O-RAN next Generation Research Group (nGRG)

Contributed Research Report

nGRG-RR-2025-05-nGRG Research Report on dApp Architecture and Interfaces

V1.0

Contributors:

Northeastern University

SoftBank

NVIDIA

Release date: 2026.02.11

Authors

Michele Polese, Salvatore D'Oro, Andrea Lacava, Niloofar Mohamadi, Northeastern University;

Yunseong Lee, SoftBank;

Davide Villa, Chris Dick, NVIDIA

Reviewers

Nishant Tiwari, 1FINITY (a Fujitsu company);

Nirlay Kundu, IMDEA Networks;

Vikas Dixit, Reliance Jio;

Nurit Sprecher, Nokia

Executive summary

This research report explores the integration of real-time AI-based control and inference into Open RAN through the development of dApps—distributed applications deployed on the same infrastructure supporting Radio Access Network (RAN) nodes. The proposed dApp framework addresses limitations in existing O-RAN architectures, i.e., lack of direct access to user plane data for edge inference and achieving sub-10ms control loop interactions. These capabilities enhance RAN optimization for use cases such as dynamic spectrum sharing and real-time positioning.

To address the limitations highlighted above, the framework introduces a new E3 interface, facilitating seamless real-time communication between dApps and other O-RAN components, including E2 nodes and the RAN Intelligent Controllers (RICs), and exposure of user-plane data units. A prototype implementation, built on OpenAirInterface (OAI), demonstrates the potential of dApps for over-the-air spectrum sensing and positioning, showcasing the effectiveness of real-time resource management.

The results highlight the feasibility of real-time control loops within RAN systems, reducing communication overhead and improving performance through Unix domain socket-based data exchange. This approach presents a scalable alternative to real-time RIC architectures proposed in the literature by minimizing complexity while leveraging existing O-RAN interfaces. The work concludes with insights into dApp lifecycle management and a comparison with other real-time RAN solutions, setting the foundation for further research and development in this area.

Table of Contents

Authors 2
Reviewers 2
Disclaimer 2
Copyright 2
Executive summary 3
List of abbreviations 6
List of figures 7
List of tables 7
1 Introduction 8
2 Role of dApps in the Hierarchical O-RAN Control Architecture 10
3 RAN Node and dApps Architecture 13
3.1 Integrating dApps for Real-time Data Access and Control 13
3.2 Service-based dApps Integration via E3 API in the O-RAN Architecture 14
3.3 dApp Architecture with Multiple RAN Functions 14
3.4 Security Aspects 15
4 dApp E3 Interface 16
4.1 Functional Description 16
4.2 Architectural Components 16
4.3 Deployment Considerations 17
4.4 dApp and RAN Interactions over E3 17
4.4.1 dApp Registration Procedure 17
4.4.2 E3 Subscription Procedure 18
4.4.3 E3 Control Message Procedure 19
4.4.4 E3 Indication and Event Notification Procedure 19
4.4.5 dApp Unsubscription Procedure 20
4.4.6 dApp Deregistration Procedure 20
4.5 Integration with xApps via E2SM-DAPP 21
4.5.1 E2SM-DAPP Registration Procedure 22
4.5.2 E2SM-DAPP Telemetry Subscription Procedure 22
4.5.3 E2SM-DAPP Control Procedure 23
4.5.4 E2SM-DAPP Indication Procedure 23

4.5.5 E2SM-DAPP Deregistration Procedure 24
5 dApp Lifecycle and Interaction with Near-RT RIC, Non-RT RIC, SMO 25
5.1 Development 25
5.2 Onboarding 26
5.3 Deployment 26
5.4 Interaction with Near-RT RIC, Non-RT RIC, and SMO 27
6 A Reference dApp Prototype – Lessons Learned 29
6.1 dApp Class 29
6.2 E3Interface Class 30
6.3 E3Connector Class 30
6.4 E3Connector Implementations 31
6.4.1 Transport Protocols in E3Connector 33
6.4.2 Distributed Deployment Considerations 33
6.4.3 Message Formatting & Standard Compliance 33
6.5 Real-Time Capabilities of dApps 34
6.6 Use Cases and Evaluation 35
6.6.1 Spectrum Sharing dApp 35
6.6.2 Sensing and Positioning dApp 37
7 Conclusions 39
References 40

List of abbreviations

AP	Application Protocol
BSR	Buffer Status Reports
CIR	Channel Impulse Response
CNF	Cloud-Native Function
CP	Control Plane
CSI	Channel State Information
CQI	Channel Quality Information
CU	Central Unit
DCI	Downlink Control Information
DMS	Deployment Management Service
DU	Distributed Unit
FIFO	First-In First-Out
IE	Information Element
IMS	Infrastructure Management Service
IPC	Inter-Process Communication
ISAC	Integrated Sensing and Communications
KPI	Key Performance Indicator
KPM	Key Performance Measurement
LCM	Life Cycle Management
MIMO	Multiple Input Multiple output
NF	Network Function
RF	Radio Frequency
RIC	Radio Intelligent Controller
RLC	Radio Link Control
RU	Radio Unit
SDLC	Software Development Lifecycle
SLA	Service Level Agreements
SM	Service Model
SMO	Service Management and Orchestration
SNR	Signal-to-Noise Ratio
SRS	Sounding Reference Signals
PDCP	Packet Data Convergence Protocol

PDU	Packet Data Units
UCI	Uplink Control Information
UE	User Equipment
UP	User Plane
QoS	Quality of Service

List of figures

Figure 1: O-RAN control loops, limitations, and extensions to the real-time and user-plane domains.......... 11
Figure 2: O-RAN near-RT RIC integration with dApps and a generic RAN function 13
Figure 3 dApp architecture with multiple RAN functions 15
Figure 4 Message exchanges for a real time control loop between a dApp and the RAN function through the E3 interface.. 17
Figure 5 Interactions between a dApp, the E2 RAN function, and an xApp through the E3 interface and the E2SM-DAPP Custom Service Model. 21
Figure 6: dApp LCM diagram 25
Figure 7 Design of the E3Connector sockets. 34
Figure 8 Intelligent real-time control loop for the PRB blacklisting. We omit in this figure the initial E3AP ... 36
Figure 9 Setup for the Sensing and Positioning dApp and UL CIR extracted by the dApp.............................. 38

List of tables

Table 1: dApp Deployment.. 26

1 Introduction

xApps and rApps, in the Near-Real-Time (RT) and Non-RT RAN Intelligent Controllers (RICs), enable a variety of use cases where the RAN can be dynamically controlled to optimally handle and adapt to varying network conditions. Wireless systems are characterized by rapidly changing channel conditions, dynamic traffic patterns, user mobility, and periodic or seasonal patterns in network utilization, to name a few. Examples of RIC-enabled control include network slicing, traffic steering, beamforming and mobility management, advanced sleep modes, anomaly detection, and spectrum and radio resource allocation.

The current O-RAN architecture, though, comes with two key limitations: (i) xApps and rApps are primarily designed to interact with data and operations associated with the RAN control-plane, thus not considering user-plane data, such as I/Q samples and packets for inference and optimization (e.g., critical to sensing operations [1]); and (ii) does not enable control loops at timescales below the 10 ms enabled by xApps. Indeed, as we will discuss in later sections of this report, and as also discussed in the next-Generation Research Group (nGRG) Research Report “dApps for Real-Time RAN Control: Use Cases and Requirements” [2], being unable to process user-plane data from RAN nodes and perform inference and control below the 10 ms timescale limits the application of O-RAN technologies at the lower levels of the protocol stack, as well as the introduction of novel use cases and applications (e.g., RF fingerprinting, one-shot beam steering, anomaly and attack detection, spectrum sensing and incumbent detection, joint sensing and communications, to name a few) which will be at the center of 6G.

To overcome these limitations, this research report delves into the concept of dApps, originally proposed in [3], and further [4]. Similarly to xApps and rApps, dApps are software components that can execute as microservices, designed to be co-located with Centralized Units (CUs, or O-CU-CP for control plane and O-CU-UP for user plane) and Distributed Units (O-DUs), where user-plane data related to RAN operations (e.g., I/Q samples, transport blocks, Radio Link Control (RLC) and Packet Data Convergence Protocol (PDCP) packets, among others) is readily available. The advantages introduced by dApps are manifold: (i) they can execute real-time operations at the O-DU/O-CU-CP/O-CU-UP directly to achieve real-time control and monitoring of L2 and L1 RAN functionalities without the need to involve the RICs; (ii) they have a lightweight lifecycle management, and can be instantiated and deleted seamlessly to provide functionalities as-a-service based on operator requirements; and (iii) they have access to user-plane data that cannot leave the O-DU/O-CU-CP/O-CU-UP due to privacy concerns (e.g., user-related data) or impracticality (e.g., I/Q streams require high bandwidth in the order of Gbps, which would generate congestion over the other O-RAN interfaces).

How to bring AI/ML to O-RAN and push inference toward the real-time domain has received increasing interest in the last few years. Apart from dApps, the literature also offers different proposals that include the concepts of real-time RIC and μApps, zApps, tApps [2, 5, 5]. These are applications that, in principle, are similar to dApps but use different architectures and interfaces, therefore representing possible alternatives to dApps for enabling real-time control and inference in the RAN.

In this direction, O-RAN nGRG recently released a research report, part of a broader research item on dApps, with input and collaborations across academia and industry aimed at exploring use cases and applications that would benefit from dApps. These include real-time scheduler reconfiguration, spectrum sensing, compute resource scaling for energy savings, channel equalization, beam management and many others. Despite the well-defined use cases, an architecture for implementing dApps in cellular networks is still lacking.

Scope and Contributions. In this technical report, we fill this gap and propose a reference architecture for dApps with the goal of fostering design and prototyping of dApp-based use cases and applications for O-RAN systems. Specifically, we propose an architecture that can be directly integrated with the existing O-RAN architecture with minimal procedural changes. This minimizes the impact that dApps have on O-RAN procedures, while still making their use feasible and practical. We introduce a novel E3 interface to allow dApps to interact in real time with O-DUs and O-CU-UPs/O-CU-CPs, for data and control exchange. We also propose an extension of the O-RAN Lifecycle Management (LCM) for the deployment and management of dApps in the O-RAN architecture, besides xApps and rApps. Once deployed, we show that dApps can leverage already existing interfaces and procedures to exchange data with the Near-RT RIC over the E2 interface, to enable coordination of local and centralized approaches (e.g., for localized sensing and coordinated control [4]). The E2 interface can also be used to perform monitoring and control tasks using a custom E2 Service Model (E2SM), i.e., E2SM-DAPP.

We also describe the development of an open-source framework for dApps integrated with the popular OpenAirInterface (OAI) 5G RAN framework, an open-source software that implements a 5G stack for RAN and UEs. OAI was selected because of its support for the E2 interface, integration with a Near-RT RIC out of the box, and ability to extract I/Q streams from the RF chain. The dApp prototype has been demonstrated to operate across multiple platforms within real-time control intervals, efficiently processing both vector data (e.g., I/Q arrays extracted from the RAN) and scalar values in loops taking less than 450 microseconds. This makes our dApp prototype able to operate well below the 10 ms threshold required for real-time operations.

Additionally, we describe the implementation of two distinct dApp use cases using the proposed framework for Integrated Sensing and Communications (ISAC): positioning and spectrum sharing.

We believe that the combination of architectural framework, open-source reference library for dApps, and the thorough performance evaluation will inspire further research and development efforts in the real-time control domain.

2 Role of dApps in the Hierarchical O-RAN Control Architecture

The programmability of the protocol stack through closed-loop control represents a key innovation of the O-RAN architecture and the broader Open RAN vision. This mechanism enables xApps and rApps running on the RICs to access RAN telemetry and Key Performance Metrics (KPMs), process them, and determine optimal configurations that meet specific objectives. These configurations are applied to the RAN functions either as control actions or policies, helping the system achieve its desired operational state.

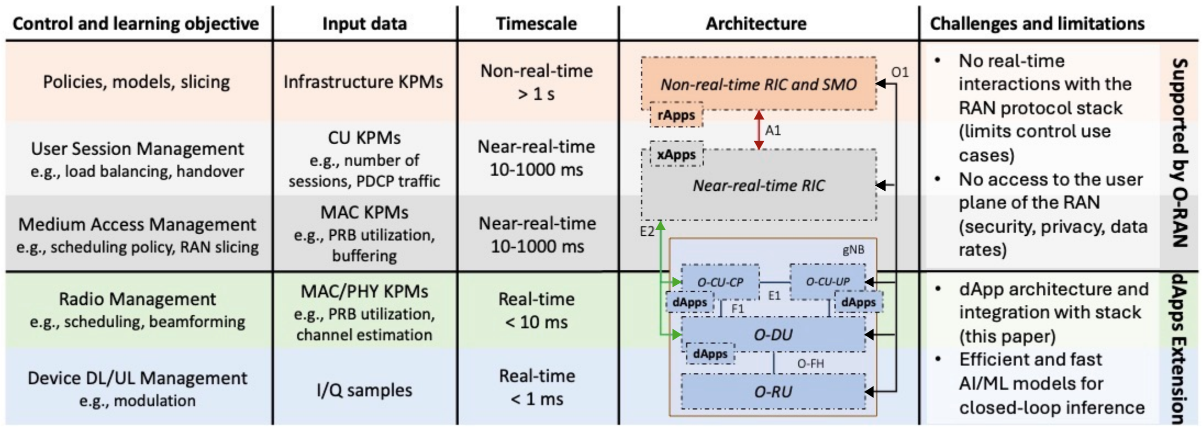
Figure 1 provides a simplified view of the O-RAN control loops, which have an impact on several components in the O-RAN architecture: the RAN's O-CU-CP and O-CU-UP handling the control and user planes, the O-DU, and the O-RU and associated interfaces.

The current O-RAN design incorporates two control loops: the non-real-time loop, managed by the Non-RT RIC and the Service Management and Orchestration (SMO), and the near-real-time loop, managed by the Near-RT RIC. The Non-RT RIC loop operates at intervals of 1 second or longer, using KPMs from thousands of devices to define policies. The Near-RT RIC loop operates within a 10 ms to 1-second range, focusing on finer control over smaller clusters of base stations and hundreds of devices. However, two key limitations affect these loops: (1) the absence of control loops with periodicity faster than 10 ms and (2) the lack of programmability and interaction at the user plane level.

Addressing these challenges requires real-time interactions, which enable new, fine-grained inference and control capabilities. These include beam management, scheduling profile selection, dynamic spectrum access, packet tagging, and QoS enforcement. Furthermore, access to user plane data, such as waveforms and PDUs, would unlock powerful inference and control capabilities based on the information they carry. Processing raw I/Q samples can improve anomaly detection, spectrum sensing, RF fingerprinting, and beam management. Similarly, access to user plane data at higher layers (e.g., RLC, PDCP) supports advanced tasks such as traffic classification and slice identification. It is also worth mentioning that extracting user plane data introduces computation and storage costs as data needs to be stored and accessed timely to perform real-time inference. However, the actual cost depends on how much and how frequently data is extracted, and how data is encoded (e.g., floating point, integer, structures).

However, the O-RAN architecture does not currently support real-time loops through RICs. Privacy, security, and performance (due to bandwidth, latency) constraints make it impractical to move large datasets, such as I/Q samples, out of the RAN. For instance, transmitting I/Q samples over the E2 interface could introduce delays of several seconds, making real-time operations infeasible. To overcome these constraints, programmable elements within RAN functions that expose APIs and can be instantiated as lightweight microservices are essential.

As illustrated in the lower part of Figure 1, dApps—lightweight, plug-and-play microservices—can address these challenges. Deployed directly within the O-CU-CP, O-CU-UP, or O-DU, dApps provide secure, real-time access to the protocol stack without the overhead of an additional real-time RIC component. This makes dApps ideal for resource-constrained RAN nodes, enabling direct interaction with user plane data and real-time control without adding unnecessary latency.



Control and learning objective	Input data	Timescale	Architecture	Challenges and limitations	
Policies, models, slicing	Infrastructure KPMs	Non-real-time > 1 s		• No real-time interactions with the RAN protocol stack (limits control use cases) • No access to the user plane of the RAN (security, privacy, data rates)	Supported by O-RAN
User Session Management e.g., load balancing, handover	CU KPMs e.g., number of sessions, PDCP traffic	Near-real-time 10-1000 ms			
Medium Access Management e.g., scheduling policy, RAN slicing	MAC KPMs e.g., PRB utilization, buffering	Near-real-time 10-1000 ms			
Radio Management e.g., scheduling, beamforming	MAC/PHY KPMs e.g., PRB utilization, channel estimation	Real-time < 10 ms		• dApp architecture and integration with stack (this paper) • Efficient and fast AI/ML models for closed-loop inference	dApps Extension
Device DL/UL Management e.g., modulation	I/Q samples	Real-time < 1 ms			

Figure 1: O-RAN control loops, limitations, and extensions to the real-time and user-plane domains.

The use cases for dApps can be classified into four main categories:

1. **Direct processing of waveforms and PDUs:** Access to I/Q samples can be used for physical layer security (e.g., anomaly detection to identify spoofed base stations or users), RF fingerprinting [6] for secure authentication, and spectrum sensing for interference detection. This information, processed by dApps, can also be shared with xApps in the Near-RT RIC to coordinate responses across multiple RAN nodes.
2. **Real-time scheduling and beam management:** dApps can embed ad hoc policies and SLAs for specific traffic patterns or slices, accelerate scheduling for massive MIMO, and manage multi-cell coordination [7]. They can also enhance beam management by directly manipulating synchronization signals within the O-DU [8] [9].
3. **RAN node and fronthaul configuration:** dApps can optimize energy efficiency by dynamically adjusting CPU states (e.g., sleep, active, halt) or pinning based on real-time RAN telemetry. They can also adapt the fronthaul compression based on signal conditions, conserving resources when SNR is high and providing uncompressed streams when users are at the cell edge.
4. **Augmented sensing and channel estimation:** dApps can dynamically compress CSI in coordination with UE and deploy custom AI/ML models for channel estimation tailored to specific network conditions [10]. They can also process synchronization signals for non-traditional use cases, such as positioning and environmental sensing [11].

A complete research report on use cases is provided in [2].

By enabling dApps, the O-RAN architecture can unlock real-time inference and control, addressing its current limitations while preparing the RAN for 6G applications. These microservices seamlessly integrate with existing RAN infrastructure, providing a scalable, flexible solution for future wireless networks.

3 RAN Node and dApps Architecture

This section explores how the O-RAN architecture can be enhanced to integrate dApps as custom, pluggable services within RAN nodes. Key elements include:

1. **RAN function**: Enhanced with a service-based E3 API (exposed by the RAN functions and consumed by the dApp) for added functionality.
2. **E2 interface**: Facilitates communication with the Near-RT RIC.

Further details on the dApps lifecycle and the role of the O1 interface are discussed in Section 4. A prototype implementation using the open-source OAI stack is presented in Section 5 as a reference. Figure 2 illustrates dApps as independent, pluggable services interfacing southbound with RAN functions (O-DU, O-CU-CP, O-CU-UP network functions) and northbound with O-RAN E2 for xApp collaboration (to enable coordination between centralized and local approaches). These interfaces enable flexible data access through streaming or polling with varying periodicity and granularity, supporting diverse telemetry and data plane insights from elements like I/Q samples, Buffer Status Reports, and Channel Quality Information. From an architectural point of view, this design enables interoperability and distributed control as multi-vendor RAN units can expose standardized high-level APIs to regulate parameter observability and control.

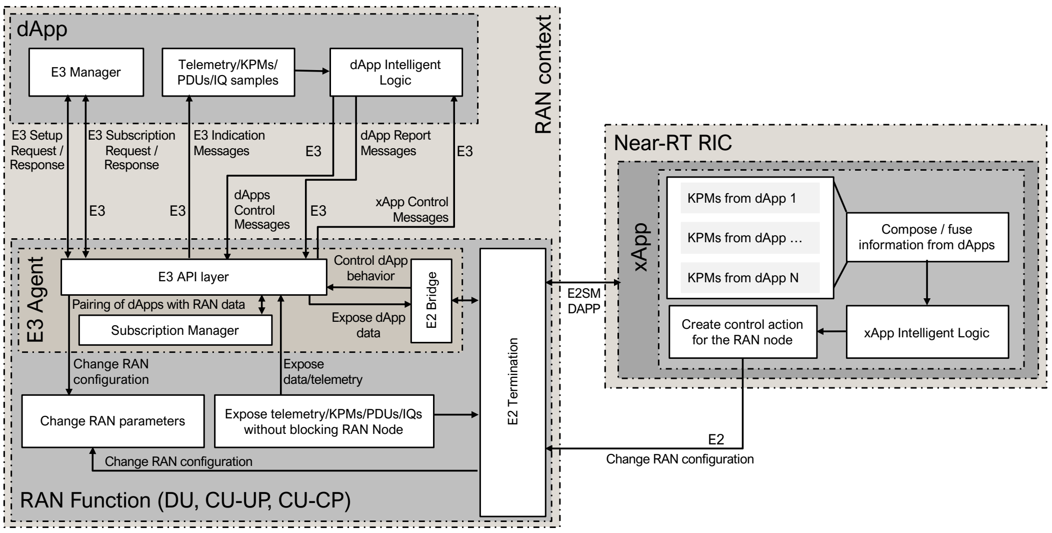


Figure 2: O-RAN near-RT RIC integration with dApps and a generic RAN function

3.1 Integrating dApps for Real-time Data Access and Control

dApps are designed as modular, pluggable services with interfaces to RAN functions—specifically, O-DU, O-CU-CP, and O-CU-UP functions—and to the O-RAN E2 interface for collaboration with xApps. The southbound interface (E3), structured as a service-based API, allows dApps to gather data and telemetry from RAN functions using various methods like streaming or polling with configurable intervals. The O-DU can provide granular data such as I/Q samples, BSRs, and CQI, essential for tasks like resource allocation and adaptive modulation. Additionally, uplink SRS and other data from channels like PUCCH and PUSCH can be accessed, either pre- or post-equalization, based on the needs of the dApps. Transport blocks and PDUs from layers such as MAC, RLC, PDCP, and SDAP can also be streamed or polled according to policies set by the dApps.

Control-plane data, including DCI and UCI, offers real-time insights into resource scheduling and allocation. Compute telemetry—monitoring CPU, RAM, and accelerator utilization—can be streamed at high frequencies or polled for performance assessments. Fronthaul configuration data is also available for alignment with dApp requirements. In the reverse direction, dApps can send control commands to the RAN functions within tight time constraints (e.g., 0.5 ms) to adjust elements such as MAC scheduling policies, beamforming weights, SINR-to-MCS mapping tables, and network slices dynamically. This bidirectional interaction enables precise, real-time optimization of RAN performance based on the use cases identified earlier in the discussion.

3.2 Service-based dApps Integration via E3 API in the O-RAN Architecture

Figure 2 shows dApps deployed alongside a RAN function, with an E3 agent exposing RAN functions through an API layer for configuration, data streaming, and control, enabling seamless interaction with both the RAN and xApps. The E3 API endpoint (or E3 agent) allows dApps to interact seamlessly with RAN functions, abstracting internal RAN functions and exposing them through a shared API. This API layer provides essential features, including dApp setup, configuration, data access (via streaming or polling), and control, ensuring the dApps remain implementation-agnostic. Additionally, the E3 agent coordinates with the E2 termination to facilitate communication between dApps and xApps in the Near-RT RIC, enabling control and adaptation across varying time scales. The architecture supports multiple dApps interacting with the same RAN function, with conflict mitigation and resource management handled as part of the dApp lifecycle, ensuring the network can meet its SLAs.

The E3 API leverages efficient IPC mechanisms, such as shared memory and logical FIFO queues, avoiding latency associated with network-based communications. The implementation of the E3 API in the RAN function controls data (and/or control) exposure for isolation and access control. Northbound data exposed by dApps is managed using a custom E2 service model (SM) identified by unique RAN function IDs during the E2 setup process. xApps on the Near-RT RIC can interact with dApps by sending subscription requests, enabling dynamic data collection and policy control throughout their lifecycle. This architecture allows the deployment of new dApps through SMO layer, ensuring ongoing adaptability to evolving network requirements.

3.3 dApp Architecture with Multiple RAN Functions

Figure 3 shows the dApp framework in scenarios involving multiple RAN functions. In this case, each vendor implements its own standardized E3 Agent using E3AP and E3SM procedures. The E3 Manager can subscribe to one or more agents simultaneously and correlates data across different vendors and RAN functions.

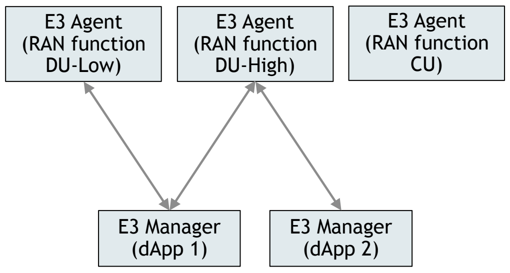

E3 Agent (RAN function DU-Low)
E3 Agent (RAN function DU-High)
E3 Agent (RAN function CU)
E3 Manager (dApp 1)
E3 Manager (dApp 2)

Figure 3 dApp architecture with multiple RAN functions

3.4 Security Aspects

The proper integration of dApps within the O-RAN architecture requires a security-by-design approach, particularly regarding access to sensitive user-plane data such as I/Q samples, PDU packets, and other telemetry data that can be directly or indirectly tied to users. While this research report does not mandate specific security mechanisms, it is essential to implement a robust security framework to protect user privacy and network integrity, along the following lines:

- Such a framework must ensure that user data payloads are never decoded or processed beyond the scope necessary for the dApp's designated function. Additionally, all accessed data must be deleted immediately after processing to prevent unauthorized retention or potential data breaches. Critical to this security model is the requirement that user-plane data remains strictly within the O-CU-CP/O-CU-UP/O-DU operational scope and is never forwarded to external systems or entities outside the RAN boundary without authorization.
- dApps can provide edge inference (thus without control interaction with the RAN) or dynamic real-time control. For the latter, as part of the conflict management and mitigation plane, the E3 agent on the RAN function implements sanity and boundary checks on dApp input.
- Authentication and verification mechanisms represent another crucial layer of security for dApp deployment and operation. The security framework must include comprehensive authentication procedures to verify that dApps are legitimate applications with proper authorization to access specific data types and perform designated control functions. This includes implementing digital signature verification during the dApp onboarding process and continuous runtime verification to ensure dApps maintain their authorized scope of operation. Furthermore, the system must incorporate verification processes that actively monitor and guarantee that dApps do not store, cache, or decode data beyond their permitted operational parameters. These verification mechanisms should include real-time monitoring of dApp behavior, periodic security audits, and automated detection systems that can identify and terminate dApps exhibiting unauthorized data handling practices. Additional verification steps can include pre-deployment testing in sandbox environments.

4 dApp E3 Interface

The dApps service-based architecture introduces the E3 interface, a standardized API that facilitates communication between dApps and RAN functions. The E3 API ensures direct integration of dApps with the O-RAN architecture, supporting telemetry, control, and data exchange. This architecture extends the existing O-RAN framework by enabling direct programmability of RAN functions at timescales below 10 ms, complementing existing xApps and rApps functionalities.

4.1 Functional Description

The E3 API enables dApps to operate as independent microservices that access user-plane data and apply real-time control logic to optimize RAN operations. The key functionalities supported by the E3 API include:

- **E3 Setup:** Establishes communication between a dApp and the RAN function, authenticating the dApp and registering its capabilities.
- **E3 Subscription:** Enables dApps to subscribe to telemetry data streams, polling mechanisms, and control actions from the RAN.
- **E3 Indication Messages:** Facilitates real-time exposure of data and telemetry to the dApps.
- **E3 Control Messages:** Enables dApps to issue control commands to modify RAN parameters dynamically.

The E3 interface is implemented as an application protocol (E3AP), which defines procedural exchanges between dApps and the RAN function. This ensures minimal latency and supports high-frequency interactions critical for real-time RAN optimization.

4.2 Architectural Components

The E3 interface consists of the following key components (also shown in Figure 2):

1. **E3 Agent** (within the RAN function): An intermediary service residing within the RAN function that manages communication between dApps and the RAN stack. The E3 Agent includes:
    a. **Subscription Manager:** Ensures optimized data retrieval and reduces redundant calls to the RAN function.
    b. **E3 API Layer**: Implements standard Inter-Process Communication (IPC) mechanisms for low-latency message exchange between dApps and RAN functions.
2. **E3 Manager** (within the dApp): Responsible for handling dApp registration, subscription, and control message dispatch.

The E3 Agent abstracts lower-layer interactions and enables secure, real-time communication with dApps, independent of vendor-specific RAN implementations. Additionally, it coordinates with the E2 termination to facilitate dApp-xApp interactions for hierarchical control.

4.3 Deployment Considerations

dApps are deployed as services co-located with O-DU/O-CU-CP/O-CU-UP functions, ensuring minimal communication overhead. The E3 API supports IPC methods such as shared memory and FIFO queues, optimizing data exchange latency while ensuring compliance with O-RAN architectural principles.

4.4 dApp and RAN Interactions over E3

The interaction between dApps and the RAN follows a structured messaging protocol, leveraging the E3 Application Protocol (E3AP) to establish control loops. This section defines the procedures for dApp registration, telemetry subscription, control messaging, and data exchange with the RAN, ensuring secure and efficient communication while maintaining the integrity of RAN operations.

Figure 4 illustrates the dApp interaction flow over the E3 interface, depicting the sequence of message exchanges between dApps and RAN functions.

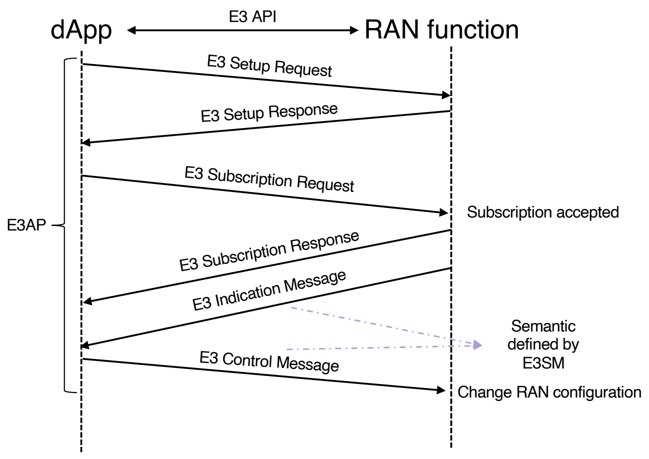


Figure 4 Message exchanges for a real time control loop between a dApp and the RAN function through the E3 interface.

4.4.1 dApp Registration Procedure

Procedure Description

This procedure allows a dApp to register with the E3 API and establish communication with the RAN function.

Procedure Steps

1. **E3 Setup Request**
    - If not registered already, the dApp sends an E3 Setup Request to the RAN function via the E3 API.
    - The request includes:
        - Supported telemetry data types
        - Required control permissions
        - Security authentication parameters
2. **E3 Setup Response**
    - The RAN function validates the setup request and sends an E3 Setup Response.
    - The response includes:
        - Confirmation of dApp registration
        - dApp identifier
        - Available telemetry data streams
        - Approved control actions
        - Subscription validity period

4.4.2 E3 Subscription Procedure

Procedure Description

This procedure enables a dApp to subscribe to telemetry data streams from the RAN function.

Procedure Steps

1. **E3 Subscription Request**
    - The dApp sends an E3 Subscription Request to the RAN function.
    - The request specifies:
        - Telemetry data types (e.g., CQI, scheduling decisions, interference metrics)
        - Data granularity and frequency
        - Subscription duration
2. **E3 Subscription Response**
    - The RAN function validates the request and responds with:
        - Acknowledgment of subscription
        - Approved data types and intervals
        - Subscription expiration details
        - Subscription Identifier
3. **E3 Data Transmission**
    - The RAN function begins streaming telemetry data to the dApp based on the agreed parameters.

4. **Subscription Refresh (Optional)**
   - If the dApp requires continued access, it sends a Subscription Refresh Request before expiration.
   - A default refresh interval can be set up by the vendor.
   - The RAN function either approves the extension or requires re-subscription.

4.4.3 E3 Control Message Procedure

Procedure Description

This procedure enables a dApp to issue control commands to the RAN function.

Procedure Steps

1. **E3 Control Request**
   - The dApp sends an E3 Control Message to the RAN function via the E3 API.
   - The message includes:
     - Target RAN function (e.g., scheduler, MAC, PHY)
     - Control parameters (e.g., beamforming weights, power control settings, resource block allocation)
     - Execution priority and timing
2. **RAN Function Execution**
   - The RAN function applies the requested modifications, ensuring compliance with network policies.
   - If the control action conflicts with other dApps, the RAN function may initiate conflict resolution.
3. **E3 Control Acknowledgment**
   - The RAN function sends an acknowledgment confirming successful execution or rejection with reasons.

4.4.4 E3 Indication and Event Notification Procedure

Procedure Description

This procedure enables the RAN function to send asynchronous notifications to dApps based on network events.

Procedure Steps

1. **Event Triggering**
   - The RAN function detects an event (e.g., congestion, mobility change, QoS violation).
2. **E3 Indication Message**

   - The RAN function sends an E3 Indication Message to the relevant dApp(s), including:
     - Event type
     - Affected RAN functions
     - Recommended dApp actions (if applicable)
3. **dApp Response (Optional)**
   - The dApp may acknowledge or act on the event notification, triggering a control message if necessary.

4.4.5 dApp Unsubscription Procedure

Procedure Description

This procedure enables a dApp to unsubscribe to telemetry data streams from the RAN function before the end of the subscription duration.

Procedure Steps

1. **E3 Unsubscription Request**
   - The dApp sends an E3 Unsubscription request to the RAN function to terminate the current subscription and streaming of telemetry data.
   - The request specifies:
     - Subscription Identifier
     - dApp Identifier
2. **E3 Unsubscription Acknowledgment**
   - The RAN function validates the request, removes the subscription, and responds with an acknowledgment of unsubscription.

4.4.6 dApp Deregistration Procedure

Procedure Description

This procedure allows a dApp to deregister from the RAN function when it is no longer required.

Procedure Steps

1. **E3 Deregistration Request**
   - The dApp sends a deregistration request to the RAN function, indicating termination of its operations.
2. **E3 Deregistration Acknowledgment**
   - The RAN function removes the dApp from the list of active services and stops sending telemetry or accepting control messages from it.
3. **Final Notification (Optional)**
   - If necessary, the RAN function notifies other dApps that relied on the deregistered dApp's data.

4.5 Integration with xApps via E2SM-DAPP

The E2 Service Model for dApps (E2SM-DAPP) defines a structured framework for interaction between dApps, xApps, and the RAN. This model enables dApps to expose real-time telemetry, AI-driven insights, and control capabilities through the E2 interface, facilitating optimized network management. The E2SM-DAPP supports seamless coordination between dApps and xApps to enhance the intelligence of Near-RT RIC while maintaining real-time adaptability and avoiding conflicts. This SM could also inspire extensions of existing SMs for relevant KPMs or control. Although conflict management and avoidance heavily depend on objectives and policies, conflict handling must account for priorities among different objectives and KPMs (e.g., prioritize control actions that favor energy savings versus actions that increase peak throughput, or vice versa).

The primary objectives of E2SM-DAPP include:

- Providing a standardized interface for dApps to interact with xApps and the RAN.
- Enabling efficient telemetry data sharing between dApps and xApps.

Supporting AI-driven decision-making and control actions.

Ensuring that dApps can act as data producers while xApps consume and interpret this data for advanced decision-making.

Figure 5 illustrates the procedural flow for dApp and xApp interactions over the E2 interface, highlighting the message exchange sequence between these components.

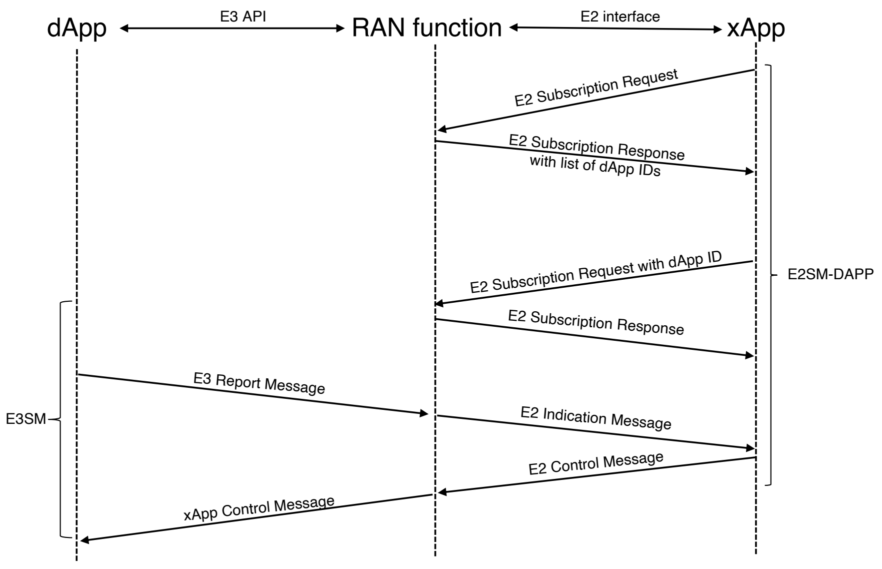


Figure 5 Interactions between a dApp, the E2 RAN function, and an xApp through the E3 interface and the E2SM-DAPP Custom Service Model.

4.5.1 E2SM-DAPP Registration Procedure

Procedure Description

This procedure mimics how RAN functions are registered to the Near-RT RIC via the E2 interface. Specifically, this procedure defines the steps required for a dApp to act as a RAN function and register with the E2 interface, allowing it to participate in data exchange and control actions with xApps and RAN functions.

Procedure Steps

1. **E2SM-DAPP Setup Request**
   - The dApp sends an E2SM-DAPP Setup Request to the Near-RT RIC.
   - The request includes:
     - dApp identifier
     - Supported telemetry data streams
     - Required control functions
     - Data exchange capabilities (push/pull model)
2. **E2SM-DAPP Setup Response**
   - The Near-RT RIC validates the request and responds with:
     - Confirmation of dApp registration
     - Approved telemetry streams and control actions
     - Subscription duration and renewal policies
3. **E2SM-DAPP Setup Acknowledgment**
   - The dApp acknowledges the registration and prepares for data exchange.

If a dApp registers after an xApp initiates a subscription request, the RAN function should send a subscription update message to notify that a new dApp has registered.

4.5.2 E2SM-DAPP Telemetry Subscription Procedure

Procedure Description

This procedure allows xApps to subscribe to telemetry data streams from dApps using the E2 interface.

Procedure Steps

1. **E2 Subscription Request**
   - The xApp sends an E2 Subscription Request to the Near-RT RIC to access dApp-generated telemetry.
   - The request specifies:
     - Required telemetry types (e.g., signal quality, interference levels, resource allocation metrics)
     - Subscription frequency and duration

2. **E2 Subscription Response**
   - The Near-RT RIC validates the request and responds with:
     - Approved telemetry streams
     - Data refresh intervals
     - Subscription expiration details
3. **Telemetry Data Transmission**
   - The dApp begins transmitting telemetry data to the xApp via the Near-RT RIC.
   - Data is delivered through:
     - Real-time streaming
     - Periodic polling
     - Event-driven notifications
4. **Subscription Modification (Optional)**
   - The xApp may modify the subscription by adjusting the data granularity or frequency.

4.5.3 E2SM-DAPP Control Procedure

Procedure Description

This procedure enables xApps to send control messages to dApps, modifying their operational parameters.

Procedure Steps

1. **E2 Control Request**
   - The xApp sends an E2 Control Message to the dApp via the Near-RT RIC.
   - The message includes:
     - Target control function (e.g., scheduler, resource allocator)
     - Adjustment parameters (e.g., priority weights, power levels, bandwidth allocation)
     - Execution priority
2. **dApp Control Execution**
   - The dApp processes the control request and applies the changes.
   - If conflicts arise, the Near-RT RIC resolves them before execution.
3. **E2 Control Acknowledgment**
   - The dApp sends an acknowledgment confirming successful execution or rejection with a reason.

4.5.4 E2SM-DAPP Indication Procedure

Procedure Description

This procedure allows dApps to notify xApps about network events that require immediate attention.

Procedure Steps

1. **Event Detection**
   - The dApp identifies a significant event (e.g., congestion, degradation in quality, sudden traffic spikes).
2. **E2 Indication Message**
   - The dApp sends an E2 Indication Message to the xApp via the Near-RT RIC.
   - The message contains:
     - Event type
     - Affected RAN functions
     - Recommended xApp actions
3. **xApp Response (Optional)**
   - The xApp may acknowledge or react to the event by sending a control message.

4.5.5 E2SM-DAPP Deregistration Procedure

Procedure Description

This procedure allows a dApp to deregister from the Near-RT RIC and terminate its operations.

Procedure Steps

1. **E2SM-DAPP Deregistration Request**
   - The dApp sends a deregistration request to the Near-RT RIC, indicating that it is terminating its services.
2. **E2SM-DAPP Deregistration Acknowledgment**
   - The Near-RT RIC removes the dApp from the list of active services and stops telemetry transmission.
3. **Final Notification (Optional)**
   - If required, the Near-RT RIC informs xApps relying on the deregistered dApp’s data.

5 dApp Lifecycle and Interaction with Near-RT RIC, Non-RT RIC, SMO

This section outlines how the life-cycle management (LCM) of dApps can be aligned with and integrated into the LCM practices defined within the O-RAN architecture. Building on and extending the guidelines provided in [12, 13], we explore the standard procedures for managing O-RAN applications, focusing on cloud-native deployments, orchestration, and virtualization of RAN solutions. The LCM framework follows a structured seven-phase SDLC model—comprising Needs, Ideation, Analysis, Development, Delivery, Validation, and Operation—ensuring that all applications are consistently developed, onboarded, and managed. The LCM model follows the same three-stage structure as considered in [12], i.e., Development, Onboarding, and Operations.

Figure 6 presents an overview of the LCM process for dApps, with a specific focus on the onboarding and operational phases. Although the development process applies universally to xApps, rApps, and dApps, the figure highlights the distinctive aspects of dApp onboarding and operation. The following sections further adapt and extend these stages to the specific requirements of the dApp lifecycle. In short, the LCM of dApps differs from LCM of xApps and rApps in that it (i) involves deployment and monitoring of software components at the O-DU/O-CU-CP/O-CU-UP; and (ii) it extends to the E3 interface.

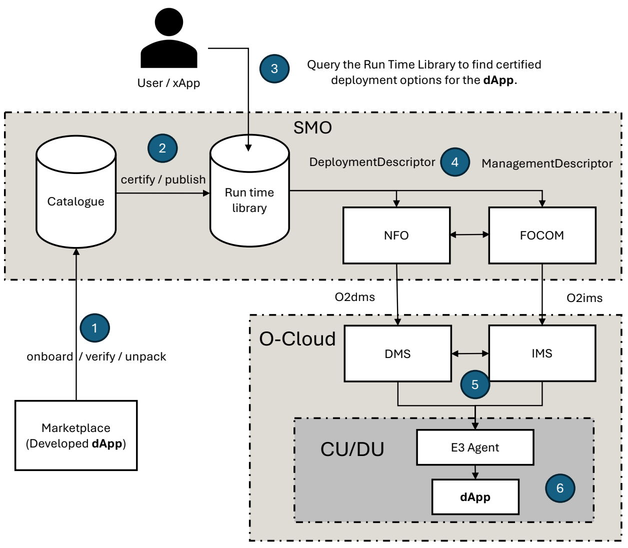


Figure 6: dApp LCM diagram

5.1 Development

The development phase follows the Solution App and AppPackage Lifecycles, ensuring smooth design, deployment, and management. It begins with defining the dApp's use case, requirements, and KPIs, followed by integrating logic and telemetry. The dApp is then containerized (e.g., with Docker) for deployment on cloud platforms like OpenShift. A Solution AppPackage, containing key descriptors (e.g., DeploymentDescriptor), ensures proper onboarding and orchestration. Digital signatures and checksums secure the package, verifying its integrity throughout deployment.

5.2 Onboarding

In the onboarding phase, the dApp AppPackage, prepared during development, is retrieved from the marketplace where it was initially stored. The LCM is the same both in the case where the marketplace is centralized or when it is vendor-specific. The package undergoes security checks and validation to ensure its authenticity. Once approved, its components are unpacked and organized in the SMO’s catalog, as illustrated in step 1 of Figure 6. Afterward, each recommended configuration is certified and added to the runtime library, as shown in step 2. Operators can initiate the onboarding process by searching the runtime library for certified deployment options. Alternatively, xApps can also initiate dApp deployments by accessing the runtime library through the O1 interface and requesting deployment via the same interface, as depicted in step 3. This enables xApps to leverage dApps for real-time operations.

5.3 Deployment

Unlike xApps and rApps, which are deployed within the Near-RT RIC and Non-RT RIC, respectively, dApps follow a distinct deployment path, being hosted directly on the O-DU/O-CU-CP/O-CU-UP. These dApps are implemented as Cloud-Native Functions (CNFs) [13], aligning with the O-RAN ALLIANCE’s standards for Network Function (NF) deployment. Managed through the SMO framework and the O-Cloud, the dApp’s deployment and lifecycle are facilitated via the O2 interface, enabling seamless interaction between the SMO and the Deployment Management Service (DMS) of the O-DU/O-CU-CP/O-CU-UP.

This deployment strategy allows dApps to harness the real-time processing power of the O-DU/O-CU-CP/O-CU-UP while benefiting from the scalability and flexibility of the O-Cloud infrastructure. Unlike the RIC-hosted xApps and rApps, dApps operate independently of the RIC environments, focusing on real-time operations and efficient resource management directly within the O-DU/O-CU-CP/O-CU-UP. This cloud-native approach ensures smooth integration with the O-DU/O-CU-CP/O-CU-UP while maintaining optimal network function performance. The deployment steps for dApps are outlined in steps 4, 5, and 6 of Figure 6, with the detailed information flow described in Table 1.

Table 1: dApp Deployment

Use Case Stage	Evaluation/Specification
Goal	To instantiate and configure a dApp on the O-DU/O-CU-CP/O-CU-UP, facilitating its deployment, management, and operationalization directly via the SMO and the O-DU/O-CU-CP/O-CU-UP.
Actors and Roles	• SMO (Service Management and Orchestration): Orchestrates the deployment, configuration, and lifecycle management of the dApp. • O-DU (Distributed Unit): Possible host of the dApp and provides necessary network and compute resources for its operation. • O-CU-CP and O-CU-UP (Centralized Unit): Possible host of the dApp and provides necessary network and compute resources for its operation.

	• Network Function Orchestration (NFO): Interfaces with the O-DU for deploying and configuring the dApp. • Deployment Management Services (DMS): Manages the deployment resources on the O-DU.
Assumptions	• The SMO and O-DU/O-CU-CP/O-CU-UP are available and operational. • The dApp package has been validated, verified, and cataloged in the SMO's runtime library. • The O-DU is pre-configured and ready to host dApp instances.
Preconditions	• The dApp is onboarded and certified within the SMO. • The O-DU/O-CU-CP/O-CU-UP has the necessary resources and access to the container images required for the dApp deployment.
Begins When	A Network Function Install Project Manager initiates a request to the SMO for deploying a new dApp instance on the O-DU/O-CU-CP/O-CU-UP.
Step 1 (M)	The SMO receives a service request to deploy the dApp instance on the O-DU/O-CU-CP/O-CU-UP.
Step 2 (M)	The SMO decomposes the service request and identifies all dApps to be deployed and their deployment order.
Step 3 (M)	The SMO determines which O-DU or O-CU-CP/O-CU-UP and deployment parameters to use, based on policies or explicit input from the Network Function Install Project Manager.
Step 4 (M)	The SMO retrieves the CloudNativeDescriptor for the dApp from the runtime library.
Step 5 (M)	The SMO directs the O-Cloud DMS using O2 to create the dApp deployment.
Step 6 (M)	DMS allocates the necessary compute, storage, and network resources on the O-DU/O-CU-CP/O-CU-UP as per the dApp deployment request.
Step 7 (M)	The SMO sets up the initial configuration for the dApp, such as environment variables, network policies, and access parameters.
Step 8 (M)	DMS deploys the dApp container(s) on the O-DU/O-CU-CP/O-CU-UP, setting up the necessary resources and connectivity.
Step 9 (M)	DMS notifies the SMO that the dApp deployment has been successfully instantiated and provides a Deployment ID.
Step 10 (M)	The SMO updates its dApp inventory with the new Deployment ID and deployment status.
Step 11 (M)	The deployed dApp instance reads its initial configuration from the provided parameters and begins its operation.
Step 12 (O)	The SMO continuously monitors the dApp's health, performance, and connectivity. It can also perform scaling, updates, or redeployment as required.
Step 13 (M)	The SMO informs the Network Function Install Project Manager of the overall success or failure of the request.
Ends When	The dApp instance is successfully deployed, configured, and operational on the O-DU/O-CU-CP/O-CU-UP, with the SMO actively managing its lifecycle.
Post Conditions	• The dApp is actively running and functional on the O-DU/O-CU-CP/O-CU-UP. • The SMO has an updated inventory reflecting the deployed dApp, and lifecycle management is in place.

5.4 Interaction with Near-RT RIC, Non-RT RIC, and SMO

Since dApps operate internally within the O-DU or O-CU-CP/O-CU-UP, their data must be made accessible through the standardized interfaces defined by the O-RAN ALLIANCE. These interfaces facilitate the exchange of management and telemetry data between the O-DU/O-CU-CP/O-CU-UP and other O-RAN components, including xApps, rApps, and the SMO. As illustrated in Figure 6, the E3 Agent bridges the dApp's data with the E2 and O1 interfaces, ensuring that the information flows smoothly to the Near-RT RIC and SMO. Through this setup, rApps access dApp-generated data via the O1 interface, using the R1 interface for data consumption.

To ensure seamless communication and data handling, O-RAN working groups must further refine these standards. WG10 needs to enhance the O1 interface to efficiently store and process dApp data, while WG3 must define appropriate E2 IEs to support dApp operations. Although dApp data can be shared as part of the telemetry feed, more detailed standards and protocols are required to manage this data flow effectively. These refinements will ensure full interoperability within the O-RAN ecosystem, allowing dApps to operate cohesively alongside other network functions and components.

6 A Reference dApp Prototype – Lessons Learned

This section defines the dApp Framework, a structured and modular implementation designed to integrate real-time control applications (dApps) into the Open RAN ecosystem. The framework follows the E3 Application Protocol (E3AP), enabling dApps to interact with RAN functions via a well-defined interface. Moreover, modularity makes it possible to reuse software components and combine several functionalities (potentially from multiple vendors) to generate dApps executing complex tasks. The framework consists of three primary components:

1. **dApp Class** – Encapsulates the logic of each dApp instance, defining its core functionality, execution lifecycle, and control actions.
2. **E3Interface Class** – Manages interactions between the dApp and the underlying RAN function, ensuring data exchange and execution of control loops.
3. **E3Connector Class** – Provides communication mechanisms, supporting multiple transport and link-layer protocols for efficient, real-time message exchange.

The framework is designed to support multiple dApps running concurrently on a RAN function while ensuring low-latency control execution. It follows a service-based architecture where dApps operate as microservices, facilitating dynamic deployment and integration with existing Open RAN components such as xApps and Near-RT RIC.

6.1 dApp Class

The dApp class is an abstract base class that serves as the foundation for implementing various dApps in Open RAN. It provides a structured framework for managing E3 Service Model (E3SM) operations, ensuring that each dApp is equipped with essential functionalities for data acquisition, processing, and control execution.

Each specific dApp use case extends this class to incorporate its own set of functionalities, parameters, and operational logic. By encapsulating all E3SM-related functionalities into a single entity, the dApp class streamlines the creation of new dApp instances while ensuring modularity and scalability.

To facilitate seamless interaction with the RAN function, the dApp class integrates with the E3Interface class. This enables efficient publish-subscribe communication between the dApp and the RAN, where dApps register callbacks to receive real-time updates from specific RAN function IDs. The E3Interface is implemented as a private variable within the dApp class, ensuring controlled access to RAN telemetry and maintaining structured data flow.

The dApp class provides two key methods for its operation:

- **setup_connection()** – Establishes communication with the RAN function via the **E3Interface** and registers callbacks for data updates.
- **control_loop()** – Implements a continuous polling mechanism that triggers the dApp's core processing logic at predefined intervals, ensuring real-time responsiveness.

By leveraging this architecture, the dApp class ensures that all dApps maintain a standardized structure while allowing flexibility for custom implementations based on specific network optimization and control requirements.

6.2 E3Interface Class

The E3Interface class serves as the core communication layer between dApps and the RAN function, providing a standardized API for accessing and managing the E3 Application Protocol (E3AP)**.** This class is responsible for handling data exchange, subscription management, and control message dispatch, ensuring efficient interaction between dApps and the network.

To maintain consistency and prevent redundant connections, the E3Interface class follows a singleton design pattern, ensuring that all dApps deployed on the same RAN function share a single E3 interface instance. This design optimizes resource utilization and maintains synchronization across multiple dApps operating within the system.

The dApp class interacts with the E3Interface through dedicated methods, enabling seamless integration with real-time Open RAN operations. Key functionalities include:

- **add_callback()** – Registers a dApp to receive updates from specific RAN function IDs.
- **remove_callback()** – Unsubscribes a dApp from RAN telemetry data.
- **schedule_control()** – Sends E3 Control Actions to the RAN, allowing dApps to modify network parameters dynamically.
- **schedule_report()** – Transmits E3 Report Messages to xApps in the Near-RT RIC for policy updates and decision-making.
- **handle_incoming_data()** – Processes incoming RAN telemetry and forwards it to registered dApps for analysis.

By implementing these functions, the E3Interface class acts as a bridge between dApps and the RAN, ensuring low-latency data flow and real-time control execution**.**

6.3 E3Connector Class

Establishing fast and reliable connectivity between dApps, the RAN function, and xApps is critical for enabling real-time network monitoring and control. The efficiency of this connectivity directly impacts the latency and responsiveness of the dApp-based control loops. To determine the optimal trade-offs between performance, reliability, and protocol efficiency, we have implemented and evaluated multiple connectivity solutions.

The E3Connector class serves as an abstract communication layer within the E3Interface, responsible for establishing and managing socket connections between dApps and the RAN function. It provides a standardized set of methods to:

- **Establish initial pairing** between the dApp and the RAN function.
- **Manage real-time message exchange**, ensuring minimal latency.
- **Handle inbound telemetry and control messages** from the RAN.
- **Transmit control and report messages** from dApps back to the RAN or xApps.

To facilitate these operations efficiently, the E3Connector class manages three distinct socket connections, each dedicated to a specific communication function. These three socket connections are illustrated in Table 2 and execute the following functionalities:

1. **Setup & Subscription Exchange**
    - Handles the E3 Setup Request-Response and **E3 Subscription Request-Response** exchanges.
    - Establishes a secure and reliable connection between the dApp and the RAN function.
2. **Inbound Data Reception**
    - Processes real-time telemetry updates from the RAN.
    - Handles E3 Indication Messages carrying RAN measurements.
    - Manages xApp Control Messages forwarded from the Near-RT RIC.
3. **Outbound Control & Reporting**
    - Sends E3 Control Messages to the RAN for real-time reconfiguration.
    - Transmits E3 Report Messages to xApps for decision-making and policy adaptation.

By implementing this structured socket management system, the E3Connector class ensures efficient, low-latency message handling, enabling real-time interactions between dApps, RAN functions, and xApps while supporting various transport-layer protocols optimized for Open RAN environments.

6.4 E3Connector Implementations

The E3Connector class has two specialized child classes—**POSIXConnector and ZMQConnector**—each designed to support different data link layers and transport protocols for dApp-to-RAN communication. These implementations ensure low-latency and reliable message exchange, tailored for real-time Open RAN control loops.

Table 2: Connector types

Connector Type	Link Layer	Transport Protocols	Communication Pattern	Key Features
POSIXConnector	POSIX System Functions	TCP, Unix Domain Sockets (IPC), SCTP	**Client-Server Model**: The receiving entity (RAN) is the **server**, while the sending entity (dApp) is the **client**	- Uses **low-level POSIX APIs** (shared memory, pipes, message queues, and sockets). - Requires **manual handling of synchronization, serialization, and transport.** - Optimized for **tight-coupled systems** with high performance but limited flexibility across distributed networks.
ZMQConnector	**ZeroMQ Messaging Library**	TCP, Unix Domain Sockets (IPC), SCTP	**Hybrid Model**: 1. **Request-Reply (REQ-REP)** for the **E3 Setup Request-Response** (one-time initialization). 2. **Publish-Subscribe (PUB-SUB)** for continuous inbound/outbound data exchanges.	- Provides **asynchronous, high-performance messaging** for both intra-machine and inter-machine communication. - Abstracts connection management and **data serialization** for simplified API calls. - Enables

				flexible built-in communication patterns to optimize real-time data handling. - Supports **SCTP**, despite ZeroMQ's current limitations with full SCTP compatibility.

6.4.1 Transport Protocols in E3Connector

Both POSIXConnector and ZMQConnector support three transport-layer protocols to accommodate diverse communication requirements:

1. TCP – Reliable, connection-oriented communication for wide-area networking.
2. Unix Domain Sockets (IPC) – Optimized for local inter-process communication, providing the lowest latency.
3. SCTP – Designed for multi-stream transport, improving resilience in networked environments.

While additional protocols could be implemented, UDP is explicitly excluded due to its lack of reliable message delivery, which is not suitable for RAN operations requiring low error rates and guaranteed data integrity.

6.4.2 Distributed Deployment Considerations

1. The ZMQConnector enables dApps to communicate with RAN functions located on remote virtual hosts, aligning with the O-RAN vision of network function disaggregation.
2. Data received via the E3Connector socket (regardless of the transport mechanism) is processed using a callback-based subscription manager, ensuring real-time data availability while avoiding unnecessary duplication.

6.4.3 Message Formatting & Standard Compliance

1. Communication between the RAN function and dApps adheres to 3GPP and O-RAN standard message exchange protocols.

2. ASN.1 (Abstract Syntax Notation One) is used for message formatting in E3AP and E3SM, ensuring compact, efficient, and rapid serialization of control and telemetry data.
3. Each dApp implementation must define its own SM and intelligent control logic, extending the dApp abstract class for specific network optimization tasks.

Figure 7 illustrates the different logical pairings of the E3Connector configurations based on the chosen transport protocols and communication patterns.

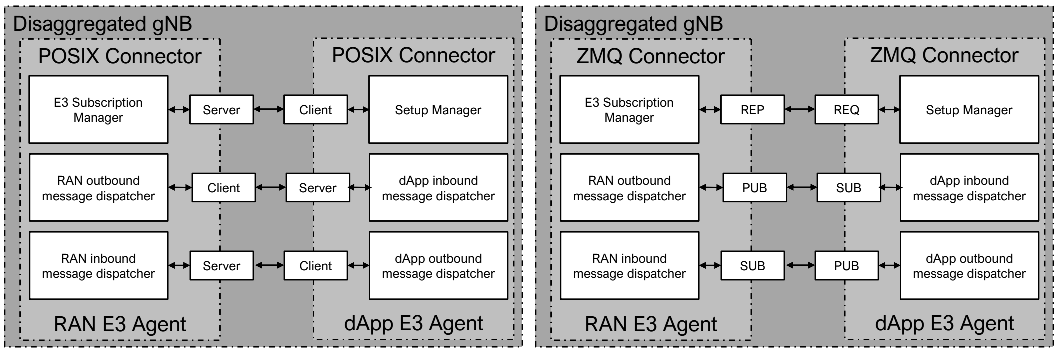


Figure 7 Design of the E3Connector sockets.

6.5 Real-Time Capabilities of dApps

The real-time capabilities of dApps were demonstrated through the integration of OpenAirInterface (OAI) and T-tracer, with modifications to optimize data exchange and reduce latency. A dedicated E3 Agent module was introduced in OAI to facilitate seamless communication between dApps and RAN components using the E3AP interface. The T-tracer tool was extended to support inter-process communication (IPC) via Unix domain sockets, reducing overhead and improving real-time metric extraction. Additionally, a C-based implementation of the E3Connector was developed and integrated into the OAI codebase, ensuring efficient message handling without external dependencies.

To evaluate performance, experiments were conducted on the O-RAN testbeds deployed at Northeastern University Colosseum and Arena, leveraging the SpectrumSharingDApp to benchmark real-time control loop execution. The setup included Dell PowerEdge R730 servers, Linux Ubuntu 20.04 OS, USRP x310 SDRs, and an OAI gNB operating in Frequency Range 1 (FR1) at 3.6192 GHz with a 40 MHz bandwidth. In all use cases, dApps execute on the same node hosting the OAI O-DU/O-CU-CP/O-CU-UP software which executes functionalities that need tight latency bounds (e.g., scheduling, channel estimation, coding and decoding). Each test focused on a five-minute window of active UE transmissions, measuring the efficiency of dApps in executing real-time control decisions. Results showed that the dApps framework maintained control loop latencies well below 1 ms, significantly outperforming the 10 ms threshold required for real-time RAN operations.

Comparisons of different transport protocols revealed that IPC introduced the least overhead, making it the most efficient method for real-time data exchange, whereas TCP and SCTP incurred additional latency. These findings validate the feasibility of dApps in enabling ultra-low-latency AI-driven control within Open RAN environments. Future studies will also focus on scalability aspects related to extraction, processing and storage of KPMs and data at high rates (e.g., I/Q samples, user-plane PDU packets) in large-scale deployments. This will facilitate the definition of safe thresholds to determine the maximum allowable rate at which a certain KPM or parameter can be retrieved by dApps while guaranteeing real-time inference.

6.6 Use Cases and Evaluation

This section presents two dApp-based use cases developed to demonstrate the feasibility and performance of real-time Open RAN control. The use cases leverage the E3 interface and dApp framework to interact with the RAN in real-time. The evaluation includes performance analysis based on latency, accuracy, and reliability of the proposed dApps. Additional literature includes evaluation of dApp use cases, including further considerations on performance, isolation, and implementation. We refer the interested reader to [10] [14] [15], among others.

6.6.1 Spectrum Sharing dApp

Use Case Objective

The Spectrum Sharing dApp enables dynamic spectrum allocation in Open RAN by detecting incumbent users and adjusting transmission parameters accordingly. The objective is to maximize spectrum utilization while ensuring interference mitigation for primary users.

This use case focuses on:

- **Real-time spectrum sensing** using RAN telemetry data.
- **Dynamic spectrum reallocation** for 5G and beyond.
- **Integration with xApps for spectrum policy adaptation.**

Setup

The Spectrum Sharing dApp connects to the gNB via the E3 interface and performs real-time spectrum sensing, detection, and control adjustments. The dApp is configured to retrieve I/Q samples with a tunable sensing periodicity. Experimental results in [4] show that this prototype has a maximum sensing periodicity of 400 microseconds. The key steps are:

1. **I/Q Sample Extraction** – The dApp registers a callback with the E3Interface to continuously receive I/Q samples from sensing symbols.
2. **Incumbent Detection** – An inference algorithm analyzes the **sample magnitudes**, comparing them against a calibrated **threshold** to detect **incumbent users**.
3. **PRB Assessment** – The dApp identifies affected PRBs and compiles an exclusion list to prevent gNB scheduling on those PRBs.
4. **Control Execution** – The PRB exclusion list is sent to the RAN function via the E3 interface, ensuring interference-free transmission.

This **real-time PRB optimization** enhances spectrum efficiency while minimizing interference with incumbent users. The set up illustrated in Figure 8.

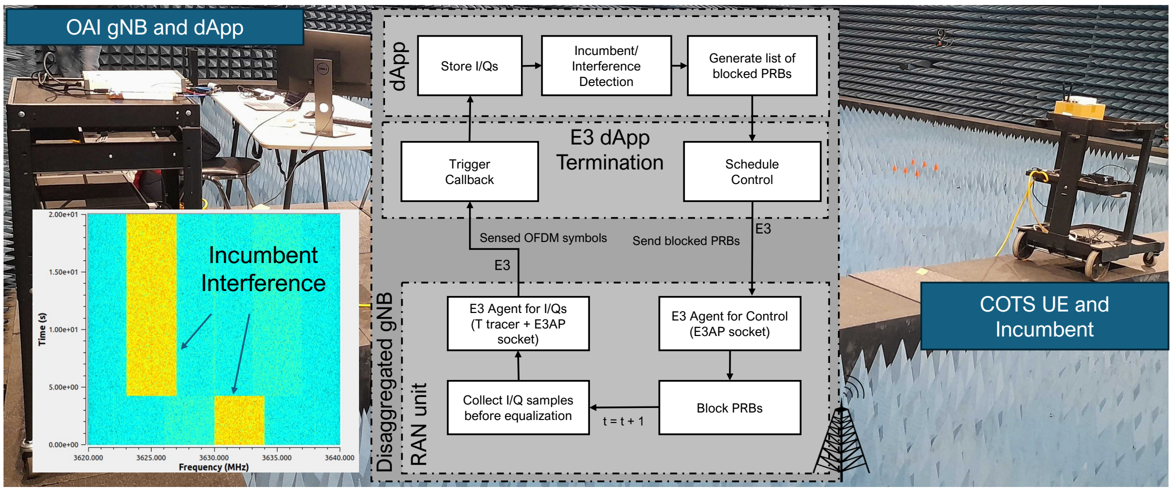


Figure 8 Intelligent real-time control loop for the PRB blacklisting. We omit in this figure the initial E3AP

Results

The evaluation of the Spectrum Sharing dApp was conducted using a Keysight EXA Spectrum Analyzer N9010B to analyze three different spectrum configurations. In the baseline scenario, the gNB utilized all available PRBs without restriction. When an incumbent user occupied a portion of the spectrum, interference degraded gNB-to-UE performance. The dApp detected the incumbent, identified the affected PRBs, and instructed the RAN to vacate them, allowing coexistence with reduced interference. The tests were conducted in Arena and Colosseum testbeds, with customized noise floor calibrations (20 dBm for Arena, 53 dBm for Colosseum) to optimize PRB blocking thresholds. The results showed minimal impact on throughput when the dApp was enabled, with Colosseum throughput remaining at ~71 Mbps and Arena throughput around ~76 Mbps, demonstrating that the dApp's operations introduced negligible overhead. When the incumbent was present and spectrum sharing was disabled, UE throughput dropped by nearly 50% due to interference. However, with the dApp-enabled PRB blocking, interference was mitigated, improving throughput and reducing retransmissions. The findings confirm that real-time PRB adaptation effectively enables spectrum sharing while ensuring stable UE performance. A possible extension to this use case considers federated dApps that can coordinate with each other to control interference and block PRBs by leveraging information exchanged via message passing.

6.6.2 Sensing and Positioning dApp

Use Case Objective

With the adoption of higher frequency bands (FR2, FR3, THz) and massive antenna arrays, beyond 5G systems are expected to provide enhanced data rates, high-resolution positioning, and advanced environment sensing. Current uplink-based positioning methods rely on the gNB extracting 3GPP-defined positioning metrics from uplink channel estimates and forwarding them to the Location Management Function (LMF) in the 5G core. These conventional techniques prioritize low-complexity signal processing for real-time feasibility but are limited in precision and adaptability.

Emerging super-resolution and ML-based positioning algorithms offer superior accuracy but require detailed RAN-internal data, such as Channel Impulse Response (CIR), which is traditionally inaccessible outside the RAN. The Positioning dApp addresses this limitation by enabling real-time CIR extraction and processing within the RAN, unlocking advanced UE localization and sensing capabilities beyond existing 3GPP methods, thus acting as complement to 3GPP defined approaches.

Setup

The Positioning dApp is tested in a real-world indoor lab environment, where a single-antenna gNB and a UE are positioned 10 meters away from each other and communicate over a line-of-sight (LOS) channel. The experimental setup includes:

- **Hardware & Software**:
  - **gNB & UE**: Both rely on the **OAI protocol stack** and **USRP B210 software-defined radios (SDRs)**.
  - **RF Front-End**: An **SC2430 NR signal conditioning module** is used at the gNB for enhanced signal processing.
  - **Operating Parameters**:
    - **Band**: n78
    - **Carrier Frequency**: 3.69 GHz
    - **Bandwidth**: 40 MHz
- **CIR Extraction & Processing**:
  - The **UL CIR** is continuously extracted by the **Positioning dApp**, as shown in Figure 9, to derive the **UE's position**.

- **Real-time CIR analysis** enables tracking of multipath components, improving **localization accuracy** beyond traditional 3GPP methods.

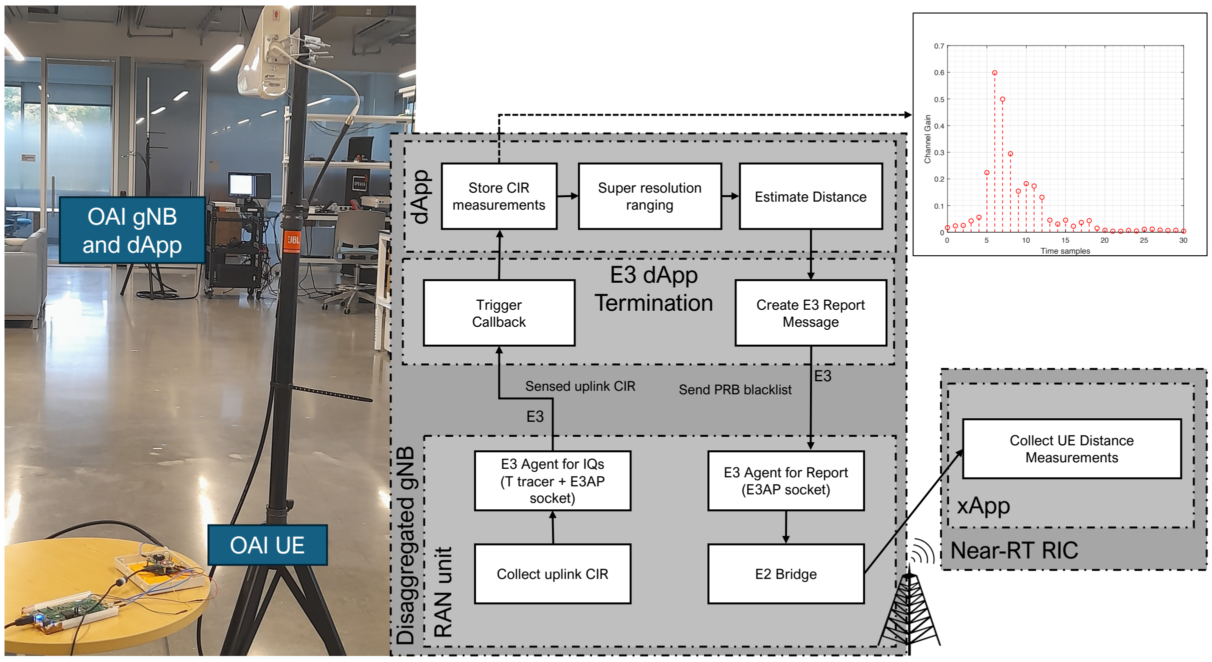


Figure 9 Setup for the Sensing and Positioning dApp and UL CIR extracted by the dApp

Results

The results demonstrate that channel gain and CIR characteristics are highly dependent on UE position, as path loss, shadowing, and multipath effects vary with distance and environmental changes. Closer UEs exhibit higher gain and reduced path loss, while the CIR structure shifts dynamically due to obstacles and reflections. As the UE moves, large-scale fading (caused by obstructions) and small-scale fading (from multipath interference) alter the channel response. The Positioning dApp effectively tracks these variations, proving its ability to support real-time UE localization. These findings confirm that CIR-based dApp positioning offers a robust and precise alternative to existing 3GPP location techniques, making it a powerful tool for Open RAN-based localization and sensing applications.

7 Conclusions

The current O-RAN architecture does not provide tools to achieve efficient real-time observability and control in O-RAN. dApps have been proposed as a way to enable such capabilities in previous research reports, with a focus on feasibility studies and use cases. In this technical report, we have presented a comprehensive reference architecture for dApps in O-RAN systems, addressing the critical need for sub-10 ms real-time control and user-plane data processing capabilities in next-generation cellular networks. Through the proposed architecture, which includes the novel E3 interface and E2SM-DAPP service model, we have shown that dApps can be directly integrated into existing O-RAN deployments with minimal impact on current standardization efforts. We also provide directions toward a secure-by-design dApp framework, including LCM, authentication, validation, and interface design.

Our open-source implementation and extensive performance evaluation on the Colosseum and Arena platforms validate the feasibility of dApps, showing that they can efficiently process I/Q samples and perform complex inference tasks within 450 microseconds—well below the real-time threshold. The successful implementation of Integrated Sensing and Communications (ISAC) use cases for positioning and spectrum sharing further illustrates the practical benefits of dApps in enabling advanced 6G applications that require ultra-low latency, minimum throughput levels and direct access to user-plane data. As the O-RAN nGRG continues to explore the adoption of dApps, this technical report provides the foundational framework, implementation guidelines, and performance benchmarks necessary to advance real-time AI/ML capabilities at the RAN edge, ultimately paving the way for more intelligent, adaptive, and efficient cellular networks.

References

[1] L. Baldesi, F. Restuccia and T. Melodia, "ChARM: NextG spectrum sharing through data-driven real-time O-RAN dynamic control," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, 2022.

[2] Northeastern University, NVIDIA, Mavenir, MITRE, Qualcomm, "dApps for Real-Time RAN Control: Use Cases and Requirements," O-RAN ALLIANCE nGRG, 2024.

[3] S. D'Oro, M. Polese, L. Bonati, H. Cheng and T. Melodia, "dApps: Distributed applications for real-time inference and control in O-RAN," *IEEE Communications Magazine,* vol. 8, pp. 52-58, 2022.

[4] A. Lacava, L. Bonati, N. Mohamadi, R. Gangula, F. Kaltenberger, P. Johari, S. D'Oro, F. Cuomo, M. Polese and T. Melodia, "dApps: Enabling real-time AI-based Open RAN control," *Computer Networks,* p. 111342, 2025.

[5] W.-H. Ko, U. Ghosh, U. Dinesha, R. Wu, S. Shakkottai and D. Bharadia, "EdgeRIC: Empowering Realtime Intelligent Optimization and Control in NextG Networks," in *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, 2024.

[6] S.-D. Wang, H.-M. Wang, C. Feng and V. Leung, "Sequential anomaly detection against demodulation reference signal spoofing in 5G NR," *IEEE Transactions on Vehicular Technology,* vol. 72, 2022.

[7] Y. Chen, T. Y. Hou, L. Wenjing, J. Reed and S. Kompella, "M 3: A sub-millisecond scheduler for multi-cell mimo networks under c-ran architecture," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, 2022.

[8] M. Giordani, M. Polese, A. Roy, D. Castor and M. Zorzi, "A tutorial on beam management for 3GPP NR at mmWave frequencies," *IEEE Communications Surveys & Tutorials,* 2018.

[9] Qualcomm Technologies and Dell, "Spectrum Sharing based on Shared O-RUs," O-RAN next Generation Research Group (nGRG), Research Report, 10 2023, report ID: RR-2023-05, 2023.

[10] H. Cheng, P. Johari, M. A. Arfaoui, F. Periard, P. Pietraski, G. Zhang and T. Melodia, "Real-Time AI-Enabled CSI Feedback Experimentation with Open RAN," in *2024 19th Wireless On-Demand Network Systems and Services Conference (WONS)*, 2024.

[11] I. Palama, S. Bartoletti, G. Bianchi and N. Blefari Melazzi, "5G positioning with SDR-based open-source platforms: Where do we stand?," in *2022 IEEE 11th*

IFIP International Conference on Performance Evaluation and Modeling in Wireless and Wired Networks (PEMWN), 2022.

[12] O-RAN Working Group 10, "O-RAN operations and maintenance architecture," ORAN-WG10O.OAM-Architecture-R004 Technical Specification, 2024.

[13] O-RAN Working Group 6, "Cloudification and orchestration use cases and requirements for O-RAN virtualized RAN," O-RAN-WG6.ORCH-USE-CASES-R003-v11.00 Technical Specification, 2024.

[14] D. Villa, M. Belgiovine, N. Hedberg, M. Polese, C. Dick and T. Melodia, "Programmable and GPU-Accelerated Edge Inference for Real-Time ISAC on NVIDIA ARC-OTA," in *Submitted to an ACM conference*, 2025.

[15] N. N. Santhi, D. Villa, M. Polese and T. Melodia, "InterfO-RAN: Real-Time In-band Cellular Uplink Interference Detection with GPU-Accelerated dApps," in *ACM MobiHoc*, 2025.

[16] A. Al-Shawabka, F. Restuccia, S. D'Oro, T. Jian, B. Costa Rendon, N. Soltani, J. Dy, S. Ioannidis, K. Chowdhury and T. Melodia, "Exposing the fingerprint: Dissecting the impact of the wireless channel on radio fingerprinting," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, 2020.

[17] M. Polese, F. Restuccia and T. Melodia, "DeepBeam: Deep waveform learning for coordination-free beam management in mmWave networks," in *Proceedings of the Twenty-second International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computin*, 2021.

[18] D. Villa, D. Uvaydov, L. Bonati, P. Johari, J. M. Jornet and T. Melodia, "Twinning Commercial Radio Waveforms in the Colosseum Wireless Network Emulator," in *Proceedings of the 17th ACM Workshop on Wireless Network Testbeds, Experimental evaluation \& Characterization*, 2023.

[19] D. Uvaydov, S. D'Oro, F. Restuccia and T. Melodia, "Deepsense: Fast wideband spectrum sensing through real-time in-the-loop deep learning," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, 2021.