AI/ML Data-driven Control Loop for Managing O-RAN SDR-based RANs

Jaswanth S. R. Mallu, Joao F. Santos, Aloizio P. da Silva, Prateek Sethi, Vikas Radhakrishnan, Luiz DaSilva
Commonwealth Cyber Initiative, Virginia Tech
{jaswanthsaireddy, joaosantos, aloiziops, prateek20, vikaskrishnan, ldasilva}@vt.edu

***Abstract*—Open Radio Access Network (O-RAN) introduced a common control and management overlay, allowing mobile network operators to embed networking intelligence using different types of third-party applications: xApps for real-time control loops, and rApps for Artificial Intelligence (AI)/Machine Learning (ML)-based classification and decision-making. However, the development of reference implementations for rApps lags behind the progress in other O-RAN-related standardization efforts. In this demonstration, we showcase a proof-of-concept rApp capable of generating policies to steer the behavior of xApps, and detail how we extended a RAN slicing xApp to react to such policies, creating the first experimental ML-based RAN slicing platform based on O-RAN.**

***Index Terms*—Open Radio Access Network, rApp, xApp, RAN Intelligent Controller, RAN Orchestration**

I. Introduction

The Open Radio Access Network (O-RAN) Alliance introduced a paradigm shift that includes the disaggregation of RAN components, the softwarization of RAN functionality, and the openness of RAN control interfaces and management Application Programming Interfaces (APIs) [1]. One of the key features of the O-RAN ecosystem is its common control and management overlay, allowing mobile network operators to orchestrate their entire RAN using custom control logic via standardized third-party applications [2]. This common control and management overlay is functionally split into two RAN Intelligent Controllers (RICs), according to the timescale of their operation. The Near Real-Time RAN Intelligent Controller (Near-RT RIC) hosts time-sensitive applications, known as xApps, that perform closed-loop control over RAN components in the order of 10-–1000 ms, e.g., load balancing and scheduling [3]. The Non Real-Time RAN Intelligent Controller (Non-RT RIC) hosts applications known as rApps, with a lasting effect based on historical data trends, taking longer than 1000 ms, e.g., data analytics and Artificial Intelligence (AI)/Machine Learning (ML) model training to optimize RAN parameters [4].

Both xApps and rApps are cloud-based microservices, discrete and modular software components that provide specific functionality [1]. While of complementary nature and conceptually similar, the standardization and prototyping of xApps and rApps are moving at very different paces. The O-RAN specifications that standardize xApps and their interfaces are mature and accompanied by open-source reference implementations that guide the development of new xApps by academia and industry. The availability of these resources has led to a number of research efforts that design, simulate, and experimentally verify xApp operation in practical settings [4]. Conversely, the standardization of rApps, their interfaces, and their generation of policies to interact with xApps lag significantly behind. Consequently, research efforts on rApps (and their interaction with xApps) have been primarily limited to theoretical works that model the performance or simulate their operation. To the best of our knowledge, there are no existing open-source prototypes of rApps able to interface with the Near-RT RIC to control the behaviour of the RAN.

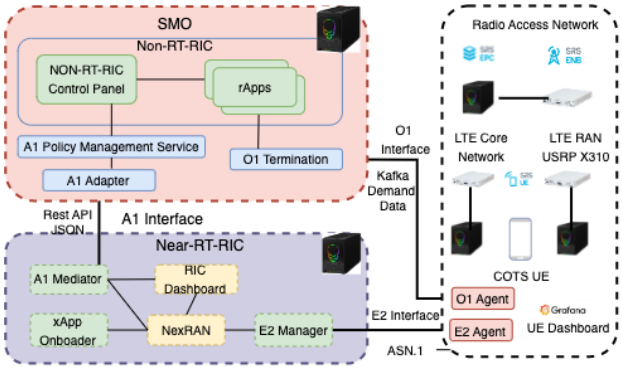


Fig. 1: Architectural components of our data-driven control loop for managing O-RAN SDR-based RANs.

In this demo, we showcase the first proof-of-concept rApp capable of generating policies and controlling the behavior of xApps, and its integration with the NexRAN RAN slicing xApp described in [3], creating the first experimental ML-based RAN slicing platform based on O-RAN. Our platform allows us to experiment with different ML-based strategies for RAN slicing that leverage information about user demand, evaluate their overall performance in over-the-air scenarios using Software Defined Radios (SDRs) and open-source radio stacks, and demonstrate the autonomous creation and adaptations of RAN slices in real-time.

II. Demo Design and Architecture

The system design of our demonstration is illustrated in Figure 1. It includes three layers (a) Service Management and Orchestration (SMO) and Non-RT RIC; (b) Near-RT RIC; and (c) RAN. All layers are based on the O-RAN Software Community E-Release. To integrate these layers [4], we enable the key open interfaces specified by O-RAN Alliance, namely the A1, E2, and O1 interfaces. In particular, we have implemented the O1 interface using Kafka. We describe each of these layers next.

Fig. 2: Demo setup of our AI/ML data-driven control loop.

A) SMO and Non-RT RIC: They are responsible for the orchestration of applications and control loop-based operations with reaction time greater than $1s$. This layer is deployed as a single Kubernetes cluster using a small-factor computer (Intel NUC). Our rApp implementing AI/ML-based decision-making is responsible for policy selection and its logic includes policy-based RAN slicing that uses two policies to allocate resources based on the presence of priority users in the network. The policies are enforced by the xApp and rApp, which autonomously determine the proportion of resources to allocate to each slice based on information from the RAN.

B) Near-RT RIC: It is responsible for near-real-time control and optimization of the RAN resources by leveraging data collection and RAN actions over the E2 interface. The Near-RT RIC interfaces with the Non-RT RIC over the HTTP-based A1 interface to obtain fine-grained control information in the form of policies that steer the optimization goals of xApps. For this demo, our Near-RT RIC hosts the NexRAN xApp [3], which performs closed-loop RAN slicing by changing the proportions of subframes available to each user and collecting metrics about their performance.

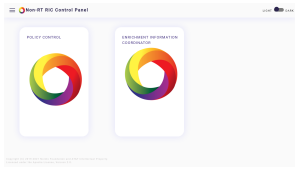
C) RAN: We created a 4G RAN using srsRAN[1], an open-source radio stack, and three USRP X310 SDRs: one base station, and two User Equipments (UEs) (representing general users). Each SDR is connected to a small-factor computer (Intel NUC) for baseband processing, and the computer running the base station also hosts the 4G core network. In addition, we used one Samsung S21, Commercial Off-The-Shelf (COTS) UE (representing a priority user).

III. Demonstration

We demonstrate, for the first time, an end-to-end SDR-based O-RAN closed-loop system that consists of Non-RT RIC, Near-RT RIC, and RAN. This demo shows the interaction between rApp and xApp, highlighting the AI/ML capability of dynamically modifying the RAN's behavior. Figure 2 shows the demo setup and its main components.

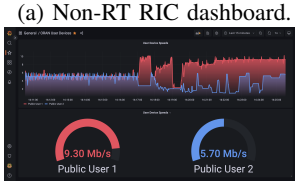
The demo begins with collecting demand data from the UEs and sending this data to the rApp as input via the O1 interface. The rApp generates policies using an ML model trained using the input from the users, and these policies are sent to the xApp. The RAN changes its behavior based on input from the xApp. To illustrate the closed-loop

[1]https://www.srsran.com/

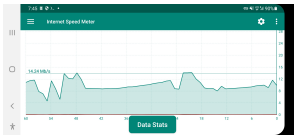


(a) Non-RT RIC dashboard.

(b) Near-RT RIC dashboard.



(c) General UE dashboard.



(d) Priority UE dashboard.

Fig. 3: Dashboards shown during our end-to-end demo.

experiment, we choose a use case where RAN resources are dynamically allocated by the rApp according to the type of user. In Figure 1, two SDRs represent general commercial subscribers, and the COTS UE represents a high-priority user, such as a first-responder.

The traffic demand from each user is generated using *Iperf* and sent to the SMO via the O1 interface. Once the SMO receives the demand from the UEs, it is forwarded to the rApp in the Non-RT RIC. Our rApp performs a regression algorithm and a trained machine-learning model that determine what resources are to be allocated to each user. The rApp autonomously selects the most appropriate policy to be enforced by the NexRAN [3] xApp. The NexRAN xApp receives policies from the A1 mediator and, through the E2 interface, applies them in the RAN.

The end-to-end demo workflow can be visualized through four main dashboards as shown in Figures 2 and 3. On the left is the Non-RT RIC dashboard (Fig. 3a) that allows us to visualize the rApp behavior and the selected policy. In the middle is the Near-RT RIC dashboard (Fig. 3b) that shows the NexRAN xApp behavior after receiving the policy from the rApp via the A1 interface. It also shows relevant information about UEs, e.g., the UE's International Mobile Subscriber Identity (IMSI), and E2 node, e.g., node type and the number of slices. On the right, the UE dashboard (Fig. 3c) depicts the throughput achieved by general subscribers. Finally, below the three previous dashboards, we have the high-priority user dashboard (Fig. 3d), displaying its throughput. xApp.

Acknowledgements

This work received support from the Commonwealth Cyber Initiative. For more information: www.cyberinitiative.org.

References

[1] M. Polese *et al.*, "Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges," *arXiv preprint arXiv:2202.01032*, 2022.

[2] S. Niknam *et al.*, "Intelligent O-RAN for beyond 5G and 6G wireless networks," *arXiv preprint arXiv:2005.08374*, 2020.

[3] D. Johnson *et al.*, "NexRAN: Closed-loop RAN Slicing in POWDER-A top-to-bottom Open-source Open-RAN Use Case," in *ACM Workshop on Wireless Network Testbeds, Experimental evaluation & CHaracterization (WiNTECH)*, 2022, pp. 17–23.

[4] S. D'Oro *et al.*, "OrchestRAN: Network Automation through Orchestrated Intelligence in the Open RAN," in *IEEE Conference on Computer Communications (INFOCOM)*, 2022, pp. 270–279.