2023 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN) | 979-8-3503-0254-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/NFV-SDN59219.2023.10329597

An Open RAN Framework for the Dynamic Control of 5G Service Level Agreements

Eugenio Moro*, Michele Polese‡, Antonio Capone*, Tommaso Melodia‡

*Department of Electronics, Information and Bioengineering, Polytechnic University of Milan, Italy
‡Institute for the Wireless Internet of Things, Northeastern University, Boston, MA, U.S.A
*{eugenio.moro, antonio.capone}@polimi.it, ‡{m.polese, t.melodia}@northeastern.edu

***Abstract*—The heterogeneity of use cases that next-generation wireless systems need to support calls for flexible and programmable networks that can autonomously adapt to the application requirements. Specifically, traffic flows that support critical applications (e.g., vehicular control or safety communications) often come with a requirement in terms of guaranteed performance. At the same time, others are more elastic and can adapt to the resources made available by the network (e.g., video streaming). To this end, the Open Radio Access Network (RAN) paradigm is seen as an enabler of dynamic control and adaptation of the protocol stack of 3rd Generation Partnership Project (3GPP) networks in the 5th generation (5G) and beyond. Through its embodiment in the O-RAN Alliance specifications, it introduces the RAN Intelligent Controllers (RICs), which enable closed-loop control, leveraging a rich set of RAN Key Performance Measurements (KPMs) to build a representation of the network and enforcing dynamic control through the configuration of 3GPP-defined stack parameters. In this paper, we leverage the Open RAN closed-loop control capabilities to design, implement, and evaluate multiple data-driven and dynamic Service Level Agreement (SLA) enforcement policies, capable of adapting the RAN semi-persistent scheduling patterns to match users' requirements. To do so, we implement semi-persistent scheduling capabilities in the OpenAirInterface (OAI) 5G stack, as well as an easily extensible and customizable version of the Open RAN E2 interface that connects the OAI base stations to the near-real-time RIC. We deploy and test our framework on Colosseum, a large-scale hardware-in-the-loop channel emulator. Results confirm the effectiveness of the proposed Open RAN-based solution in managing SLA in near-real-time.**

I. Introduction

As wireless networks improve toward ultra-high data rates, low latency, and high reliability, they also become essential to countless applications and use cases in our digital society. In a trend that started with the 5th generation (5G) of mobile networks and is continuing toward the 6th generation (6G), the same air interface is used to serve extremely heterogeneous traffic patterns, with dynamic constraints and traffic loads [1]. As an example, cellular Radio Access Networks (RANs) are designed to support Enhanced Mobile Broadband (eMBB) applications with an over-the-air frame structure, waveform, and protocol stack tailored to long-running data-rate-hungry streams (e.g., video streaming), but also more bursty traffic with low latency constraints or Machine-type Communications (MTC) communications, thanks to configurations of the frame structure that favor short over-the-air bursts [2]. Similarly, elastic applications can use any amount of resources made available by the system, while other traffic flows require tight Service Level Agreement (SLA) and performance guarantees.

However, while the capabilities to support different traffic requirements are part of the technical specifications for 3GPP NR, the RAN for 5G systems, the actual commercial implementations lack the capability to dynamically and optimally switch between such configurations and to adapt to the actual user patterns and demand on the fly. This does not allow for efficient exploitation of the scarce spectrum resources available to wireless networks, and leads to a mismatch between user expectations and achievable performance [3]. A typical example is represented by the significant body of literature on how to optimally select waveform parameters [4], [5] and enforce SLA constraints [6]–[8] in cellular systems, which however is often not deployable in real scenarios and RAN stacks due to their inflexibility and limited adaptability.

The recent paradigm shift introduced by the Open RAN vision is implementing practical primitives for the dynamic adaptation and optimization of the RAN configurations, enabling the adoption of more flexible solutions for the support of different traffic requirements [9]. Specifically, Open RAN introduces new components, the RAN Intelligent Controllers (RICs), which interact with 3GPP-compliant base stations through open interfaces, and have the capability to (i) receive telemetry and Key Performance Measurements (KPMs) from the RAN; (ii) infer the status of the system using data-driven approaches; and (iii) apply new configurations to the radio resource management process and adapt the RAN behavior to the actual conditions on the ground [10]. The Open RAN vision, and its embodiment in the O-RAN Alliance specifications, include two RICs for near-real-time (10 ms—1 s) and non-real-time (more than 1 s) closed-loop control, implemented through xApps and rApps, respectively.

In this paper, we build on the Open RAN vision and identify and compare two control strategies that an operator can use to enforce SLA for non-elastic traffic, to be deployed as custom control logic in the near-real-time RIC. The strategies build on 3GPP- and O-RAN-compliant parameters that allow specifying semi-persistent scheduling patterns in 5G base

This paper was partially supported by the NSF under Grant CNS-2117814.

stations, and thus they do not require modifications in protocol stacks that comply with technical specifications. Specifically, the two SLA management solutions embody either a *strict* or a *soft* SLA enforcement policy, which can be selected according to the network operator objectives.

The second contribution of this paper is an implementation of the experimental infrastructure required to prototype, test, and evaluate such control logic in an end-to-end, programmable framework for the design of custom Open RAN closed-loop control. Specifically, we extend the OpenRAN Gym platform, first introduced in [11], to connect the open-source OpenAirInterface (OAI) 5G RAN implementation [12] to a near-real-time RIC based on the O-RAN Software Community distribution [13]. This combines a state-of-the-art Open RAN platform with the feature-rich and standard-compliant 5G OAI implementation. Our implementation of the E2 interface connecting the RIC to the RAN is designed to be easily extensible, thanks to an abstraction of its functionalities based on human-readable data structures that can be automatically compiled into serializable buffers. We also extend the OAI stack to support the semi-persistent scheduling control and integrate it with our E2 implementation.

Finally, we profile the performance of the combined infrastructure and control logic in Colosseum, the world's largest wireless network emulator with hardware in the loop. We show how the entire framework, comprising the RAN nodes, the near-RT RIC and the xApps, is highly effective at controlling the SLA enforcement at the timescale of 100 ms.

The rest of the paper is organized as follows. Section II presents an overview of the Open RAN (O-RAN) paradigm and its associated architecture. Section III details the modifications to OpenAirInterface (OAI), the xApp Software Development Kit (SDK) and their integration in OpenRAN Gym as a complete experimental framework. Section IV details the two SLA policies and their implementation as O-RAN control micro-services. Finally, Section V presents the numerical evaluation of the entire framework on a large-scale channel emulator with hardware in the loop.

II. Open RAN - A Primer

The Open RAN paradigm is implemented by the O-RAN Alliance, an industry and academic consortium with more than 300 members, in the architecture shown in Fig. 1. In this, the RAN Next Generation Node Bases (gNBs) are disaggregated and split into multiple nodes with different functionalities based on the layers of the protocol stack they host. The Service Data Adaptation Protocol (SDAP), Packet Data Convergence Protocol (PDCP), and Radio Resource Control (RRC), i.e., the higher layers for the User Plane (UP) and the Control Plane (CP), are in the Central Unit (CU)-UP and CU-CP, respectively. The Distributed Unit (DU) hosts three layers that operate in a tightly synchronous fashion, the Radio Link Control (RLC), the Medium Access Control (MAC), and the higher part of the physical layer. Finally, the Radio Unit (RU) features the lower part of the physical layer and the Radio Frequency (RF) frontend. These nodes are connected to each

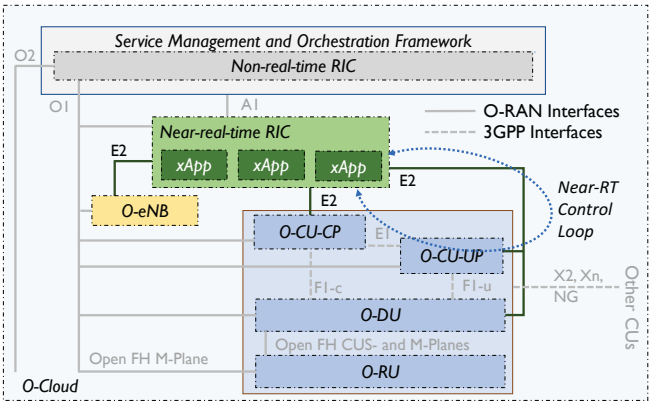


Fig. 1: Logical view of the O-RAN architecture, adapted from [10]. The focus is on the E2 interface, connecting the near-RT RIC and the RAN nodes for the near-RT control applications discussed in this paper.

other using 3GPP-defined interfaces and the Open Fronthaul interface from the O-RAN Alliance.

In addition, they are connected through the E2 and the O1 interfaces to the near-RT and non-RT RICs, respectively. These intelligent controllers can onboard plug-and-play control logic (i.e., xApps and rApps) to extend the functionalities of RAN nodes with custom control loops. Specifically, the non-RT RIC, embedded in the Service Management and Orchestration (SMO), relies on rApps to perform policy and control updates in non-real time, i.e., with a time granularity higher than 1 s. This is to dynamically update configurations in the RAN nodes and control high-level policies and parameters, e.g., cell sleeping patterns in the CU/DU and beam codebooks at the RU. The near-RT RIC, instead, uses xApps to implement control loops that execute in less than 1 s, but more than 10 ms. The near-RT RIC closes the control loop to perform radio resource management in the DU and CU. Consequently, it operates on parameters that need to be updated with more stringent deadlines compared to the non-RT control loop.

The near-RT RIC relies on the E2 interface: a logical point-to-point interface running on top of SCTP. The E2 interface features two components. The first is an application protocol, or E2AP, which manages the connection between each DU/CU (also known as E2 nodes) and the near-RT RIC, including setup, monitoring, and teardown. The E2AP offers a set of primitives (e.g., indication, report, and control messages) assembled to build custom E2 *service models*, or E2SMs. The E2SM is a component implemented on top of E2AP which provides the semantics to the E2 interface, e.g., reporting of KPMs from the RAN with E2SM KPM, or control of RAN parameters with E2SM RAN Control (RC). Specifically, RC has been designed to interact with protocol stack parameters defined in 3GPP specifications, introducing an elegant solution to decouple control from the actual RAN implementation.

III. A Framework for 5G Open RAN Closed-Loop Control

This section discusses how we implemented an easily extensible E2 interface for the OAI 5G protocol stack and

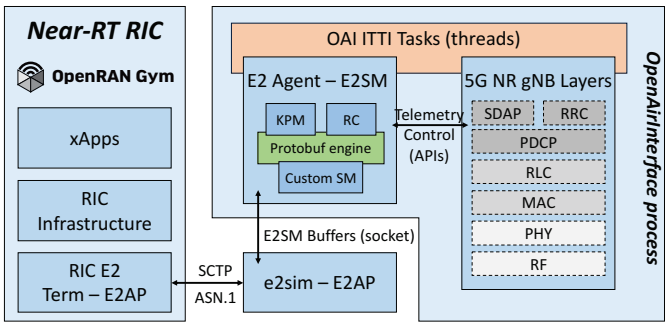


Fig. 2: Integration of the E2 interface in OAI.

a companion xApp SDK which leverages O-RAN-compliant E2AP and custom E2SMs. Both are open-source and publicly available,[1] and enable design and testing the dynamic SLA policies we propose and evaluate in this paper.

A. Integrating OpenRAN Gym and OpenAirInterface

OAI is an open-source project that provides the implementation of a 3rd Generation Partnership Project (3GPP) Release 15 NR RAN and 5G Core [12]. The RAN base stations can be deployed on generic compute hardware and leverage software-defined radios via multiple drivers, or O-RAN RUs with a 7.2x fronthaul implementation. Being open, programmable, and easily extensible, OAI can be leveraged to implement and test O-RAN closed-loop control in a 5G-compliant end-to-end environment.

OpenRAN Gym represents the first publicly accessible platform that features O-RAN-based data collection and control frameworks for data-driven experimentation at scale [11]. In particular, OpenRAN Gym frameworks package the entire software chain required to deploy the O-RAN components presented in Fig. 1, namely the base stations, the near-RT RIC, and the xApps. The RIC is based on the O-RAN Software Community (OSC) RIC implementation, which has been adapted for deployment on a variety of open testbeds, including Colosseum (Sec. V) and the Platforms for Advanced Wireless Research (PAWR) testbeds COSMOS and POWDER.

Figure 2 illustrates how we designed and implemented the integration between OAI and the OpenRAN Gym framework through the E2 interface, enabling experiments with a 3GPP-compliant, O-RAN-enabled 5G standalone (SA) deployment. As discussed in Sec. II, the E2 interface is functionally split into 2 sub-protocols: E2AP and E2SM. Similarly, our custom E2 agent has been split into 2 components, according to the principle of separation of responsibilities. The E2AP component runs as a standalone application, and it is based on E2AP libraries extracted from OSC E2 simulator library (`e2sim`) [14]. This component manages the E2AP connection lifecycle with the near-RT RIC. It is standard-compliant, i.e., it encodes and decodes E2AP messages based on O-RAN ASN.1 definitions and an SCTP transport layer.

The component that provides E2SM functionalities is integrated into OAI codebase and runs as a task inside the gNB process, similarly to the implementation in [15], [16]. Consequently, the E2 agent implementing the E2SM has direct access to the gNB data structures and processes and can effectively perform data collection and apply control actions by directly interacting with the variables that define the relevant 3GPP parameters to tune.

The two components run as independent processes in the same machine (potentially providing a resilient E2 interface in case of failures in the gNB), and they communicate through UDP sockets. When the E2AP component receives an E2AP message from the near-RT RIC, the E2SM payload is extracted, decoded according to the ASN.1 schema, and sent to the OAI E2 agent. Here the E2SM payload is further decoded and processed. Similarly, any E2SM payload produced by the E2SM component in the gNB is sent to `e2sim`, which handles E2AP encapsulation and message delivery.

In this architecture, the E2SM payload is a buffer of bytes without additional specifications or requirements. Therefore, it can be an O-RAN-compliant E2SM ASN.1-encoded payload, a custom unstructured string, or—as we propose in this paper—a protobuf buffer. Protobuf, or Protocol Buffers, is a data serialization format developed by Google to efficiently exchange structured data between different systems [17]. It uses a language-agnostic schema to define the data structures, which can then be compiled into specific implementations in different programming languages (e.g., C for the OAI E2 agent and Python for the xApp described in Sec. III-B). Compared to ASN.1, protobuf data structures are more user-friendly to extend, compile, and integrate into the code, making it a practical tool for developing custom E2 service models for research and testing.

B. A Python Framework for Flexible RAN Control xApps

To streamline and simplify the xApp development process, we developed and made publicly available[2] an xApp SDK in Python, a popular programming language which is also used in several state-of-the-art frameworks for data-driven inference [18]. Figure 3 illustrates at a high level the structure of the xApp and its SDK, where the xApp logic leverages the SDK for operations related to data encoding/decoding and interaction with the rest of the RIC platform.

We use the OSC Python xApp framework as a library to wrap primitives for the communication with the other internal RIC components, i.e., the E2 termination to the RAN nodes, the RIC Message Router (RMR), which dispatches internal messages across xApps and the RIC infrastructure, and the Shared Data Layer (SDL), containing multiple data repositories with information on the E2 nodes (e.g., the RAN Network Information Base (NIB) (R-NIB)). The xApp SDK wraps the code required to interact with these components into functions that the developer can easily embed in their xApp logic. Some examples are shown in Fig. 3. The `get_gnb_id_list()`

[1]Refer to the openrangym.com website for links to code and tutorials.

[2]https://github.com/ANTLab-polimi/xapp-e2ap-py

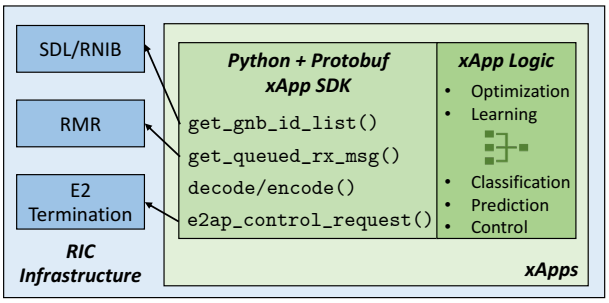


Fig. 3: Python framework for flexible xApp development.

Application Programming Interface (API) retrieves the list of RAN nodes connected over E2, which can be either leveraged to retrieve KPMs or as targets for the control and optimization; `get_queued_rx_msg()` is a non-blocking method that parses the queue of incoming E2 messages for the xApp; and `e2ap_control_request()` sends a message to the E2 termination with a control request in its buffer.

In addition, as discussed in Sec. III-A, the xApp natively embeds the APIs to create, encode, and decode messages following the protobuf format. The same definition can be added to the xApp and the E2 agent in the RAN and properly compiled in the desired programming language.

For this paper, we implement a service model equivalent to E2SM RC to control and optimize the semi-persistent scheduling patterns in the gNB. Further details are provided in Sec. IV, where we describe the optimization policies and their enforcement through the RC-based service model.

IV. Open RAN Control — SLA Enforcement Policies and Control

This section describes the two SLA enforcement strategies we implement on top of the Open RAN framework described in Sec. III, leveraging the implementation of an E2SM based on RC for Guaranteed Bitrate (GBR) control in OAI (Sec. IV-A) and a data-driven optimization framework implemented the SLA xApp (Sec. IV-B).

A. RC for GBR in OAI

The SLA xApp leverages data collection and control primitives to provide a guaranteed bitrate to the User Equipments (UEs). The xApp needs information on the cell resource utilization and the per-UE channel quality, and it needs to inform the gNB MAC layer about how many resources should be allocated to each UE. This is achieved through E2 service models.

The O-RAN RC and KPM service models provide primitives to query telemetry from the gNB. Specifically, RC provides the Transport Block Size (TBS) information, namely the bits that are exchanged between the gNB and each UE in a Transmission Time Interval (TTI). This information can be used in the xApp to estimate the UE MAC throughput with extreme precision. In 5G NR, the gNB resources are partitioned in Physical Resource Blocks (PRBs), i.e., the minimum unit of spectrum and time resources that can be allocated to UEs at each TTI. Baseides the TBS information in RC, KPM exposes the PRB allocation information at the UE level, which the xApp requires to know the gNB resource distribution.

By combining the MAC throughput and PRB allocation information, the xApp can compute the *per-PRB* throughput for each UE. This measure is a proxy for the UE spectral efficiency, and it can be used to inform the allocation decisions behind any SLA management policy.

Once the GBR allocation decision is taken in the xApp (with the policies discussed in Sec. IV-B), it must be enforced in the gNB. To this end, we leverage the E2SM RC capabilities of controlling the Semi-Persistent Scheduling (SPS) process. In 5G NR, SPS is a mechanism that allows the gNB to schedule parts of the resources on a fixed and persistent basis, as opposed to the more traditional dynamic scheduler where resources are granted based on traffic conditions. By properly configuring SPS, the xApp can reserve the fixed resource portion required by each UE to obtain the desired GBR. Note that this is an NR-compliant feature, thus, by controlling this with E2SM RC, it is possible to practically implement dynamic GBR policies for SLA enforcement in 5G gNBs.

The OAI 5G MAC implementation does not support SPS, neither it supports the necessary APIs to control it through RC. As previously mentioned, we have implemented a lightweight RC service model that exposes the required data collection and control knobs into OAI. In particular, the PRB allocation and TBS information are collected inside OAI MAC implementation by adding the required hooks in both the downlink and uplink schedulers. Additionally, we provide SPS support in OAI by modifying the aforementioned schedulers so that the TBS of any UE can be fixed through our custom RC control messages, at any time. This effectively results in a fixed per-UE resource allocation, as one would have with SPS.

B. Dynamic SLA Enforcement Strategies

We leverage the dynamic closed-loop control framework provided by O-RAN to design, implement, and evaluate two SLA enforcement strategies that dynamically manage radio resources with the goal of providing a guaranteed bitrate to UEs and data flows that require it. Through the data collection capabilities of the custom RC service model, the xApp can enforce allocations for a GBR, as long as enough resources are available. Whenever this is not true, the system cannot support all the GBR requests, and the resources must be managed according to specific resource contention resolution policies to minimize the SLA violation. For a specific UE u, we define its SLA violation as the difference v_u between the throughput SLA_u requested as the guaranteed GBR value and the experienced throughput $p_u \eta_u$, with p_u amount of allocated PRBs and η_u the per-PRB throughput.

We propose and evaluate two policies. The first is a flexible SLA enforcement (policy *Soft*). When resource contention resolution is required, the xApp dynamically allocates the available PRBs such that the overall sum of the per-UE

SLA violation v_u is minimized. This policy can be expressed through the following linear program:

$$\min \sum_{u \in \mathcal{U}} v_u, \tag{1}$$

$$\sum_{u \in \mathcal{U}} p_u \leq C, \tag{2}$$

$$v_u \geq \text{SLA}_u - p_u \eta_u \qquad \forall u \in \mathcal{U}, \tag{3}$$

$$v_u, p_u \geq 0 \qquad \forall u \in \mathcal{U}. \tag{4}$$

Here $\mathcal{U}$ represents the set of GBR UEs associated with the gNB, and $v_u, \text{SLA}_u, p_u, \eta_u$ are defined as above. Constraint (2) limits the overall allocated PRBs to the maximum amount C of PRBs available to the gNB.

The second policy is based on a more rigid GBR enforcement (policy *Strict*). According to this policy, each UE u is assigned with a *weight* w_u representing priority, economic value, or other importance metrics. In case of resource contention, the policy selects a subset of UEs to continue to serve with GBR such that the overall weight (i.e., the sum of selected UEs weights) is maximized. The UEs outside the optimal subset are either given the remaining resources or disconnected from the gNB. This policy can be mathematically expressed through a knapsack formulation [19] using the previous notation, as follows:

$$\max \sum_{u \in \mathcal{U}} x_u w_u, \tag{5}$$

$$\sum_{u \in \mathcal{U}} x_u c_u \leq C, \tag{6}$$

$$x_u \in \{0, 1\} \qquad \forall u \in \mathcal{U}. \tag{7}$$

Here x_u is a binary variable indicating whether the UE is inside the GBR-enforced subset or not. Parameter c_u represents the gNB resources required to guarantee the SLA for UE u, considered in the capacity constraint (6).

V. Experimental Evaluation

This section describes the experimental setup and evaluation of the dynamic SLA policies on Colosseum and OAI.

A. The Colosseum Testbed

We evaluated the solutions discussed in Sec. IV—together with the xApp-based control framework for OAI—on Colosseum, the world's largest wireless network emulator with hardware-in-the-loop [20]. The Colosseum testbed provides users with access to 128 pairs of servers and USRP X310 (collectively defined as Standard Radio Nodes (SRNs)). The server can load an LXC container with a custom image provided by the user (e.g., the OpenRAN Gym near-RT RIC, the OAI gNB, and OAI UE). The radio is connected to Colosseum's Massive Channel Emulator (MCHEM), which implements virtual RF scenarios with path loss, fading, and interference by filtering the transmitted signal from the SRN radio with the channel impulse response. MCHEM leverages a bank of 64 Field Programmable Gate Arrays (FPGAs) and can emulate channels with up to 4 multi path components and pre-defined mobility for the nodes.

Colosseum has already been widely used to evaluate custom control logic for Open RAN systems through OpenRAN Gym. Here, we deploy a network with 6 nodes, including a 5G Core Network (CN) node, one gNB, 3 UEs, and a near-RT RIC. The RF scenario emulates a typical laboratory testing environment, with a fixed pathloss among the RAN nodes. The gNB is configured for transmitting at 3.6 GHz in band n78, with a bandwidth of 40 MHz and numerology 1. *iperf* generates full-buffer downlink TCP traffic. We limit our analysis to the downlink only case. The xApp is configured to adjust the resource allocation every 100 ms.

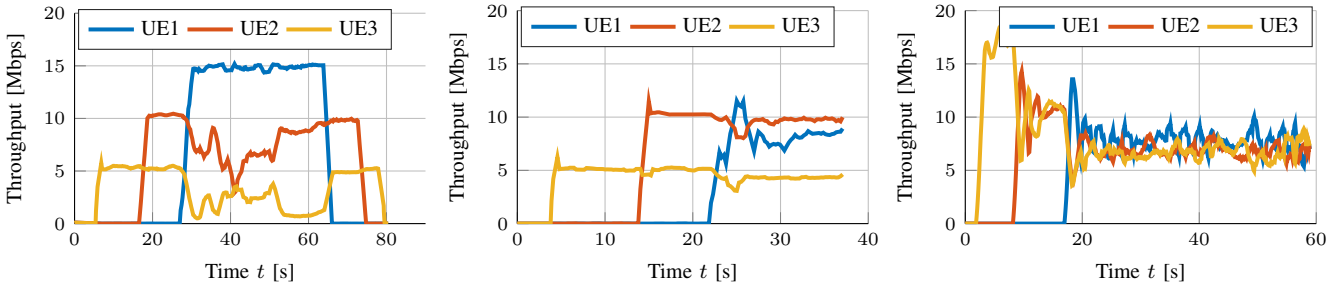
B. Results

We configure the three UEs with different GBR values, i.e., 15 Mbps for UE1, 10 Mbps for UE2, and 5 Mbps for UE3. The gNB can allocate 65 PRBs, which are not sufficient to satisfy SLA for all the UEs. We sequentially generate full-buffer traffic (starting from UE1, to UE2, and then UE3) and analyze the system performance. During the experiments, the xApp actively enforces a policy described in Sec. IV. Additionally, we have included a baseline in which the xApp is inactive and the UEs allocations follow a proportional fairness scheduler. In every case, we measure the experienced throughput and the SLA violation, i.e., the difference between the nominal GBR value and the actual throughput.

We start by analyzing the system behaviour when policy *Soft* is active (Fig. 4a). At the beginning of the experiment, only UE3 is receiving traffic and its SLA is met. UE2 becomes active at time $t = 19$ s, but the resources are still sufficient to meet both SLAs. When UE1 becomes active, it requests resources for 15 Mbps, exceeding the available PRBs and activating the contention resolution mechanism of the xApp. In this case, the optimization formulation of policy *Soft* selects UE1 to obtain the full GBR, while UE2 and UE3 obtain less-than-required resources, but the overall throughput degradation is limited, following the goal of policy *Soft* (minimize the overall SLA violation, Eq. (1)). This is also confirmed by Fig. 5, which reports the per-UE and total SLA violation and shows how policy *Soft* has a smaller overall violation with respect to policy *Strict* and the baseline, at the expense of higher per-UE violation for UE2 and UE3.

Similarly, for policy *Strict* (Fig. 4b), the xApp contention resolution mechanism is required at time $t = 21$ s, when UE1 starts exchanging traffic. In this case, however, policy *Strict* results in UE2 and UE3 being served their full GBR, while the remaining gNB resources are left to UE1. This is expected since all the UEs are set with the same weight and, thus, the xApp simply maximizes the number of satisfied UEs. As shown in Fig. 5, for policy *Strict*, most of the overall violation comes from UE3, while UE1 and UE2 have a negligible level of SLA violation.

Finally, we compare the two policies with the baseline, where no bitrate is guaranteed, and the resources are allocated according to proportional fairness scheduling (Fig. 4c). In this



(a) Policy *Soft* (b) Policy *Strict* (c) Porportional fairness (no SLA enforcement)

Fig. 4: Evolution of the per-UE throughput during different experiments, where traffic is started in sequence for UE3, UE2, and UE1.

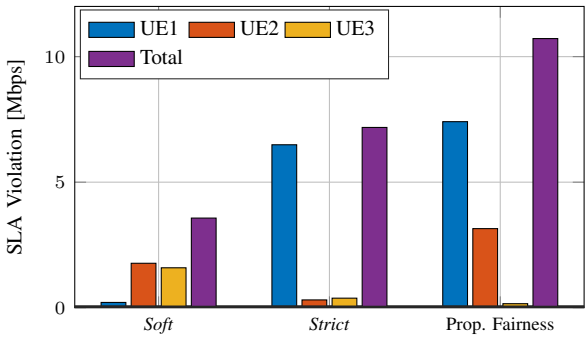


Fig. 5: SLA violation, i.e., difference between requested GBR and actual measured throughput, per UE and total.

case, all the UEs obtain the same average throughput of around 7.5 Mbps. Consequently, UE1 and UE2 experience a high SLA violation, while UE3 does not, as its throughput is higher than the GBR value for most of the duration of the experiment.

Overall, these experiments show how the proposed framework enables an effective implementation of near-RT dynamic SLA management mechanisms, which can be directly interfaced with real 5G RAN deployments.

VI. Conclusions

This paper presented experimental capabilities that enable data-driven O-RAN experimentation at scale with an open-source 5G implementation based on OAI. This has been packaged into the OpenRAN Gym framework, including an O-RAN compatible extension of OAI, with an easily extensible E2 agent, and an xApp SDK compatible with the OSC near-RT RIC. We have leveraged on this framework to build a near-real-time O-RAN-based dynamic GBR SLA management solution, where two different resource contention resolution policies are implemented as xApps. We have deployed the framework on Colosseum, and results demonstrate the effectiveness of both the proposed SLA management solution and the framework as a whole. Future works can include more refined SLA enforcement policies and

References

[1] J. Navarro-Ortiz *et al.*, "A Survey on 5G Usage Scenarios and Traffic Models," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 905–929, Second quarter 2020.

[2] E. Dahlman, S. Parkvall, and J. Skold, *5G NR: The next generation wireless access technology*. Academic Press, 2020.

[3] Z. Zhang *et al.*, "Dependent Misconfigurations in 5G/4.5 G Radio Resource Control," *ACM CoNext 2023 (Proceedings of the ACM on Networking (PACMNET))*, 2023.

[4] N. Patriciello, S. Lagen, B. Bojovic, and L. Giupponi, "An E2E simulator for 5G NR networks," *Simulation Modelling Practice and Theory*, vol. 96, no. 101933, November 2019.

[5] S.-Y. Lien *et al.*, "5G New Radio: Waveform, Frame Structure, Multiple Access, and Initial Access," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 64–71, June 2017.

[6] Q. Liu, N. Choi, and T. Han, "Onslicing: Online end-to-end network slicing with reinforcement learning," in *Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '21, 2021, p. 141–153.

[7] H.-T. Chien, Y.-D. Lin, C.-L. Lai, and C.-T. Wang, "End-to-End Slicing With Optimized Communication and Computing Resource Allocation in Multi-Tenant 5G Systems," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 2079–2091, Feb 2020.

[8] A. Abouaomar, A. Taik, A. Filali, and S. Cherkaoui, "Federated Deep Reinforcement Learning for Open RAN Slicing in 6G Networks," *IEEE Communications Magazine*, vol. 61, no. 2, pp. 126–132, February 2023.

[9] 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.

[10] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding O-RAN: Architecture, interfaces, algorithms, security, and research challenges," *IEEE Communications Surveys & Tutorials*, 2023.

[11] L. Bonati, M. Polese, S. D'Oro, S. Basagni, and T. Melodia, "OpenRAN Gym: AI/ML development, data collection, and testing for O-RAN on PAWR platforms," *Computer Networks*, vol. 220, p. 109502, 2023.

[12] 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.

[13] 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.

[14] O.-R. Alliance. (2020) "e2 simulator". [Online]. Available: https://wiki.o-ran-sc.org/display/ORANSDK/E2+Simulator

[15] C.-C. Chen, M. Irazabal, C.-Y. Chang, A. Mohammadi, and N. Nikaein, "FlexApp: Flexible and Low-Latency xApp Framework for RAN Intelligent Controller," in *IEEE International Conference on Communications (ICC)*, 2023.

[16] R. Schmidt, M. Irazabal, and N. Nikaein, "FlexRIC: An SDK for Next-Generation SD-RANs," in *Proceedings of ACM CoNEXT*, Virtual Conference, December 2021.

[17] C. Currier, "Protocol buffers," in *Mobile Forensics–The File Format Handbook: Common File Formats and File Systems Used in Mobile Devices*. Springer, 2022, pp. 223–260.

[18] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[19] H. M. Salkin and C. A. De Kluyver, "The knapsack problem: a survey," *Naval Research Logistics Quarterly*, vol. 22, no. 1, pp. 127–144, 1975.

[20] L. Bonati *et al.*, "Colosseum: Large-Scale Wireless Experimentation Through Hardware-in-the-Loop Network Emulation," in *Proceedings of IEEE DySPAN*, Virtual Conference, December 2021.