology. We present a literature survey in Section 3 to provide context and discuss existing contributions relevant to this work. In Section 4, we describe our model and its integration with ML. In Section 5, we describe how the ML integration offered by our model can be used to better control the RAN and improve network performance. Finally, in Section 6, we summarize the key elements of this paper and discuss future work.

2 O-RAN OVERVIEW

This section gives an overview of the O-RAN architecture. First, a general description of the architecture will be given to provide a high-level view of the main O-RAN components. Then, a more detailed description of the Near-Real-Time RAN Intelligent Controller (RIC) (Near-RT RIC) and E2 interface is given to coincide with the implementation of our model in ns-3. These more detailed descriptions will cover the components and functions that make up the Near-RT RIC, as well as the use of the E2 interface that facilitates the exchange of information and control indications.

2.1 Architecture

At its highest level, the O-RAN architecture consists of three main bodies, depicted in Figure 1: Service Management and Orchestration (SMO) Framework, Network Functions, and O-RAN Cloud (O-Cloud). The SMO Framework houses the Non-Real-Time RIC (Non-RT RIC) that is responsible for top-level control (or orchestration) of the network, primarily through policy management. The O-Cloud is a pool of physical servers that provide resources to the SMO framework, through the O2 interface, which includes the ability to host virtualized components, such as the Centralized Unit Control Plane (CU-CP). The Network Functions body is responsible for control at the micro level when compared to the macro level of the SMO Framework, as it consists of a Near-RT RIC, which is controlled by the Non-RT RIC through the A1 interface, and the many E2 nodes that the Near-RT RIC is managing. An E2 node is any RAN component, such as a Fourth Generation (4G) LTE Evolved Node B (eNodeB) or Fifth Generation (5G) New Radio (NR) Next Generation Node B (gNB), that possess an E2 interface between itself and the Near-RT RIC. Moreover, E2 nodes are connected to the Near-RT RIC, can perform O-RAN related actions requested by the Near-RT RIC, and/or provide information to the Near-RT RIC about network status and performance. Control loops are also associated with the components in each body. These control loops are constraints that restrict how long it should take for a component to collect information and then use it when it is time to perform a useful action. Further details related to the architecture can be found in [17, 19–25].

2.2 Near-RT RIC

To fulfil its role, there are many components that make up the Near-RT RIC, and a diagram of those components can be found in Figure 2. First and foremost, the Near-RT RIC may house one or many xApps. An xApp is software running on the Near-RT RIC that utilizes many of the other functions included in the Near-RT RIC so that it can process data and determine what changes should

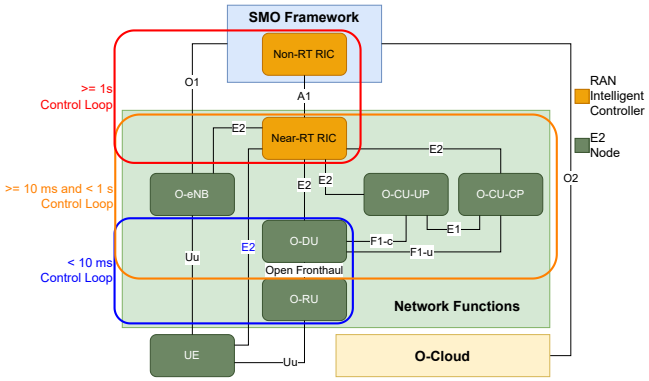


Figure 1: High-Level View of O-RAN Architecture

be made in the RAN, if any. Typically, an xApp is the entity that digests and realizes the policies pushed to the Near-RT RIC by the Non-RT RIC, and is responsible for analyzing data stored at the Near-RT RIC to make useful decisions. For example, there may be an xApp designated to use sensing data to enforce spectrum sharing, while another xApp analyzes UE location and traffic data to manage cell selection. The Database (DB) is where all of the data is stored that may need to be accessed by the Near-RT RIC and its components, such as the xApps. The Conflict Mitigation (CM) module is a filter that processes all of the actions that xApps plan to carry out in the RAN to handle direct, indirect, and implicit conflicts. This is necessary since it is possible for differences in xApp goals or settings to result in multiple actions that affect or depend on the same configuration or outcome, and it is the responsibility of the CM to both discover these conflicts and form resolutions. The ML module provides ML support so that xApps can train models and use them to make inferences. Additional details regarding the Near-RT RIC and its components can be found in [26, 29].

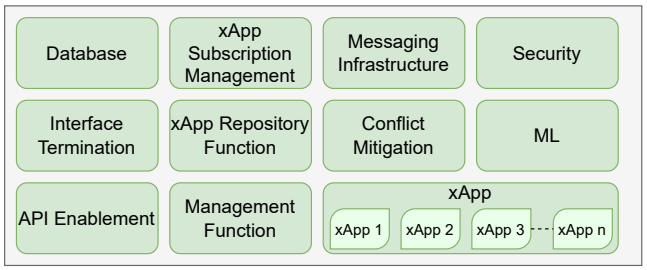


Figure 2: Near-RT RIC Components

2.3 E2 Interface

The E2 interface is the logical link between E2 nodes and the Near-RT RIC. This interface provides the transport for two protocols: the E2 Application Protocol (E2AP) and E2 Service Model (E2SM). The E2AP defines the control procedures and signals that allow E2 nodes to establish a link with the Near-RT RIC to support the E2SM. This includes setup request signals sent from an E2 node to the Near-RT RIC to identify an E2 node and the functions

that it supports, and setup response signals that are sent from the Near-RT RIC to the E2 node to mainly indicate whether the setup was successful or not. Once a successful link is created, the E2SM can be used by the E2 node to report network-related information to the Near-RT RIC via report and insert messages, while the Near-RT RIC can send control and policy messages to the E2 node to request an action or convey an overall policy that should be followed. Further details regarding the E2 interface can be found in [26–28].

3 LITERATURE SURVEY

This section provides a literature survey of existing works relevant to this paper. This includes publications on existing emulated/simulated O-RAN architectures and ML integration with ns-3. We also focus on the means of each result/capability so that we may consider them later when we describe what is necessary to use our contribution.

3.1 Existing O-RAN Tools

Since O-RAN is a relatively new topic with the potential to create significant advances in mobile networks, a recent surge in literature exists in its area. This includes various works that discuss the performance gains of O-RAN-based solutions [7, 32, 33], as well as frameworks and testbeds specifically designed to facilitate the implementation and evaluation of O-RAN components and features [2, 4, 5, 11, 31].

In particular, the authors in [32] provide a thorough overview of the O-RAN architecture and its potential applications. These include, but are not limited to, how O-RAN-based solutions can provide performance gains via autonomous load balancing, the optimization of radio resource allocations, Quality of Service (QoS)-based control targets, sensing solutions, and the optimization of O-RAN deployments themselves. The authors in [7] propose a cell selection algorithm that uses global network state instead of the heuristic, LTE-based method that relies on UE reported signal quality, on top of a combination of tools that include the 5G-EmPOWER Near-RT RIC, srsRan Software Defined Radio (SDR) suite, and Open5GS LTE core to implement and evaluate their algorithm. The authors of [33] use a Kubernetes cluster, several laptops, SDRs, GNU Radio, a JavaScript application, and Python implemented xApps to demonstrate how an O-RAN-based solution can provide a decent sensing-based algorithm.

The authors in [4] present a tool called, Channel-Aware Reactive Mechanism (ChARM), that can be used to effectively share spectrum among unlicensed LTE and Wireless Fidelity (Wi-Fi) bands using srsRAN, the Colosseum wireless testbed, and System-on-Chip (SoC) boards to show how effective ChARM is at exploiting the available channels. The strategy described in [31] also makes use of the Colosseum testbed, but instead provides a framework called, ColO-RAN, that is combined with srsRAN, Linux Containers (LXCs), and custom components to provide ML integration, programmable BSs, and the generation of large-scale data sets. In [5], the authors present an O-RAN-centric toolbox that combines the Near-RT RIC provided by ColO-RAN with the SCOPE framework from [2], which is a ready-to-deploy LXC that provides software-based, 3GPP compliant BSs, data collection tools, and a set of sample scenarios, for an "end-to-end" O-RAN design, specifically for the development and testing of ML solutions. Finally, the work from [11] requires the use of Docker containers and/or a true implementation of a Near-RT RIC, the E2 interface, and remaining O-RAN components, in combination with a specialized version of ns-3, so that the RAN of an LTE network can be simulated. This tool, which is a part of the OpenRAN GYM[2] toolbox, allowed the authors in [11] to demonstrate how ML may be leveraged to perform Traffic Steering (TS) with throughput and spectral efficiency gains of over 50 % when compared to the traditional handover triggers.

3.2 ML Integration in ns-3

There are also two more tools that are relevant to this work, ns3-gym [9] and ns3-ai [35], as they provide a means to integrate ML with ns-3. More specifically, ns3-gym is an extension of ns-3 that uses Protocol buffers (Protobuf) to encode the information (e.g., environment state, rewards, penalties, etc.) exchanged between ns-3 and a Reinforcement Learning (RL) model in Python. Communication between these two processes is carried out using Transmission Control Protocol (TCP) sockets. This allows for remote executions of the ML model and ns-3 at the expense of significant delays and the overhead of having to keep two processes running and synchronized. ns3-ai is an inspiration of ns3-gym that supports Deep Learning (DL) in addition to RL, that also provides a faster message exchange between the ns-3 simulation and a Python model through the use of shared memory when compared to Protobuf. However, ns3-ai still requires muxes/semaphores to regulate access to shared memory, which means that at any point, only one process is running. This limits flexibility as one needs to run both processes in the same system and keep them synchronized.

4 MODEL

Before we describe the details of our model, it is important to note our contribution's advantages. All of the solutions mentioned in Section 3 are useful, however they rely on multi-component setups, intimate knowledge of several tools, and/or costly pieces of hardware. For example, the components necessary to use the solution from [11] are depicted in Figure 3, while the components required for our contribution are depicted in Figure 4. From these figures we can see that the work presented in [11] requires several components, including a specialized version of ns-3 and real/emulated hardware. However, our model only requires two components that are typical of any ns-3 extension: the base version of ns-3 and the module that we provide. Therefore, our solution has the advantage of providing all of the necessary O-RAN components, through software, in one module, to design and evaluate O-RAN-based solutions using simulation. Since this solution does not rely on real or emulated components, but simply models them via the ns-3 paradigm, this approach is for the researcher that is still in the exploratory or design phase with no need to overcommit to more expensive and complex solutions that may be more attractive to vendors and operators who intend to implement and later deploy their solutions to a production environment.

Furthermore, this contribution employs the direct linkage of ML libraries with ns-3 through the use of Open Neural Network

[2] https://openrangym.com/

Exchange (ONNX) that reduces the overhead of exchanging information between the ML model and ns-3. Thus, the flexibility of this approach does not require the use of additional hardware or shared memory, so simulations can easily be run in parallel on the same machine with fewer resources. This solution also naturally invites the simulation as a part of ML model development, as a researcher can quickly iterate between training and evaluating a model. More specifically, researchers can generate data using ns-3 or an external source, train the model outside of ns-3, make inferences directly within ns-3, evaluate performance using simulation output, and repeat. In summary, our contribution is the first and only fully simulated, O-RAN-based extension for ns-3 that supports ML.

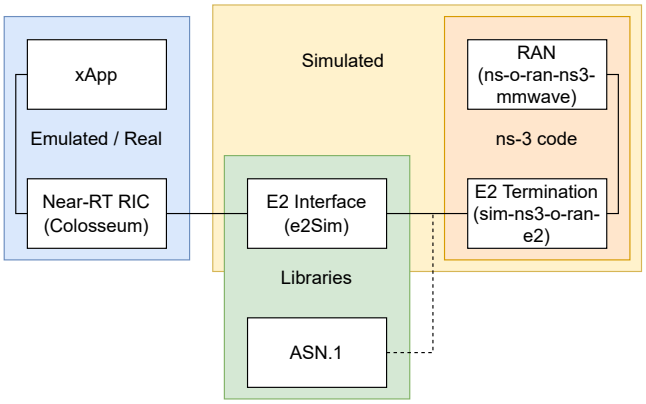


Figure 3: Components Necessary to Use the Solution Described in [11]

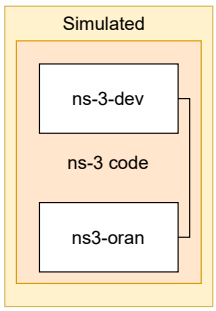


Figure 4: Components Necessary to Use the Contribution Described in this Paper

4.1 O-RAN ns-3 Module

The O-RAN module for ns-3 implements the classes required to model a network architecture based on the O-RAN Alliance specifications. This includes a RIC that is functionally equivalent to O-RAN's Near-RT RIC and reporting modules that attach to network nodes and serve as communication endpoints with the RIC similar to an E2 node in O-RAN. In its current state, the SMO Framework and O-Cloud are not considered as policy-level design and RAN virtualization are out of the scope of our model.

These models are designed to provide the infrastructure and access to data so that developers and researchers can focus on implementing their solutions and minimize the time and effort spent on handling interactions between models. With this in mind, all the components that contain logic that end users may modify are modeled hierarchically. The ns3-oran model uses a Data Repository (DR) to store all the information exchanged between the RIC and the modules, as well as to serve as a logging endpoint. By default, this requires the use of an SQLite [10] storage backend for the DB which is supported out-of-the-box by ns-3, but still requires SQLite to be properly installed on the system for use. The DB file is accessible after the simulation and can be accessed by any SQLite-compatible tool and interface. Modeling of the reporting and communication models for the network nodes is implemented using existing traces and methods, which means there is no need to modify the models provided by the ns-3 distribution to make use of the full capabilities of this module.

4.1.1 Architecture. The Near-RT RIC is modeled as a container that houses all of the individual components that provide the different functionalities of the RIC, as seen in Figure 5. This includes a DR for data storage, Logic Modules (LMs) to model xApps, a CM module, report triggers, and an E2 terminator. In this context, an E2 terminator is simply an endpoint (either on the Near-RT RIC or E2 node) of the E2 interface to facilitate communication. Moreover, an E2 node is any network node with an E2 terminator that connects it with the Near-RT RIC. The Near-RT RIC allows each of its components to be instantiated independently, but takes care of providing references between internal components, the collection of reports, and the dissemination of commands. Additionally, the Near-RT RIC can activate or deactivate all of its associated components when itself is activated or deactivated.

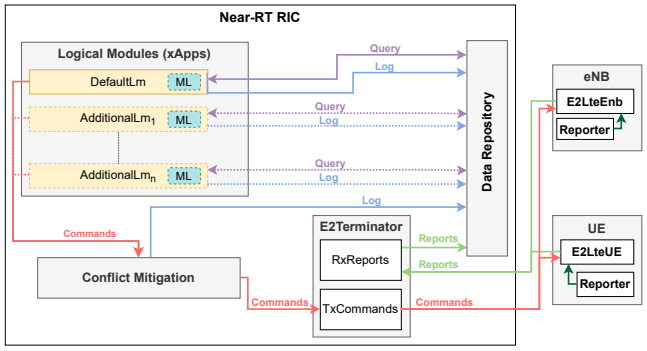


Figure 5: Block Diagram Showing the Communication Between Nodes and the Near-RT RIC

The DR model serves as the access point to the actual storage instance, which can be a database, a text file, or any other storage structure. This abstraction also serves as a single-point definition of the data access Application Programming Interface (API) that will be implemented by a storage instance. This ensures that all the other components will work the same from one simulation to another, regardless of the storage mechanism used. The operations defined by the DR API can be grouped into three categories: report storage, network status, and logging. Report storage is for storing registration requests and reports sent by network nodes through

their E2 terminator. The network status queries allow for retrieving information from node reports. Finally, the logging functions store log-related information generated by the CM module and LMs.

The LMs are the components that implement the intelligence of the RIC. It is expected that each LM will implement unique logic for issuing commands to nodes, depending on the network status. LMs use the DR to retrieve the information reported by network nodes and generate sets of commands that will help achieve the goal defined in their logic. The LMs do not decide when their code is executed, but instead the Near-RT RIC will query all of the deployed LMs for the sets of commands they want to issue. The Near-RT RIC may perform this invocation at fixed intervals or it may be triggered by a report received from a network node. The Near-RT RIC must always reference at least one LM, which serves as the default LM for the RIC. Additional LMs can be deployed as needed. The CM module processes all the commands generated by the deployed LMs to minimize potential conflicts between them. The final component of the Near-RT RIC is the E2 terminator. In the RIC model, the E2 terminator is the entity that interacts with the network nodes, receiving reports from the E2 terminators of network nodes and sending commands filtered by the CM module. Both the received reports and sent commands are logged in the DR.

The modules that enable a network node to interact with the Near-RT RIC are an E2 terminator and one or more reporters. This is depicted in Figure 5. The reporters are modules that attach to existing traces in the node, build reports with the values being traced, and pass them to the E2 terminator for transmission to the RIC. Each reporter has an associated trigger that tells the reporter when to collect the information and generate the report. These triggers may be periodic or based on events. Each report includes the identity of the node, and the time at which it is generated, so even if the transmission to the RIC is delayed, the RIC will know when the values in the report were captured. When a reporter is instantiated, it is linked to an E2 terminator in the network node which allows the reporter to obtain the node identity from the E2 terminator and the E2 terminator to store the reports generated by the reporter.

The E2 terminator in the network nodes is analogous to the E2 terminator in the RIC, in that it is the entity that communicates with the Near-RT RIC by exchanging reports and commands. All of the E2 terminators periodically send the reports generated by their reporters to the Near-RT RIC's E2 terminator. Additionally, the network nodes' E2 terminators are in charge of receiving commands from the RIC and translating them into the appropriate function calls. For example, an LTE handover command can be translated into a function call to start an X2 handover in the associated eNodeB of ns-3. Due to the disparity of the commands that may be accepted by each type of node, there are multiple E2 terminator subclasses, each capable of processing its own set of commands.

The communication between the network node E2 terminators and the Near-RT RIC E2 terminator is modeled using virtual interfaces. This means that there is no network link between these entities, instead, during the configuration, the E2 terminators in the network nodes are given a pointer to the Near-RT RIC. With this pointer, the network node terminators can contact the Near-RT RIC terminator directly when they are activated, allowing the Near-RT RIC to keep track of all the E2 terminators. Transmission delays for both reports and commands are simulated using Random Variables (RVs): one RV in the E2 terminator of the Near-RT RIC generates the delays incurred by commands sent to network nodes, and another RV in each network node's E2 terminator generates the delays incurred by reports sent to the Near-RT RIC.

4.1.2 O-RAN Model Operation Workflow. This section documents the sequence of events generated by the most common operations of the O-RAN models during a simulation: activation of the models, reporting from the nodes to the Near-RT RIC, LM logic invocation, and command issuing from the Near-RT RIC to the nodes. As shown in Figure 6, when the Near-RT RIC is activated using its public API, the activation signal is propagated to the rest of its components: DR, LMs, CM module, and E2 terminator. On the nodes' side, the activation of the E2 terminator will trigger the activation of all the reporters and report triggers attached to it, and it also triggers a registration request to the Near-RT RIC. If the Near-RT RIC has already been activated, the registration request will be recorded in the DR. If the Near-RT RIC is not active, the registration request will be ignored, and periodic retransmissions of the request will occur.

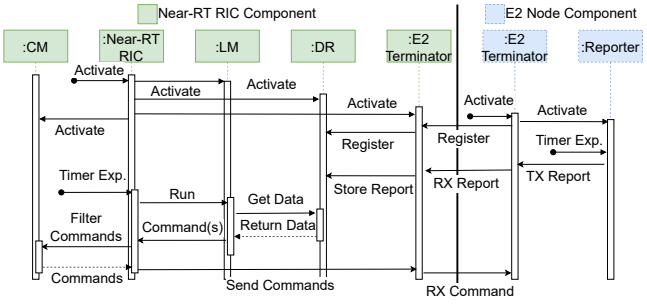


Figure 6: Sequence Diagram Showing a Typical Flow of Events and Signal Exchanges Between O-RAN Objects

Once the Near-RT RIC and network nodes' E2 terminators are activated and a registration request is successfully recorded in the DR, the report triggers in the node will signal when the reporters should generate reports. The node's E2 terminator, equipped with a timer, will periodically send the collected reports to the Near-RT RIC. When the Near-RT RIC's E2 terminator receives a report, it will be stored in the DR, as shown in Figure 6, and the Near-RT RIC will be notified so it can be examined in case a trigger should be invoked.

Figure 6 also shows how LMs in the Near-RT RIC can be triggered by a periodic timer or by an event, such as the reception of a specific report. In either case, the Near-RT RIC will request all deployed LMs to run their logic and generate commands for the network nodes. For each LM to process this request, it will retrieve information from the DR that it is interested in and then use this information to decide, based on its custom logic and goals, if any change needs to be made to the network, and if so, the appropriate commands are issued. These commands will be passed to the Near-RT RIC once a timer in the LM expires to simulate processing delays.

After all of the LMs run their logic and generate commands, or a maximum processing timer expires in the Near-RT RIC, all collected

commands are sent to the CM module. Here, the CM module filters the commands according to its own logic and returns the final set of commands that are to be sent out to the network nodes. This set is passed to the Near-RT RIC's E2 terminator which will log the individual commands in the DR, and then eventually pass each command to the appropriate network node's E2 terminator.

4.2 ML Framework

In order to implement ML, ns3-oran utilizes the ONNX standard [1]. ONNX is an open-source standard for representing DL models, designed to facilitate interoperability between various DL frameworks and tools. This standard defines both a file format and a set of operators for constructing a computational graph that represents the neural network. ONNX is natively supported by a growing number of ML frameworks, such as PyTorch [30] and MXNet [6]. Additionally, it can also be utilized through converters for TensorFlow [3] and CoreML [14]. The main benefits of using ONNX include the following:

- Framework Interoperability: ONNX brings the ability to easily import a model trained in one ML framework and perform inferences using any other ML framework that supports the ONNX format, without the need for retraining the model from scratch. This not only saves a significant amount of time but also simplifies model deployment across a wide range of platforms. For example, one can train a model in Matlab [12] and deploy it in ns-3.
- Better Performance: Multiple vendors implement hardware optimizations specifically for ONNX to increase inference speed. By utilizing ONNX as a common interface, access to these optimizations is facilitated for any ML framework that is compatible with ONNX. Additionally, the collective effort of multiple vendors to improve the performance of a single framework, ONNX, results in greater optimization when compared to targeting multiple ML frameworks independently.

The use of ONNX in ns3-oran is enabled by ONNX Runtime [13]. ONNX Runtime supports a wide range of platforms and operating systems. It is capable of importing ONNX-trained models and performing ONNX inferences using C++. It is integrated with our model by linking it through the ns-3 build process. This allows the simulation to access ONNX models and query them directly via function calls, as if they were any other C++ object. With this, the simulation can perform inferences, generate dynamic training information, and/or use reward structures for RL. For added flexibility, since the model is loosely coupled with the library, as it is only used at runtime, one can produce a trained model that is stored in a file on either the same system or a different system that is set up to run ns-3. Lastly, as ns-3 is C++ based, the C++ API used to query the ONNX model provides improved performance as the model can be accessed directly and there is no need to implement semaphores or muxes.

5 CASE STUDY

In this section we will walk through a case study to demonstrate an application of ns3-oran, as we consider several network configurations in the scenario depicted in Figure 7. This will include separate simulations for both non-O-RAN and O-RAN-based approaches. First, we will discuss the overall details of the scenario and the different approaches we consider. Then we will discuss the simulation results collected from each of the simulations and compare network performance.

5.1 Description

The topology consists of two static, single-cell eNodeBs and four mobile UEs: UE 1, UE 2, UE 3, and UE 4. Each circle in Figure 7 indicates the coverage area for a given eNodeB, which is the area in which it can serve a single UE without losing a single packet. The movements of UE 1 and UE 4 are restricted to the coverage of a single eNodeB cell (eNodeB 1 for UE 1 and eNodeB 2 for UE 4), while UE 2 and UE 3 move into an area covered by both cells. The areas in which the UEs can move are depicted by each rectangle in Figure 7. Each UE's data traffic is configured as a video stream download with a rate of approximately 290 kbits/s at the application layer, and the streams are active for a total of 200 s between the simulation times of 2 s and 202 s. We then compare the use of three different approaches to see what gains are achievable, if any, from the baseline scenario. This includes a "heuristic" approach that relies on UE signal quality, a "distance" approach that relies on UE reported location information, and an "ML" approach that relies on UE reported location and application loss.

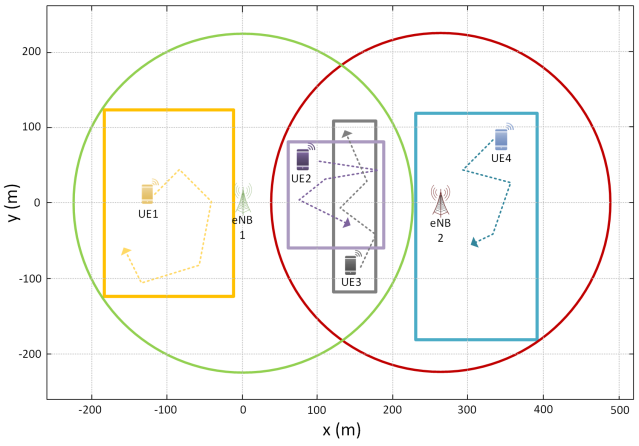


Figure 7: Topology of Case Study

The baseline scenario considers the setup of "Configuration 1" that Figure 8 depicts, with UE 1 and UE 2 connected to eNodeB 1, while UE 3 and UE 4 are connected to eNodeB 2. In the baseline scenario, the UEs are mobile but no handover mechanism is considered. Therefore, this configuration persists for the entire 210 s simulation, regardless of the location of a UE or its signal quality. In this case, the Near-RT RIC is equipped with a "no-op" LM that does nothing, however, the UEs still report their locations and packet loss.

The purpose of using ML is to use a DL model to efficiently trigger UE 2 and UE 3 handovers to maximize performance in terms of packet loss. The DL model used is shown in Figure 9. It is a simple classifier composed of a 4-layer neural network with

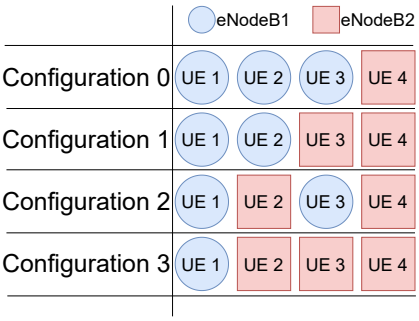


Figure 8: The Four Possible UE to eNodeB Configurations Considered in this Case Study

Rectified Linear Unit (RELU) activation for the first three layers. The classifier has four configurations, summarized in Figure 8, based on the eNodeBs to which UE 2 and UE 3 are connected. The input of the model is the distance of each UE to each eNodeB and the observed normalized packet loss of the UEs during the last 1 s. The classifier's output is a ranking of each configuration, which allows the selection of the one providing the lowest overall packet loss for the next 1 s. Moreover, in the O-RAN model itself, the Near-RT RIC is equipped with a single LM that, when queried, collects the locations reported by each UE and eNodeB from the DB, as well as the packet loss reported by each UE, calculates the distance between each UE and eNodeB, passes those inputs to the DL model, and then selects the configuration based on the DL model's inference. The data used to train this model is generated by running a simulation for each configuration without the use of any handover algorithms, and then post-processing the data to determine which configuration is best at any given time. In the case when the DL model is used, we will refer to it as the "ML" approach.

As a reference, we also employ two non-ML approaches that we refer to as the "heuristic" and "distance" approaches. The heuristic approach does not make use of any information collected via O-RAN, but instead makes use of the handover algorithm described in [8] that is included in ns-3's LTE module. The method used for handover is known as the "traditional power budget algorithm." This algorithm regularly monitors the Reference Signal Received Power (RSRP) of a UE's serving cell and neighboring cells so that once the RSRP of a neighboring cell is greater than the serving cell's, the UE is handed over to that neighboring cell. The distance approach does make use of the O-RAN reports similar to the ML approach, however, the LM uses the reported locations to calculate the distance between each UE and eNodeB, and then performs a handover for a UE if its distance to the neighboring cell is less than that of the serving cell.

A subset of parameters used in the simulations are described in Table 1, and the performance metrics considered are:

- The packet loss rate for each UE, defined as $PL = \frac{N_{Tx} - N_{Rx}}{N_{Tx}}$ where N_{Tx} is the total number of packets sent by the server and N_{Tx} is the total number of packets received at the UE.
- The aggregated throughput defined as $Thr = \sum_{i=1}^{n} Thr_{UE_i}$ where Thr_{UE_i} is the total throughput observed at the UE i in the last second for a given simulation time.

Table 1: Parameters of the Simulated Scenario

Parameter	*Value*	*Unit*
Simulated Time	210	s
Traffic	290	kbits/s
Pathloss Model	Cost231	NA
Max UE-to-eNodeB Distance	461	m
Scheduler	Round Robin	NA
O-RAN dependent parameters		
LM Invocation Interval	1	s
UE/eNodeB Report Interval	1	s
ML training parameters		
Epochs	3	NA
Training Set Size	666 667	NA
Verification Set Size	333 333	NA

- The total number of handovers that take place.

5.2 Simulation Results

Figure 10a shows the packet loss of each UE for all four approaches. On the y-axis, the total packet loss is represented as a percentage, the UE that the loss corresponds to is on the x-axis, and the color of the series denotes which approach the loss is recorded for. In all four cases neither UE 1 nor UE 4 experience any packet loss, thus we will only focus on UE 2 and UE 3. For the baseline, the total packet loss for UE 2 and UE 3 is 35 % and 9.2 %, respectively, resulting in an average packet loss of 22.1 %. With the heuristic approach the packet loss for UE 2 and UE 3 is reduced to 6.3 % and 2.8 %, resulting in an average packet loss of 4.5 %. The distance approach brings the losses down to 1.9 % and 6.4 % for an average packet loss of 4.2 %. Lastly, with the ML approach we see the greatest improvement as the packet loss recorded for UE 2 and UE 3 is 1.6 % and 5.3 %, resulting in an average packet loss of 3.5 %. Therefore, each approach reduces the total packet loss when compared to the baseline. However, the greatest improvements come from the O-RAN-based approaches, with the ML approach providing the greatest reduction in packet loss.

Figure 10c shows the aggregated throughput of all four UE over 1 s intervals at a given simulation time for each approach. The total throughput, in Mbits/s, is on the y-axis and the simulation time, in seconds, is on the x-axis. Looking at the chart, we can see that overall trends show the distance and ML approaches providing an overall higher throughput throughout the entire simulation. We also observe these two O-RAN-based approaches typically providing a throughput that is greater than 1 Mbit/s when compared to the heuristic approach and the baseline as the throughput in these two cases dip below 610 kbits/s. These dips also indicate that the signal quality between eNodeB 1 and UE 3 degrades significantly between 60 s and 69 s. The O-RAN-based solutions can avoid this drop as both the ML and distance approaches decide to handover UE 3 to eNodeB 2 at 61 s and 62 s, respectively. This behavior also indicates that the degradation in signal quality for UE 2 is due to its distance from eNodeB 1. However, with the heuristic approach no action is taken until the signal quality is degraded even further, with a handover taking place at 72 s to again realize a total throughput

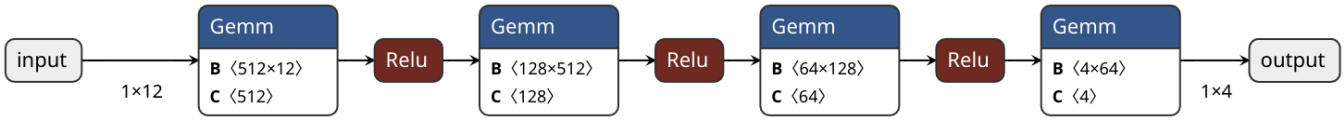
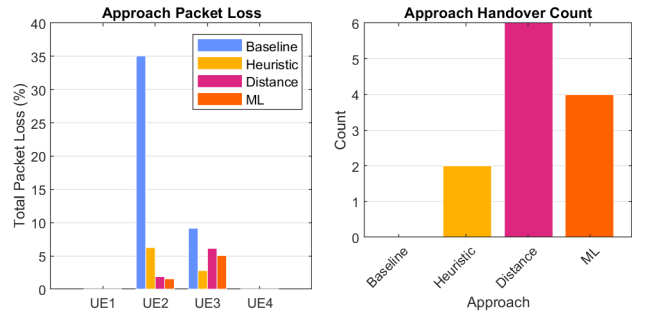


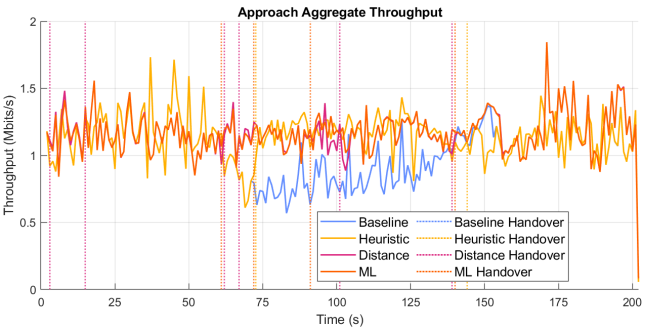
Figure 9: DL Model Used for Handover Management

of over 1 Mbit/s. While it appears that the most significant time for handovers to take place is between 61 s and 72 s, additional handovers also took place, and the totals for each approach is shown in Figure 10b.

Figure 10b shows the total number of handovers triggered for each approach. The count is shown on the y-axis, and the approach that the count corresponds to is on the x-axis. Notice that the improvement in performance achieved by the O-RAN-based approaches over the heuristic approach, requires additional handovers since the heuristic approach only performs two handovers. However, the ML approach is able to provide increased performance but with fewer handovers than the distance approach, as the ML approach only triggers four handovers while the distance approach triggers six.



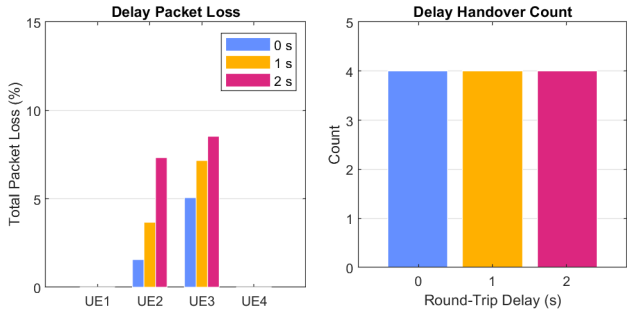
(a) Packet Loss Rate Per UE (b) Total Number of Handovers



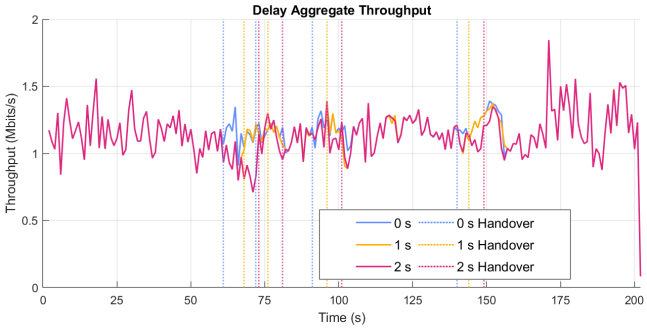
(c) Throughput Evolution Over Simulation Time

Figure 10: Simulation Results Comparing the Baseline with the Heuristic, Distance, and ML Approaches

Figures 11a, 11b, and 11c show how the link delay between the Near-RT RIC and the E2 nodes impacts the ML approach's performance. In this case we simply reuse the ML approach but introduce a constant delay on the E2 interface. Moreover, if the link delay is 1 s, then when a report is sent by an E2 node a total of 1 s in simulation time will pass before the Near-RT RIC receives the report, and the same delay would be incurred by a command sent from the Near-RT RIC to an E2 node. Increasing the link delay means that any information exchanged, whether it be a report or a command, is more representative of the past than it is of the current state of the network. Note that the "0 s" series in these plots means that there is no link delay, and it is representative of the ML series discussed in the previous plots. Figure 11a shows that the packet loss increases for both UE 2 and UE 3 as more delay is introduced on the link. Figure 11b shows that the number of handovers is not changing, which indicates that the impact on performance is related to timing since the same actions are taken across all three simulations but at later times with increasing link delays. Figure 11c also indicates that increasing the link delay hinders performance, as the overall trends show that the aggregated throughput is often lower with increases in the link delay.



(a) Packet Loss Rate Per UE (b) Total Number of Handovers



(c) Throughput Evolution Over Simulation Time

Figure 11: Simulation Results that Depict the Impact of Increasing the Link Delay of the E2 Interface

6 CONCLUSION

This paper gives the reader a high-level overview of O-RAN and presents a fully simulated model for studying O-RAN-based solutions. This includes a presentation of our ns-3 extension, as well as

an efficient integration of ML libraries with the simulation. We also show that these models and libraries make it possible to quickly define, evaluate, and tune solutions and mechanisms for improving the RAN performance based on the O-RAN concept. In terms of future work, we intend to use our model to implement existing and original case studies, and to explore the development of ML models further. Finally, we plan to improve the model by implementing some of the abstracted capabilities, such as the virtual E2 interface.

REFERENCES

[1] [n. d.]. *Open Neural Network Exchange Intermediate Representation (ONNX IR) Specification.* https://github.com/onnx/onnx/blob/main/docs/IR.md

[2] 2021. *MobiSys '21: Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services* (Virtual Event, Wisconsin). Association for Computing Machinery, New York, NY, USA.

[3] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ Software available from tensorflow.org.

[4] Luca Baldesi, Francesco Restuccia, and Tommaso Melodia. 2022. ChARM: NextG Spectrum Sharing Through Data-Driven Real-Time O-RAN Dynamic Control. In *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*. 240–249. https://doi.org/10.1109/INFOCOM48880.2022.9796985

[5] 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 *2022 IEEE Wireless Communications and Networking Conference (WCNC 2022)*. 518–523. https://doi.org/10.1109/WCNC51071.2022.9771908

[6] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. 2015. MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems. *CoRR* abs/1512.01274 (2015). http://dblp.uni-trier.de/db/journals/corr/corr1512.html#ChenLLLWWXXZZ15

[7] Estefanía Coronado, Shuaib Siddiqui, and Roberto Riggio. 2022. Roadrunner: O-RAN-based Cell Selection in Beyond 5G Networks. In *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*. 1–7. https://doi.org/10.1109/NOMS54207.2022.9789832

[8] Konstantinos Dimou, Min Wang, Yu Yang, Muhammmad Kazmi, Anna Larmo, Jonas Pettersson, Walter Muller, and Ylva Timner. 2009. Handover within 3GPP LTE: Design Principles and Performance. In *2009 IEEE 70th Vehicular Technology Conference Fall*. 1–5. https://doi.org/10.1109/VETECF.2009.5378909

[9] 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* (Miami Beach, FL, USA) *(MSWIM '19)*. Association for Computing Machinery, New York, NY, USA, 113–120. https://doi.org/10.1145/3345768.3355908

[10] Richard D Hipp. [n. d.]. *SQLite.* https://www.sqlite.org/index.html

[11] Andrea Lacava, Michele Polese, Rajarajan Sivaraj, Rahul Soundrarajan, Bhawani Shanker Bhati, Tarunjeet Singh, Tommaso Zugno, Francesca Cuomo, and Tommaso Melodia. 2022. Programmable and Customized Intelligence for Traffic Steering in 5G Networks Using Open RAN Architectures. *arXiv preprint arXiv:2209.14171* (2022).

[12] MATLAB. 2010. *version 7.10.0 (R2010a).* The MathWorks Inc., Natick, Massachusetts.

[13] Microsoft. [n. d.]. ONNX Runtime. https://onnxruntime.ai/.

[14] Subhadeep Mukhopadhyay. 2022. InfoGram and Admissible Machine Learning. *Machine Learning* 111, 1 (01 Jan 2022), 205–242. https://doi.org/10.1007/s10994-021-06121-4

[15] O-RAN Alliance. [n. d.]. *O-RAN Alliance.* https://www.o-ran.org

[16] O-RAN Alliance. 2018. *O-RAN: Towards an Open and Smart RAN.* White Paper. Open RAN (O-RAN) Alliance. https://assets-global.website-files.com/60b4ffd4ca081979751b5ed2/60e5afb502810a0947b3b9d0_O-RAN%2BWP%2BFInal%2B181017.pdf

[17] O-RAN Alliance. 2020. *O-RAN Use Cases and Deployment Scenarios.* White Paper. Open RAN (O-RAN) Alliance. https://assets-global.website-files.com/60b4ffd4ca081979751b5ed2/60e5aff9fc5c8d496515d7fe_O-RAN%2BUse%2BCases%2Band%2BDeployment%2BScenarios%2BWhitepaper%2BFebruary%2B2020.pdf

[18] O-RAN Alliance. 2021. *O-RAN Minimum Viable Plan and Acceleration towards Commercialization.* White Paper. Open RAN (O-RAN) Alliance. https://assets-global.website-files.com/60b4ffd4ca081979751b5ed2/61199f8adc85474118cf6969_O-RAN%20Minimum%20Viable%20Plan%20and%20Acceleration%20towards%20Commercialization%20White%20Paper%2029%20June%202021.pdf

[19] O-RAN Working Group 1. 2021. *O-RAN Information Model and Data Models 1.0.* Technical Specification. Open RAN (O-RAN) Alliance.

[20] O-RAN Working Group 1. 2021. *O-RAN Operations and Maintenance Architecture.* Technical Specification. Open RAN (O-RAN) Alliance.

[21] O-RAN Working Group 1. 2021. *O-RAN Operations and Maintenance Interface.* Technical Specification. Open RAN (O-RAN) Alliance.

[22] O-RAN Working Group 1. 2022. *O-RAN Architecture Description.* Technical Specification. Open RAN (O-RAN) Alliance. Version 7.0.

[23] O-RAN Working Group 1. 2022. *Use Cases Detailed Specification.* Technical Specification. Open RAN (O-RAN) Alliance. Version 9.0.

[24] O-RAN Working Group 2. 2022. *O-RAN A1 Interface: Application Protocol.* Technical Specification. Open RAN (O-RAN) Alliance.

[25] O-RAN Working Group 2. 2022. *O-RAN Non-RT RIC Architecture.* Technical Specification. Open RAN (O-RAN) Alliance.

[26] O-RAN Working Group 3. 2022. *Near-Real-time RAN Intelligent Controller Architecture & E2 General Aspects and Principles.* Technical Specification. Open RAN (O-RAN) Alliance.

[27] O-RAN Working Group 3. 2022. *O-RAN E2 Application Protocol (E2AP).* Technical Specification. Open RAN (O-RAN) Alliance.

[28] O-RAN Working Group 3. 2022. *O-RAN E2 Service Model (E2SM) KPM.* Technical Specification. Open RAN (O-RAN) Alliance.

[29] O-RAN Working Group 3. 2022. *O-RAN Near-RT RIC Architecture.* Technical Specification. Open RAN (O-RAN) Alliance.

[30] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[31] Michele Polese, Leonardo Bonati, Salvatore DOro, Stefano Basagni, and Tommaso Melodia. 2022. ColO-RAN: Developing Machine Learning-based xApps for Open RAN Closed-loop Control on Programmable Experimental Platforms. *IEEE Transactions on Mobile Computing* (2022), 1–14. https://doi.org/10.1109/TMC.2022.3188013

[32] Michele Polese, Leonardo Bonati, Salvatore D'oro, Stefano Basagni, and Tommaso Melodia. 2022. Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges. *ArXiv* abs/2202.01032 (2022).

[33] Iago Rego, Lucas Medeiros, Pedro Alves, Mateus Goldbarg, Vitor Lopes, Daniel Flor, Wysterlanya Barros, Vinícius Filho, Vicente Sousa, Eduardo Aranha, Allan Martins, Marcelo Fernandes, Ramon Fontes, and Augusto Neto. 2022. Prototyping Near-Real Time RIC O-RAN xApps for Flexible ML-based Spectrum Sensing. In *2022 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. 137–142. https://doi.org/10.1109/NFV-SDN56302.2022.9974940

[34] George F. Riley and Thomas R. Henderson. 2010. *The ns-3 Network Simulator.* Springer Berlin Heidelberg, Berlin, Heidelberg, 15–34. https://doi.org/10.1007/978-3-642-12331-3_2

[35] 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