ACCEPTED FROM OPEN CALL

O-RAN Performance Analyzer: Platform Design, Development, and Deployment

Mohammadreza Kouchaki, Seyed Bagher Hashemi Natanzi, Minglong Zhang, Bo Tang, Vuk Marojevic

The authors introduce a comprehensive network performance analyzer, tailored for the near-real time radio access network intelligent controller.

Abstract

The open radio access network (O-RAN) represents a paradigm shift in RAN architecture, integrating intelligence into communication networks via xApps — control applications for managing RAN resources. This integration facilitates the adoption of AI for network optimization and resource management. However, there is a notable gap in practical network performance analyzers capable of assessing the functionality and efficiency of xApps in near real-time within operational networks. Addressing this gap, this article introduces a comprehensive network performance analyzer, tailored for the near-real time RAN intelligent controller. We present the design, development, and application scenarios for this testing framework, including its components, software, and tools, providing an end-to-end solution for evaluating the performance of xApps in O-RAN environments.

Introduction

The Open Radio Access Network (O-RAN) builds on the foundations of the Open RAN concept. Open RAN provides the notion of openness, transparency, and flexibility for spurring innovation in the cellular industry. O-RAN refers to a specific architecture, components, processes, and interfaces, defined and maintained by the O-RAN Alliance for enabling standardized development, testing, integration, deployment, and operation. The goals of O-RAN are facilitating vendor diversity and network intelligence through open interfaces, network softwarization and virtualization, and artificial intelligence/machine learning (AI/ML) [1]. The closed, rigid structure of the traditional radio access network (RAN) has historically stifled innovation and posed entry barriers for newcomers. In contrast, O-RAN's openness and disaggregation transforms the protocols and interfaces governing RAN components, thereby fostering greater flexibility and innovation when compared to conventional RAN implementation [2]. Service providers perceive O-RAN as an opportunity to tailor their RAN networks to their precise needs, allowing them to selectively incorporate components that align with their domain expertise. This flexibility also encourages collaboration with third-party vendors, thus fostering diversity in RAN deployment.

The O-RAN architecture, as shown in Fig. 1, includes the Radio Unit (O-RU), a logical node situated near the antenna, hosting the lower part of the physical (PHY) layer and radio frequency (RF) processing based on the lower layer functional split and communicates with the Distributed Unit (O-DU) via the fronthaul interface. The O-DU implements lower-layer protocols including Medium Access Control (MAC), the upper part of the Physical (PHY) layer, and Radio Link Control (RLC). This unit is connected to the Central Unit (O-CU) for higher protocol layer processing through the midhaul interface [1].

The O-RAN architecture introduces two logical RAN Intelligent Controllers (RICs), operating at two different timescales: The near-real time (RT) RIC serves as the platform for deploying software solutions that enable near-RT control actions through xApps. It facilitates communication with RAN components via the E2 interface. The near-RT RIC supports actions with a 10 ms–1 s latency to promptly adapt to changing network dynamics. It comprises an internal messaging infrastructure, known as the RIC Message Router (RMR) [1], and an internal database that can be integrated with a Shared Data Layer (SDL) for data sharing among xApps. It offers services for handling security, managing subscriptions, resolving conflicts, and enabling logging.

In contrast to the near-RT RIC, the non-RT RIC hosts rApps, and its principal functions include long-term network optimization, resource management, higher layer procedures, policy optimization, and AI/ML model training at a time scale of above one second. Beyond deploying trained models via the A1 interface for the rapid inference in the near-RT RIC, the non-RT RIC provides a platform for the continuous evolution of AI controllers. rApps leverage the R1 interface to engage in model training, inference, and updates within the non-RT RIC, aligning with the comprehensive AI/ML workflow that is essential for their operation.

There is a gap in fully automated and intelligent frameworks that facilitate analyzing the performance of O-RAN and the near-RT RIC, in particular. The Third Generation Partnership Project (3GPP) as well as the O-RAN Alliance lay out testing methods and metrics for performance and conformance certification, but these are insufficient for verifying the performance of xApps.

Mohammadreza Kouchaki, Minglong Zhang, and Vuk Marojevic are with Mississippi State University; Seyed Bagher Hashemi Natanzi and Bo Tang are with Worcester Polytechnic Institute, USA.

Digital Object Identifier: 10.1109/MCOM.002.2300822

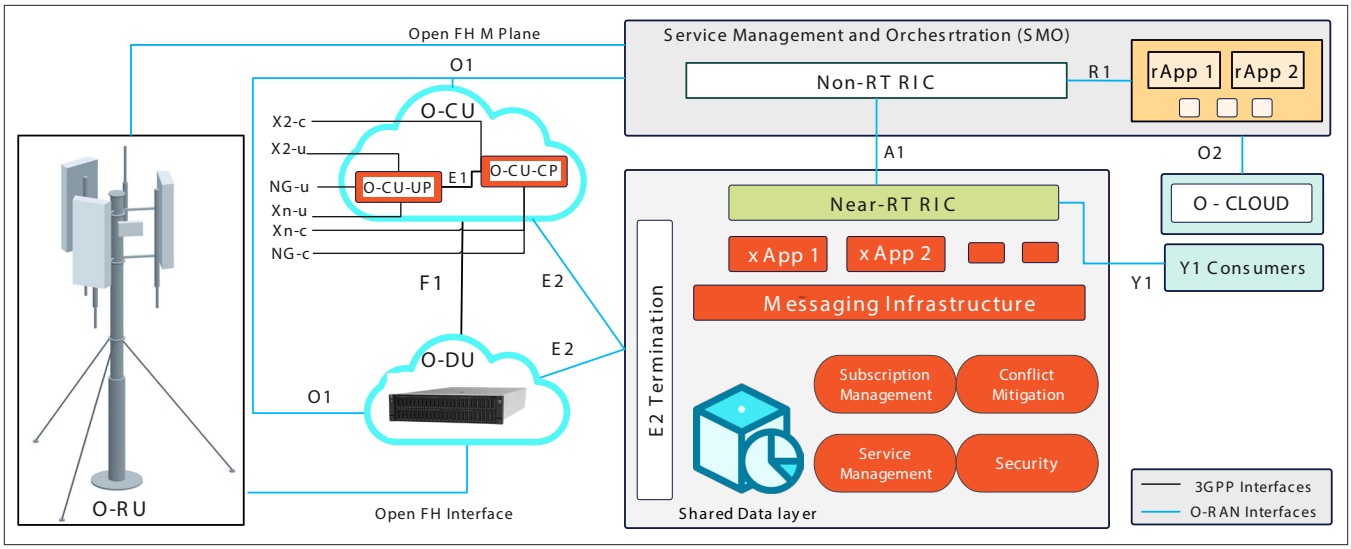

FIGURE 1. O-RAN architecture.

The complexity of open networks introduces new challenges and threats, making it increasingly difficult to identify and trace problems [3]. For instance, conflicts arising from the parallel operation of multiple RAN controller applications may lead to vendor blame-shifting.

It thus becomes imperative to advance the development of O-RAN tools and establish a comprehensive framework that facilitates the testing, analysis, monitoring, and verification of xApps deployed in O-RAN environments. This framework should not be limited to performance analysis but also facilitate defining new key performance indicators (KPIs) to track critical metrics and the operational history. Despite the existence of open-source software tools for collecting O-RAN node metrics and data, none of these are tailored to collect xApp data from the near-RT RIC, nor do they offer a platform for employing these metrics and data for comprehensive data and performance analysis.

This article addresses this gap and presents a novel O-RAN network analyzer framework, primarily focused on monitoring and evaluating the performance of xApps deployed within near-RT RIC. The proposed analyzer commences its operation cycle with the activation of an xApp within the near-RT RIC, enabling it to track alterations in network parameters and identify any significant performance or policy changes in the network resulting from the xApp's operation.

The remainder of this article is structured as follows: The next section covers background information. Following that, we explain the design principles, architecture, and key components of the O-RAN performance analyzer. Then we detail the deployment process and results, discuss challenges and future research opportunities, and finally conclude the article.

Why Do We Need an O-RAN Performance Analyzer

Existing Gaps and Challenges

The O-RAN architecture and its inherent features offer advantages in terms of interoperability, performance monitoring, data collection, and cloud integration, which are all orchestrated through open interfaces, closed-loop control, and data-driven methodologies. However, there is a lack of innovative tools for automatic network testing, performance analysis, and the introduction and collection of new KPIs. We identify the following gaps.

Performance Assessment: The integration of AI/ML modules from specialized providers may enhance RAN operations but it introduces significant complexities in comprehensively assessing the end-to-end O-RAN performance, limiting the awareness of potential issues or failures and hindering the development of AI-driven O-RAN networks [4].

Functionality Evaluation: Current testing procedures and metrics from 3GPP and O-RAN Alliance fall short in evaluating the functionality of the RICs, where the complexity of open networks adds further challenges in problem identification and tracing.

Monitoring and Analysis: There is a notable absence of a fully automated and intelligent framework and software tools for comprehensive O-RAN performance monitoring and analysis. Such tools should enable the identification and mitigation of potential threats while optimizing network performance. The open nature of O-RAN, while allowing for customization and optimization, requires a deep understanding of the underlying processes and their inter-dependencies, emphasizing the need for sophisticated performance analyzers and testing tools.

KPIs: The introduction of new KPIs will be essential for the comprehensive characterization of O-RAN performance, including metrics for capturing network node operations, energy efficiency, resource utilization, data privacy compliance, and computational and operational efficiency of xApps. Continuous monitoring and evaluation of xApp activities are imperative to prevent undesired or detrimental actions.

Related Studies

Open-source O-RAN testing and monitoring solutions have been recently introduced by industry and academia as shown in Table 1. For example, KPIMON [5], a product of the O-RAN Software Community (OSC), records a limited number of metrics from the RAN. It does not support in-depth analysis, novel performance metrics, and comprehensive network evaluation. It lacks a ded-

Framework	KPIMON [5]	ONOS-KPIMON [6]	POWDER-RAN [7]	ColO-RAN [8]	aiOS [9]	This Work
Key Features	Captures E2 Metrics	Captures E2 Metrics	Offers Experimentation	Metric Collection	Modularity	Full-Stack Testing
Data Source	Only E2 Nodes	Only E2 Nodes	Only E2 Nodes	xApp and E2 Nodes	xApp	xApp and E2 Nodes
Limitations	Lack of Analysis	Lack of Analysis	Lack of xApp Metrics	Lack of xApp Metrics	Lack of xApp Metrics	None
Functionality Comparison						
Resource Block Usage	✓	✓	✓	✓	✓	✓
Reliability	×	×	✓	✓	✓	✓
Peak Data Rate	✓	✓	×	✓	✓	✓
Spectrum Efficiency	×	×	✓	×	×	✓
Ultra-Low Latency	×	×	✓	✓	✓	✓
Extra Wide Bandwidth	×	×	✓	×	✓	✓
Network-Level Traffic Capacity	×	×	×	×	✓	✓

TABLE 1. Comparative analysis of O-RAN test platforms.

icated KPI tracking and performance monitoring dashboard, but rather offers a basic mechanism to display KPIs. Similarly, ONOS-KPIMON [6], interacting with ONOS Software-Defined RAN (SD-RAN) micro-services, focuses on recording E2 node metrics. It also limits the ability for comprehensive O-RAN performance monitoring, testing, and evaluation. While KPIMON and ONOS-KPIMON focus on RAN performance monitoring, they do not provide the granularity needed to assess xApp performance within the near-RT RIC.

To bridge these gaps, the Platform for Open Wireless Data-driven Experimental Research (POWDER) [7] provides an open-source platform and experimental capabilities for 5G, O-RAN, and other advanced wireless systems implemented on open software-defined radios (SDRs). Despite POWDER's proficiency in executing various RAN scenarios, it lacks specialized functionality for O-RAN network KPI collection and analysis. POWDER can, however, be used as a deployment platform for the O-RAN analyzer proposed in this article. ColO-RAN has contributed a thorough open-source O-RAN testing framework that allows RAN metric collection via an xApp deployed within the near-RT RIC, combined with ML solutions. Unlike over-the-air radio signaling, ColO-RAN deploys the framework on a testbed emulating the radio environment. The ColO-RAN framework offers an alternate research platform for O-RAN system prototyping and testing [8].

Other contributions, such as aiOS, have introduced AI-based platforms aimed at enabling autonomous management for Software-Defined Wireless Local Area Networks (SD-WLANs). Although aiOS employs the O-RAN architecture, its focus remains solely on SD-WLANs. There is a notable gap of O-RAN performance monitoring and analysis solutions tailored to 5G environments [9]. Open AI Cellular (OAIC) offers an open research and development platform designed for experimentation with AI-based O-RAN resource management algorithms [10]. OAIC-T provides a testing framework for O-RAN systems [11]. The platform proposed in this research is an open-source extension of OAIC-T and can be accessed through [12].

The *Sample-Collector xApp* is introduced to gather detailed metrics related to the operation of individual xApps. This includes tracking their resource usage, energy efficiency, and specific performance indicators not covered by KPIMON. Table 1 highlights this. Our work augments existing solutions, proposing an innovative and holistic approach to O-RAN testing and monitoring. By creating an innovative testing framework tailored to the O-RAN/near-RT RIC architecture and support processes, we counteract the limitations of prior approaches and provide sophisticated capabilities for testing, analysis, and monitoring of xApp behavior. The proposed framework contributes toward crafting proficient and holistic O-RAN testing and analysis tools.

O-RAN Performance Analyzer

Design Principles

Various network KPIs need to be collected to effectively analyze the performance of an O-RAN network. To this end, relevant xApps are developed for deployment on the near-RT RIC. The development involves creating containerized applications on Kubernetes [13], defining all RIC functions, and importing relevant libraries. The *xApp framework* can be used for meeting the O-RAN deployment requirements. This framework streamlines the development of xApps by providing essential functionalities, such as internal messaging, alarms, and a health-check probe handler.

KPIs

Collecting relevant KPIs is indispensable for a network performance analyzer because they reflect multi-dimensional information of a network, such as capability, efficiency, and robustness. A pivotal KPI, *xApp Access Nodes*, is introduced to evaluate the E2 nodes where each xApp conducts its operations. Another critical metric, *xApp Resource Efficiency*, assesses the energy efficiency and resource utilization of xApps, providing insights into their power consumption and resource utilization relative to their performance. *Data Privacy Compliance* gauges the extent to which xApps handle user and network data in alignment with privacy regulations and *xApp Efficiency* quantifies the computational and operational efficiency of xApps.

Architectural Components

The test framework is structured around three primary architectural components, each serving a distinct yet interconnected role.

Configuration Files and XML Test Scripts are the foundational elements that outline specific test conditions and scenarios. These files and scripts are imported into the system, setting the stage for the subsequent test procedures. They act as guiding documents, informing the Test Server on how to configure and initiate various test scenarios.

The Test Server is responsible for configuring and starting all test runs. Utilizing the input from the configuration files and XML test scripts, it acts as the command center, steering the test workflow according to predefined parameters.

Actors are modular, specialized components tasked with the execution of test operations. Actors work under the directives issued by the Test Server, executing the tasks and collecting the necessary data. Each actor is equipped with a test executor, an AI core, an actor-manager, and two adapters — simulator (SIM) and SDR.

Test Workflow

The proposed O-RAN Performance Analyzer commences its operation cycle with the activation of an xApp within the near-RT RIC. Its operational workflow is divided into a sequence of interconnected tasks: metric collection, data sharing with the test framework, test orchestration, data acquisition, in-depth analysis, and final visualization. These tasks are executed iteratively in alignment with predefined test scenarios, ensuring both comprehensive and robust analysis. The process is illustrated in Fig. 2.

To collect essential metrics, an xApp called the *Sample-Collector xApp* is developed. This xApp systematically gathers relevant metrics of the O-RAN environment and the target xApps. These metrics are harvested through the SDL and the RMR. The RMR is configured to route specific messages and data flows between the *Sample-Collector xApp* and other xApps. This involves setting up subscription models within the near-RT RIC where the *Sample-Collector xApp* subscribes to specific message types or topics generated by other xApps.

Upon establishing a connection with the *Sample-Collector xApp*, data traffic is allowed to flow through the network as per the RIC policies. Herein, the *Test Server* orchestrates the creation and execution of various testing tasks based on

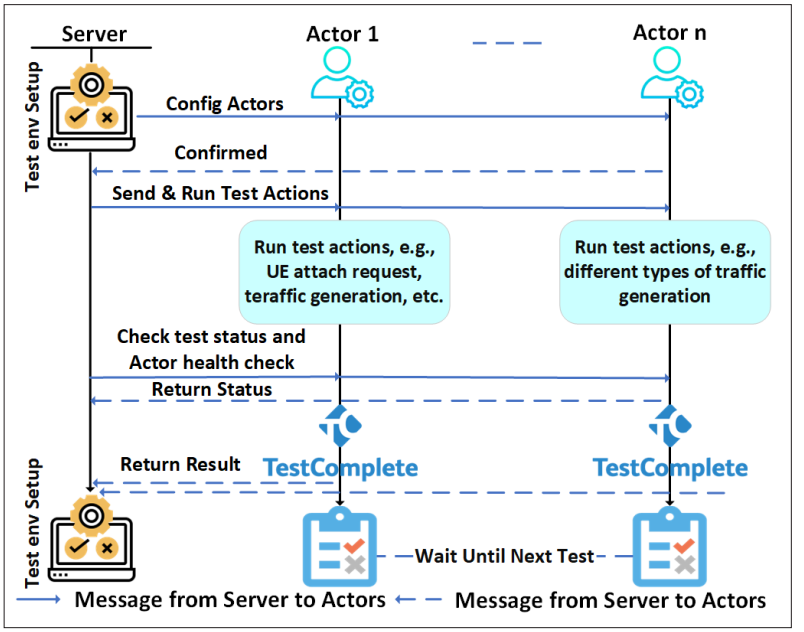

FIGURE 2. The workflow of the performance analyzer and its operation cycle [11].

pre-configured actors and the provided XML test scripts. Users have the flexibility to create an unlimited number of test scripts and actors, each identified by a unique name for management purposes.

As a final step in each iteration of the workflow, the *Test Server* is programmed to automatically generate comprehensive reports after each test cycle. These reports not only offer a holistic view of various performance metrics but also provide granular insights into the test setup, the obtained results, and the overall network performance. For ease of retrieval and future analysis, these reports are archived in designated directories.

Design Details

The design is built around the core xApp — the *Sample-Collector xApp* — as depicted in Fig. 3. This xApp has a dual purpose. On the one hand, it interfaces with the target xApp executing operations within the O-RAN nodes. On the other hand, it actively harvests the RAN metrics from operations preceding and succeeding the deployment of the target xApp. Additionally, it fetches metrics and channels them to the test platform for detailed inspection and visualization.

To achieve this, two main architectural changes are implemented. First, we integrate the SDL using InfluxDB in the near-RT RIC. Next, we enhance the KPIMON xApp, originally introduced by OSC [5]. The central goal behind this upgrade is to infuse our proposed RAN metrics and streamline the connections and data transfers to the modified SDL.

Integration of InfluxDB as SDL: There are three main steps to integrate InfluxDB in the near-RT RIC.

Step 1: Specify a dedicated pod with its unique namespace: The dedicated pod ensures logical partitioning of cluster resources between multiple users.

Step 2: Deploy InfluxDB with Persistence in Kubernetes: This ensures that the data remains intact even if the database pod restarts or migrates.

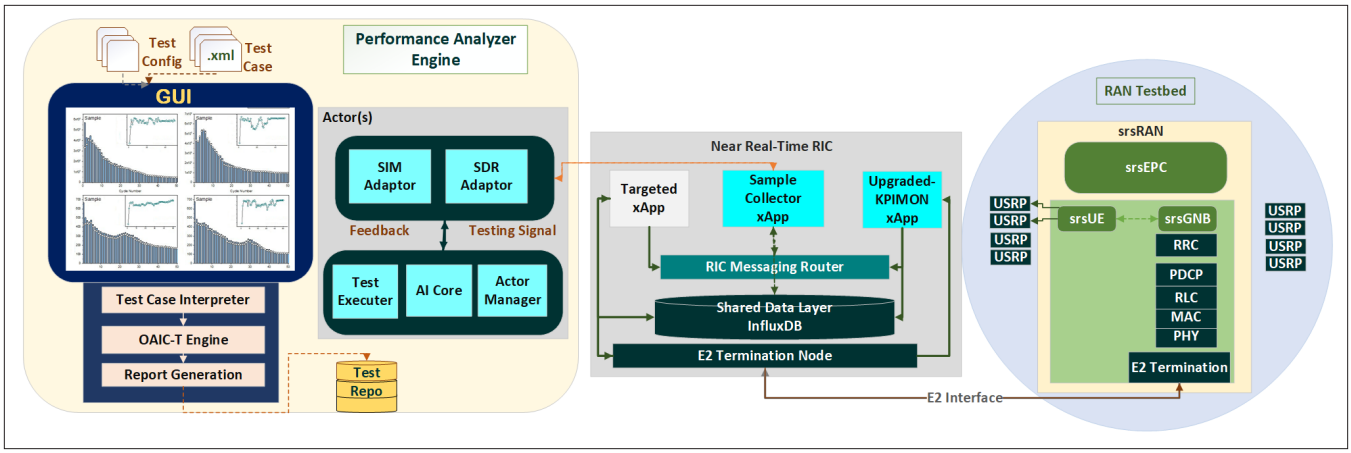

FIGURE 3. Detailed system architecture of the proposed O-RAN performance analyzer. The Performance Analyzer includes Performance Analyzer Engine, *Sample-Collector xApp*, and upgraded KPIMON xApp.

Step 3: Utilize a Network File System (NFS) within Kubernetes: It runs an NFS server within the cluster and dynamically provisions NFS-backed persistent volumes.

KPIMON Enhancement: Originally, KPIMON employed the Redis database for its data collection and sharing needs within the near-RT RIC. However, in the newly upgraded model, this has been replaced with InfluxDB. As a time-series open-source database, InfluxDB is crafted for handling extensive query loads on vast data that is continually evolving, thus making it a fitting choice for RAN metrics. This not only streamlines data collection and sharing but also augments the efficiency of real-time data analytics, which are critical near-RT RIC operations. Moreover, since we use InfluxDB as our SDL, KPIMON needs to adopt InfluxDB as well, ensuring consistency in our data management approach.

Optimizing Data Flow and Reducing Computational Overhead

We harness the robust capabilities of socket programming to facilitate seamless and real-time data flow within the proposed platform and its integral components. This ensures that various actors, each representing a particular task, test, or data subset, can effectively communicate with the O-RAN environment.

Many servers would employ a multi-thread method to respond to multiple requests simultaneously, dynamically allocating a dedicated thread upon each incoming request. However, this approach may lead to memory saturation, imposing potential operational delays, especially when managing an extensive number of concurrent tasks. We adopted a thread pool pattern to address this challenge. Rather than perpetually instantiating new threads, this pattern pre-allocates a defined pool of threads at the onset of the application. These threads remain instantiated and are kept in a ready state to accommodate incoming tasks. By doing so, we bypass the computational overhead and latency associated with on-the-fly thread generation. Consequently, this brings significant benefits in terms of system responsiveness and overall performance, while also preserving memory resources. In addition, the thread pool pattern also offers a predictable and controlled thread management system. This is critical for maintaining system stability, especially when subjected to heavy loads or during peak operation.

Graphical User Interface

A user-friendly Graphical User Interface (GUI) is engineered to facilitate real-time monitoring. This interface allows users to track ongoing tests, KPIs, resource utilization rates, and the status of socket connections. To augment this, alert notifications are systematically issued when an actor is activated.

Given these requirements, our GUI development leverages PyQt, a combination of the Qt framework, primarily implemented in C++, and Python. A salient characteristic of PyQt is its ability to expose the full range of the Qt API to Python, essentially enabling Python scripts to harness the comprehensive features of the Qt GUI. For our O-RAN Performance Analyzer, the GUI plays the role of managing actors and orchestrating scripts. Upon its launch via the server file, the GUI presents a comprehensive view of the system's state, as depicted in Fig. 4. This view offers insights into actor status, test sequences, and ongoing performance metrics, providing a user centric perspective of the dynamics of the O-RAN environment.

Test Scripts

As indicated in Fig. 2, scripts execute sequential test actions implementing customized test flows. Actions are categorized as atomic or composite. Atomic actions represent modular operations like sending requests or taking measurements. Multiple atomic actions are combined into reusable composite actions.

Test scripts can be generated for a wide range of scenarios in the OAIC-T framework to automate validation and testing. Scripts are generated by specifying the test logic as parameterized actions in JSON files. The generator parses these to create tailored scripts for verifying protocols and evaluating AI models. Test scripts can, for instance, verify successful user equipment (UE) resource provisioning in O-RAN. Such script executes actions to configure the UE, initiate the provisioning procedure, send provisioning data, and validate the procedure completion. The corresponding composite action encapsulates the end-to-end provisioning sequence between the UE and network functions such as the O-CU control plane (O-CU-CP). Executing such scripts allows testing of UE resources provisioning and analysis of KPIs. Automated testing is critical for validating the robustness of provisioning procedures during

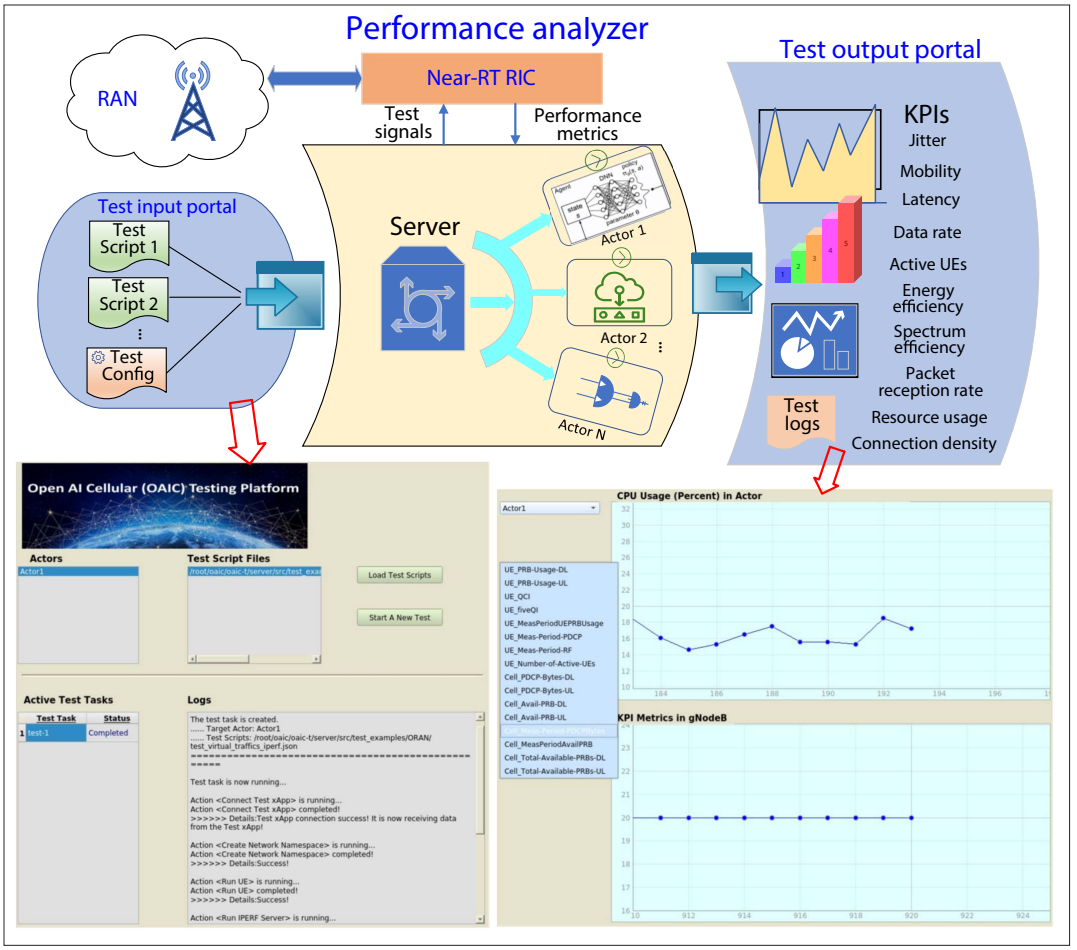

FIGURE 4. The O-RAN performance analyzer test platform with the input portal, output portal, and GUI.

O-RAN integration. The entire process from test objective identification to test script execution encapsulates a comprehensive approach to validating network operations.

The test design procedure begins by identifying the objective network procedure to validate, setting the foundation for testing. This is followed by defining test parameters, including nodes, data, and configurations, creating a precise framework for evaluation. Implementation of atomic test actions, such as sending requests and waiting for responses, assesses the network's basic response capabilities. These actions are then encapsulated into composite actions, adding complexity by simulating real-world operations through detailed procedure flows and message sequences. Building on this, the construction of procedural test logic triggers these predefined actions and validates the outcomes against expected conditions, an important step for ensuring the network's performance aligns with the standard. This logic forms the basis for generating executable test scripts, translating complex definitions into automated sequences that streamline the testing process. The procedure culminates with configuring the test environment to reflect actual network conditions and executing the test script. This execution not only verifies the network's functionality and performance through KPIs but also pinpoints areas for enhancement. The process offers a comprehensive approach from the identification of objectives to the execution of scripts and data analysis for network performance testing and evaluation as the basis for enhancing O-RAN robustness and reliability.

O-RAN Deployment

The developed system encompasses a diverse set of software stacks and platforms, which require careful integration. This process consists of deploying three main components: the O-RAN environment, which includes the near-RT RIC and the RAN, the *Sample-Collector xApp*, and the test platform.

The OAIC platform, a fully open-source implementation of the O-RAN environment with all the essential modules [12], facilitates O-RAN deployment. This deployment features the near-RT RIC, based on OSC's E-Release and augmented with the E2 Termination, which is installed using Docker containers. The near-RT RIC can operate on either a host computer or a virtual machine. For the RAN deployment, we integrate the 5G RAN with E2 Node Termination. The RAN can operate using ZeroMQ or over-the-air with SDR peripherals. Once the RIC cluster featuring the E2 interface is set up, one needs to configure and launch the 5G core, 5G RAN, and UEs.

Sample-Collector xApp Deployment

We utilize the *xapp-onboarder* to deploy the *Sample-Collector xApp* as a dockerized application. After hosting the xApp descriptor on the server, a .url file should be created to direct to the xApp descriptor within the server in order to generate

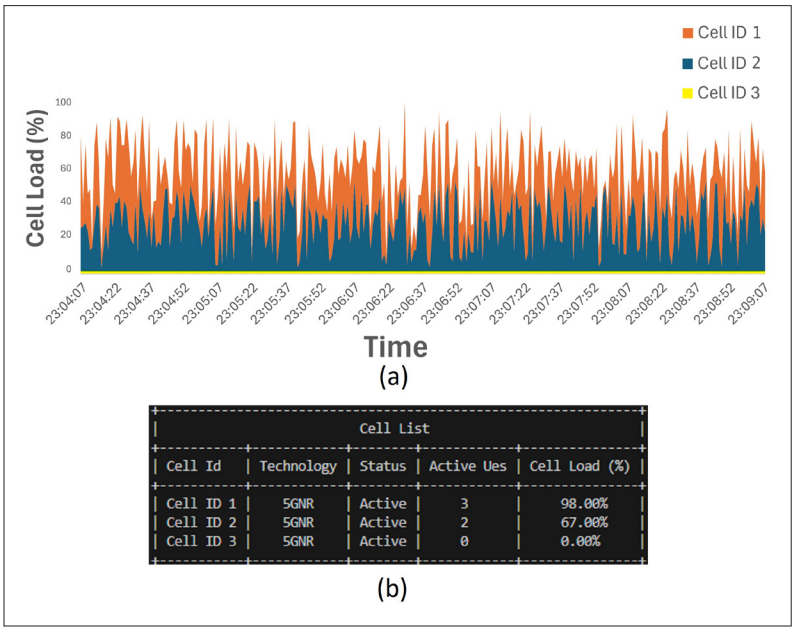

FIGURE 5. Collected KPIs: a) Instantaneous Cell Load (%) per cell over time; b) cell status.

the Helm charts, which aid in the deployment of the xApp on the O-RAN architecture built on Kubernetes. Once onboarded, the xApp is ready to establish connections and handle data transactions with both the SDL and the test platform.

Test Platform

The test framework consists of two primary components, a server and an actor, which operate together to enable testing and analysis of target xApps in an O-RAN environment. Network traffic is enabled upon the connection between the xApp and the test platform. Following the configuration of the server and actor based on the RIC parameters, users can design and implement tasks through predefined or new test scripts. When an actor is started, a notice appears on the GUI that an actor has been registered. Subsequently, communication with the server for the execution of various test scripts is possible via the console as illustrated in Fig. 4.

Test Scenario and Results

A test scenario is designed and deployed to show the capabilities of the proposed O-RAN performance analyzer in tracing xApp operations. The setup allows a detailed analysis of how xApp actions affect network performance, especially in terms of load distribution in a 5G network. In this test scenario, a traffic steering xApp is deployed in the near-RT RIC to manage traffic across a simulated 5G RAN, RANFusion, which is a RAN simulator in the OAIC-T framework, featuring three cells and five UEs. The iPerf3 traffic generator is employed to simulate uplink and downlink traffic, with each UE transferring 10 MB of data via the TCP protocol. So, a test script is generated based on the test scenario by specifying the test logic as parameterized actions in JSON files to automate validation and testing. In this case, the test scripts run the xApp and check evaluation metrics, which are defined here as cell load. The performance analyzer starts to monitor the network states as all five UEs concurrently transmit uplink data.

Initially, the five UEs are uniformly distributed across the three cells, with each cell handling a near-equal share of the traffic load. The traffic steering xApp is designed to dynamically allocate resources to balance the traffic based on real-time conditions. However, due to a misconfiguration in the xApp, it directs a disproportionate amount of traffic to Cell 1 causing congestion as shown in Fig. 5a. The O-RAN performance analyzer provides traces of KPIs resulting from xApp operation. It captures the maximum load across all cells and detects congestion in Cell 1, as illustrated in Fig. 5b. These measurements can be used for further analysis and actions to flag or terminate xApps causing inefficient network configurations.

The O-RAN performance analyzer monitors the CPU and memory utilization to assess the computational efficiency of the deployed xApps. The CPU utilization is captured by the GUI as shown in Fig 4. The energy efficiency is evaluated by measuring the power consumption of the near-RT RIC while the xApp is active. Results indicate that the traffic steering xApp operates with high computational efficiency and low resource overhead, which is critical for large-scale deployments.

Limitations, Challenges, and Opportunities

As O-RAN continues to grow in importance, one of the critical aspects identified by the O-RAN Alliance is performance measurements to help understand how network and computing resources are being used. The proposed performance analyzer is designed as a sophisticated tool to monitor and analyze KPIs of O-RAN operating environments.

There are still several limitations of the proposed solution requiring further research and development. For instance, OAIC is currently deployed in a laboratory testbed with limited network nodes and UEs. Deploying the proposed O-RAN performance analyzer in a large-scale network environment and across several target xApps will provide many additional data points to evaluate O-RAN. Another limitation stems from a significant shortage of xApps that are openly available in the O-RAN marketplace, due to the limited supply from operators, vendors, and the community. A diverse selection of xApps helps understand O-RAN's performance and potential issues. As a result of this shortage, it is difficult to evaluate and resolve potential issues thoroughly.

Navigating the complexity of AI model behaviors in general and, particularly, when deployed as RAN controllers in O-RAN environments presents a significant challenge . Evaluating AI performance across parameters such as correctness, robustness, and efficiency, and considering vulnerabilities of AI models cause enormous difficulties and are resource intensive.

On the other hand, the proposed performance analyzer opens up various research opportunities. These opportunities involve investigating novel techniques and algorithms for efficient performance analysis, exploring new approaches for monitoring and optimizing network and computational resources, and developing intelligent mechanisms for automated network control. For example, a more advanced KPI analyzer may provide real time performance analysis of each xApp, enabling swift identification and resolution of conflicts and other issues. By integrating AI to

perform predictive analytics, it may foresee and flag potential problems that need to be addressed to improve network stability. Moreover, another KPI analyzer may monitor the quality of essential O-RAN services, contributing to higher customer satisfaction and network reliability.

Conclusions

The inherent nature and architecture of O-RAN enable performance and resource optimization through data-driven, AI-based control. While this can theoretically lead to more efficient network operation and improved service provisioning for end users — facilitated by flexible, software-defined infrastructure, enhanced services, and open interfaces — practical demonstrations are lagging. Despite the ongoing testing and evaluation efforts for testing O-RAN components, subsystems, and systems, there is still a gap in automatically and intelligently monitoring, measuring, and analyzing the performance of O-RAN and xApps deployed in the the near-RT RIC, in particular. Motivated by this gap, this research proposes a novel O-RAN network analyzer. Through a test framework, the analyzer can effectively collect and visualize relevant information about O-RAN operations, including monitoring and exposing diverse network KPIs, xApp functionalities, and interactions with the O-RAN environment. We illustrate the design and working principles, deployment, and initial results on the OAIC platform, a fully open-source community research platform. Future developments will extend the O-RAN performance analyzer to include RAN slice monitoring, AI/ML-based security analysis of xApps, and integration with Prometheus or Grafana to enhance network operation visualization.

Acknowledgment

This work was supported in part by NSF award CNS-2120442.

References

[1] M. Polese *et al.*, "Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges," *Commun. Surveys Tuts.*, vol. 25, no. 2, Jan. 2023, pp. 1376–11.
[2] E. Amiri *et al.*, "Optimizing Virtual Network Function Splitting in Open-RAN Environments," *Proc. 2022 IEEE 47th Conf. Local Computer Networks*, 2022, pp. 422–29.
[3] S. Soltani *et al.*, "Poisoning Bearer Context Migration in O-RAN 5G Network," *IEEE Wireless Commun. Letters*, vol. 12, no. 3, 2023.
[4] ETSI, "Artificial Intelligence (AI) in Test Systems, Testing AI Models and ETSI GANA Model's Cognitive Decision Elements (DEs) via a Generic Test Framework for Testing GANA Multi-Layer Autonomics & their AI Algorithms for Closed-Loop Network Automation," White Paper No. 5, Mar. 2020.
[5] O-RAN Software Community, "O-RAN SC KPI Monitoring," https: //docs.o-ran-sc.org/projects/o-ran-sc-ric-app-kpimon/en/latest/, 2023; Accessed, Aug. 2024.
[6] X. Foukas *et al.*, *FlexRAN: A Flexible and Programmable Platform for Software-Defined Radio Access Networks*, Association for Computing Machinery, 2016, p. 427–41.
[7] J. Breen *et al.*, "POWDER: Platform for Open Wireless Datadriven Experimental Research," Computer Networks, vol. 197, 2021, p. 108281; accessed, Aug. 2024; available: https://www.sciencedirect.com/science/article/pii/S1389128621003017.
[8] M. Polese *et al.*, "ColORAN: Developing Machine Learning-Based xApp. For Open RAN Closed-Loop Control on Programmable Experimental Platforms," *IEEE Trans. Mobile Computing*, July 2022, pp. 1–14.
[9] E. Coronado *et al.*, "AI-Empowered Software-Defined WLANs," *IEEE Commun. Mag.*, vol. 59, no. 3, 2021, pp. 54–60.
[10] P. S. Upadhyaya *et al.*, "Open AI Cellular (OAIC): An Open Source 5G O-RAN Testbed for Design and Testing of AI-Based RAN Management Algorithms," *IEEE Network*, 2023, vol. 37, no. 5, pp. 7–15.
[11] B. Tang *et al.*, "AI Testing Framework for Next-G O-RAN Networks: Requirements, Design, and Research Opportunities," *IEEE Wireless Commun.*, vol. 30, no. 1, 2023, pp. 70–77.
[12] Open AI Cellular (OAIC), "Open AI Cellular Software Repository," https://github.com/openaicellular; accessed, Aug. 2024.
[13] M. Kouchaki and V. Marojevic, "Actor-Critic Network for O-RAN Resource Allocation: xApp Design, Deployment, and Analysis," *Proc. 2022 IEEE Globecom Workshops*, 2022, pp. 968–73.

Biographies

Mohammadreza Kouchaki (mk1682@msstate.edu) is a PhD student in electrical and computer engineering at Mississippi State University, MS, USA. His research interests include development and integration of AI/Machine Learning solutions in communication networks, O-RAN, and network security.

Seyed Bagher Hashemi Natanzi (snatanzi@wpi.edu) is a Ph.D. student in the Electrical and Computer Engineering Department at Worcester Polytechnic Institute (WPI), MA, USA. His research focuses on O-RAN, AI, and security solutions in 5G/6G wireless networks.

Minglong Zhang (mz354@msstate.edu) is an assistant research professor at Mississippi State University, Mississippi State, MS, USA. His research topics include O-RAN, 5G/6G V2X communications and wireless communications and networks.

Bo Tang (btang1@wpi.edu) is an associate professor in Department of Electrical and Computer Engineering at Worcester Polytechnic Institute, Worcester, MA, USA. His research interests include machine learning, AI security, and wireless communications.

Vuk Marojevic (vuk.marojevic@msstate.edu) is a professor in electrical and computer engineering at Mississippi State University, Mississippi State, MS, USA. His research interests include software radios, vehicle-to-everything communications, and wireless security with application to cellular communications, O-RAN, mission-critical networks, and unmanned aircraft systems.

The inherent nature and architecture of O-RAN enable performance and resource optimization through data-driven, AI-based control. While this can theoretically lead to more efficient network operation and improved service provisioning for end users — facilitated by flexible, software-defined infrastructure, enhanced services, and open interfaces — practical demonstrations are lagging.