xSTART: xApp Simulated Evaluation Environment for Developers

Juan Luis Herrera Gonzalez, Sofia Montebugnoli, Domenico Scotece, Luca Foschini
Department of Computer Science and Engineering, University of Bologna, Bologna, Italy
Email: {juanluis.herrera, sofia.montebugnoli3, domenico.scotece, luca.foschini}@unibo.it

***Abstract*—The advent of the Open Radio Access Network (Open RAN) in 5G, as delineated in the standards introduced by 3GPP and O-RAN Alliance, posed a pivotal shift in revolutionizing the telecommunications landscape. Central to this transformation is the Open RAN control-plane architecture. Control plane encompasses containerized network functions, operating as intelligent controllers for RAN nodes and resources with non-Real-Time and near-Real-Time constraints. These functions, called Near Real-Time RAN Intelligent Controller (Near-RT RIC) and Non Real-Time RAN Intelligent Controller (Non-RT RIC), include operators-defined applications, respectively named xApps and rApps, serving as common cloud-native applications controlling and optimizing Open RAN elements. Understanding the entire development process and runtime behaviour for these applications becomes an essential prerequisite for providing support for next-generation networks. However, the lack of ready-to-use tools that allow developers to test and evaluate their applications in simulated and real environments makes it difficult to study and experiment with xApps. In this work, we propose *xSTART*, a ready-to-deploy environment based on Docker technology that allows xApp developers to quickly deploy and incorporate their xApps in a simulated ns-3 environment to test and evaluate their functionalities. Then, we evaluate an IIoT Network use-case scenario showing a machine learning (ML)-based xApp applied to the O-RAN environment. The results compare the use of different ML techniques and show the correct behavior of the simulation. Finally, we make xSTART available to the community to ease the development and evaluation of future xApps.**

***Index Terms*—Open RAN, xApp, DevOps, Industrial IoT, 5G**

I. Introduction

The Industrial Internet of Things (IIoT) revolutionizes industrial processes by integrating a multitude of heterogeneous devices and sensors within a single network [1]. These devices, generate a vast amount of data critical for monitoring and optimizing industrial operations. To ensure seamless data collection and high-performance communication, IIoT networks often leverage network standards like 5G, offering elevated Quality of Service (QoS). However, achieving high availability and avoiding production downtime remains paramount in industrial scenarios. This necessitates the utilization of predictive tools capable of analyzing and testing network behaviour under various conditions [2].

Network simulation emerges as a powerful solution for evaluating innovative network implementations within the IIoT 5G domain [3]. However, deploying existing simulation tools requires several efforts, particularly for Open Radio Access Network (Open RAN) simulations, which include numerous components like gNodeBs (gNBs), the Near-Real Time RAN Intelligence Controller (Near-RT RIC), xApps, Non-Real Time RAN Intelligence Controller (Non-RT RIC), rApps, and all the involved interfaces. In particular, Near-RT RIC simulations offer a valuable testing ground for operator-defined control applications, called xApps, which are designed to quickly learn and adjust the behaviour of RAN components to User Equipment (UE) devices, like IIoT sensors, within an industrial environment [4]. Nonetheless, existing simulation tools often lack the necessary ease of deployment and adaptability required for efficient testing across diverse industrial networks [5].

This paper addresses these challenges by proposing xSTART, a novel environment specifically designed for the easy deployment of 5G Open RAN simulations in IIoT network contexts. xSTART prioritizes rapid deployment across heterogeneous industrial settings, ensuring adaptability to a wide range of network configurations. Furthermore, we propose novel Machine Learning (ML)-based xApps specifically tailored for seamless integration with network data, facilitating efficient data processing and analysis within the simulated network environment. Through the implementation of our proposed framework and ML-based xApps, we aim to empower researchers and engineers to effectively test and optimize network performance within simulated IIoT and industrial network environments, ultimately contributing to enhanced reliability and efficiency in modern industrial operations.

The remainder of this paper is structured as follows. Section II provides the necessary background to fully understand the proposal as well as an overview of the state of the art in O-RAN. Section IV describes the architecture of xSTART and details our deployment proposal for a simulated evaluation environment in IIoT networks. Section V shows a set of performance results, showing the usefulness of xSTART through the evaluation of ML-based xApps. Finally, Section VI concludes the paper.

II. Background

In this section, we briefly introduce background concepts about the Open RAN framework and the Near-RT RIC architecture, in Section II-A. Finally, we discuss the research directions of the literature in the area of Open RAN simulation in Section II-B.

A. Open RAN and Near-RT RIC Architecture

The definition of Open RAN represents a revolutionary approach within the realm of RANs, fostering seamless interoperability among cellular network equipment from diverse vendors. It pursues a revolution of the conventional, hardware-focused design of RANs by embracing a modular framework comprised of distinct components, each featuring open and standardized interfaces. To accommodate more flexibility in the design of RANs, the O-RAN Alliance has developed O-RAN protocols, allowing the base stations or gNBs to be split into three different modules and their protocol layers [6], [7]. In particular, the Radio Unit (RU) processes the Radio Frequency and lower part of the physical layer, while the Distributed Unit (DU) takes on tasks of the upper part of the physical layer, Medium Access Control, and Radio Link Control. Finally, the Central Unit (CU) manages the Packet Data Convergence Protocol, Service Data Adaptation Protocol, and Radio Resource Control entities.

The Near-RT RIC constitutes a network function within the Open RAN architecture. It is designed to deliver near real-time control and optimization of services and resources associated with the E2-managed gNBs [8]. The primary functionalities of the Near-RT RIC encompass fine-grained data collection through the E2 interface, which involves gathering data about the gNBs, and fast data aggregation and inference within xApps. The collected data undergoes rapid aggregation and analysis using xApps, that leverage inference techniques to extract meaningful insights from the data. Moreover, the Near-RT RIC encompasses data exchange with Non-RT RIC via the A1 interface [9]. This enables the Non-RT RIC to perform elaborate analysis on the Near-RT RIC data, albeit without the constraints of real-time processing. Nonetheless, due to its smaller timeframe (10-1000 ms), which is of higher relevance in the IIoT environment, and its higher Technology Readiness Level [10], we focus on the Near-RT RIC and xApps. Essentially, the Near-RT RIC plays a pivotal role within the RAN control loop, since it facilitates the steering of RAN behavior through the implementation of xApps, which allow O-RAN operators and xApp developers to have full control of the RAN. Moreover, The reason for the importance of near-RT RIC for IIoT networks lies in the shared time constraints [4]. Industrial networks require efficient automation, monitoring, and optimization of industrial processes, alongside the sensitivity of protocols that are required to maintain consistent communication amongst devices without interruption [1]. The majority of this communication is microsecond dependent, making the xApps the best container for hosting low latency network-centered industrial applications in the 5G stack [4]. Figure 1 illustrates the interaction of Open RAN components with industrial machinery.

B. Network Simulation

Communication simulators are pivotal tools for industrial experimentation, enabling the exploration of diverse network architectures, configurations, and controlled environments.

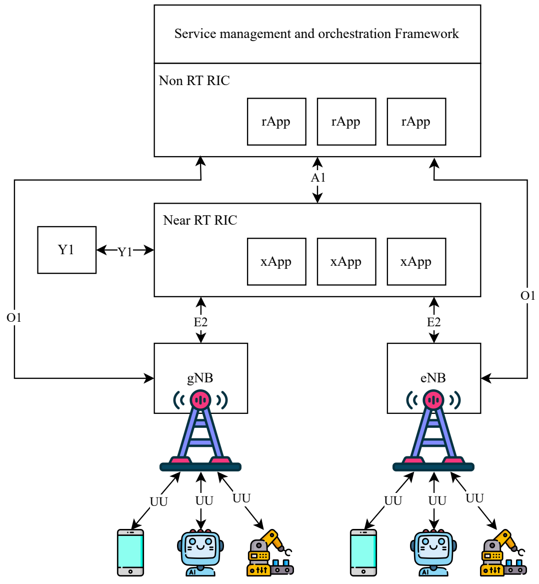


Fig. 1: O-RAN Architecture in an industrial scenario, with network functions integration with industrial devices and machinery

They furnish a platform for analyzing network behavior, identifying potential bottlenecks, and evaluating the performance of network protocols across a spectrum of network conditions, all without the need for costly physical deployments. Network simulation is crucial in industrial networks since is capable of preventing chaotic behaviors in which the introduction of small changes in the network architecture can cause major communication fails, offering a testbed for the overall deployment. Within the context of O-RAN, the Near-RT RIC, and xApps, network simulation becomes a key enabler. As xApps are being developed, it is necessary to test them with *instrumented tests*, i.e., using a similar environment to its real application. Moreover, especially when xApps that rely on automated techniques, such as ML, are being developed, it becomes key to evaluate them in a comparable environment, both to assess their performance compared to other systems and to obtain some estimations on their future performance in the real O-RAN IIoT network. However, performing these tests and evaluations in a real environment can prove complicated and costly, as one would need to build a separate testbed that replicates the real O-RAN IIoT network. Furthermore, not performing these tests can be even worse, as deploying xApps with unknown performance and behavior to the production IIoT network can prove fatal for its effectiveness. Hence, using a network simulator for the testing and evaluation of xApps can be a relevant alternative to a real testbed.

III. Related works

Recent academic interest has led to the development of evaluation environments tailored for xApp deployment and their assessment within the Near-RT RIC framework. In this section, we review pertinent literature contributing to xApp development and integration within this ecosystem.

In their paper [11], the authors delve into the challenges confronted by third-party application developers within the Open RAN paradigm. Their study conducts a comparative analysis of various open platforms designed for xApp development and testing, elucidating encountered obstacles.

Meanwhile, [10] presents a comprehensive framework for the implementation and deployment of xApps within the Near-RT RIC environment. This framework encompasses xApp lifecycle management facilitated through an xApp software development kit and HELM charts. Furthermore, the study evaluates the platform utilizing metrics such as session establishment time, configuration establishment time, and indication processing time.

Furthermore, in [12], the authors explore the design considerations for AI/ML applications aimed at intelligent closed-loop control of the Open RAN. This paper offers practical guidelines and insights drawn from exemplary solutions illustrating the integration of AI/ML applications into xApps instantiated on the O-RAN Near-RT RIC. Notably, the authors introduce OpenRAN Gym, a toolbox facilitating data-driven O-RAN experimentation at scale.

Our study introduces an assessment framework for xApps based on Docker for easily incorporating xApps in Near RT RIC, leveraging the advantages of isolation and reproducibility offered by containerization. Moreover, our experimentation on an ML xApp specifically tailors to an IIoT scenario. Therefore, we employ simulation to evaluate the performances of xApps under real-world conditions, before their release.

IV. xSTART: O-RAN Simulated Environment

To support the developers of xApps in IIoT O-RAN networks, we propose the xSTART evaluation environment. The aim of xSTART is, on the one hand, the provision of a unified environment where developers can test and compare their xApps and, on the other hand, the simplification of the testing and evaluation process of xApps. We envision a future where developers of IIoT-oriented xApps are tested and evaluated using xSTART, in a similar manner to how current developers test and evaluate their Android apps with the AVD simulated environment [13]. The overall architecture of xSTART is shown in Fig. 2, and each of its components will be detailed in the remainder of this section. Concretely, Section IV-A provides details on the xApp development and usage with xSTART, while Section IV-B presents the near-RT RIC and its components, and Section IV-C details how the RAN is simulated.

A. xApps

The first element we find, at the top of Fig. 2, are three xApps. It is important to note that xSTART does not impose any concrete constraints on the number of xApps that may be deployed in its environment: it is possible to evaluate a single xApp or two xApps, as well as more than three xApps, within a single xSTART environment. xApps are an important part of the architecture, as, while one example xApp is provided with xSTART, xApps are not envisioned to be directly part of the environment. Instead, xApp developers should be the ones that provide the xApp software, simply integrating it with the existing xSTART environment.

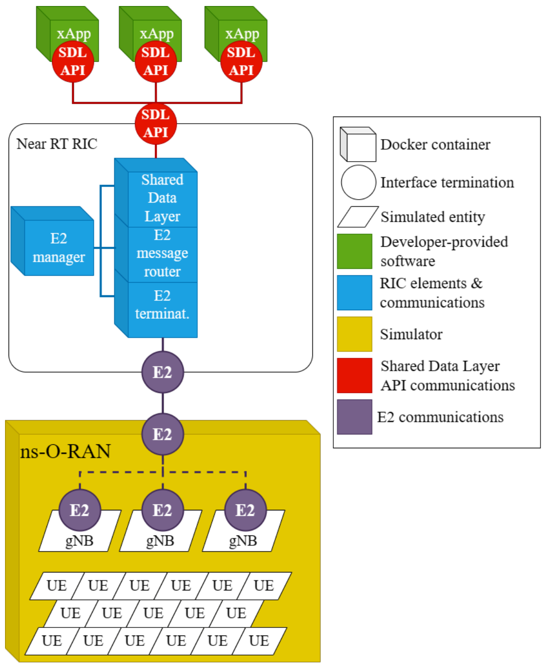


Fig. 2: Architecture of the xSTART environment with 3 sample xApps.

To maintain a clean environment, xSTART strongly recommends developing and deploying xApps as illustrated: each xApp should be provided in a separate Docker container, neither deploying multiple xApps in a single container nor deploying a single xApp that deploys multiple containers. xApps that need to communicate with each other should either do it using the RIC, or using external, out-of-band communications, but it is not recommended to perform these communications between processes in the same container. Similarly, if an xApp must communicate with multiple components, it is recommended to make a single container with the xApp and its dependencies. These are, nonetheless, recommendations that are not directly enforced by xSTART, as we understand there may be niche use cases where the one-to-one relationship between containers and xApps cannot be achieved or are otherwise undesirable (e.g., multiple xApps may need to communicate with common components to share information, and thus, these common components are given an additional container that xApps must communicate with).

Similarly, xSTART does not impose any constraint on the programming language or frameworks used to develop xApps. On the one hand, this design decision is aimed at enabling xSTART to cover a wide range of use cases, not necessarily limited to ML-based xApps implemented with traditionally ML-aimed technologies (e.g., Python, TensorFlow). On the other hand, it is also aimed at minimizing the limitations to the development of new xApps, ensuring they are not constrained to any given framework or programming language and can freely choose the tools that are best suited for their use case. To be compatible with xSTART, these xApps must, nonetheless, be able to communicate with the near-RT RIC through its' Shared Data Layer (SDL). In the case of xSTART, the API offered by the near-RT RIC's SDL is the Redis API [14]. Hence, as long as the xApps can provide communication with the Redis API, either through libraries for Redis, xApp development frameworks or by directly implementing it, they can be tested and evaluated with xSTART.

B. Near-RT RIC

The next elements we find in Fig. 2 are encompassed in a block: the *near-RT RIC*. From the perspective of a developer, the near-RT RIC must fulfill two roles: it must provide xApps with information about the RAN through the SDL, and it must also serve as a broker between the xApps and the RAN. xApps should, thus, not be concerned with the RAN itself (e.g., controlling directly each gNB, filtering the messages aimed at the xApp from messages for other xApps, implementing and controlling the E2 interfaces, getting notifications on messages from the RAN), and it should just interface with it through the SDL to gather the necessary information and subscribe to the relevant messages. It is the near-RT RIC's responsibility to ensure the information is gathered and updated from the RAN, that messages from xApps are delivered to the correct gNBs, that messages from the RAN are routed only to the xApps that subscribed to them, and to implement, control, and maintain an E2 termination to control the RAN. For this task, xSTART makes use of the state-of-the-art ColO-RAN RIC [14]. This RIC was chosen for three reasons: it is the state-of-the-art RIC that best supports developer-provided software, it provides an easier deployment, as it only requires Docker, and it is fully O-RAN compliant [14]. The ColO-RAN RIC comprises four different elements, each deployed in a separate Docker container, and its role is detailed in the remainder of this section.

The first element of the ColO-RAN RIC is the SDL itself. It has two roles: on the one hand, it serves as a traditional SDL, storing information from the RAN as the RIC gathers it and updating it accordingly, and on the other hand, it offers a publish-subscribe mechanism that is leveraged by xApps to communicate with the RIC. In the case of ColO-RAN, this SDL is a Redis instance, and hence the SDL API is the Redis API. The next element is the E2 message router, which determines the gNB or gNBs to which each xApp message should go, as well as which xApp or xApps should receive each of the messages sent by the RAN. These two elements, however, do not collaborate directly, and instead, the E2 manager is the element that controls it. The E2 manager can be seen as the overall controller of the RIC: it must control the E2 communications, put the messages through the router, and send them to the SDL appropriately. Moreover, obtaining and updating the information stored in the SDL is also the role of the E2 manager. Finally, the E2 interface is not directly implemented in the E2 manager or the E2 message router. Instead, a dedicated E2 termination element is tasked with the implementation of the E2 interface, including its connection to the E2 terminations of gNBs. This design allows the rest of the near-RT RIC modules to remain independent: the E2 termination provides the gNBs in the RAN for a single endpoint to connect to, regardless of whether the ColO-RAN RIC elements are deployed in the same node or different, distributed notes.

C. RAN simulation

The final component of xSTART is the RAN simulator. This simulation is key for xSTART to fulfill its purpose, as it gives xApps a realistic environment where they can be tested. Moreover, as the xApps are interfacing with an O-RAN compliant near-RT RIC, which at the same time interfaces with the simulated RAN, xApps tested in xSTART should also work in real O-RAN testbeds: if the simulated RAN is changed by a real one, the near-RT RIC will be able to manage it, and thus, the xApps will not experience any change, as they only interface directly with the RIC.

To play the crucial role of simulation in xSTART, we propose the use of a state-of-the-art simulator: ns-O-RAN [3]. This simulator is based on the well-known ns-3 simulator, a prominent open-source, discrete-event network simulator tailored to replicate modern communication networks and protocols. Distinguished by a modular, object-oriented design, ns-3 facilitates the creation of bespoke network components and protocols. It offers realistic modeling capabilities for various network elements, including routers, switches, links, and channels, configurable to emulate real-world attributes such as bandwidth, delay, and packet loss. Additionally, ns-3 accommodates the instantiation of traffic patterns and movement patterns for the users. The simulation process encompasses the collection of detailed statistics on diverse network metrics on the data plane, comprehending throughput, delay, jitter, and packet loss. ns-O-RAN provides additional functionalities to ns-3 that enable it to simulate O-RAN networks. In terms of the simulation of the RAN environment, ns-O-RAN builds on top of the ns-3-mmWave modules, which provide the simulation of the data plane of RAN networks, from the simulation of UEs and their antennas to the simulation of the physical plane of the RAN and the network core. Moreover, as it is integrated with ns-3, other traditional networks, such as IP networks, can also be simulated as part of a scenario. ns-O-RAN also provides the O-RAN interface to ns-3, simulating not only its data plane but its control plane. gNBs simulated within ns-O-RAN can provide information and receive commands through a simulated E2 interface, which is also O-RAN compliant.

However, the simulation is a single process, and as such, ns-O-RAN is a single Docker container, and thus, it can only expose one E2 interface. To solve this issue, ns-O-RAN integrates a modified version of the E2 simulator *e2sim*, which enables this single termination to become a *multiplexed* E2 termination. Hence, as depicted in Fig. 2, there is no need for each gNB to have its own E2 interface. Instead, the simulated E2 interfaces of all gNBs are all connected to a single E2 termination, ready to manage this multiplexed situation. This design is very interesting for our purpose as, while the data plane is fully simulated, the control plane is partially simulated, and partially implemented in a real manner. Therefore, O-RAN-compliant software can directly control the simulated RAN without the need for specific integration with ns-3 or ns-O-RAN. Instead, they can be fully decoupled and evaluated as if the simulated RAN was a real testbed.

In summary, by simulating a RAN using ns-O-RAN, which provides a realistic control plane implementation, and by making use of the ColO-RAN RIC, which serves as a broker between the RAN and the xAppps, it is possible to perform realistic testing and evaluation of xApps, and thus, xSTART can provide a valuable environment for xApp developers.

V. Experimental Results

This section pursues the validation of xSTART by showing evaluation results within START of different ML-based xApps, aimed at comparing the performance and precision of ML techniques in the prediction of RAN measurements. The environment used for this evaluation is discussed in Sec. V-A, while the obtained results are described in Sec. V-B.

A. Evaluation setup

To evaluate xSTART, we developed different xApps whose role is to predict the average Signal-to-Interference-plus-Noise Ratio (SINR) experienced UEs in a cell, using as features the timestamp and the ID of the physical cell. These xApps would allow, within the IIoT field, to predict interferences in given cells to lead to more efficient handovers, hence maintaining the necessary QoS. There are, however, different ML models that could perform such prediction tasks, and it is complex to know which one is most suitable *a priori*. Hence, we developed a total of 6 xApps, each of them able to perform these predictions through a different machine learning model: Ridge, Support Vector Regressor (also known as Support Vector Machine), K-Nearest Neighbors, Radius-based K-Nearest Neighbors, Multi-Layer Perceptron (also known as Neural Network), and Histogram-based Gradient Boosting. Each of these models has been trained using data obtained from the simulated RAN, and their hyperparameters have been tuned using ten-fold cross-validation techniques to ensure their quality. Finally, these xApps have been evaluated using xSTART's default configuration. The xSTART implementation used for this evaluation can be publicly accessed in its repository[1].

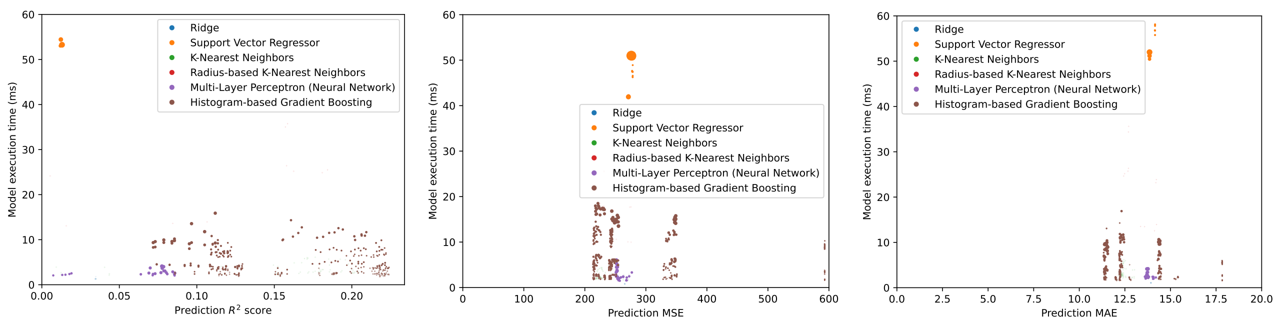
The objective of this evaluation is to show the usefulness of xSTART in the evaluation of xApps. To do so, however, it is necessary to rely on the evaluation of xApps themselves. Hence, the secondary objectives of the evaluations are to assess the precision and execution times of the six evaluated ML models using xSTART, to thus show the usefulness of xSTART. Nonetheless, it is key to understand that the scoring of the ML models themselves, whether they are high or low, are not relevant in our evaluation of xSTART, as our proposal are not ML models to perform these predictions, but the xSTART framework.

[1]https://gitlab.com/MMw_Unibo/o-ran-public/xstart

B. Evaluation results

The obtained results are depicted as scatter plots in Fig. 3. Each of the predictions made by an ML model is shown as a point within the scatter plot, where the size of the point refers to the time taken to train the model. On the other hand, the time represented on the Y-axis refers to the time the model requires to execute, i.e., to perform a prediction. Finally, the X-axis depicts the score of the predictions in terms of different metrics: R^2 score in Fig. 3a, Mean Square Error (MSE) in Fig. 3b, and Mean Absolute Error (MAE) in Fig. 3c. The first model we find is thus the Support Vector Regressor, which exhibits the worst results across all six models: its predictions have an average R^2 score of 0.012, lower than any other model, with an MSE of 16002.81 and an MAE of 49.41, as some of the values do not appear on the plots due to their huge difference to the rest. Moreover, it is the slowest model to train (13.33 seconds on average), as well as the one providing the slowest predictions (53.58 ms on average, 2.03 ms standard deviation). The next model, in this case, would be the Ridge model, which, although still the second-worst model, provides much better precision than the Support Vector Regressor (0.035 average R^2 score, 265.41 MSE, 13.93 MAE), while being the fastest model to train (2 ms on average) and among the fastest providing predictions (1.35 ms on average, 0.25 ms standard deviation). The next model in terms of precision is the Multi-Layer Perceptron, with an average R^2 score of 0.063, an average MSE of 257.84, and an average MAE of 13.93. It is, however, notably slow to train, being the second slowest model to train (3.422 s on average), although fairly fast providing its predictions (2.71 ms on average, 0.29 ms standard deviation). Radius-based K-Nearest Neighbors is next, with a two-fold improvement in precision (0.121 average R^2 score, 247.24 MSE, 12.28 MAE). It is also fast to train this model (3.12 ms on average), although it is the second slowest model to predict (23.75 ms on average, 1.16 ms standard deviation). The second best model in terms of precision is K-Nearest Neighbors, with an average R^2 score of 0.160, an average MSE of 230.93, and an average MAE of 12.43, while being both fast to train (2.87 ms on average) and use (3.49 ms on average, 0.19 ms standard deviation). Finally, in a similar level of precision is the Histogram-based Gradient Boosting model, with an average R^2 score of 0.161, an average MSE of 309.26, and an average MAE of 13.32. It is, however, notably slower to train (719.94 ms on average) and on its predictions (4.93 ms on average, 0.61 ms standard deviation) than the K-Nearest Neighbors model.

In summary, to provide a highly reconfigurable, ML-based xApp for SINR prediction in O-RAN, K-Nearest Neighbors is the best overall model, nearly as precise as Histogram-based Gradient Boosting, but much faster to train and use. These insights are only possible thanks to xSTART, and thus, the usefulness of the environment for the evaluation of xApp has been made clear, achieving the objective of the evaluation.



(a) R^2 score (b) Mean Square Error (c) Mean Absolute Error

Fig. 3: Evaluation of the precision and execution time of the xApp.

VI. Conclusion

In summary, this paper addressed the challenge of providing xApp developers in the industrial domain with an environment to test and evaluate their xApps. To do so, we propose xSTART, an integrated, simulation-based environment for the testing and evaluation of xApps available to the community. The evaluation using ML-based xApps has shown the usefulness of xSTART during xApp development, testing, and evaluation, especially for ML-based xApp aimed at IIoT O-RAN networks. We expect xSTART to become a tool that will not only make the development of xApps easier but also enhance the adoption of O-RAN as a whole within IIoT. The first version of the xSTART simulator is available to the community at the link: *https://gitlab.com/MMw_Unibo/o-ran-public/xstart*. In the future, we plan to release a further version of xSTART that switches the ns3-based simulation environment with the OAI 5G RAN emulation environment.

Acknowledgment

This work was partially supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART") CUP: J33C22002880001

References

[1] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial internet of things: Challenges, opportunities, and directions," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 11, pp. 4724–4734, 2018.

[2] A. Mahmood, L. Beltramelli, S. Fakhrul Abedin, S. Zeb, N. I. Mowla, S. A. Hassan, E. Sisinni, and M. Gidlund, "Industrial iot in 5g-and-beyond networks: Vision, architecture, and design trends," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 6, pp. 4122–4137, 2022.

[3] A. Lacava, M. Bordin, M. Polese, R. Sivaraj, T. Zugno, F. Cuomo, and T. Melodia, "ns-o-ran: Simulating o-ran 5g systems in ns-3," in *Proceedings of the 2023 Workshop on ns-3*, ser. WNS3 2023. ACM, Jun. 2023. [Online]. Available: http://dx.doi.org/10.1145/3592149.3592161

[4] T. F. Rahman, M. Zhang, and V. Marojevic, "O-ran perspective on industrial internet of things: A swot analysis," in *2023 IEEE International Conference on Industrial Technology (ICIT)*, 2023, pp. 1–6.

[5] D. Morato, C. Pérez-Gómara, E. Magaña, and M. Izal, "Network simulation in a tcp-enabled industrial internet of things environment-reproducibility issues for performance evaluation," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 2, pp. 807–815, 2022.

[6] O-RAN Working Group 2, "O-RAN Non-RT RIC Architecture 1.0," O-RAN Alliance, Technical specification (TS) O-RAN.WG2.Non-RT-RIC-ARCH-TS-v01.00 Technical Specification, 2021.

[7] O-RAN Working Group 3, "O-RAN Near-RT RAN Intelligent Controller Near-RT RIC Architecture 2.00," O-RAN Alliance, Technical specification (TS) O-RAN.WG3.RICARCH-v02.00, 2021.

[8] O-RAN ALLIANCE e.V., "O-ran work group 1 (use cases and overall architecture), o-ran architecture description," in *Proceedings of the 2023 Workshop on ns-3*. O-RAN Alliance, Feb. 2023. [Online]. Available: https://orandownloadsweb.azurewebsites.net/specifications

[9] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding o-ran: Architecture, interfaces, algorithms, security, and research challenges," 2022.

[10] H.-T. Thieu, V.-Q. Pham, A. Kak, and N. Choi, "Demystifying the near-real time ric: Architecture, operations, and benchmarking insights," in *IEEE INFOCOM 2023-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2023, pp. 1–8.

[11] M. Hoffmann, S. Janji, A. Samorzewski, L. Kulacz, C. Adamczyk, M. Dryjański, P. Kryszkiewicz, A. Kliks, and H. Bogucka, "Open ran xapps design and evaluation: Lessons learnt and identified challenges," *IEEE Journal on Selected Areas in Communications*, vol. 42, no. 2, pp. 473–486, 2024.

[12] L. Bonati, M. Polese, S. D'Oro, S. Basagni, and T. Melodia, "Intelligent closed-loop ran control with xapps in openran gym," in *European Wireless 2022; 27th European Wireless Conference*, 2022, pp. 1–6.

[13] Google, "Android virtual device (avd) as a development platform," 2024. [Online]. Available: {https://source.android.com/docs/automotive/start/avd/android_virtual_device}

[14] 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*, 2022.