

D. Lebedev¹, Zh. Otarbay¹, A. Kyzirkanov^{1*}

¹Al-Farabi Kazakh National University, Almaty, Kazakhstan

*e-mail: abzzall@gmail.com

A REPRODUCIBLE 5G SA TESTBED FOR O-RAN MACHINE LEARNING

Abstract

Data-driven control of the radio access network (RAN) through RAN Intelligent Controllers (RICs) in the Open Radio Access Network (O-RAN) architecture depends on labelled, stack-wide measurement data, which is scarce and costly to obtain on commercial equipment. This paper describes a fully software-defined fifth-generation (5G) Standalone (SA) testbed and an automated pipeline that turns it into a generator of labelled machine-learning datasets. The testbed integrates the srsRAN gNodeB, the Open5GS core, a GNU Radio plus ZeroMQ radio emulator with runtime-controllable channel impairment, and a FlexRIC near-real-time RAN Intelligent Controller (Near-RT RIC), all on a single commodity server with central processing unit (CPU) isolation and per-user network namespaces for repeatability. A Monte-Carlo orchestrator runs up to one thousand 20-second iterations, randomizing user count, per-user path loss, traffic type and rate, and parallelism, and a synchronized parser assembles one-second snapshots into a 37-feature table spanning the physical (PHY), medium access control (MAC), radio link control (RLC), packet data convergence protocol (PDCP), and operating-system layers, augmented with five-second rolling statistics, an energy-efficiency index, and forward-looking regression and classification targets at a five-second horizon. We describe the architecture, the feature engineering, the labelling methodology, and the resulting dataset structure, and we discuss how the dataset supports predictive xApps for proactive congestion and degradation control. The platform and pipeline are designed for reproducibility and are intended as a reusable resource for the O-RAN machine-learning community.

Keywords: 5G Standalone, O-RAN, RAN Intelligent Controller, machine-learning dataset, feature engineering, srsRAN, FlexRIC, testbed, reproducibility.

Д. Лебедев¹, Ж. Отарбай¹, А. Кызырканов¹

¹Казахский национальный университет имени Аль-Фараби, г.Алматы, Казахстан

ВОСПРОИЗВОДИМЫЙ ТЕСТОВЫЙ СТЕНД 5G SA ДЛЯ МАШИННОГО ОБУЧЕНИЯ В O-RAN

Аннотация

Управление радиосетью на основе данных через интеллектуальные контроллеры O-RAN RAN Intelligent Controllers требует размеченных измерительных данных по всему стеку протоколов, которые трудно и дорого получать на коммерческом оборудовании. В данной статье описывается полностью программно-определяемый тестовый стенд 5G Standalone (SA) и автоматизированный конвейер, который превращает его в генератор размеченных датасетов для машинного обучения. Тестовый стенд объединяет srsRAN gNodeB, ядро Open5GS, радиоэмулятор GNU Radio и ZeroMQ с управляемыми в реальном времени ухудшениями канала, а также FlexRIC Near-RT RIC. Все компоненты работают на одном сервере общего назначения с изоляцией CPU и отдельными сетевыми пространствами имен для каждого пользователя, что обеспечивает повторяемость экспериментов. Monte-Carlo-оркестратор выполняет до одной тысячи итераций длительностью 20 секунд, случайным образом изменяя количество пользователей, потери в канале для каждого пользователя, тип и скорость трафика, а также степень параллелизма. Синхронизированный парсер собирает односекундные снимки состояния в таблицу из 37 признаков, охватывающих уровни PHY, MAC, RLC/PDCP и операционной системы. Таблица дополнена пятисекундными скользящими статистиками, индексом энергоэффективности, а также регрессионными и классификационными целевыми переменными с горизонтом прогнозирования пять секунд. В статье описываются архитектура, инженерия признаков, методология разметки и структура полученного датасета, а также обсуждается, как этот датасет может поддерживать предиктивные xApps для проактивного управления перегрузками и деградацией

качества. Платформа и конвейер разработаны с учетом воспроизводимости и предназначены как многоразовый ресурс для сообщества O-RAN machine learning.

Ключевые слова: 5G Standalone, O-RAN, RAN Intelligent Controller, датасет для машинного обучения, инженерия признаков, srsRAN, FlexRIC, тестовый стенд, воспроизводимость.

Д. Лебедев¹, Ж. Отарбай¹, А. Кызыркенов^{1, *}

¹Өл-Фараби атындағы Қазақ ұлттық университеті, Алматы қ., Қазақстан

O-RAN ЖҮЙЕСІНДЕГІ МАШИНАЛЫҚ ОҚЫТУҒА АРНАЛҒАН ҚАЙТА ЖАҢҒЫРТЫЛАТЫН 5G SA ТЕСТІЛІК СТЕНДІ

Аңдатпа

O-RAN жүйесіндегі RAN Intelligent Controllers арқылы радиоқолжетімділік желісін деректерге негізделген түрде басқару үшін протоколдар стегінің барлық деңгейлерін қамтитын таңбаланған өлшеу деректері қажет. Мұндай деректерді коммерциялық жабдықта алу қиын әрі қымбат. Бұл мақалада толық бағдарламалық түрде анықталған 5G Standalone (SA) тестілік стенді және оны машиналық оқытуға арналған таңбаланған датасеттер генераторына айналдыратын автоматтандырылған конвейер сипатталады. Тестілік стенд srsRAN gNodeB, Open5GS ядросын, арна параметрлерін нақты уақытта басқаруға мүмкіндік беретін GNU Radio және ZeroMQ негізіндегі радиоэмуляторды, сондай-ақ FlexRIC Near-RT RIC компонентін біріктіреді. Барлық компоненттер қайталанатын эксперименттерді қамтамасыз ету үшін CPU оқшаулауы және әр пайдаланушыға арналған жеке желілік namespace қолданылатын бір әмбебап серверде жұмыс істейді. Monte-Carlo оркестраторы ұзақтығы 20 секундқа дейінгі бір мыңға жуық итерацияны орындайды, әр итерацияда пайдаланушылар санын, әр пайдаланушыға тән жолдағы жоғалтуды, трафик түрі мен жылдамдығын және параллелизм деңгейін кездейсоқ өзгертеді. Синхрондалған парсер бір секундтық жүйе күйінің снимоктарын PHY, MAC, RLC/PDCP және операциялық жүйе деңгейлерін қамтитын 37 белгіден тұратын кестеге жинақтайды. Бұл кесте бес секундтық жылжымалы статистикалармен, энергия тиімділігі индексімен және бес секундтық болжау горизонты бар регрессиялық және классификациялық мақсатты айнымалылармен толықтырылады. Мақалада архитектура, белгілерді құрастыру, таңбалау әдістемесі және алынған датасеттің құрылымы сипатталады, сондай-ақ бұл датасеттің желідегі кептеліс пен сапа нашарлауын проактивті басқаруға арналған предиктивті xApps әзірлеуге қалай көмектесетіні талқыланады. Платформа мен конвейер қайта жаңғыртылуды қамтамасыз ету мақсатында жасалған және O-RAN машиналық оқыту қауымдастығы үшін қайта пайдалануға болатын ресурс ретінде ұсынылады.

Түйін сөздер: 5G Standalone, O-RAN, RAN Intelligent Controller, машиналық оқыту датасеті, белгілерді құрастыру, srsRAN, FlexRIC, тестілік стенд, қайта жаңғыртылу.

Introduction

Key Provisions

This study introduces a reproducible software-defined 5G Standalone testbed and an automated pipeline for generating labelled machine-learning datasets for O-RAN applications. The platform integrates srsRAN, Open5GS, GNU Radio, ZeroMQ, and FlexRIC on a single server and collects synchronized measurements from different protocol-stack layers. The resulting dataset contains approximately 19,000 labelled snapshots with more than 45 features, rolling statistics, energy-efficiency indicators, and forward-looking prediction targets. The proposed platform provides a reusable foundation for developing predictive xApps capable of detecting network congestion and performance degradation in advance.

The Open Radio Access Network (O-RAN) architecture adds programmable radio access network (RAN) Intelligent Controllers (RICs) that carry machine-learning (ML) applications (xApps) in the near-real-time RIC (Near-RT RIC), and radio applications (rApps) in the non-real-time RIC (non-RT RIC) for RAN optimization through closed-loop control [1,2]. Whether this vision becomes a reality depends on a resource that most takes it for granted: labeled high-resolution measurement data spanning the entire protocol stack and characterizing network performance under various operating conditions. Indeed, the most typical ML workflow for O-RAN starts by collecting exactly such “step data” across many RAN configurations, before any ML models can be conceived, trained or deployed [3,4]. This data is often challenging to collect: commercial equipment exposes only a

limited subset of internal telemetry, experimental deployments in live networks are prohibitive and public datasets that combine PHY, MAC, transport and hardware metrics with “forward-looking” labels are extremely rare.

Software-defined cellular stacks provides one possible solution. Thanks to the availability of open-source software-defined fifth-generation (5G) core (Open5GS) [5], gNodeB (srsRAN) [6], and O-RAN RIC (FlexRIC) [7] implementations, an end-to-end 5G Standalone (SA) network can be built using off-the-shelf hardware, and instrumented extensively [8,9,10]. Such software-defined networks have already been used for profiling RAN components [11], developing ML-based xApps for closed-loop control [12], and for deploying campus-scale O-RAN testbeds [9]. Existing work on RIC intelligence, such as traffic steering [13,14], prediction [15], and key performance indicator (KPI)-driven optimization [16], rely heavily on relevant training data, and O-RAN surveys confirm that lack of labeled data and relevant features is a practical challenge [17]. However, less focus has been dedicated to automatically generate such labeled stack-wide dataset from a testbed in a reproducible manner, with all necessary documentation.

In this paper, we tackle this problem head-on. We introduce a software-defined 5G SA testbed, and, more importantly, an automated pipeline that transforms it into a dataset generator for ML-based RAN control. Our contributions are threefold. Firstly, we describe a reproducible testbed, comprising srsRAN, Open5GS, a runtime-controllable radio emulator (GNU Radio + ZeroMQ), and a FlexRIC Near-RT RIC, deployed on a single, central processing unit (CPU)-isolated server. Secondly, we present an automated data-generation pipeline using a Monte-Carlo approach, where various parameters of the network are randomized. We synchronize a stack-wide parser that collects per-second samples and aggregates them into a table of 37 features from four different layers of the protocol stack, along with rolling statistics and an energy efficiency metric. Thirdly, we propose a forward-looking methodology for assigning ground truth labels to our generated data, both in the form of a regression target and a binary class indicative of network degradation, at a 5-second horizon, enabling proactive xApp development. We provide details on our design, the resulting data format, and its potential use cases, keeping reproducibility at the forefront of our efforts.

The remainder of this paper is structured as follows. Section 2 outlines the architecture of our testbed, data generation pipeline, and feature and label design. Section 3 presents the dataset structure, scope, reproducibility, and data-quality results. Section 4 discusses its applicability to predictive RAN control. Section 5 concludes the paper.

Research methodology

Testbed architecture

The testbed realizes a complete 5G SA network in software on a single commodity server, with no dedicated radio hardware. Four domains compose the system, as shown in Figure 1.

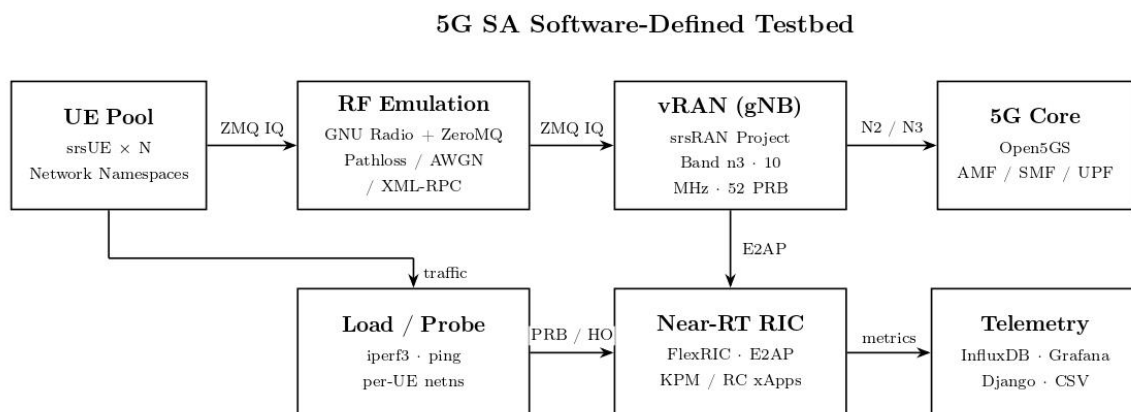


Figure 1. 5G SA testbed architecture

The core network is provided by Open5GS [5], running the access and mobility management, session management, and user-plane functions as independent daemons that communicate over standardized Third Generation Partnership Project (3GPP) interfaces on loopback addresses. The radio access network is the srsRAN Project gNodeB [6], operating in 5G New Radio (NR) Standalone mode on band n3 with a 10 MHz channel (52 physical resource blocks). The air interface is virtualized: instead of physical radios, the gNodeB and the srsUE instances exchange baseband in-phase and quadrature (IQ) samples over ZeroMQ sockets, which guarantees deterministic, repeatable experiments isolated from external interference [8]. A GNU Radio flow graph [18] interposed on the IQ path applies controllable path loss and additive white Gaussian noise (AWGN) and exposes an Extensible Markup Language Remote Procedure Call (XML-RPC) interface so that channel conditions can be altered at runtime without restarting the cell.

It takes two engineering steps to configure the platform as a measurement instrument. One step is pinning the latency sensitive physical-layer and ZeroMQ-broker threads on non-operating-system (OS) scheduler cores and isolating the network interface interrupts to non-isolated cores, making sure that the time-series have no scheduler jitter. The second step is isolating the per-user traffic generators to their own network namespaces inside the OS, emulating each user within their own Linux network namespace and thus separating the traffic generators at the network stack layer. We sample telemetry from two sources:

(1) The Near-RT RIC of FlexRIC [7] pulls the scheduler internals per second through the E2 interface and its key performance measurement (KPM) and radio control (RC) service models.

(2) A simple custom daemon is run from within the OS's root which samples the hardware and OS metrics once per second by reading from kernel pseudo-files including processor power read from the Intel Running Average Power Limit (RAPL) counters [19].

Automated dataset-generation pipeline

To collect a representative dataset rather than a handful of fixed operating points, the testbed is driven by a Monte-Carlo orchestration script that executes up to one thousand independent iterations, each lasting 20 seconds. At the start of every iteration the orchestrator draws the experimental conditions at random from defined distributions, summarized in Table 1: the number of active users, a per-user base path loss that emulates the user's distance from the cell centre (driven by a random-walk mobility process), the offered traffic rate, the transport protocol mix, and the number of parallel flows. Randomizing these factors jointly produces the diversity of network states that a robust ML model requires.

Table 1. Stochastic parameters randomized per iteration by the Monte-Carlo orchestrator

<i>Parameter</i>	<i>Range / distribution</i>	<i>Purpose</i>
<i>User count</i>	<i>Uniform 1-5</i>	<i>Dynamic attach/detach of UEs</i>
<i>Path loss, UE 1</i>	<i>25-42 dB</i>	<i>Near zone (random walk)</i>
<i>Path loss, UE 2</i>	<i>38-58 dB</i>	<i>Mid zone</i>
<i>Path loss, UE 3</i>	<i>50-70 dB</i>	<i>Far zone</i>
<i>Path loss, UE 4</i>	<i>65-82 dB</i>	<i>Cell edge / handover candidate</i>
<i>iPerf bitrate</i>	<i>2-18 Mbps</i>	<i>Within ZMQ stability range</i>
<i>Transport mix</i>	<i>80% UDP / 20% TCP</i>	<i>Streaming vs. data transfer</i>
<i>Parallel flows</i>	<i>1 or 4</i>	<i>Single vs. concurrent sockets</i>
<i>Warmup discard</i>	<i>first 3 s</i>	<i>Remove transient TCP ramp-up</i>

A carefully planned warm-up period is part of data quality: the first three seconds out of every 20-second run of a 20s iteration are filtered out from the parser. During the transient, the network is setup, establishes sessions, transmission control protocol (TCP) window grows exponentially and the scheduler heuristic are not stabilized, and filtering these data ensures that we are logging in to a stable

network environment, and not just artifacts at startup. A timestamp synchronization parser reprojects timestamps to the millisecond-level for the gNodeB, the iPerf[20], ping, and system daemon data and produce one line per second corresponding to one system state.

Feature engineering and labelling

Each snapshot row comprises 37 features extracted directly from the internal interfaces of the core, the virtualized radio access network (vRAN), and the server hardware, organized by stack layer as shown in Figure 2. The physical layer contributes channel-quality and link metrics: channel quality indicator (CQI), downlink and uplink modulation and coding scheme (MCS), reference signal received power (RSRP), physical uplink shared channel (PUSCH) signal-to-noise ratio (SNR), timing advance, the real radio bitrate, and the block error rate. The medium access control (MAC) layer contributes protocol data unit (PDU) sizes, logical-channel byte counts, and the Buffer Status Report, following the 5G NR MAC protocol specification [21], a high-value signal of pending uplink data. The radio link control (RLC) and packet data convergence protocol (PDCP) layers contribute sequence numbers, packet counts, and per-bearer encrypted volumes. The operating-system layer contributes gNodeB CPU utilization, free memory, interrupt rate, and processor power from RAPL.

ml_dataset.csv — 37 features across the protocol stack

PHY layer cqi, mcs_dl, mcs_ul, rsrp, snr, pusch, timing_advance, brate_dl, brate_ul, bler	MAC layer mac_pdu_size, lcid_4_bytes, lcid_1_bytes, bsr (buffer status report)
RLC / PDCP rlc_sn, pdcp_count, pdcp_bytes_bearer	OS / hardware cpu_util_gnb, ram_free_mb, irq_rate, power_rapl_watt
Rolling (5 s window) dl_roll5, lat_roll5, prb_roll5, cpu_max_roll5	Targets (t+5 s) target_dl_5s, target_rtt_5s, is_degraded (binary)

each row = 1-second snapshot of the full system state

Figure 2. Dataset feature structure

To give models temporal context, the parser computes five-second rolling statistics (moving average, maximum, and variance) for the key quantities bitrate, latency, physical resource block (PRB) load, and CPU peak. It also derives a network energy-efficiency index defined in Equation (1) as the radio throughput delivered per watt of processor power:

$$E_{\text{eff}} = \frac{R_{\text{radio}} \text{ (Mbps)}}{P_{\text{RAPL}} \text{ (W)}} \quad (1)$$

For predictive control, the dataset is labelled with targets shifted forward by a fixed five-second horizon: target_dl_5s and target_rtt_5s record the throughput and round-trip time the cell will exhibit five seconds later, supporting regression. A binary degradation class supports classification and is the key label for early warning xApps. As shown in Equation (2), it is set to one when the predicted round-trip time exceeds the current value by more than fifty percent:

$$y_{\text{deg}} = 1 \text{ if } \text{RTT}_{t+5} > 1.5 \cdot \text{RTT}_t, \quad \text{else } 0 \quad (2)$$

This label lets a classifier learn to anticipate the onset of queue buildup before it manifests, which is precisely the signal a proactive RIC needs to act in time. The full row schema concatenates the snapshot features, the rolling statistics, the energy index, and the targets into a single record written to a consolidated comma-separated values (CSV) file.

Results of the study

Dataset structure and scope

The pipeline yields a single consolidated file, `ml_dataset.csv`, in which every row is a one-second, timestamp-aligned snapshot tagged with the iteration's scenario, run, and randomized conditions. Table 2 summarizes the volume produced by the two collection phases. A deterministic phase over twelve profiling scenarios, executed in triplicate at 60 seconds each, contributes on the order of two thousand rows of carefully controlled states; the stochastic Monte-Carlo phase contributes the bulk of the data, on the order of seventeen thousand rows of randomized states after warmup removal. Together they give roughly nineteen thousand labelled snapshots spanning a wide range of user counts, channel conditions, and traffic mixes.

Table 2. Dataset volume by collection phase

Phase	Scenarios	Runs	Duration	Rows (est.)
Deterministic	12	3 each	60 s	~2,160
Stochastic (Monte-Carlo)	up to 1000 iter.	-	17 s (post-warmup)	~17,000
Total	-	-	-	~19,160

Table 3 lists the column schema grouped by origin. The 37 base features are complemented by four rolling-statistic columns, the energy-efficiency index, and three target columns, for a total of more than forty-five columns per row. Because the features are drawn from every layer of the stack simultaneously and aligned in time, the dataset captures cross-layer relationships for example between MAC Buffer Status Reports, PRB utilization, and subsequent latency that single-layer measurements cannot express.

Table 3. Column schema of `ml_dataset.csv` grouped by origin

Group	Representative columns	Count
PHY	<code>cqi</code> , <code>mcs_dl</code> , <code>mcs_ul</code> , <code>rsrp</code> , <code>snr_pusch</code> , <code>timing_advance</code> , <code>brate_dl</code> , <code>brate_ul</code> , <code>bler</code>	9
MAC	<code>mac_pdu_size</code> , <code>lcid_4_bytes</code> , <code>lcid_1_bytes</code> , <code>bsr</code>	4
RLC / PDCP	<code>rlc_sn</code> , <code>pdcp_count</code> , <code>pdcp_bytes_bearer</code>	3
OS / hardware	<code>cpu_util_gnb</code> , <code>ram_free_mb</code> , <code>irq_rate</code> , <code>power_rapl_watt</code>	4
Context / KPI	<code>ue_count</code> , <code>load</code> , <code>radio</code> , <code>control</code> , <code>dl</code> , <code>ul</code> , <code>lat_mean</code> , <code>lat_p95</code> , <code>loss</code> , <code>prb</code>	10+
Rolling (5 s)	<code>dl_roll5</code> , <code>lat_roll5</code> , <code>prb_roll5</code> , <code>cpu_max_roll5</code>	4
Energy	<code>e_eff</code>	1
Targets	<code>target_dl_5s</code> , <code>target_rtt_5s</code> , <code>is_degraded</code>	3

Reproducibility and data quality

Three design choices underpin the quality of the resulting data. ZeroMQ-based radio emulation makes every experiment bit-exact and free of the uncontrolled variability of over-the-air measurement, so that a given set of conditions can be reproduced precisely. CPU isolation removes scheduler-induced timing noise from the latency and throughput records, which is essential because the dataset's value rests on accurate one-second aggregation. The warmup-discard rule removes the non-stationary start-up transient, ensuring that the snapshots represent settled network states rather

than session-establishment artifacts. The per-user network namespaces guarantee that multi-user traffic is genuinely independent at the stack level rather than multiplexed through a shared interface. Together these measures yield a dataset whose labels reflect the network's intrinsic behavior rather than measurement noise.

A natural concern for any emulated platform is external validity. The ZeroMQ link imposes a throughput ceiling that does not exist in a real radio, and the path-loss-plus-AWGN channel is simpler than a full fading model. These limit the absolute realism of throughput magnitudes, but they do not undermine the dataset's purpose: the features capture the relative dynamics of buffers, scheduling, and link adaptation, which are the relationships predictive RIC models must learn. The platform is also extensible the GNU Radio graph already supports fading, shadowing, Doppler, and interference blocks so richer channel realism can be added without changing the pipeline.

Discussion

Supporting predictive RAN control

The dataset is constructed specifically to train the proactive xApps that O-RAN envisions. The forward-shifted regression targets allow a model to forecast throughput and latency five seconds ahead, while the binary degradation label of Equation (2) frames early-warning as a supervised classification problem: predict, from the current cross-layer state, whether latency is about to inflate. Gradient-boosted trees such as XGBoost [22] are a natural fit for the tabular, mixed-type feature space, while recurrent architectures such as long short-term memory (LSTM) [23] or gated recurrent unit (GRU) [24] can exploit the temporal context encoded in the rolling features and the sequence of snapshots. A model trained on this data and deployed as an xApp in the Near-RT RIC could act on the degradation signal for example by adjusting PRB allocation or limiting aggressive flows through the RC service model before queue buildup becomes user-visible. The energy-efficiency index additionally enables joint optimization of performance and power, aligning with the growing emphasis on sustainable RAN operation [16].

Far more than one lab, what makes the pipeline useful is that it is reusable. Since the testbed is constructed from completely open-source parts, and documented how to be reproduced, other researchers can also reproduce similar datasets, expand on feature coverage, or test other conditions, fostering the type of sharing and comparison-of-apples experiment that the O-RAN ML community calls for. As such, we consider the data-generating pipeline itself the primary deliverable, more so than any specific trained model. It makes the reusable testbed into a renewable source of labeled, stack-wide data to train stack-wide data-driven control systems.

Conclusion

We showcased a software-defined 5G Standalone testbed and an automated workflow to make it a stack-wide labelled ML data generator for O-RAN. The testbed uses srsRAN, Open5GS, a GNU Radio/ZeroMQ based radio emulator with runtime-controlled impairments and a FlexRIC Near-RT RIC on a single, CPU-isolated server with per-user network namespaces to achieve repeatability. A Monte-Carlo orchestrator is used to randomize the number of users, pathloss, traffic load and level of parallelism over several brief iterations and a synchronized parser combines these over one-second snapshots into a 37 feature-tablecovering PHY, MAC, RLC/PDCP and OS layers; complemented by 5s rolling-average stats, an energy efficiency index, and 5s forward looking regressors and classifiers. This yields a reproducible source of 19K-odd, labelled snapshot tables for training proactive xApps on congestion and degradation. We intend to use this dataset in future work to train and run such xApps closed loop and to further enhance the channel emulations with impairments and fading.

Acknowledgment

This research was funded by the Committee of Science of the Ministry of Science and Higher Education of the Republic of Kazakhstan (Grant No.BR24993211).

References

- [1] O-RAN Alliance, "O-RAN Architecture Description," O-RAN.WG1.OAD, v11.0, 2024. URL: <https://www.o-ran.org/specifications>.
- [2] 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*, vol. 25, no. 2, pp. 1376-1411, 2023. DOI: 10.1109/COMST.2023.3239220.
- [3] 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, 2021. DOI: 10.1109/MCOM.101.2001120.
- [4] O-RAN Working Group 2, "AI/ML workflow description and requirements," O-RAN.WG2.AIML, v1.03, 2021. URL: <https://www.o-ran.org/specifications>.
- [5] Open5GS, "Open5GS Documentation: Open-source 5G core network," 2024. [Online]. Available: <https://open5gs.org>
- [6] srsRAN Project, "srsRAN Project Documentation," Software Radio Systems, 2024. [Online]. Available: <https://docs.srsran.com>
- [7] R. Schmidt, M. Irazabal, and N. Nikaein, "FlexRIC: An SDK for next-generation SD-RANs," in *Proc. ACM CoNEXT*, 2021, pp. 411-425. DOI: 10.1145/3485983.3494870.
- [8] L. Mamushiane, A. Lysko, H. Kobo, and J. Mwangama, "Deploying a stable 5G SA testbed using srsRAN and Open5GS: UE integration and troubleshooting towards network slicing," in *Proc. IEEE AFRICON*, 2023. DOI: 10.1109/icABCD59051.2023.10220512.
- [9] C. Sun et al., "Campus5G: A campus scale private 5G Open RAN testbed," *arXiv preprint arXiv:2506.23740*, 2025. DOI: 10.48550/arXiv.2506.23740.
- [10] NIST, "Blueprint for deploying 5G O-RAN testbeds," NIST Technical Note 2311, 2024. URL: <https://doi.org/10.6028/NIST.TN.2311>.
- [11] C. Wei, A. Kak, N. Choi, and T. Wood, "5GPerf: Profiling open-source 5G RAN components under different architectural deployments," in *Proc. ACM SIGCOMM Workshop on 5G and Beyond*, 2022, pp. 43-49. DOI: 10.1145/3538394.3546044.
- [12] L. Bonati, M. Polese, 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*, vol. 22, no. 10, pp. 5787-5800, 2023. DOI: 10.1109/TMC.2022.3188013.
- [13] A. Lacava et al., "Programmable and customized intelligence for traffic steering in 5G networks using Open RAN architectures," *IEEE TMC*, vol. 23, no. 4, pp. 2882-2897, 2024. DOI: 10.1109/TMC.2023.3266642.
- [14] I. Tamim, S. Aleyadeh, and A. Shami, "Intelligent O-RAN traffic steering for URLLC through deep reinforcement learning," *arXiv preprint arXiv:2303.01960*, 2023. DOI: 10.48550/arXiv.2303.01960.
- [15] Transformer-based wireless traffic prediction and network optimization in O-RAN, *arXiv preprint arXiv:2403.10808*, 2024. DOI: 10.48550/arXiv.2403.10808.
- [16] Power-efficient RAN intelligent controllers through optimized KPI monitoring, *arXiv preprint arXiv:2501.09509*, 2025. DOI: 10.48550/arXiv.2501.09509.
- [17] Recent advances in machine learning for network automation in the O-RAN, *Sensors*, vol. 23, no. 21, art. 8792, 2023. DOI: 10.3390/s23218792.
- [18] GNU Radio Project, "GNU Radio: The free and open software radio ecosystem," 2024. [Online]. Available: <https://www.gnuradio.org>
- [19] Intel Corporation, "RAPL energy reporting," *Intel 64 and IA-32 Architectures Software Developer's Manual*, 2023. URL: <https://www.intel.com/content/www/us/en/developer/articles/technical/intel-sdm.html>.
- [20] A. Tirumala, M. Gates, et al., "iPerf: TCP/UDP bandwidth measurement tool," 2024. [Online]. Available: <https://iperf.fr>
- [21] 3GPP, "NR; Medium Access Control (MAC) protocol specification," 3GPP TS 38.321, V18.0.0, 2024. URL: https://www.3gpp.org/ftp/Specs/archive/38_series/38.321/.
- [22] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. ACM SIGKDD*, 2016, pp. 785-794. DOI: 10.1145/2939672.2939785.
- [23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, 1997. DOI: 10.1162/neco.1997.9.8.1735.
- [24] K. Cho et al., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. EMNLP*, 2014, pp. 1724-1734. DOI: 10.3115/v1/D14-1179.