Near-RT RIC and Emulated Basestation for Initial xApps Development

Lucas Costa, Rebecca Aben-Athar, Cleverson Nahum, Glauco Gonçalves, Ilan Correa and Aldebaro Klautau

Abstract— **The development of xApps for telecommunications systems is a new and open field. With the growing demand for new functionalities and the need for integration with the near Real-Time Radio Intelligent Controller (Near-RT RIC) systems and emulated base stations, it is essential to understand the challenges and opportunities involved in this process. In this context, we describe a virtualized and automated 5G environment using containers, which aims to support the whole development cycle of xApps.**

Keywords— **O-RAN, FlexRIC, Virtualization, OpenAirInterface.**

I. Introduction

The Open Radio Access Network (O-RAN) is an architecture that, aims to optimize the future generation wireless networks, increasing its efficiency. It also promotes functions disaggregation, for example, by dividing the gNB functions into Central Unit (CU), Distributed Unit (DU), and Radio Unit (RU). Additionally, it allows the integration of closed-loop intelligent systems through two types of RAN Intelligent Controllers (RICs): Near-real-time RIC and non-real-time RIC [1].

O-RAN is still under development, and there several closed and open source implementations [1]. The FlexRIC, which implements the near-real-time RIC. It consists of multiple applications that support custom logic, known as xApps, and the necessary services to execute these xApps. The advantage of using FlexRIC lies in its flexibility, lightweight design, and the pursuit of future compatibility [2].

The article [3] describes the implementation of a mobile network using OAI. However, the network elements were not automated using Docker containers. To create the BaseStation for FlexRIC, OAI will be utilized. This open-source program offers a complete software solution for all aspects of the 4G LTE and 5G system architecture, such a gNodeB. [4]

We implemented the containerization of the near-time RIC using FlexRIC software and the base station modules (CU, DU, and RU), making the execution easier. The O-RAN elements will be virtualized in containers that can be implemented using Docker. We automated the deployment of network components using Dockerfiles and Docker-compose, which are publicly available on GitHub[1]. Thus, the content of this work is a way to facilitate the creation of the environment and the integration of OpenAirInterface (OAI) with FlexRIC.

Additionally, it's worth noting that although the system can be implemented in various ways, the use of Docker significantly simplifies the replication of the environment across different machines. Docker's containerization approach allows for easy packaging and distribution, ensuring consistent behavior regardless of the host system. This characteristic simplifies the setup process and enables deployment on multiple machines, promoting greater scalability and reproducibility of the environment.

II. FlexRIC and virtualization

The near-time RIC, such as the FlexRIC software, plays an essential role in O-RAN architecture, it does the real-time orchestration and control of radio resources, ensuring efficient and dynamic allocation of radio resources to different devices and services [2].

FlexRIC is a Software Development Kit (SDK) that provides a standardized interface for programming and controlling network resources in SD-RANs (Software-Defined Radio Access Networks), designed to overcome challenges such as the complexity of network resources and technology standards.

The SDK offers tools and libraries for creating and deploying custom logic applications for the RIC called xApps. These xApps can be customized to meet different needs and functions within the SD-RAN architecture. With the FlexRIC SDK, users can define and implement their xApps based on their specific use cases. The SDK provides a framework for executing these xApps and offers the necessary services and interfaces for seamless integration with the RAN infrastructure. Additionally, it offers an open and programmable interface, facilitating potential software-defined network solutions.

Fig. 1 shows an overview of the FlexRIC architecture (Near-RT RIC) and how it connects to other standardized gNB functions (CU, DU, and RU). The FlexRIC can be segmented into two parts. On the top is the component called Specialization, which implements controller specialization through iApps (an application internal to the controller) that implements specific behavior or services that can be used by xApps (External Application). The bottom part is the FlexRIC Server Library that is an event-driven system that manages connections and messages between iApps and gNB functions.

Following current standards, telecom setups are deployed as virtualized functions [5], which can be implemented through

Lucas Costa, Rebecca Aben-Athar, Cleverson Nahum, Glauco Gonçalves, and Aldebaro Klautau are with LASSE - Telecommunications, Automation and Electronics Research and Development Center, Belém-PA, Brazil (e-mails: lucas.queiroz.costa, rebecca.athar@itec.ufpa.br, cleversonahum, glauco-goncalves, aldebaro@ufpa.br). This work was supported by the Open-RAN Brazil - Phase 2 project (MCTI grant N° A01245.014203/2021-14), SAMURAI project (FAPESP grant #20/05127-2), and CNPq-Brasil (grant 405111/2021-5).

[1]https://github.com/lasseufpa/flexric_oai.git

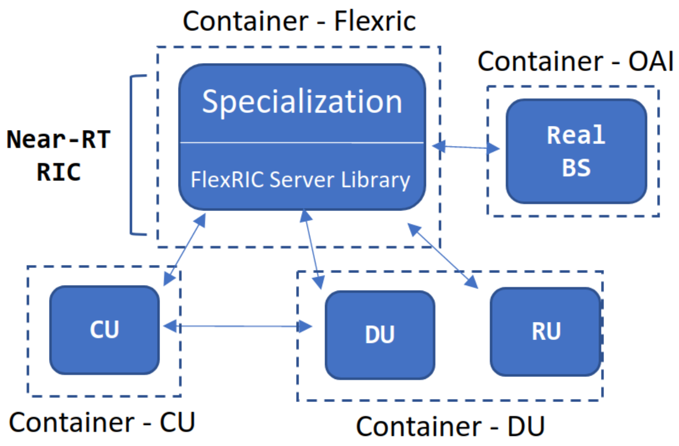
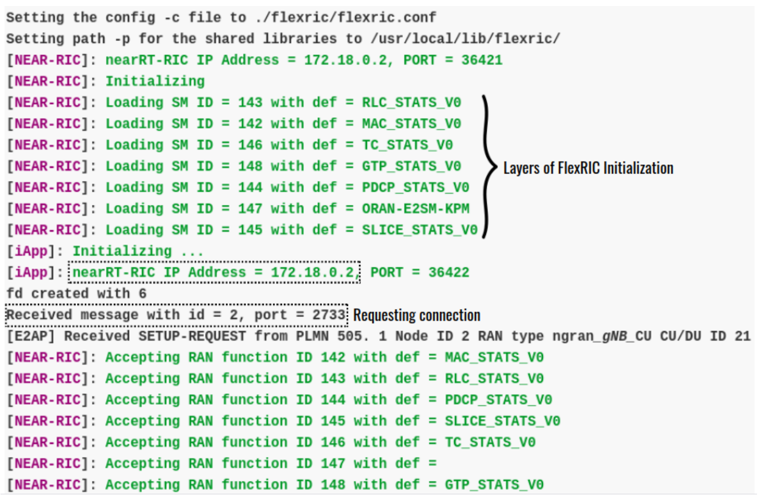


Fig. 1: Flexric architecture, gNB functions and BaseStation OAI.



```
Setting the config -c file to ./flexric/flexric.conf
Setting path -p for the shared libraries to /usr/local/lib/flexric/
[NEAR-RIC]: nearRT-RIC IP Address = 172.18.0.2, PORT = 36421
[NEAR-RIC]: Initializing
[NEAR-RIC]: Loading SM ID = 143 with def = RLC_STATS_V0
[NEAR-RIC]: Loading SM ID = 142 with def = MAC_STATS_V0
[NEAR-RIC]: Loading SM ID = 146 with def = TC_STATS_V0
[NEAR-RIC]: Loading SM ID = 148 with def = GTP_STATS_V0      Layers of FlexRIC Initialization
[NEAR-RIC]: Loading SM ID = 144 with def = PDCP_STATS_V0
[NEAR-RIC]: Loading SM ID = 147 with def = ORAN-E2SM-KPM
[NEAR-RIC]: Loading SM ID = 145 with def = SLICE_STATS_V0
[iApp]: Initializing ...
[iApp]: nearRT-RIC IP Address = 172.18.0.2, PORT = 36422
fd created with 6
Received message with id = 2, port = 2733 Requesting connection
[E2AP] Received SETUP-REQUEST from PLMN 505. 1 Node ID 2 RAN type ngran_gNB_CU CU/DU ID 21
[NEAR-RIC]: Accepting RAN function ID 142 with def = MAC_STATS_V0
[NEAR-RIC]: Accepting RAN function ID 143 with def = RLC_STATS_V0
[NEAR-RIC]: Accepting RAN function ID 144 with def = PDCP_STATS_V0
[NEAR-RIC]: Accepting RAN function ID 145 with def = SLICE_STATS_V0
[NEAR-RIC]: Accepting RAN function ID 146 with def = TC_STATS_V0
[NEAR-RIC]: Accepting RAN function ID 147 with def =
[NEAR-RIC]: Accepting RAN function ID 148 with def = GTP_STATS_V0
```

Fig. 2: FlexRIC's log when CU connects with its.

Docker containers. Two essential tools are Docker Compose and Dockerfile when working with Docker containers.

III. Implemented System Description

The setup was implemented on a server with Core i5-9400 CPU @2.90GHz and 8GB RAM, running Ubuntu 22.04. The gNB functions and FlexRIC were deployed into four containers: one for FlexRIC, another for CU, a third for emulating DU and RU (denominated DU container), and finally, the OAI BaseStation. After creating images with Dockerfiles, Docker Compose was used to run the containers. The containers are linked by a Docker network, simplifying the orchestration and defining their configuration in a single YAML file for the environment.

Detailed steps for setting up the FlexRIC environment can be found in the GitHub git repository. It is crucial to generate the corresponding images before proceeding to ensure a successful configuration of FlexRIC. The docker-compose will create separate containers for each component and establish the necessary connections with FlexRIC.

To perform the setup implementation, we need to realize some steps:

- Clone the repository that contains the Dockerfile and docker-compose files.
- We must create two images (FlexRIC and OAI). The images contain all the required files for running FlexRIC, the emulated components of O-RAN, and OAI.
- Run docker-compose to generate the containers. It will change the configuration of the components to connect them to the FlexRIC container and make requests to run NearTime-RIC, CU, DU, and OAI.

So, after these steps, FlexRIC and the components such as CU, DU, and BaseStation will be created, according image 1.

The image 2 shows what you can expect to see when emulating the connection between CU and FlexRIC. The initial lines represent the initialization of FlexRIC, and the IP address to connect to is provided at the end. In the second part, the CU requests to connect to FlexRIC by sending a message that FlexRIC understands, and thus the connection is established.

IV. Conclusions

Automated installation simplifies the configuration process, allowing for easy execution of new projects and xApp testing without worrying about the complexity of environment configuration. This enables a more efficient and productive development cycle. Dockerfile and Docker-Compose can configure the environment on nearly any computer, making it highly portable and compatible with different operating systems. This approach reduces the need for complex manual configurations and minimizes potential dependency conflicts. Moreover, it enhances the efficiency of deploying development environments, saving time and effort.

References

[1] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding o-ran: Architecture, interfaces, algorithms, security, and research challenges," *IEEE Communications Surveys & Tutorials*, 2023.

[2] R. Schmidt, M. Irazabal, and N. Nikaein, "Flexric: An sdk for next-generation sd-rans," in *Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies*, 2021.

[3] A. Mufutau, F. Guiomar, A. Oliveira, and P. Monteiro, "Software-defined radio enabled cloud radio access network implementation using openairinterface," *Wireless Personal Communications*, 2021.

[4] N. Nikaein, M. K. Marina, S. Manickam, A. Dawson, R. Knopp, and C. Bonnet, "Openairinterface: A flexible platform for 5g research," 2014.

[5] G. L. Santos, D. d. F. Bezerra, E. d. S. Rocha, *et al.*, "Service function chain placement in distributed scenarios: A systematic review," *Journal of Network and Systems Management*, 2022.