IEEE INFOCOM 2024 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS) | 979-8-3503-8447-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/INFOCOMWKSHPS61880.2024.10620861

5G RAN and Core Orchestration with ML-Driven QoS Profiling

Carlos Valente*†, Pedro Valente*, Pedro Rito*, Duarte Raposo*, Miguel Luís*‡, Susana Sargento*†
*Instituto de Telecomunicações, 3810-193 Aveiro, Portugal
†Departamento de Eletrónica, Telecomunicações e Informática, Universidade de Aveiro, 3810-193 Aveiro, Portugal
‡Instituto Superior Técnico, Universidade de Lisboa, Av. Rovisco Pais 1, Lisboa 1049-001, Portugal

***Abstract*—5G has revolutionised mobile communication networks; however, it poses significant challenges due to the increased number of connected devices and the escalating data demands from applications. The Open Radio Access Network (O-RAN) architecture has emerged as a solution, characterised by open and standardised interfaces that foster interoperability among diverse vendors and enable the implementation of innovative solutions. In this context, the RAN Intelligent Controller (RIC) emerges as an intelligent control entity that empowers the efficient management and optimisation of the 5G RAN. Central to this orchestration are xApps, which can be developed and executed within the RIC. These applications possess the potential to drive innovation and substantially enhance the operation of 5G networks. As primary objective, this paper demonstrates the feasibility of employing a monitoring xApp within the Near-RT RIC to support the 5G core. This contributes to a better selection of user profiles, resulting in a better management of allocated resources to each user, and improved Quality of Service (QoS). By collecting and analysing real-time data, an Orchestrator enables proactive management and informed decision-making to optimise the Core performance, QoS, and resource utilisation. Specifically, Machine Learning (ML)-processed data is leveraged to select QoS profiles and assign them to individual users with the assistance of the Core Network (CN) agent. The results demonstrate the system's capability to efficiently collect and process real-time RAN data, to make user profile category predictions, and to allocate resources accordingly.**

***Index Terms*—5G, Machine Learning, O-RAN, QoS Profiling, RAN, RIC, xApps**

I. Introduction

The telecommunications industry is evolving rapidly, especially with the emergence of 5th Generation (5G) and next-generation networks. Open Radio Access Network (O-RAN), as per the specifications outlined by the O-RAN Alliance, advocates virtualised and disaggregated Radio Access Network (RAN), where components are interconnected through open interfaces and managed by intelligent controllers. These networks are designed to be adaptable, welcoming components from various vendors and fostering interoperability. The advanced capabilities of RAN Intelligent Controllers (RICs) in efficiently managing RAN nodes have been a focal point of exploration.

In this context, the Near-Real-Time RIC (Near-RT RIC) is central in orchestrating and enhancing the RAN, ensuring compliance with 3GPP standards. It hosts xApps that optimise the radio resource management of E2 nodes, serving as a central hub for three essential interfaces (O1, A1, and E2). On the other hand, the Non-Real-Time RIC (Non-RT RIC) is responsible for long-term analyses of the RAN, deploying rApps and selecting xApps to regulate scheduling decisions sent to the Near-RT RICs. It is with these Apps, service models and the metrics they collect, that actions can then be carried out to perform RAN slicing [1], [2].

To enhance the slicing effectiveness and Quality of Service (QoS) in 5G networks, the integration of ML-based applications into xApp design could be explored. An example of this approach is presented in [3], where a framework based on xApp design is proposed to optimise 5G networks by improving network slicing operations. This framework leverages ML algorithms and open-source software to collect and analyse data regarding RAN and CN. In [4], the work delves into developing a user-specific O-RAN traffic steering framework utilising xApps. By employing a custom O-RAN service model, this framework provides fine-grained control over traffic flows, enabling xApps to steer traffic to different network slices and resources based on real-time conditions, in a hybrid environment where the Network Simulator ns-3 is integrated. In addressing the challenges of slow convergence and unstable performance of Deep Reinforcement Learning (DRL) algorithms in O-RAN slicing, the work in [5] proposes a hybrid transfer learning approach. This approach harnesses the benefits of policy reuse and distillation Transfer Learning (TL) methods, promoting safe and accelerated convergence. The proposed approach could enable more efficient and effective RAN slicing.

Nevertheless, the control interaction between RAN and Core presents a non-trivial challenge. Core applications need to be based on 5G Application Function (AF) principles and xApp based on O-RAN definitions and specifications. Therefore, the concept of an orchestrator is considered as a potential bridge between both sides, with the potential to influence throughout the entire 5G network. In [6], the authors introduce an orchestrator designed to dynamically place components within a disaggregated Near-RT RIC in a cloud-edge computing environment. This orchestrator effectively utilises xApp monitoring capabilities to enhance latency requirements within the disaggregated RIC. Within the same research group, the authors of [7] define a framework for developing DRL agents on large-scale, by developing software pipelines for the design, training, testing, and experimental evaluation of ML-based xApps for Open RAN closed-loop control.

The work presented here focuses on the same design principles: a Near-RT RIC and a xApp capable to manage network resources through periodic monitoring of metrics using RAN functions. These metrics are then transmitted to an orchestrator for ML processing and analysis, enabling the profiling of connected users. The process allows a better resource allocation and assignment of specific QoS profiles based on the predicted profile generated by the ML model, facilitated through a core agent that communicates with the 5G core's user database to read and write user session parameters. Additionally, in this paper, 5G O-RAN networks and existing open-source frameworks are also analysed regarding an architecture perspective. Subsequently, the chosen framework is deployed together with a 5G network, emphasising the Near-RT RIC and the orchestrator with the goal of performing resource allocation through QoS profiling in each user.

The remaining of this paper is organised as follows. Section II evaluates O-RAN frameworks by reviewing open-source projects. Section III proposes the ML pipeline, that enables user profiling and QoS. Section IV presents the results, in terms of ML selection, and the impact of the QoS profile in jitter and bitrate. Lastly, Section V concludes this paper, giving some insights for the future work.

II. xAPP Design and 5G Network Architecture

Prior to select a framework for the xAPP definition and development, in this section we conduct a comparative analysis of the architectural frameworks utilised in the RAN. Our focus rely on the Open Networking Foundation (ONF) and OpenAir-Interface Software Alliance (OSA) initiatives, in conjunction with their integration within O-RAN projects. Additionally, the influence of virtualisation and orchestration technologies on the deployment process is also explored. Finally, we present our end-to-end solution, comprising RAN and Core components selected for the ML-Driven QoS Profiling.

A. O-RAN Frameworks Evaluation

The ONF's Software Defined RAN (SD-RAN) [8] platform aligns with O-RAN architecture, focusing on a cloud-native solution within 3GPP standards, emphasizing open-source components for various aspects of disaggregated RAN. The RAN-in-a-Box (RiaB)[1] offers a diverse set of commands and procedures for the effective administration and validation of the 5G network. These commands allow the verification of the deployment status, the confirmation of user plane functions, and the status of the active User Equipments (UEs) through Radio Resource Control (RRC) connections. It also ensures the proper connectivity between network elements and inspects the Radio - Network Information Base (R-NIB) and UE - Network Information Base (UE-NIB) to gather details about the radio network elements and connected UEs. For slice management, it allows the measurement of the throughput to assess the default slice performance, as well as the creation, association, update, verification and deletion of slices to ensure effective resource allocation, load balancing, and seamless user experience. RiaB provides a comprehensive description of the diverse built-in xApps: Mobility Load Balancing, Mobility HandOver, RAN Slice Management, Key Performance Indicators Monitor, and Physical Cell Identifier.

The MOSAIC5G Project Group (M5GPG)[2] initiative, integrated in the OSA, presents open-source platforms ranging from a centralized network control to mobile edge network deployments, including an effort for flexible and programmable RANs. FlexRIC [9], featuring a real-time controller, interacts with OpenAirInterface (OAI) or the srsRAN[3] radio stacks through the O-RAN E2 interface, aiming for real-time monitoring and control in 5G deployments. Notably, FlexRIC incorporates built-in Service Models (SMs) for customizable RAN monitoring and control, essential for adapting to dynamic 5G ecosystem use cases. Additionally, monitoring xApps provide Key Performance Indicators (KPIs) from Medium Access Control (MAC), Radio Link Control (RLC) and Packet Data Convergence Protocol (PDCP) layers, as well as the monitoring of the GPRS Tunnelling Protocol (GTP). A dedicated service model was conceptualised and tested within the FlexRIC ecosystem. Creating a service model requires a solid understanding of the underlying code, and can be validated by adapting or creating an xApp. This allowed to choose to have multiple callbacks from multiple built-in SMs, instead of having a single callback from a custom-made service model. FlexRIC also explains how xApps communicate with the RAN, via the RIC and the RAN Functions. Once the setup phase is complete, xApps subscribe a given RAN function within a E2 Node. In the explored monitoring use case, this node will periodically send the metrics collected from the RAN back to the xApp where it is stored in a database.

B. Framework Selection

ONF's usage of Kubernetes and Docker enhances scalability but can introduce complexity and more intricate command structures. ONF's strength lies in its set of deployable xApps; however, the ongoing development of documentation and some hardware limitations need to be addressed. On the other hand, OSA's FlexRIC distinguishes itself by offering open-source raw code, providing users with complete access to the entire network. It facilitates the development of custom xApps and service models with the support of tutorials. Nevertheless, its reliance on mailing lists for support may pose challenges in accessing past issues and answers. The research of these two frameworks led to some conclusions, presented in Table I. The selection of the OSA framework for this work was influenced by the absence of 5G support in the ONF RiaB framework, by the availability of instructional materials and well-documented resources within the OSA community and the simplicity of entry into their micro-service architecture.

[1]https://docs.sd-ran.org/master/sdran-in-a-box/README.html

[2]https://mosaic5g.io

[3]https://www.srslte.com/

TABLE I: O-RAN framework decision

Feature	OSA	ONF
Service Models	Allows creation and has templates	Does not mention
System Connectivity	Allows external core network use while emulating CU, DU and UE	Completely OMEC and CU-CP while emulating DU and UE
Availability of xApps	Mostly Monitoring	Diversity of built-in RC xApps
NR support	Yes	No
LTE support	Yes	Yes
Documentation	Step-by-step Tutorials	Good xApp Explanations

C. End-to-end System

For the establishment of an end-to-end 5G network in which the evaluated frameworks could be implemented, there was also an evaluation, of the RAN and CN platforms. From our previous research experiments [10], Open5GS[4] is the most stable open-source core project. For the access network, two projects were evaluated: OAI and srsRAN. These projects were analysed, not only taking into consideration the features implemented, but also in the overall applicability in QoS provisioning scenarios and real UE compatibility. However, our main focus will be the integration within the FlexRIC framework.

OAI's implementation of gNodeB (gNB) E2 interface supports Key Performance Monitoring (KPM), RAN Control (RC), GTP and MAC + RLC + PDCP SMs as mentioned in their repository[5]. In the same way, the srsRAN Project is an open-source 5G CU/DU but from SRS, that implements the KPM, and RC SMs with some limitations as mentioned in their documentation[6]. These features make the OAI's gNB the platform selected to retrieve metrics through the RIC as it presents the most variety of SMs. However, the application of QoS flows into a wide variety of Commercial Off-The-Shelf (COTS) UEs is better supported by the srsRAN's project. Thus, the srsRAN project was used to evaluate the impact in a real deployment.

III. RAN QoS Profiling: Proposal

To establish QoS profiling based on the traffic perceived by the Near-RT RIC, we introduce an orchestrator with the ability to gather KPIs generated by the monitoring xApp. This orchestrator integrates a ML pipeline for ongoing training and prediction. The outcome is the attribution of a profile, resulting in the modification of the user characteristics on the core, working as an AF. Figure 1 offers a more in depth architecture overview of the developed components. Represented in blue are the components needed for the RAN, with green the OSA framework with its Near-RT RIC and xApps, and with red the components developed for the orchestration.

The xApp begins its execution by subscribing four SMs within the RIC: MAC, RLC, PDCP, and GTP. The MAC callback function manages the Radio Network Temporary Identifier (RNTI) to International Mobile Subscriber Identity (IMSI) table. This table serves as a bridge between the known

[4]https://open5gs.org/

[5]https://gitlab.eurecom.fr/oai/openairinterface5g/-/blob/develop/openair2/E2AP/README.md

[6]https://docs.srsran.com/projects/project/en/latest/

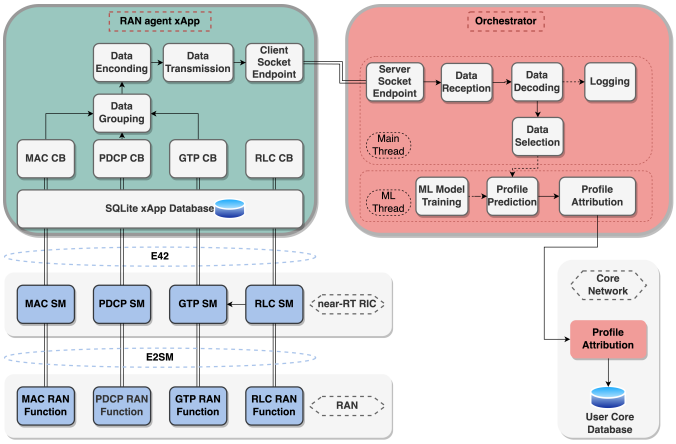


Fig. 1: 5G Network Architecture for Core-Orchestration based on ML-Driven QoS Profiling

values from the RAN (RNTI) and the core network's known values (IMSI). The PDCP callback function is responsible for processing PDCP indication messages, extracting a total of five parameters: `RNTI`, `txsdu_bytes`, `rxsdu_bytes`, `txsdu_pckts`, and `rxsdu_pckts` while the GTP callback function is responsible for parsing `RNTI` and `teidgnb` parameters. A socket handshake protocol streamlines the communication between the xApp and the orchestrator. The process begins the communication with the xApp transmitting an `ACK_TRANSMITTED` message, with value `1`, after which the orchestrator responds with an `ACK_RECEIVED`, with value `2`, allowing the xApp to transmit the entire dataset to the orchestrator. The socket also receives two other message parameters for PDCP and GTP protocols, with values `3` and `4` respectively.

The orchestrator consists of two concurrent threads: the main and the ML threads. Within the main thread, the primary role revolves around receiving, decoding, and processing data transmitted by the xApp. On the other hand, the ML thread is responsible for predicting, selecting, and associating the most suitable profile with each user based on the data gathered by the main thread. Based on the output of the ML prediction, the orchestrator attributes the correct profile to the user. This is accomplished using a custom core agent designed to interact with the Open5GS core, interfacing with the database to enact the necessary changes to the 5QI, Session Aggregate Maximum Bit Rate (AMBR) Downlink (DL), and Session AMBR UpLink (UL).

As a proof-of-concept, we defined three distinct user profiles aligned with the three verticals. The Enhanced Mobile Broadband (eMBB) is designated as the Web-Based profile, Ultra-Reliable Low-Latency Communications (URLLC) as the Sensor profile, and massive Machine-Type Communications (mMTC) as Vehicle-to-Everything (V2X) profile. Each profile is customised to address the specific applications and network requirements associated with the corresponding vertical. While the 5QI was chosen for each of these profiles based on the example services of the 5QI table, the values for the

TABLE II: Profile characteristics

	Web-Based	Sensor	V2X	Default
5QI	6	5	3	9
Resource Type	Non-GBR	Non-GBR	GBR	Non-GBR
Default Priority Level	60	10	30	90
Packet Delay Budget (ms)	300	100	50	300
Session AMBR DL (Mbps)	1000	0.1	10	1000
Session AMBR UL (Mbps)	1000	0.1	10	1000

AMBR were assigned based on the throughput of the each service. With these conditions, Table II summarises their key characteristics, where V2X profile has Guaranteed Bit Rate (GBR) and the lowest 5QI to have the highest priority, and the Web-based profile has the highest throughput. There is also a definition of a default user, created to generate unspecified traffic with no priority over the others. This profile represents the default profile normally defined by operators to a UE, where the final service is not defined. All profiles were defined according to 3GPP TS 23.501 version 15.8.0 Release 15 [11].

The last step was the generation of representative traffic, to collect a dataset to train the ML models. In our case, we evaluate the use of the K-Nearest Neighbors (KNN) and Support Vector Machine (SVM) classifiers to perform the classification of the user's profile. For the Web-Based profile, cURL commands were executed randomly, fetching various Internet images and videos. In the V2X profile, ITS-G5 messages were sent using a Message Queuing Telemetry Transport (MQTT) client to a remote broker to emulate the behaviour of V2X traffic, used in our Aveiro Tech City Living Lab (ATCLL) [12]. For the Sensor profile, a periodic message was transmitted using MQTT, representing a city traffic radar application running YOLO[7], in one of our NVIDIA Jetsons[8].

IV. Evaluation

The envisioned architecture was implemented in practice using a ML pipeline to construct a model for user classification, using OAI RAN and FlexRIC. Thereafter, the system underwent thorough evaluation within a real-world hardware testbed, comprising srsRAN (with USRP-based radio frontend) and the Open5GS 5G core.

A. ML Thread

The ML pipeline used to perform the classification problem is represented in Figure 2, and was focused on two main steps: the data preprocessing, and the model building and training. The pipeline starts with the extraction of information -*data collection*- from the three profiles (Web-based, Sensor, and V2X) collected from the monitoring xApp. For each profile, an entire hour was collected, to perform the training and testing steps for each classifier. After the data collection, it was necessary to perform a *data clean* task, to identify and correct anomalies within the dataset. The data from the three profiles is then grouped into a table, *profile data aggregation*.

The next step is the *feature selection*. A statistical data analysis was performed, aimed to reveal hidden trends and patterns

[7]https://pjreddie.com/darknet/yolo/

[8]https://developer.nvidia.com/embedded/jetson-nano-developer-kit

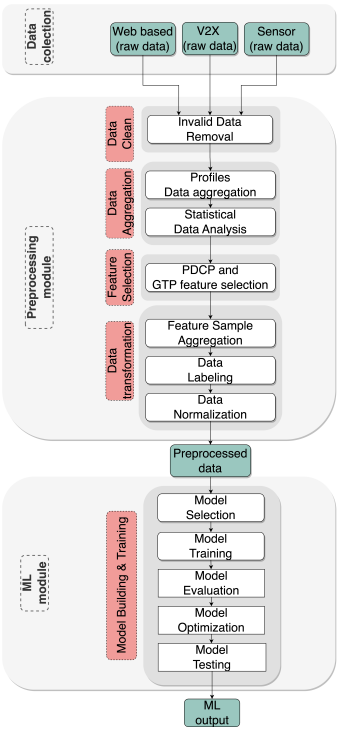
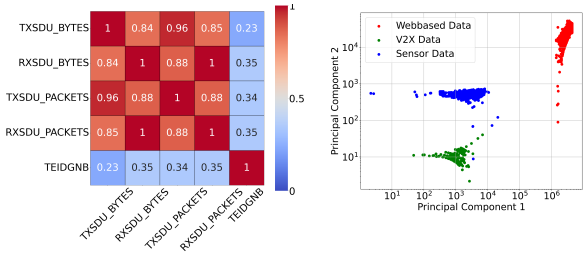


Fig. 2: ML pipeline



(a) Correlation matrix (b) Selected features PCA

Fig. 3: ML pipeline: correlation matrix and PCA

in the dataset. It included the visualisation of a heatmap to depict feature interconnections through a correlation matrix, in a *statistical data analysis*. The correlation matrix was further reduced to a simpler one with the feature selection process. This process allowed to identify the most relevant data attributes, reducing the initial hundred and twenty three features down to five. Among these, four were derived from the PDCP data, and one from the GTP data. The selection process honed in on Tx Service Data Unit (SDU) bytes, Rx SDU bytes, Tx SDU packets, and Rx SDU packets due to their presence in both the UL and DL of traffic, making them vital components. Figure 3a depicts the correlation matrix of this selected dataset, offering a visual representation of the relationships among these attributes. The relationships between Rx and Tx data (bytes and packets) for three profiles were explored to evaluate the selected features. The Web-based profile showed higher packet and byte counts, but there were certain samples intersecting the other profiles samples. A Principal Component Analysis (PCA) technique was also

TABLE III: ML models testing metrics

	KNN	SVM
Accuracy	0.9877	0.9641
Recall	0.9881	0.9632
Precision	0.9873	0.9641
F1 Score	0.9875	0.9634

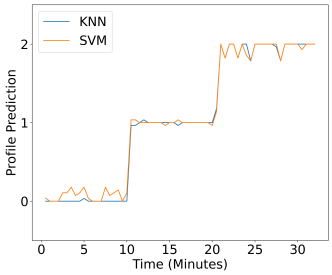


Fig. 4: Profile classification

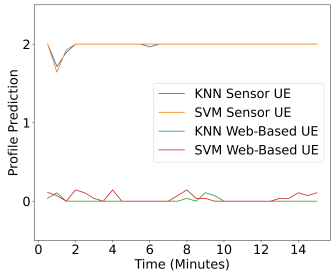


Fig. 5: Profile prediction

employed to represent the five features in an unified two-dimensional plane. Figure 3b graphically illustrates the PCA, revealing three clusters corresponding to distinct profiles, observing the clear distinction between each profile.

In order to better track the trends over a period of time, a step of *Feature Sample Aggregation* process was built, grouping these features into sets of ten, generating minute-long samples, allowing for a wider time-window of samples. Instead of having a dataset with N five-feature samples, the new dataset would contain $\frac{N}{10}$ fifty-feature samples each. The data was then classified into three categories, aligning with the previously defined profiles, and then normalised using the StandardScaler -*data normalization*. This is essential for ML algorithms reliant on distance metrics where the feature scale can profoundly impact the results.

With the pre-processing of data already done, two ML classifiers were selected, namely KNN and SVM. Subsequently, the data was carefully divided into two sets: the training and testing sets. In this process, 20% of the dataset was reserved for testing, while the remaining 80% was dedicated to training.

The *model evaluation and optimisation* included hyperparameter tuning using the GridSearchCV technique to explore hyperparameter combinations to derive the best-performing model configuration. Each parameter serves a specific role in defining the behaviour of the algorithms. The parameter grid offers a systematic approach to finding optimal hyperparameters for ML models such as KNN and SVM. Another optimisation that was implemented was in the samples time window. It was utilised ten samples per group, considering the periodic nature of Orchestrator profiling and sample availability within the timeframe. Furthermore, the training data underwent cross-validation with the newly optimised models. This process involves dividing the training dataset into five sub-datasets, each with a distinct testing section and four sections for training. The models were then trained and tested on these sub-datasets, yielding a total of five F1 scores. After training and optimisation, the models were tested on unseen data to evaluate their performance in predicting profile data. The output values for each sample were either 0, 1, or 2, and metrics like precision, accuracy, recall, and F1 score were extracted to assess the model's effectiveness.

Table III displays the evaluation results achieved with optimal parameters for each model. The KNN model shows slightly superior values, but overall, both ML models prove suitable for deployment, as already foreseen in the PCA analysis. The ML thread is now ready, with KNN classifier already trained. New incoming data should be scaled using the same scaler employed during training, and subsequently organised into groups of ten samples.

B. Profile Attribution

With the models fully trained and optimised, they were deployed into the orchestrator. Figure 4 illustrates the profile attribution made by the orchestrator for a single UE on the network. This UE experienced traffic alterations every ten minutes, following the profile sequence of Web-based, V2X, and Sensor. The graph displays the mean of all the samples from the ML output in a period of thirty seconds, verifying that the classification is consistently accurate. Despite slight fluctuations, both ML models assign accurate profiles to all generated traffic. In Figure 5, a second UE is introduced, and unlike the previous scenario, these UEs now maintain a constant profile for a duration of fifteen minutes. The orchestrator consistently predicts profiles accurately using both ML models within the specified timeframe. This classification process for multiple UEs in the network also demonstrates minimal variation in the output of the ML models. This ensures that the orchestrator can effectively classify multiple UEs into their correct profiles simultaneously.

C. QoS Flow Evaluation

For the real QoS flow evaluation, the RAN was deployed using srsRAN installed in a Intel NUC with Ubuntu 22.04 attached to a Radio Unit (RU) utilising the USRP B210. The UEs are two Smartphones Google Pixel 6a and Xiaomi Redmi Note 9T 5G with programmable sysmocom sysmoISIM-SJA2 SIM cards, as seen in Figure 6. The UEs configuration in the CN was done following Table II, with a V2X profile for GBR traffic and a default profile for the Non-Guaranteed GBR (non-GBR). Iperf3[9] was used with the V2X user generating 1 Mbps of User Datagram Protocol (UDP) traffic. To see the congestion impact, an user with a non-GBR profile was connected to the gNB, generating 20 Mbps of UDP traffic, using Iperf3.

Figures 7a and 7b both represent a comparison between two users in which one was applied the V2X profile. From the plots, it can be seen that, when a user has the GBR profile, it has priority over other users not only on the jitter but also in the bitrate, improving the QoS experienced by individual

[9]https://iperf.fr

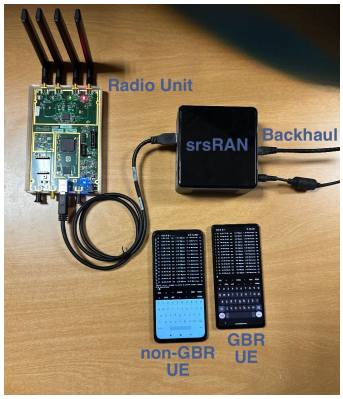
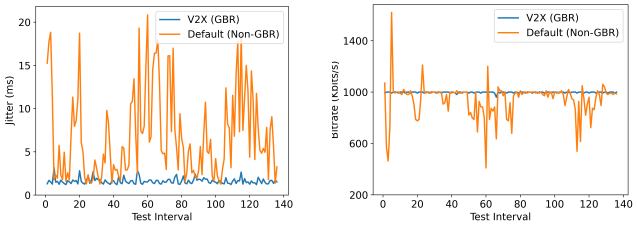


Fig. 6: Network action evaluation testbed, using srsRAN



(a) Jitter (b) Bitrate

Fig. 7: Comparison between V2X (GBR) and Default (non-GBR) UEs

service. Table IV reinforces the results shown in the plots, emphasising the standard deviation that reflects the stable connection that a GBR profile provides in both the jitter and bitrate tests. Also, the minimum and maximum values show that the edge cases are way more steep when a service has no resource allocation. This shows the impact of our solution for each service. Rather than depending on a generic QoS profile assigned by operators to each user, our solution demonstrates that using metrics collected by an xAPP on the RAN, it facilitates the dynamic configuration of the QoS profile. This dynamic approach enhances the management of 5G resources for operators.

V. Conclusion and Future Work

This paper proposed an evaluation and implementation of O-RAN frameworks, together with the development of an orchestrator capable of collecting metrics directly from the RAN, using a monitoring xAPP, and stream them through a ML pipeline for QoS profiling. The deployed orchestrator consistently predicts and assigns user profiles based on the users' static or dynamic traffic conditions. The obtained results demonstrate the accuracy of this prediction concerning the diverse traffic generations traversing the network. It was also observed that these profiles' impact on the network, optimising resources such as jitter, bitrate, and Round Trip Time (RTT).

The limits of what can be done in the current state of open-source RAN and RIC was explored. Monitoring xAPP applications provide a comprehensive set of metrics from MAC, RLC, PDCP, but some are not implemented in hardware platforms and remain static. Furthermore, due to this absence of hardware support, the control of xAPPs on the RAN is not feasible. On the core, a more in-depth study of open-source User Plane Function (UPF) QoS feature can be done, and the integration of the orchestrator as an AF, interacting directly with the Policy Control Function (PCF), and making use of the CN's APIs to change the user session.

TABLE IV: Jitter (ms) and Bitrate (kbits/s) statistics comparing V2X (GBR) and Default (non-GBR) UEs.

Test	Scenario	Mean	Std Dev	Min	$P_{75\%}$	Max
Jitter	GBR	1.61	0.36	1.20	1.73	3.20
	Non-GBR	6.86	5.07	1.26	9.50	20.84
Bitrate	GBR	997.40	5.02	960	1000	1000
	Non-GBR	941.93	134.62	410	1000	1620

Acknowledgment

This work was supported by the European Union/Next Generation EU, through Programa de Recuperação e Resiliência (PRR) [Project Nr. 29: Route 25 (02/C05-i01.01/2022.PC645463824-00000063)].

References

[1] A. Kak, V. Q. Pham, H. T. Thieu, and N. Choi, "ProSLICE: An Open RAN-based approach to Programmable RAN Slicing," in *2022 IEEE Global Communications Conference (GLOBECOM)*, 2022, pp. 197–202.

[2] A. Kak, V.-Q. Pham, H.-T. Thieu, and N. Choi, "Demo: A Disaggregated O-RAN Platform for Network Slice Deployment and Assurance," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2022, pp. 1–2.

[3] R. Ferreira *et al.*, "Demo: Enhancing Network Performance based on 5G Network Function and Slice Load Analysis," in *IEEE 24th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2023, pp. 340–342.

[4] A. Lacava *et al.*, "Programmable and Customized Intelligence for Traffic Steering in 5G Networks Using Open RAN Architectures," *IEEE Transactions on Mobile Computing*, 2023.

[5] A. M. Nagib *et al.*, "Safe and Accelerated Deep Reinforcement Learning-based O-RAN Slicing: A Hybrid Transfer Learning Approach," *IEEE Transactions on Communications*, vol. 70, no. 11, 2022.

[6] G. Z. Bruno, V. K. Radhakrishnan, G. M. Almeida, A. Huff, A. P. da Silva, K. V. Cardoso, L. A. DaSilva, and C. B. Both, "RIC-O: An Orchestrator for the Dynamic Placement of a Disaggregated RAN Intelligent Controller," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2023, pp. 1–2.

[7] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "ColO-RAN: Developing Machine Learning-based xApps for Open RAN Closed-loop Control on Programmable Experimental Platforms," *IEEE Transactions on Mobile Computing*, 10 2022.

[8] Open Networking Foundation, "SD-RAN, ONF's Software-Defined RAN Platform Consistent with the O-RAN Architecture White Paper," 2 2020. [Online]. Available: https://opennetworking.org/wp-content/uploads/2020/03/SD-RAN-White-Paper.pdf

[9] R. Schmidt, M. Irazabal, and N. Nikaein, "FlexRIC: an SDK for next-generation SD-RANs," in *Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 411–425.

[10] P. Valente, *et al.*, "Disaggregated Mobile Core for Edge City Services," in *2023 IEEE 24th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2023, pp. 157–166.

[11] TSGS, "Ts 123 501 - v15.8.0 - 5g; system architecture for the 5g system (5gs) (3gpp ts 23.501 version 15.8.0 release 15)," 2020. [Online]. Available: https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

[12] P. Rito *et al.*, "Aveiro Tech City Living Lab: A Communication, Sensing, and Computing Platform for City Environments," *IEEE Internet of Things Journal*, vol. 10, no. 15, pp. 13 489–13 510, 2023.