FOCOM-Enabled Automation and Closed-Loop Control for Multi-Cloud O-Cloud Deployments

Tse-Han Wang*†, Yi-Fan Li*, Li-Hsing Yen*, and Chien-Chao Tseng*
*Department of Computer Science, College of Computer Science, National Yang Ming Chiao Tung University, Hsinchu, Taiwan.
Email: {wangth, liyf0404, lhyen, cctseng}@cs.nycu.edu.tw
†Network Management Laboratory, Chunghwa Telecom Laboratories, Taoyuan, Taiwan.

Abstract—**The O-RAN Alliance introduces the O2 interface to support interoperable infrastructure management across heterogeneous O-Cloud environments. However, achieving unified and scalable lifecycle management remains challenging due to the diversity of cloud platforms. This paper proposes a generic, O-RAN-compliant Infrastructure Management Services (IMS) framework that integrates an abstraction-layer-based design with automation tools to support seamless multi-cloud interoperability. A FOCOM-enabled mechanism is developed to automate the initialization of both O-Cloud and IMS instances. Experimental evaluations demonstrate the effectiveness of the proposed framework in three scenarios: (1) closed-loop control for energy-efficient node management, (2) automated IMS and O-Cloud deployment, and (3) large-scale IMS performance under emulated O-Cloud topologies. Results show successful node scaling in response to workload thresholds, efficient IMS initialization within 306 seconds, and scalable IMS operation across 2500 simulated nodes. These findings validate the framework's applicability to real-world O-RAN deployments and its potential for AI/ML-driven, cross-domain resource optimization.**

Index Terms—**Open RAN, O-Cloud, Infrastructure Management, Closed-loop Control, Automation, Interoperability**

I. Introduction

Traditional telecommunications networks are often hindered by vendor lock-in, limited flexibility, and high operational expenses. To overcome these limitations, the O-RAN Alliance has spearheaded the development of Open Radio Access Networks (Open RAN), which advocate for interoperability through standardized open interfaces and leverage virtualization technologies to enhance network scalability and adaptability. A cornerstone of this architectural evolution is the disaggregation of base station functionalities into three modular network elements: Radio Unit (RU), Distributed Unit (DU), and Centralized Unit (CU). These elements are interconnected through well-defined, open interfaces, enabling multi-vendor deployments and fostering innovation.

The O-RAN architecture is fundamentally based on the NG-RAN architecture [1] defined by 3GPP, inheriting its functional split between CU and DU. However, O-RAN extends this baseline by introducing additional open interfaces (e.g., A1, E2, O1, O2) and service management entities, such as the Near-Real-Time RAN Intelligent Controller (Near-RT RIC) and Non-Real-Time RIC [2]. This design preserves backward compatibility with existing 3GPP systems while promoting openness, programmability, and end-to-end automation in RAN deployments.

By incorporating Network Function Virtualization (NFV), O-RAN enables RAN elements to be instantiated as Virtual Network Functions (VNFs) or Containerized Network Functions (CNFs), which can operate across heterogeneous cloud environments. When these environments are compliant with O-RAN specifications, they are referred to as O-Clouds. The Service Management and Orchestration (SMO) framework is responsible for orchestrating and managing the lifecycle of these network functions. To ensure seamless interaction between SMO and O-Cloud platforms, the O-RAN Alliance has defined the O2 interface, which comprises two subinterfaces: O2 Infrastructure Management Services (O2ims) for infrastructure-level monitoring and resource management, and O2 Deployment Management Services (O2dms) to orchestrate the deployment of O-RAN network functions [3].

Through the O2 interface, SMO can gain visibility into heterogeneous, multi-vendor cloud infrastructures and implement closed-loop automation for efficient resource utilization [4], ultimately reducing operational and capital expenditures (OPEX/CAPEX). However, managing such diverse environments, including public, private, and on-premise clouds, introduces substantial complexity. Thus, a generalized and cloud-agnostic Infrastructure Management Service (IMS) architecture is necessary to abstract and unify the underlying cloud platform heterogeneity. Furthermore, as the deployment of O-RAN scales, the need for reliable and automated IMS initialization and provisioning mechanisms becomes increasingly critical.

Another pivotal requirement involves enabling Federated O-Cloud Orchestration and Management (FOCOM), which allows the SMO to access real-time resource states and performance data from distributed O-Cloud domains. Seamless interoperation between FOCOM and other SMO components is vital to achieving comprehensive system awareness and coordination.

To address these challenges while adhering to O-RAN specifications, this paper makes the following key contributions:

- **Generic IMS Architecture:** A generalized IMS design based on an abstraction-layer approach, facilitating seamless integration across heterogeneous O-Cloud platforms;

- **FOCOM-enabled Automation Mechanism:** An automated O-Cloud provisioning and IMS initialization framework that leverages FOCOM to ensure consistent deployment across diverse environments;
- **FOCOM-SMO Interworking Mechanism:** A validated integration of FOCOM into the SMO framework, supporting closed-loop infrastructure resource control via the O2 interface.

The remainder of this paper is organized as follows: Sec. III details the proposed architecture and methodology. Sec. IV presents the experimental evaluation, and the last section concludes this paper.

II. Related Work

Numerous studies have investigated infrastructure management across heterogeneous and hybrid cloud environments [5], with several initiatives extending these concepts to the evolving O-RAN ecosystem.

Driver-based infrastructure management frameworks attempt to unify the control of diverse cloud platforms, spanning public clouds, private clouds, and on-premise data centers via a centralized management plane. These solutions [6]–[8] typically employ a driver-based architecture in which platform-specific adaptors (drivers) are embedded within the central manager to translate cloud-native APIs. While effective in achieving interoperability at a basic level, such architectures encounter scalability and maintainability issues. From the perspective of O-RAN, this approach contradicts its core design philosophy, which emphasizes interoperability through standardized, open interfaces such as O2. In particular, tightly coupling the SMO to vendor-specific logic undermines modularity and conflicts with O-RAN's intent of pushing compliance responsibility to the O-Cloud side, thereby promoting vendor neutrality and architectural decoupling.

The *O2 project* within the O-RAN Software Community (OSC) provides a reference implementation of the O2 interface using StarlingX [9], an open-source cloud infrastructure stack that supports compute, networking, and virtualization services. As part of this effort, the pti-o2 module [10] implements a prototype of the O2ims, deployed on Kubernetes clusters managed by StarlingX. This implementation partially supports the O2 specifications, including inventory and fault management, and communicates with StarlingX through its native APIs. However, the current prototype is limited in scope, supporting only a subset of O2 features and remaining tightly coupled to StarlingX, significantly reducing its portability and applicability across other O-Cloud implementations.

Nephio, an open-source project led by LF Networking, adopts a cloud-native approach to telecom infrastructure management using Kubernetes-native mechanisms such as Operators [11]. Nephio models infrastructure, CNFs, and configurations as Kubernetes Custom Resources (CRs) [12]. Developed in collaboration with OSC and OpenAirInterface [13], Nephio aspires to support integration with O-RAN components, including O-Cloud, O2 interfaces, and network function orchestration roles such as FOCOM and Network Function Orchestrator (NFO). While the architectural vision is aligned with O-RAN principles, Nephio remains in early development stages and has yet to deliver concrete implementations for key scenarios addressed in this paper, such as automated O-Cloud provisioning and energy-aware infrastructure optimization.

In summary, existing approaches exhibit several critical limitations: driver-based frameworks violate O-RAN's open interface principles; OSC's O2 reference design is functionally restricted and lacks cross-platform flexibility; Nephio shows potential, yet remains undeveloped, and its effectiveness has not been verified in practical O-RAN scenarios. These gaps highlight the need for a more comprehensive solution. This paper proposes a generic, extensible, and automation-ready O-Cloud management framework that conforms to O-RAN architecture and supports operationally relevant features, including cloud infrastructure initialization and closed-loop control for energy efficiency.

III. O-Cloud Genesis Automation and Management Framework

A. System Overview

This paper proposes an automated O-Cloud deployment and management framework to address the growing complexity and heterogeneity of public clouds, private clouds, and on-premise environments in O-RAN systems. Aligned with the O-RAN architecture, the framework realizes management and orchestration functionalities of the O-Cloud as defined under the SMO framework [14].

At its core, the framework integrates a generic IMS, an automated initialization mechanism, and enhanced capabilities within the FOCOM module. The IMS component is designed in compliance with the O-RAN O2ims Interface Specification v6.0 [15], ensuring standardized interoperability across multi-vendor cloud environments. By enabling O2 interface-based automation, the framework supports scalable, flexible, and closed-loop orchestration of distributed O-Cloud resources, enhancing operational efficiency and interoperability in cloud-native RAN deployments.

B. Generic Infrastructure Management Service (IMS)

To facilitate unified infrastructure management, a generic IMS architecture adheres to the O2 interface defined by the O-RAN Alliance. A key design principle is incorporating an abstraction layer between the IMS core and underlying infrastructure management platforms, allowing seamless adaptation to heterogeneous environments and minimizing integration complexity.

As shown in Fig. 1, the IMS architecture comprises three main components. The **IMS Core Services** implement O2ims functionality, exposing O2 APIs (HTTP/HTTPS) to SMO components, particularly FOCOM. Internally, the core provides gRPC-based adaptor APIs to interact with infrastructure-specific adaptor modules. It also handles service logic and persistent data storage.

Three primary O2ims services are implemented:

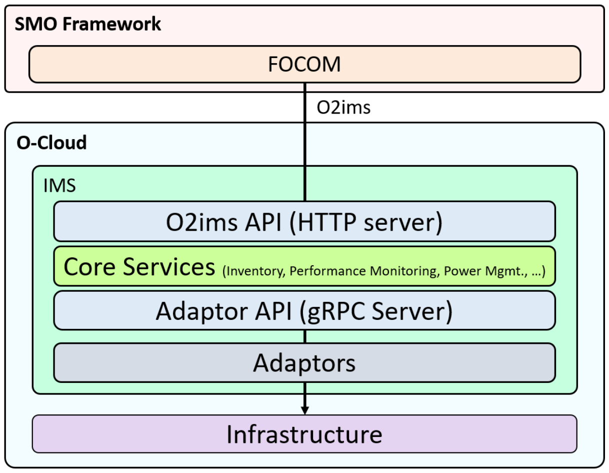
SMO Framework
FOCOM
O2ims
O-Cloud
IMS
O2ims API (HTTP server)
Core Services (Inventory, Performance Monitoring, Power Mgmt., ...)
Adaptor API (gRPC Server)
Adaptors
Infrastructure

Fig. 1. O-Cloud Genesis Automation and Management Framework

- **Inventory Service**: Collects and normalizes infrastructure resource inventory data via adaptors and stores it in a MongoDB backend. It supports event-based O2ims inventory change notifications.
- **Performance Monitoring Service**: Collects metrics such as CPU and memory usage using Prometheus as the collector and metric database.
- **Power Management Service**: An extension to the O2ims specification, this module enables remote power control through registered adaptors. It supports energy-efficient closed-loop scenarios.

The **IMS Adaptors** serve as connectors between the core and infrastructure platforms. Each adaptor translates gRPC calls into native API interactions for infrastructure management orchestrators. Adaptors collect telemetry and measurement data, execute control commands, and deliver structured responses to the core.

The **Infrastructure Layer** refers to the managed O-Cloud resources, encompassing both physical and virtual elements such as servers, VLANs, and IP address ranges.

C. FOCOM-enabled O-Cloud and IMS Initialization Automation

To streamline the deployment of O-Cloud environments and IMS components, this work introduces an automated initialization mechanism orchestrated by FOCOM. In view of the expected scale of O-RAN deployments, automation is essential to minimizing operational burden and deployment time.

The framework utilizes Ansible along with AWX, a Kubernetes-native automation frontend, to perform a three-stage initialization process: (as depicted in Fig. 2)

1) **Node Preparation**: virtual machines (VMs) are provisioned via cloud provider APIs, while bare-metal servers are initialized using IPMI, PXE, and cloud-init [16].
2) **Kubernetes Cluster Creation**: Kubernetes clusters are deployed using suitable distributions (e.g., Vanilla Kubernetes, K3s, Rancher Kubernetes Engine) based on infrastructure constraints.
3) **IMS Deployment**: The IMS software stack is installed using Helm, a Kubernetes package manager.

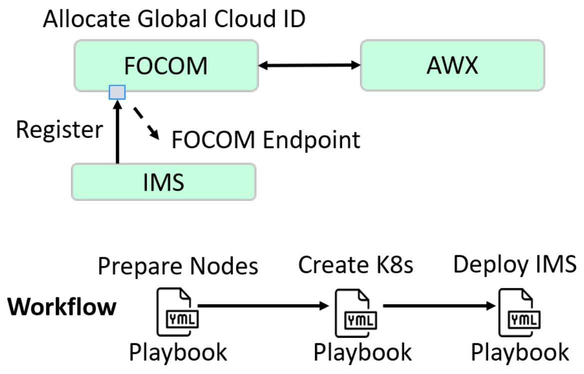

Fig. 2. O-Cloud and IMS Initialization with Ansible

FOCOM manages this pipeline by generating enriched Ansible inventories containing O-Cloud metadata (including Global Cloud ID as O-Cloud identifier and IMS endpoint IP address), configuring AWX credentials and workflows, and triggering deployment tasks. Upon successful initialization, the IMS instance self-registers with FOCOM, exposing its managed resources to SMO through the O2 interface. This modular, automation-centric approach enables reusable and scalable provisioning across diverse infrastructure platforms.

D. Federated O-Cloud Management and Closed-loop Control

The framework supports federated management of multiple O-Cloud instances through FOCOM, which acts as both the termination point and proxy for O2ims interactions. FOCOM maintains metadata (Global Cloud IDs, IMS endpoint IP addresses) for each registered O-Cloud and exposes internal HTTP APIs for SMO component integration.

When SMO components issue O2ims commands or subscriptions, FOCOM resolves the corresponding IMS endpoint using its registry and forwards the request. It also aggregates event notifications from all IMS instances and redistributes them to SMO consumers via an internal event bus, as illustrated in Fig. 3.

To validate closed-loop capabilities, this work implements the *O-Cloud Node Shutdown* use case defined by O-RAN WG1 [17]. The Energy Saving App queries time-series metrics (e.g., memory usage) from InfluxDB. When underutilized nodes are detected, a shutdown policy is triggered. The app then sends a shutdown command through FOCOM, which relays the request to the relevant IMS. The IMS then invokes its Power Management Service to power off the selected nodes. This use case demonstrates the complete awareness–analysis–decision-execution cycle, achieving energy-efficient O-Cloud operations through standardized O2 interfaces.

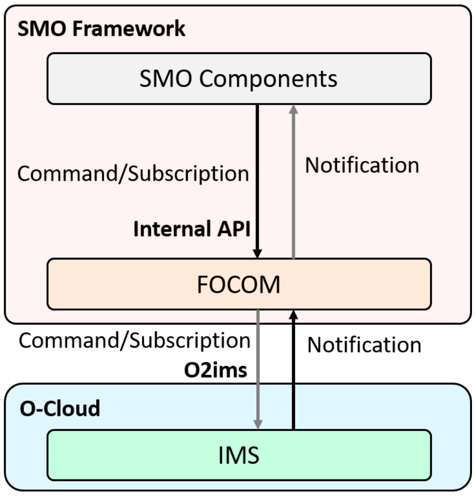

Fig. 3. Design of Federated O-Cloud Management and Closed-loop Control

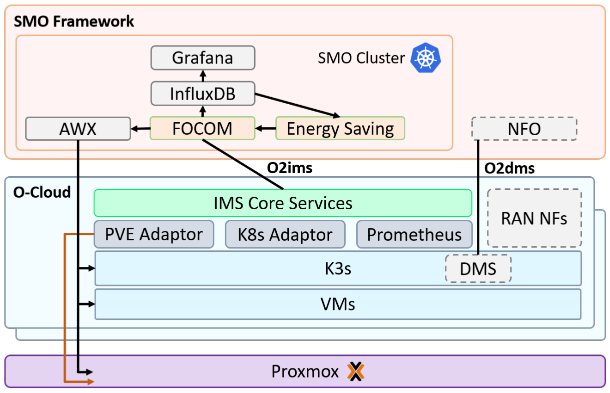

Fig. 4. Overview of Implementation Architecture

IV. Evaluation

This section presents the experimental validation of the proposed framework. The evaluation is organized into three subsections: (1) analysis of the energy-saving closed-loop control mechanism, (2) performance assessment of SMO during O-Cloud and IMS initialization, and (3) scalability analysis of the IMS under large-scale O-Cloud scenarios.

A. Environment Setup

All experiments were conducted on a single physical server using VMs to emulate SMO and O-Cloud environments, implementation architecture as shown in Fig. 4. The hardware and software configurations are detailed below:

Server Specifications

- CPU: 24 × 12th Gen Intel® Core™ i9-12900
- RAM: 128 GB
- Host OS: Proxmox VE 7 (Kernel: 5.13.19-2-pve)

SMO Configuration

- Kubernetes: Single-node cluster (Vanilla Kubernetes v1.27.5)
- OS: Ubuntu 22.04
- CPU: 8 Cores
- RAM: 32 GiB

O-Cloud Configuration

- Kubernetes: K3s-based cluster (v1.27.5+k3s1) with variable node configurations
- OS: Ubuntu 22.04
- CPU per node: 4 Cores
- RAM:
  - Single-node cluster: 8 GiB
  - Multi-node cluster:
    - Control plane node: 12 GiB
    - Worker nodes: 4 GiB

System metrics, including CPU and memory utilization, were collected from both SMO and O-Cloud nodes. For the energy-saving closed-loop control experiment, O-Cloud performance data were retrieved via the O2ims Performance Monitoring Service and FOCOM, with InfluxDB as the data store. For the SMO and IMS initialization evaluations, metrics were obtained through custom scripts that directly accessed the Kubernetes Metrics Server.

B. Results and Discussion

1) Energy Saving Closed-loop Control: To validate the proposed closed-loop control functionality, a memory usage-based node shutdown policy was tested in a simulated O-Cloud environment composed of four worker nodes. The control logic followed the *O-Cloud Node Shutdown* use case [17]. Specifically, a node was powered on if the average memory usage across all active nodes exceeded 70%, and powered off when it fell below 30%.

To emulate dynamic workload conditions, each fake network function (NF) consumed 400 MiB of memory. NFs were incrementally added (up to 30) and then removed, at 30-second intervals. The Energy Saving App evaluated control decisions every 20 seconds, with a 60-second cooldown to avoid oscillations.

The results confirmed that nodes 2, 3, and 4 were sequentially activated as usage increased and deactivated as the load decreased. Fig. 5 presents a time-series plot of memory usage and node status transitions, validating the effectiveness of the closed-loop mechanism.

2) FOCOM-Enabled O-Cloud and IMS Initialization: The automation framework for O-Cloud and IMS initialization was evaluated using a prototype deployment. The SMO, including FOCOM and AWX, was deployed on a single-node Kubernetes cluster. The target O-Cloud was instantiated as a K3s single-node cluster on a VM hosted by Proxmox.

The initialization process was divided into two phases: (1) SMO preparation and (2) O-Cloud setup and IMS deployment. The total execution time was measured at 306 seconds. Deployment of AWX and upload of Ansible Playbooks were excluded from this measurement. During this time, the maximum memory usage for the SMO node reached 3312 MiB. AWX is a more complete and complex Ansible runtime system, so it consumes more memory. Table I summarizes the timing of

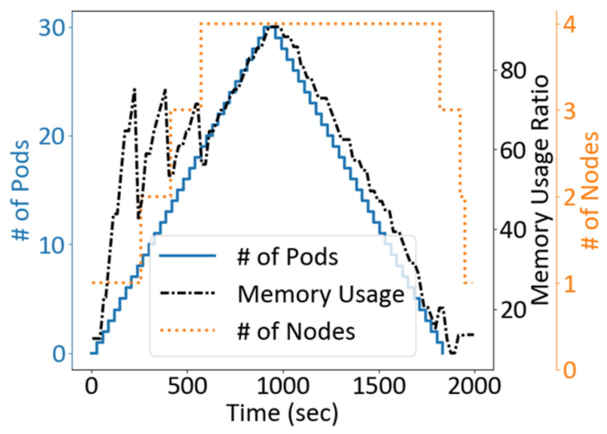

Fig. 5. Experiment Result of Energy Saving Closed-loop Control

TABLE I
EXECUTION TIME OF O-CLOUD AND IMS INITIALIZATION

Target	Details	Time (Sec.)
SMO Initialization Phase		
Deploy AWX	Deploying AWX on the SMO node.	168
Upload Ansible Playbook	Downloading required Ansible playbooks for O-Cloud deployment to AWX.	44
Deploy FOCOM	Deploying FOCOM to the SMO node.	46
O-Cloud Initialization Phase		
FOCOM Handling	FOCOM receives deployment request, uploads inventory to AWX, and completes pre-deployment tasks.	38
Workflow Execution	Time taken by AWX to execute the deployment workflow.	136
IMS Registration	AWX triggers IMS installation on K3s; time from start to IMS initialization and registration with FOCOM.	86

each phase, while Fig. 6 illustrates CPU and memory usage throughout the procedure.

These results demonstrate the efficiency of the proposed automation framework, suggesting its suitability for scalable, production-level O-RAN environments.

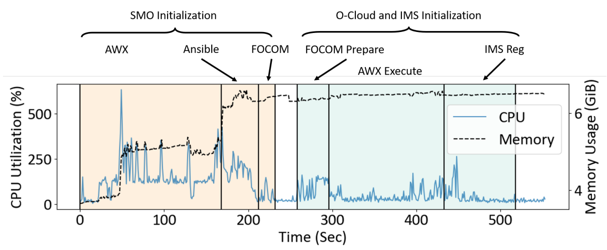

Fig. 6. O-Cloud and IMS Initialization SMO Node CPU/Memory Usage

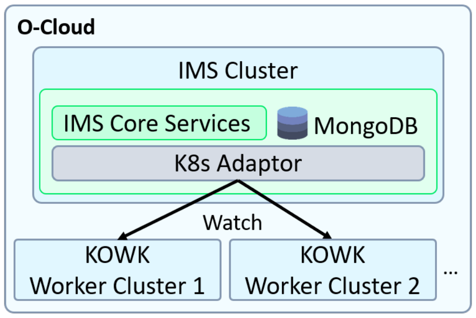

Fig. 7. IMS with Large Scale O-Cloud Experiment Architecture

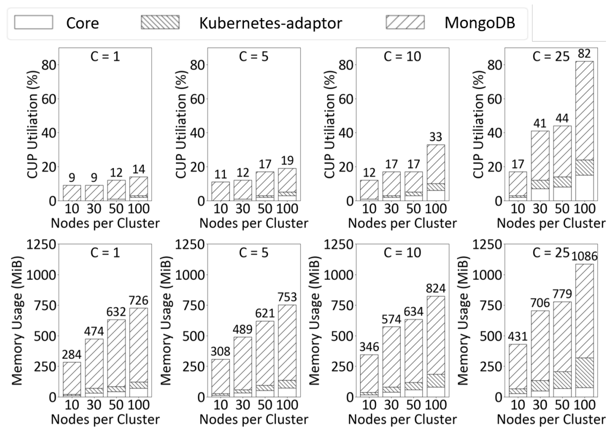

Fig. 8. Experiment Result of IMS with Large Scale O-Cloud

3) IMS Performance with Large Scale O-Cloud: To assess the scalability of IMS and identify potential performance bottlenecks, an emulated O-Cloud testbed was created using the KWOK [18] tool, which simulates Kubernetes clusters with configurable size. The IMS, including IMS Core Services, the Kubernetes adaptor, and MongoDB, was deployed on a single-node Kubernetes cluster. The experiment architecture is illustrated in Fig. 7.

The experiment varied both the number of clusters (1, 5, 10, 25) and the number of nodes in each cluster (10, 30, 50, 100). The maximum scale reached was 2,500 worker nodes. In the case of Kubernetes, this is a large-scale system [19]. The CPU and memory usage of IMS components (IMS Core Services, the Kubernetes adaptor, and MongoDB) were recorded.

The results, shown in Fig. 8, indicate that MongoDB was the dominant consumer of system resources, particularly due to write-heavy inventory synchronization. Even at the largest scale, overall system usage remained within acceptable bounds, peaking at 82% CPU and 1.09 GiB RAM, demonstrating the scalability of the proposed IMS design. These findings highlight areas for future optimization, particularly in data storage and synchronization mechanisms.

V. CONCLUSIONS

This paper presented a generic infrastructure management framework for O-RAN-compliant O-Clouds to ensure interoperability across heterogeneous deployment environments, including public, private, and on-premises infrastructures. The proposed framework is designed in full compliance with O-RAN specifications and integrates mature IT automation technologies to enable scalable, automated, and platform-agnostic infrastructure management. It supports IMS and end-to-end automation to initialize O-Cloud and IMS components.

The main contributions of this work are summarized as follows:

- **Generic IMS Architecture:** An abstraction-layer-based IMS design was proposed, enabling seamless integration with diverse infrastructure management systems. This design significantly reduces development overhead and ensures compliance with O-RAN O2 interface standards.
- **FOCOM-Enabled O-Cloud and IMS Initialization Automation:** An automated provisioning mechanism leveraging AWX and FOCOM was implemented, enabling rapid and reproducible deployment of O-Cloud environments and IMS instances across various infrastructures.
- **Federated O-Cloud Management and Closed-Loop Control:** A closed-loop control use case, specifically O-Cloud node shutdown, was implemented and validated in accordance with O-RAN specifications. This demonstrates the framework's real-time infrastructure monitoring, decision-making, and actuation capability.

Future efforts will focus on extending the current implementation to support additional O2 interface services, including the *Provision Service*, which facilitates automated deployment and lifecycle management of O-Cloud Deployment Management Service (DMS) clusters. The framework will be enhanced by incorporating the Non-RT RIC architecture, enabling more comprehensive analytics and policy-driven decision-making within the closed-loop control system.

To improve predictive resource scaling, the integration of Artificial Intelligence and Machine Learning (AI/ML) techniques is planned to enable proactive and adaptive cluster resizing based on workload forecasts. Furthermore, future development will explore inter-O-Cloud coordination mechanisms, involving collaboration among multiple clusters and Network Function Orchestration components. This will facilitate cross-domain service deployment and enable holistic resource optimization at a global scale.

References

[1] "5G; NG-RAN; architecture description," 3GPP, TS 38.401, V16.3.0, Release 16, Nov. 2020.

[2] T.-H. Wang, Y.-C. Chen, S.-J. Huang, K.-S. Hsu, and C.-H. Hu, "Design of a network management system for 5G Open RAN," in *2021 22nd Asia-Pacific Network Operations and Management Symposium (APNOMS)*, 2021, pp. 138–141.

[3] O-RAN WG6, "O-RAN O2 interface general aspects and principles 7.0," O-RAN Alliance, Tech. Rep., Jun. 2024.

[4] ——, "O-RAN cloud architecture and deployment scenarios for O-RAN virtualized RAN 7.0," O-RAN Alliance, Tech. Rep., Jun. 2024.

[5] P. Raj and A. Raman, "Multi-cloud management: Technologies, tools, and techniques," in *Software-Defined Cloud Centers: Operational and Management Technologies and Tools*, P. Raj and A. Raman, Eds. Cham: Springer International Publishing, 2018, pp. 219–240.

[6] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster, "Virtual infrastructure management in private and hybrid clouds," *IEEE Internet Computing*, vol. 13, no. 5, pp. 14–22, Sep. 2009.

[7] S. Yan, B. S. Lee, G. Zhao, D. Ma, and P. Mohamed, "Infrastructure management of hybrid cloud for enterprise users," in *2011 5th International DMTF Academic Alliance Workshop on Systems and Virtualization Management: Standards and the Cloud (SVM)*, 2011, pp. 1–6.

[8] M. Caballer, M. Antonacci, Z. Šustr, M. Perniola, and G. Moltó, "Deployment of elastic virtual hybrid clusters across cloud sites," *Journal of Grid Computing*, vol. 19, no. 1, p. 4, Feb. 2021.

[9] "Open source edge cloud computing architecture - starlingx," https://starlingx.io/, accessed: May 25, 2024.

[10] "pti/o2 repo," https://github.com/o-ran-sc/pti-o2, accessed: May 25, 2025.

[11] "Operator pattern," https://kubernetes.io/docs/concepts/extend-kubernetes/operator/, accessed: May 25, 2025.

[12] "Custom resources," https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/custom-resources/, accessed: May 25, 2025.

[13] "Openairinterface," https://openairinterface.org/, accessed: May 25, 2025.

[14] O-RAN WG1, "O-RAN decoupled smo architecture 3.0," O-RAN Alliance, Tech. Rep., Oct. 2024.

[15] O-RAN WG6, "O-ran o2ims interface specification 6.0," O-RAN Alliance, Tech. Rep., Jun. 2024.

[16] "cloud-init - the standard for customising cloud instances," https://cloud-init.io/, accessed: May 25, 2025.

[17] O-RAN WG1, "O-RAN network energy saving use cases technical report 2.0," O-RAN Alliance, Tech. Rep., Jun. 2023.

[18] "KWOK (Kubernetes WithOut Kubelet)," https://kwok.sigs.k8s.io/, accessed: May 25, 2025.

[19] "Considerations for large clusters — kubernetes," https://kubernetes.io/docs/setup/best-practices/cluster-large/, accessed: May 25, 2025.