Intelligent Closed-loop RAN Control with xApps in OpenRAN Gym

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: {bonati.l, m.polese, s.doro, s.basagni, melodia}@northeastern.edu

***Abstract*—Softwarization, programmable network control and the use of all-encompassing controllers acting at different timescales are heralded as the key drivers for the evolution to next-generation cellular networks. These technologies have fostered newly designed intelligent data-driven solutions for managing large sets of diverse cellular functionalities, basically impossible to implement in traditionally closed cellular architectures. Despite the evident interest of industry on Artificial Intelligence (AI) and Machine Learning (ML) solutions for closed-loop control of the Radio Access Network (RAN), and several research works in the field, their design is far from mainstream, and it is still a sophisticated—and often overlooked—operation. In this paper, we discuss how to design AI/ML solutions for the intelligent closed-loop control of the Open RAN, providing guidelines and insights based on exemplary solutions with high-performance record. We then show how to embed these solutions into xApps instantiated on the O-RAN near-real-time RAN Intelligent Controller (RIC) through OpenRAN Gym, the first publicly available toolbox for data-driven O-RAN experimentation at scale. We showcase a use case of an xApp developed with OpenRAN Gym and tested on a cellular network with 7 base stations and 42 users deployed on the Colosseum wireless network emulator. Our demonstration shows the high degree of flexibility of the OpenRAN Gym-based xApp development environment, which is independent of deployment scenarios and traffic demand.**

***Index Terms*—O-RAN, 5G/6G, Open RAN, AI, xApp.**

I. Introduction

Recent years have witnessed the softwarization of cellular networks and of Radio Access Network (RAN) deployments [1]. Standardization bodies and other telecom organizations have been proposing all-encompassing solutions to manage the very many different functions of next generation cellular networks. O-RAN is arguably the most noteworthy of these solutions. Network operation and control are uniformly overseen via RAN Intelligent Controllers (RICs) acting at the different timescales typical of network operations: non-real-time (or non-RT) and near-real-time (or near-RT) timescales [2]. Intelligent RAN closed-loop control is enabled in O-RAN by data-driven applications, called *xApps* on the near-RT RIC and *rApps* on the non-RT RIC, which optimize network performance based on live data received from the RAN through standardized and open interfaces [3]. Here, the rise of Artificial Intelligence (AI) and Machine Learning (ML) applications for cellular networking brings forward the need for large-scale experimental facilities where to safely design and test data-driven solutions at scale without compromising the operations of commercial deployments.

Solutions for data-driven control of the new RAN have flourished in recent years. Some works concern the design and implementation of xApps in small-size setups [4–6]. Others focus on specific use cases [7], are structured to describe specific O-RAN functionalities and capabilities [2, 8], evaluate multivendor interoperability [9, 10], or focus on the general organization of networks managed by O-RAN [11]. All these works illustrate the strategic relevance of the paradigm heralded by O-RAN as the future of cellular networking, while however addressing the challenges of its usage, e.g., the design of xApps, in piecemeal fashion and limited setups, showing results that are often difficult to replicate.

Motivated by the need of providing a ready-made environment for testing at scale, Bonati et al. introduced OpenRAN Gym, the first publicly-available research framework for data-driven O-RAN experimentation with hardware-in-the-loop [12]. OpenRAN Gym enables uniform design of AI/ML-based solutions and to implement them as xApps for an O-RAN-compliant near-RT RIC. It also provides a framework to safely test them at scale in the RIC controlling a softwarized RAN. Moreover, OpenRAN Gym provides users with the capability of performing data collection campaigns in heterogeneous environments, which is key to train data-driven solutions that can generalize to different deployment scenarios [13].

Even though OpenRAN Gym provides a streamlined open-source environment to prototype solutions at scale, the proper design, testing and validation of xApps for the Open RAN is however not trivial. Besides the need for exhaustive data collection, for generalizable solutions and for testing in controlled environments, interfacing and adapting the final xApps to the dynamics of a commercial grade production infrastructure requires additional careful steps. These include the implementation of O-RAN-compliant interfaces, procedures, and messages—used by the xApps to communicate with the RAN—and, potentially, additional online training to fine-tune the designed agent to the production infrastructure.

In this paper, we illustrate and discuss the steps for designing and testing data-driven xApps for closed-loop control and inference of a softwarized RAN through OpenRAN Gym. We first provide an overview of the OpenRAN Gym framework, and of how it can be used to deploy and test Open RAN solutions at scale. We then detail how to design AI/ML-based xApps that implement closed-loop control of the configuration

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

of the base stations based on live data from the RAN. Finally, we showcase an example of xApp designed with OpenRAN Gym for controlling a large-scale softwarized RAN with 7 base stations and 42 User Equipments (UEs) instantiated on the Colosseum wireless network emulator [14]. Our work demonstrates the adaptability of the xApp to different traffic requirements and conditions, and the role of additional online training for boosting the performance of the RAN.

The remainder of this paper is organized as follows. Section II provides an overview of OpenRAN Gym. Section III describes the design of xApps. Section IV provides an example of xApp designed with OpenRAN Gym and tested on a large-scale RAN. Section V discusses future directions and the challenges of closed-loop control. Section VI concludes the paper.

II. An Overview of OpenRAN Gym

The OpenRAN Gym framework is made up of a set of architectural components, including softwarized RAN protocol stacks like srsRAN [15] and OpenAirInterface [16], data collection and control frameworks such as SCOPE [17], and O-RAN control architectures such as ColO-RAN [13]. OpenRAN Gym also provides hooks for usage in experimental wireless platforms for testing at scale, including Colosseum [14] for emulation-based experiments, Arena for indoor testing, and the outdoor platforms of the PAWR program [18].

SCOPE is a framework for data-collection and for the run-time control of a softwarized RAN. It builds on srsRAN, which allows users to instantiate cellular protocol stacks on a generic infrastructure, and to use Software-defined Radios (SDRs) as radio front-ends. SCOPE extends srsRAN with functionalities such as the ability to instantiate multiple network slices on the same softwarized base station, to select the scheduling policy used by each slice, and to perform automatic data collection of RAN Key Performance Measurements (KPMs) (e.g., throughput, transmitted packets). It also implements control Application Programming Interfaces (APIs) to reconfigure RAN parameters at run-time, including the amount of resources for each slice, and their scheduling policy. Finally, SCOPE includes a RAN-side E2 termination—adapted from the one released by the O-RAN Software Community (OSC) [19]—that is used to interact with the near-RT RIC.

ColO-RAN implements an O-RAN-compliant near-RT RIC, which is a lightweight OSC RIC tailored to run in the containerized environments typically used in experimental platforms for wireless research [13]. It provides a Software Development Kit (SDK) to design, train, and test xApps for RAN inference and control, as well as a ready-to-use xApp skeleton, where users can plug custom AI/ML models. ColO-RAN also implements RIC messaging to handle communications with the xApps and the RAN. Examples include the *RIC Subscription Indication/Response messages*—used by the base stations to establish the initial connection with the RIC—*E2 Indication messages*—used by the base stations to transmit periodic KPM reports to the xApps—and *E2 Control messages*—used by the xApp to control functionalities exposed by the base stations (e.g., to reconfigure their scheduling and slicing configuration).

The capabilities of OpenRAN Gym for facilitating data collection campaigns at scale [13], for extending O-RAN control loops to real-time procedures via dApps [20], and for performing control and inference of large-scale softwarized RANs [21], have been demonstrated on the Colosseum testbed. The containerized solutions developed with OpenRAN Gym can also be ported to other testbeds with minor adjustments, e.g., on the Arena testbed, and on the PAWR platforms.

III. How to Design an xApp

In this section, we illustrate the steps needed to design an O-RAN-compliant data-driven xApp that can be used in OpenRAN Gym and other RICs. In general, xApps interact with the RAN nodes through a component of the E2 interface called Service Model (SM).[1] The O-RAN Alliance has defined, and is still defining, multiple SMs, to carry out different tasks through a standardized interaction between xApps and base stations. The two main components of an OpenRAN Gym xApp are shown in Figure 1 (adapted from [12]).

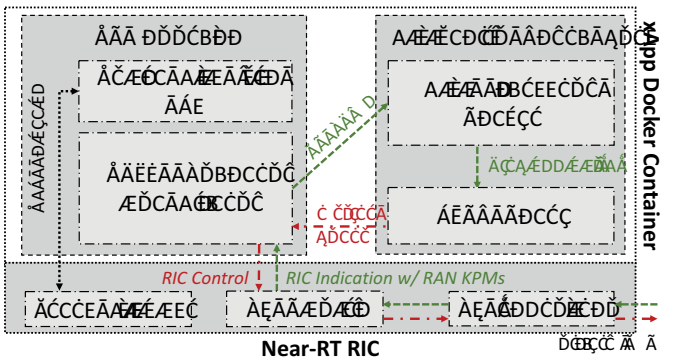


Fig. 1: Structure of an xApp [12].

As part of the OpenRAN Gym publicly available components,[2] we provide an xApp skeleton that implements the first component, *SM connector*, and has a drop-in component for the second, the *data-driven logic unit*, so that interested researchers can plug their own solutions in. In the following paragraphs, we describe how to use and, possibly, extend the SM connector, and how to design general and effective AI and ML solutions for the control of the RAN in the data-driven logic unit.

A. Service Model Connector

The SM connector handles the communication between the xApp (instantiated as a Docker container) and the near-RT RIC [22]. It relays RAN KPMs and xApp control to and from the data-driven logic unit. The connector includes multiple components for the interaction with the rest of the near-RT RIC infrastructure, including features for specific API

[1]The other component, the E2 Application Protocol (AP), provides foundations and primitives for the different SMs and basic interactions between the RIC and the RAN nodes, e.g., connection setup, teardown, etc.

[2]https://openrangym.com

ISBN 978-3-8007-6001-5

and messages parsing. For instance, the O-RAN *shared data layer APIs* are used to query the Redis database Network Information Base (NIB) (or R-NIB) deployed on the RIC, e.g., to get the list of the base stations to subscribe to.

ASN.1 Serialization. The *ASN.1 encoding and decoding* module uses the standardized ASN.1 interface description language to serialize/deserialize messages to/from the E2 manager component and to/from the E2 termination of the RIC. Examples include:

- The RIC Subscription message, used by the xApp to subscribe to the RAN base stations.
- The RIC Indication message, sent by the base stations the xApp is subscribed to, to report about events or data (e.g., KPM reporting).
- The RIC Control message, used by the xApp to send control actions to the base stations (e.g., to change the scheduling policy, the slicing configuration, etc.).

RAN side, these messages are processed through similar operations by an E2 termination component implemented by the protocol stack of the softwarized base stations. In the case of KPM reporting sent to the xApp via RIC Indication messages, after deserializing it, the SM connector forwards the received KPMs to the *data processing module* of the *data-driven logic unit* of the xApp. In case of control actions produced by the AI/ML model of the data-driven logic unit, instead, these are serialized into RIC Control messages and sent to the base station via the E2 manager/termination of the RIC. In both cases, communications internal to the xApp (i.e., between the SM connector and the data-driven logic unit) can happen in very many different ways, e.g., through sockets, as in the OpenRAN Gym stub xApp. Communications between the xApp and the near-RT RIC, or among xApps, is handled by the RIC routing manager and the RIC Message Router (RMR) protocol [23].

OpenRAN Gym Service Models. OpenRAN Gym aims at facilitating rapid prototyping of new ideas and use cases for closed-loop RAN control. As such, the first release of the OpenRAN Gym skeleton xApp provides a custom SM that is tailored to the swift development of new payloads for E2 indication and control messages, with no need to fully define ASN.1 schemes.[3] This custom SM serializes information on strings, which simplifies the process of adding, removing or customizing the information sent from the RAN to the xApp, or the control messages. This also fits well with the ASN.1 encoding of the underlying E2 message, which simply embeds the string as a sequence of bytes. This, however, requires support at RAN side. OpenRAN Gym xApps have been designed to interface with the near-RT RIC provided by ColO-RAN, and with base stations implemented through SCOPE, a framework using software-defined stacks where new control functionalities can be easily and quickly prototyped. At RAN side, SCOPE presents an E2 termination that collects and organizes multiple metrics from the base station, and converts them to the string that the SM expects. It is also possible to enable the reporting of different sets of metrics by extending the `readMetricsInteractive` method of the `csv_reader.c` file in the E2 termination implementation [24]. In the opposite direction, it is possible to ingest control messages (currently for slicing and scheduling) by extending the `write_control_policies` method of the `srs_connector.c` file [25]. The corresponding interpretation of the metrics and actions is provided by the custom data-driven logic unit, as discussed next.

[3]The development of standard-compliant SMs is on our roadmap.

B. Data-driven Logic Unit

The other main component of the xApp is the *data-driven logic unit*, shown in Figure 2 [26].

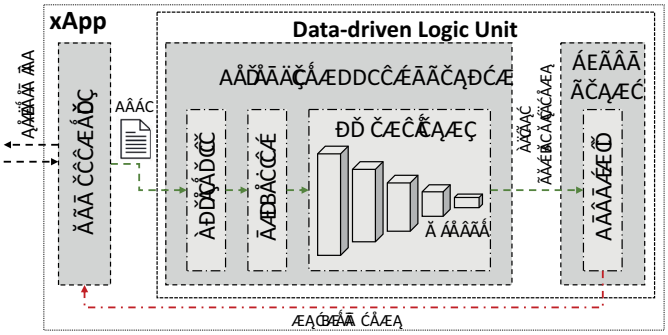


Fig. 2: The data-driven logic unit of the xApp.

The goal of this component is to use data received in near real-time over the E2 interface for online inference via data-driven algorithms, including AI and ML. The unit processes the metric strings received by the SM connector and sends back control commands. As shown in Figure 1, this component consists of the following subunits, namely, the *AI/ML Model* and the *Data Processing Module*. In the following, we review a set of best practices for the design of AI/ML models for RAN control.

AI/ML Model. This subunit hosts the models for prediction, classification and control tasks. Feature and properties relevant to our work are the following:

- *Mandatory offline training*. The O-RAN specifications mandate that any AI/ML solution must be first trained offline, and then validated and tested to avoid inefficiencies and ensure that the trained models do not detrimentally affect the performance and stability of the network [27]. In this context, the AI/ML models are trained by using data lakes storing large amounts of information that has been collected over the O1 interface. Once training is complete, the models are validated and tested in a controlled environment to verify that accuracy levels are high (in the case of classification), predictions are accurate (in the case of forecasting), and control strategies do not result in inefficiencies, or, even worse, outages and unfairness to the subscribers (in the case of control tasks).
- *Online fine-tuning*. Despite mandatory offline training, pre-trained models can still be fine-tuned in an online fashion using online data from the E2 interface. This is

ISBN 978-3-8007-6001-5

especially useful when operators want to tailor the xApp to their specific deployment scenarios. Examples include capturing only network and traffic conditions that affect the deployment area controlled by the xApp. Alternatively, there might be cases where the AI/ML models are deployed and operate under traffic and network conditions never seen during the training phase. In these scenarios, updating the weights of a Deep Reinforcement Learning (DRL) agent or of a neural network could improve the performance of the trained model under current network configurations (Section IV).

- *Chaining AI/ML models*. In many cases, controlling the RAN involves a complex pipeline of several decision-making steps. A practical example is that of two xApps, the first forecasting the evolution over time of one or more time series of KPMs, and the second taking as input the forecast KPMs and making control decisions on some network policy (e.g., on the scheduling policy). In this case, the RMR protocol running at the RIC can be used to support sequential data flows between xApps, thus effectively enabling chains of xApps [23].

Data Processing Module. This submodule is designed to process data received over the E2 interface to meet the input format and representation requirements of the AI/ML models hosted by the xApp. Among others, typical operations performed by this module include:

- *KPM extraction and reshaping*. This operation allows the xApp to feed the hosted AI/ML model with the correct amount and type of information. Specifically, since the KPM stream received from the RAN is continuous—and potentially with a different structure than the one required by the model—this operation makes it possible to extract relevant KPMs of specific size from the E2 stream. The size should match the input size of the AI/ML model. This module also performs data padding in case of missing data (e.g., when only a few data points are available).
- *Scaling*. A well-known issue of many AI/ML-based algorithms is the susceptibility against the values of the input data. While in many computer vision applications the input data is composed of images, where each pixel is typically represented by a 3D tuple with values in the 0-255 range, in cellular networks the input data are KPMs with different physical meaning. Indeed, KPMs might have positive/negative values in very many different ranges. The majority of AI/ML-based algorithms leverages gradient-based methods during the training phase. Although this has been shown to be extremely effective, it also requires proper solutions to avoid biased weight updates, where KPMs with larger values impact the resulting stochastic gradient updates much more than smaller KPMs. A well-established data processing step to avoid such bias consists in scaling the input data so that all KPMs fed to the model assume values in a common and well-defined interval.
- *Data transformation*. In cases where large amounts of data need to be processed, the data processing module can also implement more complex and advanced processing tools. Among others, autoencoders are worth mentioning. These ML tools are commonly used to generate latent representations, and to perform dimensionality reduction, of the input data [13]. These autoencoders typically have an hourglass architecture and consist of two elements, an encoder and a decoder. The former transforms the input data into its latent representation, which usually has substantially smaller dimension than that of the input. The latter, instead, is trained to reconstruct the input data from its latent representation. Ultimately, the goal of autoencoders is to reduce the size of the input while maintaining all of its relevant information. This is extremely useful in facilitating training procedures of DRL agents by reducing the size of the exploration space (i.e., by reducing the number of states that must be explored by the agent) [13].

IV. xApp Use Case

We now provide an example of data-driven closed-loop control through an xApp designed through OpenRAN Gym to jointly control the scheduling and slicing functionalities of the base stations. A schematic representation of our experimental setup is shown in Figure 3.

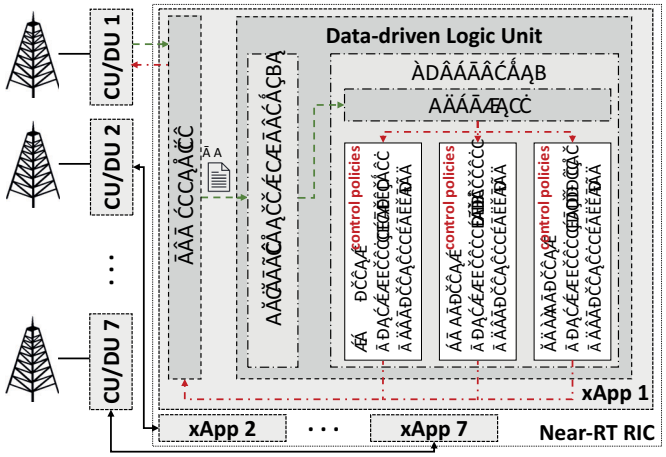


Fig. 3: The OpenRAN Gym experimental setup.

The xApp gets the RAN KPMs periodically through RIC Indication messages. It then feeds them to its data-driven logic unit (Section III-B), and makes control decisions on the scheduling and slicing policies of the base stations. Scheduling control concerns choosing the scheduling policy to run at each slice (among round-robin, waterfilling, and proportionally fair policies). The slicing policy concerns selecting the amount of Physical Resource Blocks (PRBs) allocated to each slice. Both control actions are sent to the base stations via the xApp SM connector by means of RIC Control messages (Section III-A).

We demonstrate this xApp on a softwarized network with 7 base stations and 42 UEs (6 UEs per base station) instantiated

ISBN 978-3-8007-6001-5

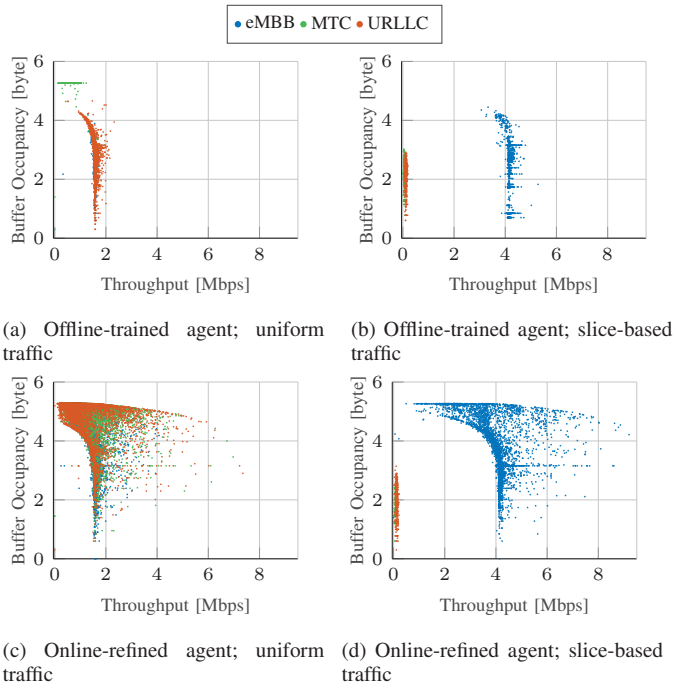
on the Colosseum wireless network emulator [14] through SCOPE. As each base station implements three network slices with diverse service requirements—namely, Enhanced Mobile Broadband (eMBB), Machine-type Communications (MTC), and Ultra Reliable and Low Latency Communications (URLLC) slices—the xApp has been designed to prioritize different metrics for different types of service. Specifically, the data-driven logic unit of the xApp has the goal of maximizing the throughput of the eMBB slice, and the amount of transmitted packets for the MTC slice. Instead, it aims at keeping the occupancy of the transmission buffer queues—used as proxy for latency—at low levels for the URLLC slice. This data-driven logic unit has been trained offline on a dataset of almost 8 GB developed on Colosseum [28]. After the training phase, we instantiated 7 instances of this xApp—one per base station—on the near-RT RIC provided by ColO-RAN, and used them to control the RAN.

After the design phase, we tested the xApp on different classes of traffic: (i) *slice-based traffic*—seen during the training—in which UEs belonging to different slices request different amounts of data (4 Mbps/UE for the eMBB slice, 44.6 kbps/UE for MTC, and 89.3 kbps/UE for URLLC), and (ii) *uniform traffic*—unseen during the training—in which UEs request data at an average rate of 1.5 Mbps. We consider the use-case in which the xApp is used as-is—in which the agent of the data-driven logic unit is used as trained offline, i.e., *offline-trained agent*—and that in which the xApp agent is fine-tuned online, namely, *online-refined agent* case.

Figure 4 shows the correlation between the throughput of the UEs of each slice, and the occupancy of their transmission buffers at the base stations in the above-mentioned traffic cases and with/without online training.

The *offline-trained agent* cases are shown in Figures 4a and 4b, while the *online-refined agent* cases in Figures 4c and 4d. Furthermore, the cases with *uniform traffic* are shown in Figures 4a and 4c, while the *slice-based traffic* in Figures 4b and 4d. By comparing the cases with agents trained offline, we notice that the xApp is able to provide the requested resources to the slice UEs—even in cases unseen in the training—by dynamically changing the configuration of the base stations.

By comparing offline and online cases with the same traffic configuration—i.e., Figures 4a and 4c, and Figures 4b and 4d—we notice that the online training phase is key in providing a superior service to the UEs, whose performance significantly exceed those of the offline-trained xApp. This is because the additional online training phase allows the xApp to adapt to specific deployments, tailoring the agent to the run-time RAN.



(a) Offline-trained agent; uniform traffic

(b) Offline-trained agent; slice-based traffic

(c) Online-refined agent; uniform traffic

(d) Online-refined agent; slice-based traffic

Fig. 4: Correlation between UE throughput and buffer occupancy across multiple slices with offline-trained and online-refined xApps for different classes of traffic: (a, c) uniform traffic (1.5 Mbps/UE), and (b, d) slice-based traffic (4 Mbps/UE for eMBB, 44.6 kbps/UE for MTC, 89.3 kbps/UE for URLLC).

V. Closed-loop Control: Future Directions

The Open RAN provides key building blocks for end-to-end, closed-loop, and automated control of the RAN. Through its embodiment by the O-RAN Alliance, it also represents a practical mechanism for embedding intelligence and AI/ML in the control of the RAN. These intelligent software-based solutions do no need to be statically baked in the network appliances, but operate as plug-ins on standardized, open platforms such as the RICs. Nonetheless, there are still several challenges that must be overcome before getting to efficient, reliable and fully autonomous RAN control and optimization.

The first concerns the state and maturity of the O-RAN specifications on closed-loop RAN control. The Working Group (WG) 3 of the O-RAN Alliance has defined an initial set of service models that operate over the E2 interface. However, further development and additional SMs are required to make RIC and xApps more effective, with control spanning a larger scope than what is currently available. This is no easy feat, as the standardization process crosses multiple domains, most notably, O-RAN and 3GPP. The O-RAN Alliance can only influence the definition of the E2 interface, while the protocol stack is under the 3GPP domain. The current approach of the O-RAN Alliance is that of providing methods to measure, tune, or adapt the values of 3GPP-defined parameters. However, while current SMs enable streaming of a comprehensive set of KPMs from the RAN defined in 3GPP documents [29], the control could be further enhanced. For example, at the time of writing, slicing support is limited in the standard, despite slicing being a key area for closed-loop optimization [4]. Overcoming this challenge would need tighter interaction and collaboration between 3GPP and O-RAN.

The second challenge concerns adoption and easy access to O-RAN implementations. The availability of RAN equipment that supports E2 integration for closed-loop control is still limited, and the situation is further complicated by an ongoing standardization process which does not provide a stable set of

ISBN 978-3-8007-6001-5

features to be implemented. The telecom industry (vendors and operators) should consider adopting more flexible and agile software-driven practices for automated and fast-rolling updates without service disruption. This would allow networks to leverage softwarization and virtualization and to reduce the time-to-market, following the cloud-native paradigms that have transformed the software industry in the last decade.

When it comes to intelligent RAN control, the most compelling challenge concerns access to data and datasets representative of networks with diverse, heterogeneous, and realistic conditions. Data availability is key to training the models to be deployed in the network (Section III). The definition of reference datasets, however, is a much more daunting task in the wireless/RAN domain than, for example, in the field of computer vision, where standardized datasets are the norm for training and comparing different algorithms. OpenRAN Gym, Colosseum, and the PAWR platforms represent a first step toward medium-to-large-scale data collection. However, their datasets capabilities are still a far cry from the scale and diversity that only production environments can offer.

VI. Conclusions

In this paper, we illustrate steps to the design of intelligent solutions for closed-loop control of the cellular Open RAN. We provide details and insights on how to design AI/ML-based solutions, and show how to use the OpenRAN Gym framework to deploy such solutions as xApp and and to test and train them on an O-RAN-compliant near-RT RIC. Our work also showcase sample xApps developed with OpenRAN Gym that can be used to control a softwarized cellular network with 7 base stations and 42 UEs instantiated on the Colosseum testbed. We emphasize their adaptiveness to different RAN deployments and traffic demands. Finally, we discuss future directions and challenges for closed-loop control of the Open RAN.

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] 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!" *IEEE Network Magazine*, May 2022.

[3] A. Garcia-Saavedra and X. Costa-Perez, "O-RAN: Disrupting the Virtualized RAN Ecosystem," *IEEE Communications Standards Magazine*, pp. 1–8, 2021.

[4] 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.

[5] L. Baldesi, F. Restuccia, and T. Melodia, "ChARM: NextG Spectrum Sharing Through Data-Driven Real-Time O-RAN Dynamic Control," in *Proceedings of IEEE INFOCOM*, May 2022.

[6] P. Li, J. Thomas, X. Wang, A. Khalil, A. Ahmad, R. Inacio, S. Kapoor, A. Parekh, A. Doufexi, A. Shojaeifard *et al.*, "RLOps: Development Life-cycle of Reinforcement Learning Aided Open RAN," *arXiv preprint arXiv:2111.06978*, 2021.

[7] 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.

[8] 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.

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

[10] O-RAN Global PlugFest 2021 Demonstrates Stronger Ecosystem and Maturing Solutions. (2021, December) https://tinyurl.com/ycxdtbzv.

[11] X. Wang, J. D. Thomas, R. J. Piechocki, S. Kapoor, R. Santos-Rodríguez, and A. Parekh, "Self-play Learning Strategies for Resource Assignment in Open-RAN Networks," *Computer Networks*, vol. 206, pp. 1–11, April 2022.

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

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

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] Platforms for Advanced Wireless Research (PAWR). https://www.advancedwireless.org. Accessed June 2022.

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

[20] S. D'Oro, M. Polese, L. Bonati, H. Cheng, and T. Melodia, "dApps: Distributed Applications for Real-time Inference and Control in O-RAN," *IEEE Communications Magazine*, pp. 1–7, 2022, in print; preprint available at https://arxiv.org/pdf/2203.02370.pdf.

[21] 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.

[22] ColO-RAN Near-RT RIC Service Model Connector. https://github.com/wineslab/colosseum-near-rt-ric/tree/e41cd25e500c527ee60fd8095bb6f40e96b7ccce/setup/xapp-sm-connector/. Accessed June 2022.

[23] O-RAN Software Community. RIC Message Router Documentation. https://docs.o-ran-sc.org/projects/o-ran-sc-ric-plt-lib-rmr. Accessed June 2022.

[24] SCOPE E2 Termination CSV Reader Source File. https://github.com/wineslab/colosseum-scope-e2/blob/f0457b39e03cadb2a93523bc73ccc2d056f2188b/src/du_app/csv_reader.c. Accessed June 2022.

[25] SCOPE E2 Termination Connector Source File. https://github.com/wineslab/colosseum-scope-e2/blob/f0457b39e03cadb2a93523bc73ccc2d056f2188b/src/du_app/srs_connector.c. Accessed June 2022.

[26] ColO-RAN Sample xApp Skeleton. https://github.com/wineslab/colosseum-near-rt-ric/tree/e41cd25e500c527ee60fd8095bb6f40e96b7ccce/setup/sample-xapp. Accessed June 2022.

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

[28] Colosseum O-RAN ColORAN Dataset. https://github.com/wineslab/colosseum-oran-coloran-dataset.

[29] O-RAN Working Group 3, "O-RAN near-real-time RAN intelligent controller E2 service model (E2SM) KPM 2.0," ORAN-WG3.E2SM-KPM-v02.00 Technical Specification, July 2021.