Demo: GitOps-Based Automation for Distributed NFs Reconfiguration in Cloud-Native O-RAN

Vitumbiko Mafeni, Phuong Bac Ta, and Younghan Kim*
Department of Electronic Engineering, *Soongsil University*, Seoul, Korea
{vitumafeni, bactp}@dcn.ssu.ac.kr, younghak@ssu.ac.kr

***Abstract*—O-RAN disaggregation introduces flexibility in deploying Central Units (CUs) and Distributed Units (DUs) as independent network functions (NFs) connected via midhaul (F1 interface). In cloud-native deployments, multiple CU instances may coexist across clusters, each with varying compute load and network distance. In current practice, CU–DU associations are typically static and do not adapt to changing runtime conditions, making it difficult to apply consistent, timely configuration changes when a CU becomes overloaded. This demo presents a GitOps-based automation framework for runtime reconfiguration of distributed NFs in cloud-native O-RAN. The framework continuously collects NF telemetry and evaluates reconfiguration triggers, applying declarative updates via GitOps workflows to ensure consistency, traceability, and rapid propagation across clusters. As an illustrative scenario, we demonstrate CU–DU coordination under compute stress: when a CU instance becomes overloaded, the system automatically initiates reconfiguration or DU reassociation to an alternative CU. During the demo, attendees will observe live telemetry, configuration commits, and protocol traces showing CU-DU reassociation in response to induced load, illustrating a practical realization of automated NF reconfiguration in cloud-native O-RAN.**

***Index Terms*—O-RAN, GitOps Automation, CU-DU Coordination**

I. Introduction

The O-RAN architecture introduces disaggregation of gNB components, splitting them into the Central Unit (CU) and Distributed Unit (DU), connected via the midhaul (F1) interface. This flexibility enables cloud-native, multi-cluster deployments where multiple CU instances may coexist across geographically distributed environments [1]. While this modularization improves manageability and interoperability, it also introduces new challenges in managing CU–DU associations and the midhaul links that interconnect them, especially under dynamic traffic and resource constraints.

In existing deployments and prior research, the association between DUs and CUs is typically configured statically at design time. These associations are either hardcoded or determined via offline optimization processes that do not respond to runtime infrastructure states [2], [3]. Even advanced proposals that jointly optimize CU/DU placement and user association rely on precomputed or periodically triggered decisions, lacking mechanisms for rapid, synchronized updates in response to changing compute or network conditions. Recent standardization efforts, such as the IETF CATS [4] — Computing-Aware Traffic Steering initiative, have underscored the need for adaptive midhaul steering but remain largely conceptual without concrete implementations.

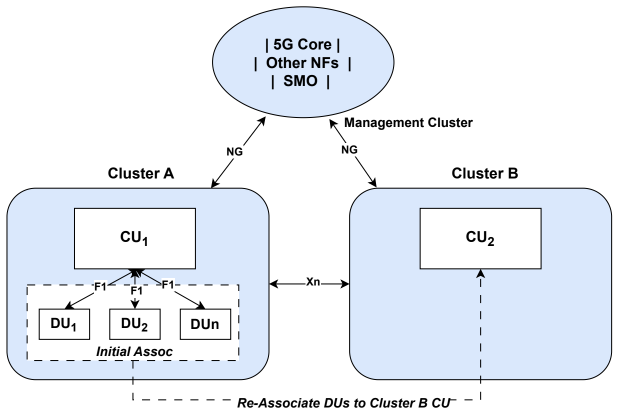

Fig. 1. System architecture overview

This demo presents a GitOps[1]-based automation framework for runtime reconfiguration of distributed network functions in cloud-native O-RAN. We demonstrate how a disaggregated CU/DU system deployed over multiple Kubernetes [2] clusters can dynamically react to compute pressure or bandwidth saturation by (i) reconfiguring CU deployment parameters, or (ii) reassociating DUs to alternative CUs based on live telemetry. Our approach leverages declarative configuration management, continuous metrics collection, and automated GitOps workflows to create a closed feedback loop between infrastructure state and NF configuration. By prototyping this end-to-end mechanism, we aim to bridge the gap between static CU–DU association schemes and practical, fully automated reconfiguration in cloud-native O-RAN environments.

II. System Design and Demonstration Setup

A. Architecture Overview

Our demo environment consists of three interconnected Kubernetes clusters as shown in Fig.1: two clusters host separate

This work was supported in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by Korean Government (MSIT) (Development of Fast and Automatic Service recovery and Transition software in Hybrid Cloud Environment and High Performance 6G Cross Cloud Infrastructure Technology) under Grant RS-2020-II200946, and Grant RS-2024-00398379

[1]https://about.gitlab.com/topics/gitops/
[2]https://kubernetes.io/

CU instances, and one management cluster capable of service management and orchestration (SMO) and midhaul association with CU-DU. All RAN and Core components are deployed using the OAI 5G software stack, containerized as network functions and orchestrated through Kubernetes with ArgoCD for declarative GitOps-based lifecycle management. The UE side is emulated using UERANSIM to generate traffic and validate end-to-end connectivity. Resource usage metrics such as CPU load and memory utilization are continuously exported from the CU/DU functions and collected by a Prometheus[3] server accessible to the orchestration layer.

To support runtime adaptability, the system relies on a CU-DU association function integrated into the non-RT RIC within the SMO layer. This component aggregates compute metrics from centralized units and network telemetry from the midhaul—such as RTT and packet loss. Based on this aggregated state, the controller evaluates the quality of ongoing DU–CU associations using a weighted scoring function. When performance degradation is detected, it takes one of two actions: for moderate degradation at the current CU, the controller triggers a capacity reconfiguration, adjusting resource allocations (e.g., CPU limits) to absorb additional load; for severe degradation, it triggers a re-association procedure, reconfiguring the DU to connect to an alternative CU offering lower expected cost.

B. Demonstration Workflow

The demonstration proceeds in two sequential phases, each illustrating a distinct adaptation strategy in response to CU-side compute pressure. At the outset, the audience is presented with a real-time dashboard displaying CU telemetry—CPU and memory usage curves, computed cost scores derived from weighted metric functions, and midhaul network statistics. Alongside these metrics, the GitOps commit history and Kubernetes event logs are shown to reveal the exact configuration changes being applied, while SCTP session states and F1-AP signaling traces make the adaptation visible at the protocol level.

1) Phase 1 - CU Reconfiguration: We first inject artificial CPU load on CU-A to simulate compute degradation. As the CPU utilization and memory pressure increase, the *GitAgent DU-CU Assoc Controller* calculates a rising cost score for CU-A based on a predefined weighted function. Upon breaching the adaptation threshold, the controller commits a declarative update to the CU-A Helm values in a GitOps repository. This update adjusts the CU's resource configuration—such as reducing the number of threads or restarting with an alternate profile. The GitAgent detects the change and applies it automatically. CU-A is gracefully restarted, and its compute score gradually recovers. During this process, the DU remains attached to CU-A, demonstrating live reconfiguration without reassociation.

2) Phase 2 - CU Selection and Reassociation: In the second phase, CU-A's load is further increased beyond its reconfigurable limits. As illustrated in Fig. 2, the controller identifies CU-B as a more suitable candidate based on its lower compute cost and better network metrics. The controller then commits a second GitOps update that modifies the DU's configuration to change its F1 association to CU-B. Once the update is applied, the DU tears down its SCTP session with CU-A and initiates a new F1 setup with CU-B. This reassociation is observable through changes in SCTP state and F1-AP message logs. From the system's perspective, RAN control-plane continuity is preserved, and signaling operations resume seamlessly through CU-B. After various experiments, the reassociation process showed reliable, consistent and successful completion within short time intervals, while the GitOps pipeline applied the rollout automatically without disruption through event based webhooks.

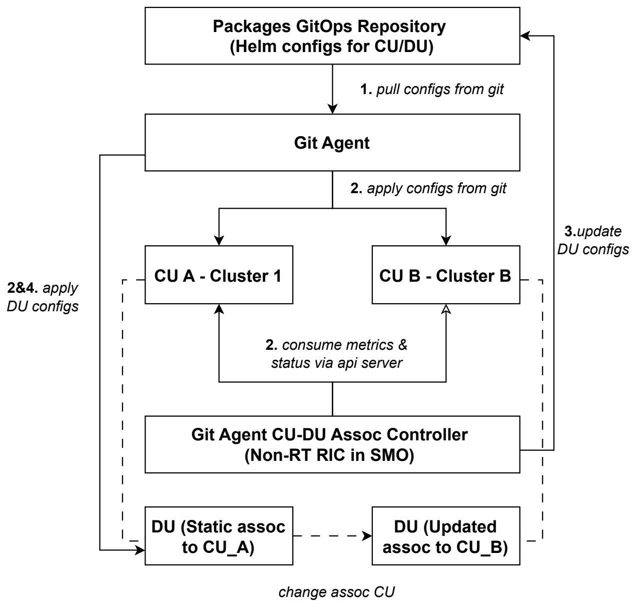
Packages GitOps Repository (Helm configs for CU/DU)
1. pull configs from git
Git Agent
2. apply configs from git
3.update DU configs
2&4. apply DU configs
CU A - Cluster 1
CU B - Cluster B
2. consume metrics & status via api server
Git Agent CU-DU Assoc Controller (Non-RT RIC in SMO)
DU (Static assoc to CU_A)
DU (Updated assoc to CU_B)
change assoc CU

Fig. 2. Live procedure demonstrating DU re-association from CU_A to CU_B triggered by compute pressure and automated via GitOps.

III. Conclusion

This demonstration showcases an automated DU–CU coordination framework designed for cloud-native O-RAN environments. By continuously monitoring real-time resource metrics at CU/DU instances, the system can dynamically trigger either a reconfiguration of the current CU or a DU reassociation to an alternative CU with lower infrastructure cost. Both adaptation paths are implemented using declarative GitOps workflows, enabling automated, transparent, and runtime-resilient midhaul operations. This approach illustrates the potential for infrastructure-aware coordination mechanisms in disaggregated RAN architectures, moving beyond static association strategies and toward more adaptive and intelligent mobile network control. Our observations show that these adaptations are achieved with low latency and consistent reliability, further reinforcing the practicality of the approach.

[3]https://prometheus.io/

REFERENCES

[1] M. Polese, L. Bonati, S. D'Oro, S. Basagni and T. Melodia, "Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges," in IEEE Communications Surveys & Tutorials, vol. 25, no. 2, pp. 1376-1411, Secondquarter 2023.

[2] Hojeij, Hiba, Guilherme Iecker Ricardo, Mahdi Sharara, Sahar Hoteit, Véronique Vèque, and Stefano Secci. "On flexible association and placement in disaggregated RAN designs." Computer Communications 238 (2025): 108166.

[3] R. Joda, T. Pamuklu, P. E. Iturria-Rivera and M. Erol-Kantarci, "Deep Reinforcement Learning-Based Joint User Association and CU–DU Placement in O-RAN," in IEEE Transactions on Network and Service Management, vol. 19, no. 4, pp. 4097-4110, Dec. 2022.

[4] IETF CATS Working Group, Compute-Aware Traffic Steering for Midhaul, Internet-Draft, July 2025. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-lcmw-cats-midhaul-03