OpenRAN Gym: An Open Toolbox for Data Collection and Experimentation with AI in O-RAN

Leonardo Bonati, Michele Polese, Salvatore D'Oro, Stefano Basagni, Tommaso Melodia
Institute for the Wireless Internet of Things, Northeastern University, Boston, MA, U.S.A.
E-mail: {l.bonati, m.polese, s.doro, s.basagni, melodia}@northeastern.edu

***Abstract*—Open Radio Access Network (RAN) architectures will enable interoperability, openness, and programmatic data-driven control in next generation cellular networks. However, developing scalable and efficient data-driven algorithms that can generalize across diverse deployments and optimize RAN performance is a complex feat, largely unaddressed as of today. Specifically, the ability to design efficient data-driven algorithms for network control and inference requires at a minimum (i) access to large, rich, and heterogeneous datasets; (ii) testing at scale in controlled but realistic environments, and (iii) software pipelines to automate data collection and experimentation. To facilitate these tasks, in this paper we propose OpenRAN Gym, a practical, open, experimental toolbox that provides end-to-end design, data collection, and testing workflows for intelligent control in next generation Open RAN systems. OpenRAN Gym builds on software frameworks for the collection of large datasets and RAN control, and on a lightweight O-RAN environment for experimental wireless platforms. We first provide an overview of OpenRAN Gym and then describe how it can be used to collect data, to design and train artificial intelligence and machine learning-based O-RAN applications (xApps), and to test xApps on a softwarized RAN. Then, we provide an example of two xApps designed with OpenRAN Gym and used to control a large-scale network with 7 base stations and 42 users deployed on the Colosseum testbed. OpenRAN Gym and its software components are open source and publicly-available to the research community.**

I. Introduction

The next generations of cellular networks will follow the Open Radio Access Network (RAN) paradigm, which promotes openness, virtualization, programmability, and data-driven control loops in the mobile environment [1]. This will help network operators support new bespoke services on the same physical infrastructure, thanks to the flexibility and reconfigurability of software-based deployments and algorithmic control. Open RAN will also decrease operational costs because of the increased efficiency enabled by virtualized and open ecosystems.

In this context, the O-RAN Alliance has developed specifications to apply the Open RAN paradigm to prevailing radio access technologies including 3GPP LTE and NR networks [2]. O-RAN specifications introduce standardized interfaces that connect new, O-RAN-specific network nodes to key RAN elements, such as the NR Central Units (CUs), Distributed Units (DUs), Radio Units (RUs), and the LTE O-RAN-compliant evolved Node Bases (eNBs) [3]. To enable programmatic closed-loop control of the RAN, O-RAN also introduces two so-called RAN Intelligent Controllers (RICs). The near-real-time (or near-RT) RIC is connected through the E2 interface to the RAN network functions (i.e., the CU and the DU) to enable control loops that operate at timescales between 10 ms and 1 s [4]. The non-real-time (non-RT) RIC, instead, is part of the service management and orchestration framework and operates at timescales larger than 1 s [5]. This component connects to one or multiple near-RT RICs through the A1 interface, used to distribute policies, external information, and to manage Artificial Intelligence (AI) and Machine Learning (ML) models. These models define intelligent network control strategies that are then run on the RICs in the form of applications—namely, xApps and rApps—that can be provided by RAN vendors, operators or third-party entities. xApps run in the near-RT RIC, while rApps run in the non-RT RIC.

The open and disaggregated O-RAN architecture enables the practical deployment of AI/ML solutions at scale. For instance, AI/ML algorithms can perform inference and traffic forecasting or configure RAN nodes based on run-time conditions and traffic requirements. The O-RAN specifications [6] discuss the typical workflow for the development and testing of AI/ML in the RAN [7]. This involves multiple steps, including: (i) *data collection*, to create rich datasets to capture the characteristics of the environment where the AI/ML solution will be deployed (e.g., wireless channel, user distributions and requirements) as well as various indicators of network performance under different configurations; (ii) *AI/ML model design*, with a selection of the model inputs and outputs, and *training and testing*, to understand its limits and effectiveness; (iii) *model deployment* as xApps or rApps; (iv) *model fine-tuning* with live data from the RAN, to adapt it to the production environment, and, finally, (v) the actual *inference, forecasting and/or control* of the RAN.

In this paper, we introduce OpenRAN Gym, an open toolbox to develop O-RAN-compliant AI/ML solutions, deploy them as xApps on the near-RT RIC, and test them on large-scale softwarized RANs controlled by the RIC. We first give an overview of OpenRAN Gym and its core components, discussing how they enable design and testing workflows of ML-based xApps. Then, we demonstrate how two xApps designed with OpenRAN Gym can be used to control a large-scale RAN instantiated on the Colosseum wireless network emulator through the SCOPE framework [8], and managed by an O-RAN near-RT RIC provided by the ColO-RAN framework [7]. To the best of our knowledge, this is the first open toolset for the end-to-end design and experimentation of data-driven O-RAN xApps on large-scale experimental platforms.

Previous experimental work has focused on the development of data-driven solutions and xApps for specific use

This work was partially supported by the U.S. National Science Foundation under Grants CNS-1925601, CNS-2120447, and CNS-2112471.

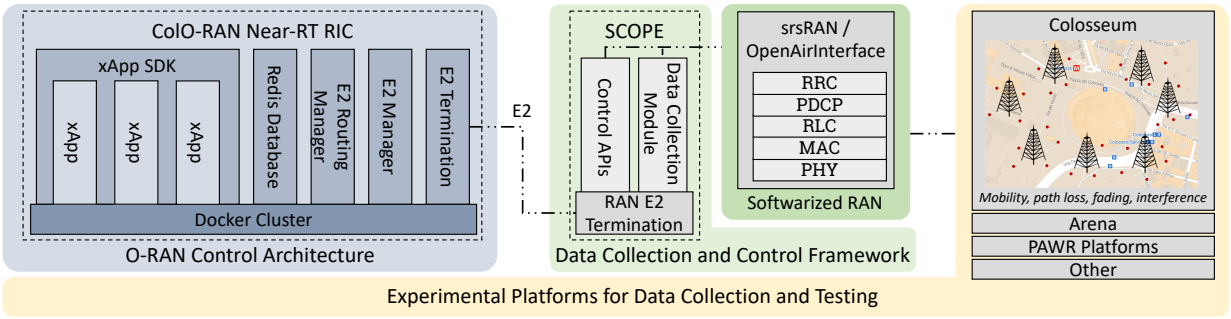


Fig. 1: OpenRAN Gym architecture.

cases [9, 10], on the description of the AI/ML capabilities of O-RAN [11, 12], on interoperability testing [13], and on orchestration [14]. Compared to the state of the art, OpenRAN Gym enables an end-to-end workflow for the design and testing of AI/ML solutions as xApps in the O-RAN ecosystem. By doing so, it empowers users with a first-of-its-kind open and publicly-available O-RAN-compliant toolbox that will unleash the potential of data-driven applications for next generation cellular networks. OpenRAN Gym aims at creating a thriving community of researchers and developers contributing to it with open source software components for experimental O-RAN-driven AI/ML research.[1]

The remainder of this paper is organized as follows. Section II provides an overview of OpenRAN Gym. A practical description of the SCOPE and ColO-RAN frameworks is given in Sections III and IV. Section V presents the xApp design and testing workflow, and provides an example of large-scale RAN control using xApps developed with OpenRAN Gym on Colosseum. Finally, Section VI concludes the paper.

II. OpenRAN Gym

The architecture of OpenRAN Gym is shown in Fig. 1. It includes: (i) one or multiple publicly-accessible *experimental platforms* for data collection and testing (e.g., Colosseum [15], Arena [16], the platforms of the PAWR program [17]); (ii) a *softwarized RAN* (e.g., implemented through srsRAN [18] or OpenAirInterface [19]); (iii) a *data collection and control framework* with Application Programming Interfaces (APIs) to control the cellular stacks and extract statistics from them (e.g., SCOPE [8]), and (iv) an *O-RAN control architecture* with interfaces to connect to the RAN and control it through AI/ML solutions (e.g., ColO-RAN [7]). The platform-independence of OpenRAN Gym allows users to collect data, design and train solutions in heterogeneous environments before deploying them on production networks. In this way, several evaluation and fine-tuning iterations can be performed in controlled setups to guarantee that the final AI/ML model behaves as expected.

At the time of this writing, OpenRAN Gym supports Arena [16] and Colosseum [15] as experimental platforms. Arena is an indoor testbed that includes a grid of 64 indoor antennas and 24 Software-defined Radios (SDRs) controlled by dedicated compute servers. Colosseum is a large-scale wireless network emulator that allows users to test solutions at scale through 128 USRP X310 SDRs and compute servers (called Standard Radio Nodes (SRNs)), and to emulate the conditions of wireless environments (e.g., path loss, fading, user mobility and signal interference) through a Massive Channel Emulator (MCHEM) that controls 128 additional USRP X310. MCHEM is capable of capturing the conditions of the wireless channel with high fidelity. Wireless channels can be modeled through ray-tracing software [20], reproduced in MCHEM through FPGA-based finite impulse response filters. In this way, Colosseum allows users to perform experiments in a multitude of different emulated terrains, as if the SDRs were operating in the real-world. Similarly, Colosseum can also emulate traffic with different characteristics and distributions through the Traffic Generator (TGEN) system, which generates and manages traffic flows between the SRNs.

In the current version of OpenRAN Gym, the softwarized RAN is based on srsRAN [18], which implements the full stack of a 3GPP eNB and controls the USRPs X310 of the SRNs that act as radio front-ends. The srsRAN protocol stack is extended by the SCOPE framework, which enhances it with additional networking and control functionalities. These include RAN slicing capabilities, additional scheduling policies, and data collection functionalities, as well as novel APIs to control such capabilities at run time. As we will discuss in Section III, this component leverages the emulation capabilities of Colosseum—which acts as a *wireless data factory*—to enable automated, large-scale data collection campaigns to create datasets with tens of hours of RAN experiments and in different wireless and traffic conditions [7, 21, 22].

Finally, the O-RAN control architecture is implemented through ColO-RAN, which extends and adapts the O-RAN Software Community (OSC) near-RT RIC to the Colosseum environment, and connects SCOPE-enabled base stations to the near-RT RIC through the O-RAN E2 interface. As we will discuss in Section IV, ColO-RAN provides an instance of an O-RAN-compliant near-RT RIC, together with an *xApp Software Development Kit (SDK)* that allows users to design and test AI/ML-based xApps (Fig. 1).

III. Data Collection and Control Framework

OpenRAN Gym leverages SCOPE [8] as the data collection and control framework. SCOPE provides a development

[1]The software components of OpenRAN Gym are publicly-available and accessible at https://openrangym.com

TABLE I: Main SCOPE slicing and scheduling configuration parameters.

Name	Description
`network-slicing`	Enable/disable network slicing
`slice-allocation`	Define the base station slice allocation
`slice-scheduling-policy`	Set the scheduling policy for each slice
`slice-users`	Assign UEs to slices

environment to design, prototype and test adaptive solutions for cellular networking, and for large-scale data collection of RAN Key Performance Measurements (KPMs). It builds on top of srsRAN [18], and extends it with novel functionalities (e.g., network slicing and additional scheduling policies), open APIs to control the RAN configuration at run time (e.g., the resources allocated to each slice), and data collection capabilities. SCOPE has powerful data collection capabilities when used in combination with Colosseum, which makes it possible to automatically perform large-scale data collection campaigns in realistic wireless environments [8]. Examples include data collection campaigns with tens of hours of experiments, and in setups with up to 49 nodes (7 base stations and 42 users) in sliced RANs with different Quality of Service (QoS) requirements, scheduling policies, slicing resources, and in different emulation scenarios [7, 21, 22]. SCOPE has been extended to incorporate a RAN-side E2 termination (based on the OSC DU [23]) to connect to the O-RAN near-RT RIC. This allows xApps running on the RIC to interface with the SCOPE APIs and control the functionalities of the base stations at run time.

In the remaining of this section, we will show how to configure the main parameters of the SCOPE base stations, and how to start SCOPE on Colosseum. It is worth mentioning that even if we specifically focus on the implementation for the Colosseum network emulator (provided as a ready-to-use container image, namely `scope-e2`), SCOPE can be ported to different platforms with minor adaptations (see [8, Section 5.3]).

A. Starting SCOPE

SCOPE allows users to configure the base stations through JSON files. The main parameters to set up the network slicing and scheduling functionalities (see Table I) are as follows.[2]

- `network-slicing`: this parameter enables/disables the network slicing capabilities of the SCOPE base station.
- `slice-allocation`: sets the Resource Block Groups (RBGs) of each slice. It takes as input {`slice:[first_rbg, last_rbg],...`}, e.g., {`0:[0,3],1:[5,7]`} assigns RBGs 0-3 to slice 0 and RBGs 5-7 to slice 1.
- `slice-scheduling-policy`: sets the scheduling policy the base station uses for each slice, e.g., `[2,0]` assigns policy 2 to slice 0 and policy 0 to slice 1. The possible values correspond to the scheduling policies supported by SCOPE (0 for round-robin, 1 for waterfilling, 2 for proportionally fair).
- `slice-users`: assigns User Equipments (UEs) to the slices. It takes as input {`slice:[ue1,ue2],...`}, e.g., {`0:[2,5],1:[3,4]`} assigns UEs 2, 5 to slice 0, and UEs 3, 4 to slice 1. The UE ID corresponds to the i-th SRN allocated to the experimenter (the base station is assumed as the first SRN).

After saving the above parameters in a JSON-formatted configuration file[3] (e.g., named `radio.conf`), experiments can be started through the commands shown in Listing 1. This

```
#!/bin/bash
cd radio_api/
python3 scope_start.py --config-file radio.conf
```

Listing 1: Commands to start SCOPE applications.

command, executed on the SRNs assigned to the user, takes care of starting base station, core network and UEs applications.

At experiment run time, the SCOPE APIs can be used to reconfigure the base station, e.g., to modify the amount of RBGs allocated to each slice, or their scheduling policy (see [8, Section 3.3]). Finally, RAN KPMs are automatically logged by the base stations and saved into CSV-formatted files. These files can either be used on-the-fly (e.g., to perform online inference or model training) or retrieved at a later time from Colosseum data storage (e.g., for offline training or data processing).

IV. O-RAN Control Architecture

The O-RAN control architecture component used by Open-RAN Gym is provided by ColO-RAN, an open development environment to design, train and test data-driven O-RAN-compliant solutions at scale [7]. ColO-RAN offers a minimalist implementation of the OSC near-RT RIC, that can be instantiated on Colosseum through Docker containers, as well as scripts for the automated deployment of the RIC components. ColO-RAN includes containers for the O-RAN *E2 termination, E2 manager* and *E2 routing manager* that handle the communications within the RIC and with the RAN nodes (e.g., SCOPE base stations), a *Redis database* that records the RAN nodes connected to the RIC, and an *xApp SDK* (Fig. 1). The latter provides software tools to design and test AI/ML-based xApps for run-time RAN inference and/or control.

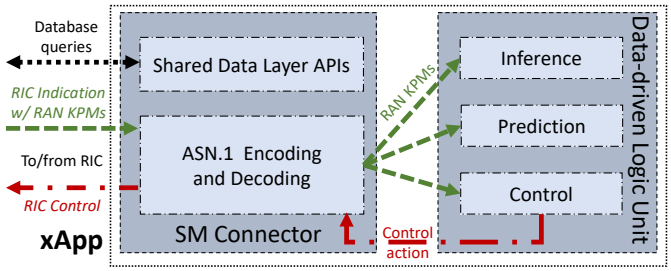


Fig. 2: ColO-RAN xApp, adapted from [7].

At a high level, ColO-RAN xApps are made of two building blocks, shown in Fig. 2: (i) the *Service Model (SM) connector*, which handles the communications to/from the near-RT RIC (e.g., to communicate with the base stations), ASN.1 message encoding/decoding, and queries to the RIC Redis database, and (ii) the *data-driven logic unit* that performs tasks based

[2]A comprehensive description of all parameters of the SCOPE APIs configuration files can be found at https://github.com/wineslab/colosseum-scope.

[3]An example of a SCOPE configuration file can be found at https://github.com/wineslab/colosseum-scope/blob/main/radio_api/radio_interactive.conf.

on KPMs received from the RAN at run time, e.g., traffic prediction and/or control of the base stations. Notice that at the time of this writing, ColO-RAN xApps use custom SMs. Standard-compliant SMs are part of our future work.

In the remainder of this section, we will show how to instantiate the ColO-RAN near-RT RIC on Colosseum (Section IV-A), connect the SCOPE base station to the RIC through the O-RAN E2 termination (Section IV-B), and start a sample xApp that interfaces with the base station (Section IV-C). Even though we focus on the implementation for the Colosseum environment (provided in a ready-to-use container image, namely `coloran-near-rt-ric`), ColO-RAN can be ported to different platforms with minor adjustments (see [7, Section 6]).

A. Starting the ColO-RAN Near-RT RIC

The Docker images of the ColO-RAN near-RT RIC can be built and started as containers through the provided `setup-ric.sh` script and the commands of Listing 2. This script (adapted from [24]) takes as argument the network interface used by the RIC to communicate with the RAN (e.g., the `col0` interface in Colosseum). First, base Docker images,

```
1 #!/bin/bash
2 cd setup-scripts/
3 ./setup-ric.sh col0
```

Listing 2: Commands to set up the ColO-RAN near-RT RIC.

used to build the RIC images, are imported. Then, the four images composing the near-RT RIC are built, and their IP addresses and ports (defined in the `setup-lib.sh` script) are configured. These images include: (i) `e2term`, which is the endpoint of the E2 messages on the RIC ("E2 Termination" in Fig. 1); (ii) `e2mgr`, which manages the messages to/from the E2 interface ("E2 Manager"); (iii) `e2rtmansim`, which leverages the RIC Message Router (RMR) protocol to route the E2 messages inside the RIC ("E2 Routing Manager"), and (iv) `db`, which maintains a database of the RAN nodes connected to the RIC ("Redis Database"). After the building process completes, the Docker images are initialized as containers on a Colosseum SRN, and listen for incoming connections from RAN nodes implementing an E2 termination endpoint. The container logs can be read through the `docker logs` command, e.g., `docker logs e2term -f` shows the logs of the E2 termination (`e2term`) container.

B. Connecting the SCOPE Base Station to ColO-RAN

After the ColO-RAN neat-RT RIC is started following the steps of Section IV-A, the SCOPE base station (set up in Section III-A) can be connected to it. To this aim, the SCOPE base station runs an instance of the O-RAN E2 termination, which we adapted from the OSC DU implementation [23]. After connecting to the near-RT RIC, this component can exchange messages with the xApps running therein. Specifically, the E2 termination of the base station can: (i) receive *RIC Subscription* messages from the xApps; (ii) send periodic RAN KPMs to the xApps through *RIC Indication* messages; (iii) receive *RIC Control* messages from the xApps, and (iv) interface with the SCOPE APIs to modify the scheduling and slicing configurations of the base station based on the received xApp control.

Listing 3 shows the steps to initialize the E2 termination on the SCOPE base station. The E2 termination is first built

```
1 #!/bin/bash
2 cd colosseum-scope-e2/
3 ./build_odu.sh clean
4 ./run_odu.sh
```

Listing 3: Commands to build and start the SCOPE E2 termination process.

through the `build_odu.sh` script of line 3, which also sets the IP address and port of the near-RT RIC to connect to, as well as the network interface used for the connection to the RIC. Then, the E2 termination can be started through the `run_odu.sh` script (line 4), which initializes the E2 termination and connects it to the near-RT RIC. The successful connection of base station and near-RT RIC can be verified by reading the logs of the `e2term` container (through the command `docker logs e2term -f`, see Section IV-A). This log shows the association messages between the RIC and the base station, together with the ID of the connected base stations (e.g., `gnb:311-048-01000501`).

C. Initializing a Sample xApp

After starting the near-RT RIC, and connecting the SCOPE base station to it, the sample xApp provided as part of ColO-RAN can be initialized. This can be done through the `setup-sample-xapp.sh` script and the commands shown in Listing 4. The script takes as input the ID of the RAN node the

```
1 #!/bin/bash
2 cd setup-scripts/
3 ./setup-sample-xapp.sh gnb:311-048-01000501
```

Listing 4: Commands to build the ColO-RAN sample xApp Docker image, and to start and configure the xApp container.

xApp subscribes to (e.g., the base station), which can be read in the logs of the ColO-RAN `e2term` Docker container (see Section IV-B). It then builds the Docker image of the sample xApp, and starts it as a Docker container on the near-RT RIC.

After the xApp container (named, for instance `sample-xapp`) has been started, the xApp process can be run with the commands shown in Listing 5. By running these commands, the

```
1 #!/bin/bash
2 docker exec -it sample-xapp
      /home/sample-xapp/run_xapp.sh
```

Listing 5: Commands to run the ColO-RAN sample xApp process.

xApp subscribes to the RAN node specified in Listing 4 (through a RIC Subscription message), and triggers periodic reports (sent through RIC Indication messages) of RAN KPMs from the node.

After performing these steps, the ColO-RAN sample xApp logs on file the KPMs received from the RAN node. Users of OpenRAN Gym can add custom intelligence (e.g., through AI/ML agents) to the xApp by modifying the template scripts in the `setup/sample-xapp/` directory, and rebuilding the xApp Docker image through the steps described in this section.

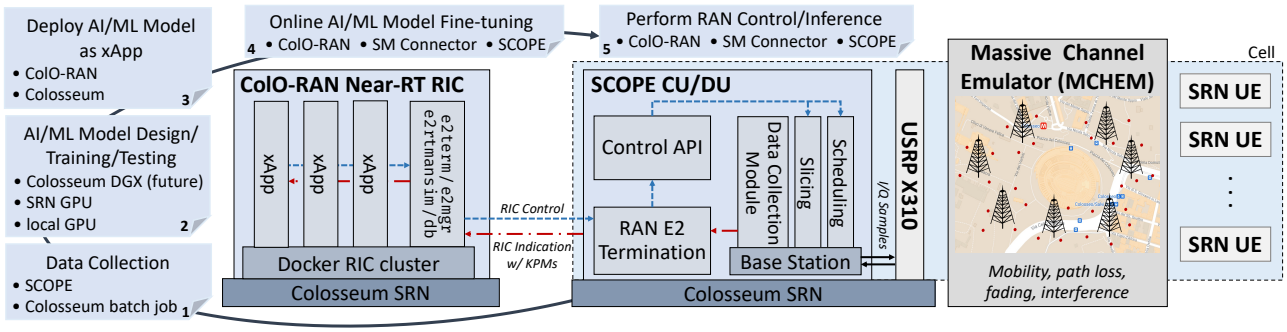


Fig. 3: xApp design and testing workflow in OpenRAN Gym.

V. xApp Design and Testing Workflow

The steps of the workflow to develop a data-driven xApp in OpenRAN Gym, shown in Fig. 3, are described next.

1) **Data collection.** Data to train the AI/ML model that will be included in the xApp needs to be collected. In Colosseum, this can be done by using SCOPE to instantiate a large-scale cellular network with multiple base stations and UEs (see Section III and [8]). During the experiment, the base stations log the KPMs relative to the performance of the served UEs, thus creating CSV-formatted datasets representative of the RAN statistics. Colosseum then transfers these datasets to its internal data storage, making them accessible to the users after the experiment ends. Data should be representative of a variety of environments and conditions, so that the model can adapt to diverse channel conditions and traffic requirements. Using OpenRAN Gym within Colosseum, this can be achieved by running several experiments that emulate different wireless and traffic scenarios (e.g., through SCOPE cellular scenarios [8]).

However, manually running hundreds of experiments at scale and in different scenarios is not trivial, and it may take a long time. To facilitate this task, users can leverage Colosseum *batch jobs*, which take care of automatically running scheduled experiments (or jobs) configured through JSON files [15, Section IV]. In this way, users can schedule experiments in different scenarios and configuration in advance, and then retrieve the generated data from Colosseum data storage.

2) **AI/ML model design, training and testing.** After collecting data in the desired wireless environments, the model can be designed. This step involves selecting the AI/ML techniques that the model will use, which data it should take as input, the reward function of the model, and the set of actions produced as output (e.g., to perform inference or control of the RAN).

After the model has been designed, it can be trained and tested offline on the data collected in step 1.[4] These phases, which usually benefit from GPU-enabled environments, can either be carried out locally (on the user's own GPUs), or on the GPUs of the SRNs. As another option to train and test AI/ML solutions, Colosseum has recently added two NVIDIA DGX servers with A100 GPUs as part of its planned extensions [15]. Although at the time of this writing these resources cannot be reserved by the users of the testbed yet, their general availability is envisioned for Q2 2022. These novel servers will significantly increase Colosseum's computational capabilities, making it, together with OpenRAN Gym, a key tool to develop data-driven solutions to control Open RAN systems.

3) **Deploy the AI/ML model as an xApp.** After the model has been designed, trained, and tested, it can be deployed as an xApp on the ColO-RAN near-RT RIC. This can be done through the procedures described in Section IV-C. Specifically, the trained AI/ML model can be included in the ColO-RAN sample xApp (as the *data-driven logic unit* of Fig. 2) modifying the template scripts in the `setup/sample-xapp/` directory. Then, the Docker image of the xApp with the user-defined logic is built with the commands of Listing 4, and instantiated on the ColO-RAN near-RT RIC through the commands of Listing 5.

4) **Online AI/ML model fine-tuning.** Upon startup, the xApp interfaces with the SCOPE base station through the ColO-RAN near-RT RIC and the O-RAN E2 termination. First, the xApp subscribes to SCOPE CU/DU by sending it RIC Subscription messages. Then, it triggers periodic reports (tunable based on the experiment requirements [7]) of RAN KPMs from the base station, sent through RIC Indication messages. The xApp may also use the KPMs in these report messages to fine-tune the model online, which allows it to adapt to the actual production environment where it will be deployed. Once the model has undergone this additional phase of online training, the xApp Docker image can be updated to save the newly trained weights of the model.

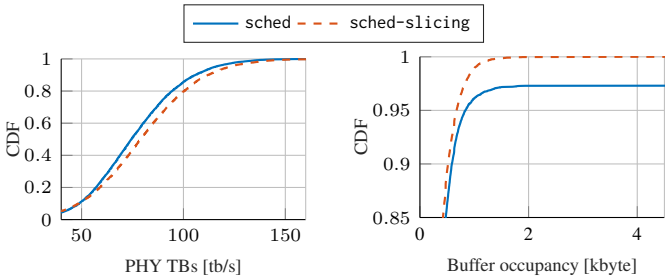
5) **Perform RAN control/inference.** Now the xApp can be used in the production infrastructure to perform run-time inference and control of the RAN. This latter step is achieved by having the xApp transmitting the actions computed by the AI/ML model (e.g., to modify the parameters and configuration of the base station) through RIC Control messages. These are sent to the base station through the O-RAN E2 interface, where they are processed by the CU/DU the xApp is subscribed to. At the CU/DU side, these messages trigger the SCOPE control APIs (see Fig. 3) that apply the newly received configuration to the protocol stack of the base station at run time.

We now showcase an example of xApps designed and tested with OpenRAN Gym, and used to control a cellular network with 7 base stations and 42 UEs (6 UEs per base station) instantiated on Colosseum. Each base station serves

[4]It is worth mentioning that the O-RAN specifications do not permit the deployment of AI/ML models that have not been pre-trained offline. This is to shield the RAN from poor performance or outages [6].

UEs with different QoS requirements, divided on three network slices: Enhanced Mobile Broadband (eMBB), Machine-type Communications (MTC), and Ultra Reliable and Low Latency Communications (URLLC) slice. We follow the workflow described in this section to design two Deep Reinforcement Learning (DRL)-based xApps—trained on a collected dataset with 3.4 GB of RAN traces, and more than 73 hours of experiments—that control the configuration of the base stations at run time using RAN KPMs, as discussed in [7]. One xApp (named `sched`) manages the scheduling policies of the base station; the other (`sched-slicing`) also allocates the amount of resources available to each slice.

Figure 4 illustrates the Cumulative Distribution Function (CDF) of the RAN when the two xApps are instantiated on the ColO-RAN near-RT RIC and used to control the network. Statistics on the transmitted Transport Blocks (TBs) for the MTC slice are shown in Fig. 4a; on the downlink buffer occupancy of the URLLC slice in Fig. 4b. In this example, both



(a) MTC transmitted packets (b) URLLC buffer occupancy

Fig. 4: Comparison of xApps developed with OpenRAN Gym.

xApps aim at maximizing the transmit rate of the MTC traffic, and at minimizing the time packets remain in the base station queues for the URLLC traffic. By acting on an additional action set (i.e., the slice resource allocation), the `sched-slicing` xApp achieves better performance both in terms of transmitted packets and of buffer occupancy.[5]

VI. Conclusions

We presented OpenRAN Gym, the first publicly-available research platform for data-driven O-RAN experimentation at scale. Building on frameworks for data collection and RAN control, OpenRAN Gym enables end-to-end design and testing of data-driven O-RAN xApps. We described the main components of OpenRAN Gym, detailing configuration options and procedures for experimenting at scale on Colosseum. We then gave an overview of the OpenRAN Gym xApp design and testing workflow, from user instantiation of their AI/ML models as xApps on the near-RT RIC, to performing inference, prediction and/or control of base stations using live KPMs from the RAN. Finally, we provided an example of two xApps designed with OpenRAN Gym and used to control a large-scale O-RAN-managed network deployed on Colosseum. OpenRAN Gym has been made publicly-available to the research community, and opened up for community contributions and additions.

[5]A detailed evaluation of OpenRAN Gym xApps, including their orchestration, and control of large-scale experimental networks can be found in [7, 14].

References

[1] L. Bonati, M. Polese, S. D'Oro, S. Basagni, and T. Melodia, "Open, programmable, and virtualized 5G networks: State-of-the-art and the road ahead," *Computer Networks*, vol. 182, pp. 1–28, December 2020.

[2] O-RAN Working Group 1, "O-RAN Architecture Description 5.00," O-RAN.WG1.O-RAN-Architecture-Description-v05.00 Technical Specification, July 2021.

[3] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges," *arXiv:2202.01032 [cs.NI]*, February 2022.

[4] O-RAN Working Group 3, "O-RAN Near-RT RAN Intelligent Controller Near-RT RIC Architecture 2.00," O-RAN.WG3.RICARCH-v02.00, March 2021.

[5] O-RAN Working Group 2, "O-RAN Non-RT RIC Architecture 1.0," O-RAN.WG2.Non-RT-RIC-ARCH-TS-v01.00 Technical Specification, July 2021.

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

[7] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "ColO-RAN: Developing Machine Learning-based xApps for Open RAN Closed-loop Control on Programmable Experimental Platforms," *arXiv:2112.09559 [cs.NI]*, December 2021.

[8] L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "SCOPE: An open and softwarized prototyping platform for NextG systems," in *Proceedings of ACM MobiSys*, June 2021.

[9] M. Dryjanski, L. Kulacz, and A. Kliks, "Toward Modular and Flexible Open RAN Implementations in 6G Networks: Traffic Steering Use Case and O-RAN xApps," *Sensors*, vol. 21, no. 24, pp. 1–14, December 2021.

[10] D. Johnson, D. Maas, and J. Van Der Merwe, "Open Source RAN Slicing on POWDER: A Top-to-Bottom O-RAN Use Case," in *Proceedings of ACM MobiSys*, June 2021.

[11] H. Lee, J. Cha, D. Kwon, M. Jeong, and I. Park, "Hosting AI/ML Workflows on O-RAN RIC Platform," in *Proceedings of IEEE GLOBECOM Workshops*, December 2020.

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

[13] O-RAN Alliance Conducts First Global Plugfest to Foster Adoption of Open and Interoperable 5G Radio Access Networks. (2019, December) https://tinyurl.com/f48auynf.

[14] S. D'Oro, L. Bonati, M. Polese, and T. Melodia, "OrchestRAN: Network Automation through Orchestrated Intelligence in the Open RAN," in *Proceedings of IEEE INFOCOM*, May 2022, arXiv:2201.05632 [cs.NI].

[15] L. Bonati, P. Johari, M. Polese, S. D'Oro, S. Mohanti, M. Tehrani-Moayyed, D. Villa, S. Shrivastava, C. Tassie, K. Yoder, A. Bagga, P. Patel, V. Petkov, M. Seltser, F. Restuccia, A. Gosain, K. R. Chowdhury, S. Basagni, and T. Melodia, "Colosseum: Large-Scale Wireless Experimentation Through Hardware-in-the-Loop Network Emulation," in *Proceedings of IEEE DySPAN*, December 2021.

[16] L. Bertizzolo, L. Bonati, E. Demirors, A. Al-Shawabka, S. D'Oro, F. Restuccia, and T. Melodia, "Arena: A 64-antenna SDR-based Ceiling Grid Testing Platform for Sub-6 GHz 5G-and-Beyond Radio Spectrum Research," *Computer Networks*, vol. 181, pp. 1–17, November 2020.

[17] Platforms for Advanced Wireless Research (PAWR). https://www.advancedwireless.org. Accessed December 2021.

[18] I. Gomez-Miguelez, A. Garcia-Saavedra, P. Sutton, P. Serrano, C. Cano, and D. Leith, "srsLTE: An open-source platform for LTE evolution and experimentation," in *Proceedings of ACM WiNTECH*, October 2016.

[19] F. Kaltenberger, A. P. Silva, A. Gosain, L. Wang, and T.-T. Nguyen, "OpenAirInterface: Democratizing innovation in the 5G era," *Computer Networks*, no. 107284, May 2020.

[20] M. Tehrani-Moayyed, L. Bonati, P. Johari, T. Melodia, and S. Basagni, "Creating RF Scenarios for Large-Scale, Real-Time Wireless Channel Emulators," in *Proceedings of IEEE MedComNet*, June 2021.

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

[22] B. Casasole, L. Bonati, S. D'Oro, S. Basagni, A. Capone, and T. Melodia, "QCell: Self-optimization of Softwarized 5G Networks through Deep Q-learning," in *Proceedings of IEEE GLOBECOM*, December 2021.

[23] O-RAN Software Community. O-DU L2 Repository. https://github.com/o-ran-sc/o-du-l2. Accessed December 2021.

[24] David Johnson. POWDER RIC Profile Repository. https://gitlab.flux.utah.edu/johnsond/ric-profile. Accessed December 2021.