The xApp Store: A Framework for xApp Onboarding and Deployment in O-RAN

Philip Rodgers
School of Computer Science
University of Glasgow
Glasgow, UK
philip.rodgers@glasgow.ac.uk

Paul Harvey
School of Computer Science
University of Glasgow
Glasgow, UK
paul.harvey@glasgow.ac.uk

***Abstract*—5G and beyond mobile telecommunication networks are increasingly embracing software technologies in their operation and control, similar to what has powered the growth of the cloud. This is most recently seen in the radio access network (RAN). In this new approach, the RAN is increasingly controlled by software applications known as *xApps*, and opens the door to third party development of xApps bringing diversity to the ecosystem, similar to mobile phone apps. This model aligns closely with the controllers in the ITU-T architecture for autonomous networks, and provides a pathway towards autonomous operation in the RAN. Unfortunately, no marketplace to host or supply xApps currently exists.**

This work describes our experiences in leveraging open-source O-RAN implementations to design and develop an xApp store.

***Index Terms*—O-RAN, RIC, xApps, Manifest, autonomous networks**

I. Introduction

A 5G network is able to communicate with mobile phones by sending and receiving radio waves. This part of the telephone network is called the radio access network (RAN). Traditionally, the RAN has been deployed as a series of dedicated hardware elements which are strongly associated with the vendors that provided them. More recently, these hardware elements are being increasingly replaced by software virtualisation [1], similar to what was seen in the cloud domain.

A key element of this new software-based approach is the introduction of modular applications known as xApps [2], which are promoted by the Open Radio Access Network (O-RAN) Alliance (section II). These dedicated software components are responsible for monitoring and controlling user-defined functions and operations within the RAN, and have already been identified as key enablers of autonomous network behaviour [3]. By decoupling control logic from the virtualisation runtime, xApps allow network functionality to be updated independently of the underlying hardware. This architecture also opens the door to third-party development, similar to that of the mobile app ecosystem for smartphones. We describe the role and characteristics of xApps in more detail in section III. At present, however, no equivalent of the mobile app store exists to support discovery, onboarding, and deployment of xApps in the O-RAN ecosystem.

This paper describes our experiences in trying to leverage open source O-RAN implementations to develop a prototype xApp Store (section IV). The xApp Store proposed in this work not only facilitates third-party innovation and rapid deployment of new functionalities, but also lays the groundwork for automated and self-managing RAN components. To the best of our knowledge, this is the first attempt to create such a store. Our objective was to define and implement xApp descriptions, validate the conformance of xApps to their descriptions, and run acceptance testing on an O-RAN platform.

A related body of work explores similar marketplace models for nApps, which are non-real-time applications designed to interface with the 5G core and the Non-RT RIC. Notably, the EVOLVED-5G project developed and validated an nApp marketplace to support vertical integration through standardised APIs and onboarding procedures [4]. While nApps and xApps operate in different timescales and architectural domains, both efforts highlight a growing need for structured, open ecosystems for third-party applications. We discuss this work more in section VIII.

II. Open Radio Access Network

Open Radio Access Network [5] is designed to address technical and non-technical challenges in traditional RAN systems. The high cost of these systems and vendor lock-in is bad for competition and innovation, preventing network operators the flexibility to build cost-effective solutions [6]. By fostering a multi-vendor ecosystem, O-RAN aims to reduce costs, allow for third-party innovation, and enable operators to scale and adapt network components to various deployment scenarios [7].

At its core, O-RAN is built on two fundamental principles: *vendor-agnostic interoperability* and *open architecture*. The vendor-agnostic nature of O-RAN is achieved through the definition of open interfaces for different tasks, such as RAN control via the E2, A1 or O1 interfaces. These open interfaces enable communication between components from different vendors [5]. The intention is that network operators are no longer limited by proprietary ecosystems, and are able to mix and match hardware and software components to suit their needs [8]. Open interfaces also facilitate the development

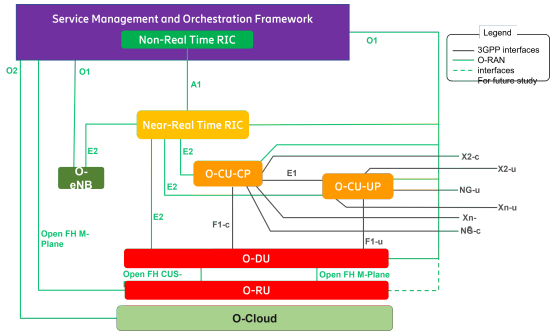

Fig. 1. O-RAN Architecture [12]

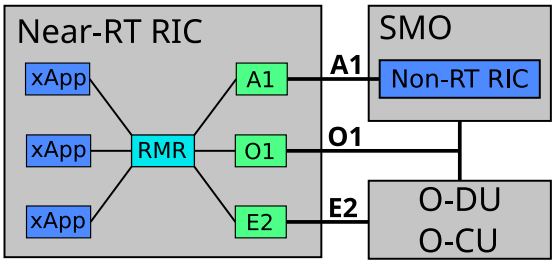

Fig. 2. xApps within the near-real-time RIC

of standardised service models, such as Key Performance Metrics (KPM) and RAN Control (RC), driving competition and enabling smaller, more specialised, vendors to enter the market [9].

The open architecture of O-RAN disaggregates the traditional RAN into distinct components, including the Open Radio Unit (O-RU), Open Distributed Unit (O-DU), and Open Central Unit (O-CU) [7], [10]. These components communicate through standardised interfaces, allowing for independent upgrades and replacements. An overview of O-RAN components and interfaces can be seen in Figure 1.

Central to O-RAN's architecture are the ran intelligent controllers (RIC). There are two variants: the Near-Real-Time RAN Intelligent Controller (Near-RT RIC) and the Non-Real-Time RAN Intelligent Controller (Non-RT RIC) [11]. The Near-RT RIC is intended to support control operations with a timescale of ten milliseconds to one second, enabling time-sensitive control through modular applications called *xApps* [5]. In contrast, the Non-RT RIC focuses on long-term network optimisation and analytics, leveraging *rApps* to inform policy and decision making [10]. As our work is focused on xApps we only consider the near-real-time RIC, however, we believe that our approach would be transferable to rApps. We will refer the near-real-time RIC as RIC in the remainder.

III. xApps

xApps are software applications designed to run on the RIC in O-RAN architectures. They separate control logic from the virtualisation runtime, allowing network functionality to be managed and updated independently of the underlying hardware. xApp examples include monitoring and control of key network operations such as traffic management [13], interference reduction [14], and performance monitoring [15]. By isolating these functions, network operators can update or replace specific control features without disrupting the entire system.

Developed as containerised applications or stand-alone binaries, xApps use standard interfaces (e.g. E2 interface) via RIC Message Router (RMR)[1] to ensure they work across different RIC implementations. When xApps are deployed to the RIC, they register with RMR with the message types it will be sending and receiving, see Figure 2. While not a publish/subscribe system, RMR routes messages using the message types. Multiple endpoints can register interest in certain message types so that a single message can be routed to multiple recipients. This approach is intended to support interoperability and simplify the process of adding or removing network functions as needed. The xApps' only requirements are to be able to communicate with RMR and be able to formulate and understand the interface messages. An example workflow would be a traffic steering xApp receiving strategy updates from the Non-Real-time RIC via the A1 interface. It could then send control messages to the radio units via the E2 interface [16].

The term "xApp" is more than just a label for what an application is intended to do. It comes with a formal definition as part of the O-RAN Alliance architecture[2]. As xApps are designed to run on RICs, they are expected to interact with other O-RAN components using the defined interfaces and protocols. This standardisation is intended to ensure interoperability across the ecosystem.

If an application bypasses the expected messaging framework (RMR) and communicates directly with radio hardware, for example, it is deviating from the formal specifications for xApps. So while the intended functionality might remain, from a standards and technical standpoint an app that does not adhere to the specified communication methods would no longer be an xApp in the formal sense.

Unfortunately, the current state of the standard interfaces often requires xApp developers to bypass the standard interfaces in order to realise their functionality [2] (section VII).

IV. xApp Store

Our goal for this work was to design and implement a prototype *xApp Store*. Similar to a mobile phone app store, the xApp Store is envisaged as a digital distribution platform specifically designed for hosting and managing xApps within an O-RAN ecosystem.

An xApp Store would be a logically centralised repository where developers can publish their xApps and users can browse, download, and install them. An xApp Store would provide publishing tools that allow developers to upload

[1]https://docs.o-ran-sc.org/projects/o-ran-sc-ric-plt-lib-rmr

[2]https://docs.o-ran-sc.org/en/latest/

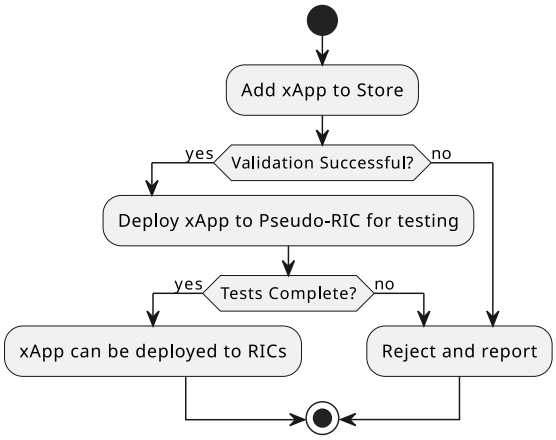

Fig. 3. xApp Lifecycle

xApps, manage updates, and monitor performance and usage statistics, managed licensing and payments, and become a marketplace for individual or several network operators.

We believe the creation of an xApp Store is vitally important for O-RAN adoption. If realised, such a system would enable the deployment and interoperability of xApps in O-RAN by creating a vendor-neutral platform where xApps can be stored, discovered, and deployed across various RIC implementations. The goal is to provide a standarised description and packaging of xApps, ensuring compatibility across different RIC implementations, and establishing a marketplace for third-party xApp developers. The xApp Store and the xApps it hosts directly map to the knowledge base and controllers of the ITU-T autonomous network architecture [17], respectively.

The lifecycle of an xApp in the xApp Store is shown in Figure 3.

A. Manifest-based validation

In order to be accepted as valid by the xApp Store, xApps must come bundled with a *manifest file*, similar to the manifest files that come with mobile apps for mobile app stores[3]. This file should include information about components, permissions, requirements, and runtime configurations. This manifest file contains the information necessary to achieve automated deployment and validation of xApps within an O-RAN environment. It allows RIC platforms to verify compatibility and assign necessary resources before running the xApp. It also allows the RICs to orchestrate deployments based on the interdependence of xApps.

Building on our initial concept [3], the manifest includes:

- Metadata: Name, version, author, license, and contact information.
- Runtime Configuration: Compatible RIC versions and resource requirements.
- Interface Definitions: Supported E2 service models and message types.
- Dependencies and Security Settings.

[3]https://developers.google.com/apps-script/concepts/manifests

Some xApps, such as OSC's KPIMON[4], include a JSON file which is used when deploying to a RIC. This file includes metadata such as author and version, Docker container information and health probes, and messaging information such as which message types the xApp will be sending and receiving.

In its current form, the xApp Store uses the JSON file from the xApp. This file does not include the all of the features we discussed in our previous work, as we have focused on creating an initial implementation, but this will be extended in the future. An example of a complete JSON file, based on a combination of our previous work and that of the OSC, is available online and validated against a JSON Schema.[5]

B. xApp Testing

In order to test xApps in an environment we controlled, we had to build our own RIC. We called this testing environment the *Pseudo-RIC*. Using the Pseudo-RIC, we can determine if an xApp is deployable and will interact with the RIC components and the simulated RAN components. We identified and tried four RIC implementation, described more in section VII. We chose to go with the OSC reference implementation because we believed this would be the most standards compliant, and because the O-RAN Alliance provides a great deal of technical specification documents [18]. Ideally, we would implement all four of the RIC environments outlined in section VII in order to test the xApps in each, but as the xApps we were using to test our system were built for OSC RICs we only implemented an OSC Pseudo-RIC. In this work, a lack of portability of xApps emerged as a key challenge.

Once validated against its manifest, an xApp is deployed to a *Pseudo-RIC* environment. The behaviour of the xApp is tested using a simulated 5G scenario as described in section VI. The behaviour is compared to the expected behaviour of the xApp determined by the manifest file. If an xApp fails validation or testing, a report is generated. This gives the xApp developer the information required to make their xApp O-RAN compliant.

C. Deployment

After successful testing on the Pseudo-RIC, xApps become available for deployment to production RICs. In order to do this automatically, the RIC infrastructure must allow the xApp Store the required access. This would likely be in the form of a software daemon or agent running on the RIC through which the xApp Store interacts with the RIC. In our prototype system, this is done via a web API provided by *RICMON*. RICMON is a web front end which incorporates many RIC tools and is discussed in section V.

V. RICMON

In the course of this work, we discovered that there is a significant amount of fragmentation in the RIC ecosystem, see section VII. Consequently, it was necessary to create a

[4]https://github.com/o-ran-sc/ric-app-kpimon-go

[5]https://github.com/philrod1/ric-stuff/tree/main/files/manifest

standardised way of interacting with RIC implementations. To this end, we created the *RICMON* tool[6]. RICMON is a web application built to simplify interactions with a RIC. RICMON is able to provide straight-forward access to the status and logging information from a RIC via REST API calls. It allows for easy control of RIC functions such as container monitoring and management.

Aside from monitoring the RIC, RICMON's main use is to simplify the deploying and removal of xApps. To deploy an xApp, RICMON takes a Git repository URL for the xApp. The URL can be passed to RICMON via its API or its web interface. It will then *automatically* clone the repository and organise the files into a standardised format. With all the required files in their correct location, the xApp is *automatically* built using Docker and saved to a locally hosted Docker repository. We can then use the tools provided by the O-RAN Alliance to onboard and install the xApp to the RIC.

VI. Scenario Simulation

Beyond validation against a manifest and deployment of the xApp, the xApp store also seeks runtime verification of the xApps behaviour, and eventually benchmarking. To the best of our knowledge, no standard xApp benchmarking suite exists.

As a first step, a simple mobile network simulation environment was created. The goal of the simulation environment is to test how the xApps interact with the radio systems, such as Next Generation Node B (gNB) and User Equipment (UE) elements they are built to monitor and control. The simulation environment is composed of four parts: -

- Logic and mobility simulation, written in JavaScript as an extension of RICMON.
- Networking and connectivity implemented using *GNURadio*.
- The 5G gNB and UE entities implemented using *srsRAN*.
- Network traffic generated using *iperf*.

The logic and mobility simulation is responsible for the visualisation and location of the entities. Each entity is modelled using srsRAN [19] to handle 5G configurations such as frequencies, handover, etc., with network connectivity realised using GNURadio[7]. A simplified diagram of GNURadio connectivity can be seen in Figure 4.

We use the simulation to check communication between xApps and the radio equipment they are controlling or monitoring. At the time of writing, the xApps can monitor the simulation but they cannot control it. This is due to a combination of lack of support in the form of E2 service models, and also seems to be a limitation of srsRAN software implementations. The srsRAN documentation shows that real hardware would accept and react to commands sent via E2.

[6]https://github.com/philrod1/ricmon
[7]https://www.gnuradio.org/

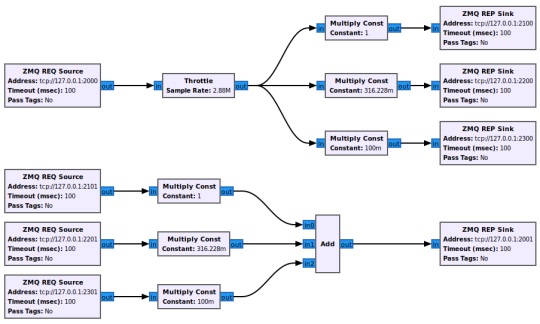

Fig. 4. A GNURadio flow chart with three UEs connected to one gNB

VII. Discussion

In developing the xApp store it was necessary to deploy a RIC. We now discuss the different options we considered to understand their appropriateness.

The RIC ecosystem is still evolving, but in this section we focus on four prominent RIC environments from the available literature: OSC RIC, µONOS-RIC, FlexRIC, and OAIC RIC. Throughout this project we built examples of each of these RIC environments, sometimes multiple examples, with varying levels of difficulty and success.

A. Open-source RICs

1) OSC RIC: The OSC RIC[8] is the de-facto reference implementation provided by the O-RAN Alliance. It is widely adopted in academic research and initial xApp development due to its adherence to published O-RAN standards. However, the OSC RIC suffers from several practical limitations. Its dependency on legacy software versions (e.g., Ubuntu 20.04 and GCC 9.3.0) and rigid installation scripts pose significant challenges for scalability and integration with more modern tools [20]. Moreover, the standard E2 interface implemented in OSC RIC is often found to be inflexible, limiting the ability to support active control functions such as dynamic traffic steering [21].

2) µONOS-RIC: µONOS-RIC[9] is a production-oriented platform designed for large-scale deployments. Its modular design and enhanced scalability make it attractive for industry applications. However, while µONOS-RIC promises improved performance, it tends to deviate from strict adherence to O-RAN standards. This divergence necessitates further testing to fully understand its trade-offs in terms of interoperability and compliance [20], [21]. As a result, µONOS-RIC may offer better performance in real-world scenarios, but its long-term viability depends on further refinement and industry collaboration.

3) FlexRIC: FlexRIC[10] is a platform that prioritises low latency and flexibility in xApp interactions. One of its distinguishing features is an extended version of the E2 interface (often referred to as E2*), which permits xApps to

[8]https://docs.o-ran-sc.org/en/latest/
[9]https://docs.onosproject.org/
[10]https://gitlab.eurecom.fr/mosaic5g/flexric

bypass certain RIC functions and interact directly with RAN components. This approach reduces latency, making FlexRIC particularly suitable for time-sensitive applications. While this design choice improves performance, it also raises questions about compatibility with other O-RAN-compliant systems, as FlexRIC's modifications may not be universally supported [22].

4) OAIC RIC: OAIC RIC[11], based on the *E release* of the OSC RIC, provides a middle ground between strict standards compliance and necessary modifications for enhanced functionality. Although it retains a high degree of compatibility with O-RAN specifications, OAIC RIC incorporates adjustments that enable better support for testing environments, such as integration with srsRAN [] components. Despite these modifications, OAIC RIC is still perceived as less robust compared to production-focused platforms like µONOS-RIC, and it faces challenges similar to OSC RIC in terms of scalability and long-term maintenance [21].

5) Summary: In summary, while OSC RIC remains the baseline for academic research due to its standards compliance, its practical limitations have lead to the development of alternative platforms. µONOS-RIC offers potential improvements for production use, FlexRIC provides enhanced low-latency performance through interface extensions, and OAIC RIC represents a compromise with additional support for testing scenarios. The divergence between these academic implementations and production-ready systems underscores the need for further collaboration and refinement of O-RAN standards. An overview comparison of the four main RIC environments can be seen in Table I.

TABLE I
COMPARISON OF RIC IMPLEMENTATIONS

Feature	OSC RIC	µONOS-RIC	FlexRIC	OAIC
Open-Source	✓	✓	✓	✓
Standards Compliance	✓	✓	Partial	Partial
Interface Support	E2, A1, O1	E2, A1, O1	E2	E2
xApp Modularity	✓	✓	✓	✓
Production-Ready	×	✓	×	×
AI/ML Support	Partial	✓	✓	✓
Good for Research	✓	✓	✓	✓
Industrial Deployment	✓	✓	×	×
Ease of Integration	Moderate	High	Moderate	High

B. Standards Conformance

A common observation was that many existing xApps are bypassing the RIC platform entirely for communication with the RAN, instead directly communicating with the E2 interface for performance reasons. This is orthogonal to the RIC design intentions. From the xApp store perspective, this creates a significant challenge in validating xApps if arbitrary communication channels exist between the xApp and the RAN interfaces. To attempt to evaluate this, we used EdgeShark[12] to sniff network packets being sent in the containerised Pseudo-RIC environment. Unfortunately, we were unable to observe meaningful behaviour.

The initial tests indicate that a standardised manifest approach simplifies xApp validation and onboarding as the xApps need to provide the necessary information to build, test and deploy them. The xApp store automates testing and reduces manual steps. However, reliance on older software and the limited capabilities of the current E2 interface hinder full functionality. These issues suggest that while the framework is well-suited for academic research and early-stage development, further enhancements are needed. Ideally, large telcos who are actively deploying O-RAN solutions will upstream their implementations.

[11]https://openaicellular.github.io/oaic/

[12]https://edgeshark.siemens.io/

Our investigations reveal several insights and challenges:

- **Integration Challenges:** Integrating a reference OSC RIC with modern components such as srsRAN proved to be more complex than anticipated. The OSC RIC's dependency on older software versions (e.g., Ubuntu 20.04 with GCC 9.3.0) restricted our ability to incorporate up-to-date libraries and tools. Such issues raise concerns about long-term viability of the reference implementation.
- **xApp Validation and Testing:** The introduction of an xApp manifest file and an associated validation process is effective in standardising descriptions and deployment requirements. The xApp Store was able to automate initial validation steps; however, the testing phase on the Pseudo-RIC uncovered limitations in current service model support. Although passive operations (such as KPM monitoring) were successful, active functions (such as traffic steering) could not be fully tested due to limitations of the E2 interface implementation.
- **Toolchain and Ecosystem Constraints:** RICMON and the xApp Store provide useful functionality for managing deployments and monitoring states. Yet, our experience underscored the need for a more cohesive ecosystem that minimises manual interventions. The necessity to modify source code for features like dynamic registration of message types (mtypes) highlights a broader issue: the current state of open RAN implementations often requires ad hoc solutions to bridge gaps between evolving standards and existing deployments [21].
- **Implications for Industry Adoption:** The challenges we encountered suggest that while the OSC RIC and associated tools offer a promising testbed for academic research and early-stage xApp development, there may be significant challenges for large-scale, production-level deployments. This dichotomy between research-friendly and production-grade platforms must be addressed to realise the full potential of open, vendor-neutral RAN architectures.

C. Recommendations

Based on our experience, to support future open-source development and engagement of this work we recommend the following: -

- Modernising RIC implementations to support current operating systems and toolchains.

- Enhancing the E2 interface to better support active functions such as traffic steering.
- Expanding testing environments to include real-time scenarios and dynamic message registration.
- Fostering industry collaboration to further standardise xApp descriptions and deployment processes.

VIII. Related Work

In the non-real-time domain, the EVOLVED-5G project introduced and validated a comprehensive marketplace for nApps—software modules that interface with the 5G core via standard northbound APIs such as NEF and CAPIF. This marketplace includes mechanisms for onboarding, automated enrolment, and service discovery, as well as deployment support for both operator-managed and third-party domains [4]. A key feature of this approach is the use of semantic descriptors and the TMF Open API framework to expose nApps as catalogue entries.

Additionally, Charpentier et al. [23] provide a broader view of nApps within the 5G and beyond ecosystem. They describe the evolution of plug-and-play interfaces such as NEF, CAPIF, and SEAL, and introduce the concept of a Network Application Package. This aligns with our manifest-based description for xApps, reinforcing the idea that network programmability should be supported by standard packaging and deployment metadata.

IX. Conclusions

This work describes the design and implementation of a prototype open-source xApp store to host, test, and deploy O-RAN xApps to support ecosystem development and greater autonomous operation of the network. Leveraging open-source O-RAN implementations, we were able to perform validation, testing and deployment of xApps, noting that our manifest-based approach streamlines the onboarding and testing process. Simulation results demonstrate that the framework reliably supports passive functions, such as KPI monitoring. Based on our experience, we identified several challenges with the state of the open-source O-RAN ecosystem, which are described and recommendations provided to bridge the gap between academic prototypes and production-ready systems.

Acknowledgment

This work was supported by EPSRC IAA award EP/X5257161/1.

References

[1] M. Condoluci and T. Mahmoodi, "Softwarization and virtualization in 5g mobile networks: Benefits, trends and challenges," *Computer Networks*, vol. 146, pp. 65–84, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128618302500

[2] 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.

[3] A. KLIKS, M. DRYJANSKI, V. Ram OV, L. Wong, and P. Harvey, "Towards autonomous open radio access networks," *ITU Journal on Future and Evolving Technologies*, vol. 4, no. 2, pp. 251–268, 2023.

[4] X. Li, A. Mesodiakaki, M. Gatzianas, G. Kalfas, A.-S. Charismiadis, D. Tsolkas, L. Zanzi, A. Salkintzis, D. Warren, A. Gavrielides, M. Sophocleous, M. Gramaglia, A. Gavras, T. Cogalan, H. Lee, and O. Bulakci, "Integrating napps in 5g for verticals: Architecture innovation and technology enablers," *IEEE Communications Magazine*, vol. 63, no. 1, pp. 161–167, 2025.

[5] O-RAN Alliance, "O-ran white papers and resources," https://www.o-ran.org/o-ran-resources, accessed: 2025-01-08.

[6] VIAVI Solutions, "O-ran: An open ecosystem to power 5g applications," https://www.viavisolutions.com/en-us/literature/o-ran-open-ecosystem-power-5g-applications-white-papers-books-en.pdf, accessed: 2025-01-08.

[7] O-RAN Alliance, "O-ran: Towards an open and smart ran," https://assets-global.website-files.com/60b4ffd4ca081979751b5ed2/60e5afb502810a0947b3b9d0_O-RAN%2BWP%2BFInal%2B181017.pdf, accessed: 2025-01-08.

[8] ——, "O-ran use cases and deployment scenarios," https://mediastorage.o-ran.org/white-papers/O-RAN.WG1.Use-Cases-and-Deployment-Scenarios-White-Paper-2020-02.pdf, accessed: 2025-01-08.

[9] ——, "O-ran map," https://map.o-ran.org/, accessed: 2025-01-08.

[10] ——, "O-ran minimum viable plan and the acceleration towards commercialization," https://mediastorage.o-ran.org/white-papers/O-RAN.Minimum-Viable-Plan-and-Acceleration-towards-Commercialization-white-paper-202.pdf, accessed: 2025-01-08.

[11] S. Niknam, A. Roy, H. S. Dhillon *et al.*, "Intelligent o-ran for beyond 5g and 6g wireless networks," *arXiv preprint arXiv:2005.08374*, 2020. [Online]. Available: https://arxiv.org/abs/2005.08374

[12] O-RAN Software Community, "O-RAN Architecture Documentation," 2025, accessed: 2025-01-21. [Online]. Available: https://docs.o-ran-sc.org/en/j-release/architecture/architecture.html

[13] P. Sroka, L. Kulacz, S. Janji, M. Dryjański, and A. Kliks, "Policy-based traffic steering and load balancing in o-ran-based vehicle-to-network communications," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 7, pp. 9356–9369, 2024.

[14] H. Yang, C. Mcpeak, R. Nagampally, N. Tripathi, G. Anderson, J. H. Reed, Y. Yang, and D. J. Jakubisin, "Agile 5g networks: Advance traffic steering xapp for interference mitigation," in *MILCOM 2024 - 2024 IEEE Military Communications Conference (MILCOM)*, Oct 2024, pp. 300–305.

[15] M. Kouchaki, S. B. H. Natanzi, M. Zhang, B. Tang, and V. Marojevic, "O-ran performance analyzer: Platform design, development, and deployment," *IEEE Communications Magazine*, vol. 63, no. 2, pp. 152–159, February 2025.

[16] RimeDo Labs, "Policy-based traffic steering: xapp implementation within o-ran," https://rimedolabs.com/blog/policy-based-traffic-steering-xapp-implementation-within-o-ran/, accessed: 2025-03-06; n.d.

[17] ITU-T, "Autonomous Networks -Architecture Framework," United Nations International Telecommunication Union, Geneva, CH, Standard, Dec. 2023.

[18] O-RAN Alliance, "O-RAN Specifications," 2025, accessed: 2025-01-21. [Online]. Available: https://specifications.o-ran.org/specifications

[19] W. Flakowski, M. Krasicki, and R. Krenz, "Implementation of a 4g/5g base station using the srsran software and the usrp software radio module," *JTIT*, vol. 3, no. 2023, pp. 30–40, 2023.

[20] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding o-ran: Architecture, interfaces, algorithms, security, and research challenges," *IEEE Communications Surveys and Tutorials*, vol. 25, no. 2, pp. 1376–1411, 2023.

[21] J. F. Santos, A. Huff, D. Campos, K. V. Cardoso, C. B. Both, and L. A. DaSilva, "Managing o-ran networks: xapp development from zero to hero," *arXiv preprint*, 2024, arXiv:2407.09619.

[22] C.-C. Chen, M. Irazabal, C.-Y. Chang, A. Mohammadi, and N. Nikaein, "Flexapp: Flexible and low-latency xapp framework for ran intelligent controller," in *ICC 2023 - IEEE International Conference on Communications*. Rome, Italy: IEEE, 2023, pp. 5450–5456.

[23] V. Charpentier, G. Landi, E. Giannopoulou, J. Brenes, M. Camelo, J. M. Marquez-Barja, and N. Slamnik-Kriještorac, "Advancing vertical services for 6g: Future directions and innovations," *IEEE Network*, pp. 1–1, 2025.