IEEE INFOCOM 2023 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS) | 978-1-6654-9427-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/INFOCOMWKSHPS57453.2023.10225753

RIC-O: An Orchestrator for the Dynamic Placement of a Disaggregated RAN Intelligent Controller

Gustavo Z. Bruno*, Vikas Krishnan Radhakrishnan†, Gabriel M. Almeida‡, Alexandre Huff‖,
Aloízio P. da Silva†, Kleber V. Cardoso‡, Luiz A. DaSilva†, Cristiano Bonato Both*
*Sinos River Valley University (UNISINOS), Brazil, †Commonwealth Cyber Initiative, Virginia Tech (VT), USA,
‡Universidade Federal de Goiás (UFG), Brazil, ‖Federal Technological University of Paraná (UTFPR), Brazil
{zanattabruno, cbboth}@unisinos.br*, {vikaskrishnan, aloiziops, ldasilva}@vt.edu†,
{gabrielmatheus, kleber}@inf.ufg.br‡, alexandrehuff@utfpr.edu.br‖

***Abstract*—In this demonstration, we present the RIC Orchestrator (RIC-O), a system that optimizes the deployment of Near Real-time RAN Intelligent Controller (Near-RT RIC) components across cloud and edge computing nodes. RIC-O quickly and efficiently adapts to sudden changes and redeploys components as needed. We describe small-scale real-world experiments using RIC-O and a disaggregated Near-RT RIC within a Kubernetes deployment to demonstrate its effectiveness.**

I. Introduction

The Open RAN (O-RAN) Alliance [1] has established a vision of open, virtualized, fully interoperable, and intelligent mobile networks [2]. This vision can transform the Radio Access Network (RAN) industry by lowering barriers to entrance, decreasing vendor lock-in, and fostering innovation [3]. O-RAN controllers run Artificial Intelligence (AI) or Machine Learning (ML) based applications that establish control loops with the RAN nodes. In this context, the O-RAN architecture defines two RAN controllers: Non Real-time RAN Intelligent Controller (Non-RT RIC) and Near Real-time RAN Intelligent Controller (Near-RT RIC). The Non-RT RIC runs rApps with control loops with time intervals above 1s. The Near-RT RIC runs applications called xApps with control loops with time intervals between 10ms and 1s. The specific time constraint of a control loop depends on the RAN function being managed by the corresponding xApp. In a large RAN, the Near-RT RIC (or some of its components) and latency-sensitive xApps (i.e., control loop with latency as low as 10ms) must be replicated and assigned to manage a limited set of RAN nodes. Orchestrating multiple Near-RT RIC instances and determining where their components must run is a challenging resource allocation problem, considering the mobile network dynamics.

This work demonstrates the RIC Orchestrator (RIC-O) [4], which deploys components of a Near-RT RIC platform in a cloud-edge computing environment. The RIC-O is designed to minimize overall deployment costs while meeting stringent latency requirements. Moreover, RIC-O can dynamically redeploy components in response to sudden changes in the network infrastructure running on an extended Kubernetes (K8s).

MCTIC/CGI.br/FAPESP supported this work through the projects: SAMURAI (No. 2020/05127-2) and PORVIR-5G (No. 2020/05182-3). This work also received support from the Commonwealth Cyber Initiative. Visit CCI: www.cyberinitiative.org.

II. RIC-O Design Choices and Implementation

RIC-O is an orchestrator enabling efficient and dynamic placement of the components of disaggregated Near-RT RIC in a cloud-edge computing environment. RIC-O comprises two main building blocks: *Monitoring System* running on Near-RT RIC and RIC-O Components running on Non-RT RIC. Figure 1 shows a high-level view of the main building blocks composing our demonstration scenario. The RIC-O Components building block includes *RIC-O Optimizer* and *RIC-O Deployer* components. The K8s Control Plane manages the worker nodes and pods in the cluster. The *Monitoring System* analyzes the resource usage of the Near-RT RIC components, E2 Nodes, and Open Cloud (O-Cloud) infrastructure. Moreover, it monitors the control loop latency of E2 Nodes and alerts the *RIC-O Optimizer* component when control loop thresholds are violated. The control loop consists of a message exchange between the E2 Node (source) and a given xApp (destination) through the E2 Termination (E2T) component. The xApps control and optimize the RAN.

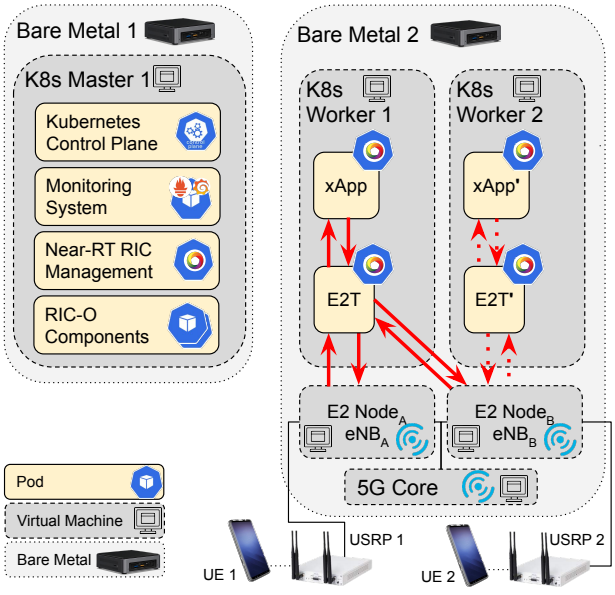

Fig. 1. Scenario of RIC-O demonstration with main architectural blocks.

Near-RT RIC components are split up according to their latency requirements. The components that execute tasks

978-1-6654-9427-4/23/$31.00 ©2023 IEEE

without stringent latency requirements are grouped into the Near-RT RIC Management building block. However, certain xApps and E2T components need to be placed in the cloud-edge infrastructure to meet their stringent latency requirements. The xApps provide services for each Software Defined Radio (SDR)-based E2 Node connected to their corresponding E2T. Each User Equipment (UE) connects to a Universal Software Radio Peripheral (USRP) X310 SDR networked to a given E2 Node. Given the computing nodes on which the Near-RT RIC components are deployed, it is possible to calculate the signaling latency between an E2 Node and an xApp (and potentially other Near-RT RIC components). This control loop round-trip time of a latency-sensitive xApp is monitored on the E2 Node.

When a control loop threshold violation is detected, the *Monitoring System* alerts the *RIC-O Optimizer* to compute a new placement strategy for the Near-RT RIC components. The goal is to minimize the overall cost of running the Near-RT RIC. The optimization strategy adopted by the *RIC-O Optimizer* is described in [4]. After computing an efficient solution, the *RIC-O Optimizer* sends the outcome to the *RIC-O Deployer*, which redeploys the Near-RT RIC components in case the solution differs from the previous one. The *RIC-O Deployer* uses the O2 interface defined by O-RAN to communicate with the cloud-native infrastructure and apply the new placement solution.

Two variants of E2 Nodes are considered for the demonstration of RIC-O. The first includes an enhanced version of the open-source E2 Node simulator. The second involves an O-RAN-compatible end-to-end software-based RAN implementation using the open-source Software Radio Systems RAN (srsRAN) suite [5]. A Long-term Evolution (LTE) mobile network is set up using srsRAN's Evolved Packet Core (EPC) and srsRAN's Evolved Node B (eNB). This E2 Node communicates with E2T using an E2 agent over the SCTP-based E2 interface. Moreover, Commercial Off-The-Shelf (COTS) UEs connect to the LTE network, and trigger RAN actions that impact the control plane signaling latency between the E2 Node and E2T.

III. Demonstration Flow

We demonstrate RIC-O in two scenarios using two bare metal computers and six Virtual Machines (VMs). Three VMs host a K8s cluster running Near-RT RIC, and three VMs run the mobile network. Moreover, two COTS mobile phones work as UEs. The first scenario involves a sudden increase in latency in the path serving a specific E2 Node. Initially, we have two E2 Nodes (A and B) attached to the same E2T, which connects in xApp with an enhanced version of the Bouncer service in K8s Worker 1. In this context, the control loop is represented by the solid red arrows in Fig. 1, and the 10ms latency constraint is met. Then, we insert latency in the link between E2 Node_B and K8s Worker 1 to demonstrate the orchestration of RIC-O. This latency can be observed in the "Control loop failure" event in Fig. 2. After 10s, the *Monitoring System* determines that this event is a consistent control loop violation and notifies RIC-O to compute a new solution, as indicated by "Optimization trigger". RIC-O finds a solution and applies the new placement nearly 5s later, as noted in "Start to redeploy".

RIC-O installs new E2T′ and xApp′ instances in K8s Worker 2. Moreover, a new connection of the control loop is established between E2 Node_B and these instances, represented by the red dotted lines in Fig. 1. In this scenario, the Bouncer xApp service continues to be provided by the E2T to the E2 Node_A and by xApp′ through E2T′ to the E2 Node_B. Therefore, the control loop operates within the 10ms threshold again, as indicated by "Control loop satisfied" in Fig. 2.

In the second scenario, K8s Worker 2 running the E2T′ and xApp′ serving E2 Node_B becomes suddenly unavailable. As a result, the latency-sensitive control loop is disrupted. As soon as the *Monitoring System* detects this event and notifies RIC-O, it orchestrates the new placement of Near-RT RIC components, directing E2 Node_B back to E2T.

Fig. 2. Real-time monitoring visualization.

IV. Concluding Remarks

This demonstration shows how RIC Orchestrator (RIC-O) efficiently orchestrates the Near-RT RIC components so they can meet stringent latency requirements and provide failover capacity. RIC-O efficiently orchestrates the Near-RT RIC components across the edge-cloud computing nodes in these scenarios.

References

[1] O-RAN Alliance, "O-RAN Alliance," 2022. [Online]. Available: https://www.o-ran.org/

[2] A. Garcia-Saavedra and X. Costa-Pérez, "O-RAN: Disrupting the Virtualized RAN Ecosystem," *IEEE Communications Standards Magazine*, vol. 5, no. 4, pp. 96–103, 2021.

[3] V. S. Pana, O. P. Babalola, and V. Balyan, "5G radio access networks: A survey," *Array*, vol. 14, p. 100170, 2022.

[4] G. M. Almeida *et al.*, "RIC-O: Efficient placement of a disaggregated and distributed RAN Intelligent Controller with dynamic clustering of radio nodes," 2023. [Online]. Available: http://arxiv.org/abs/2301.02760

[5] srsRAN, "srsRAN - Your own mobile network," 2022, Last Accessed: 12/11/2022. [Online]. Available: https://www.srslte.com/