RAN Slicing using OpenAirInterface and FlexRAN in a Virtualized Scenario

Lucas Nóvoa, Virgínia Tavares, Cleverson Nahum, Pedro Batista and Aldebaro Klautau

***Abstract*— Network Slicing is considered a key mechanism to serve all network clients in a flexible and cost-efficient manner. This concept defines a real network into virtual and independent logical networks allocated according to services and requirements. The software-defined RAN (Radio Access Network) platforms allow the control and administration of RAN, thus providing functionalities such as network slicing and RAN programmability. This paper presents an implementation of a software-defined RAN platform in a Centralized RAN architecture using FlexRAN and OpenAirInterface platform. Furthermore, the network elements were containerized using Docker in an orchestrated scenario using Kubernetes.**

***Keywords*— Network slicing, OpenAirInterface, FlexRAN, SD-RAN**

I. Introduction

The centralized radio access network (C-RAN) consists of a centralized architecture based on cloud computing for radio access network (RAN) [1]. C-RAN splits evolved nodeB (eNodeB) functionalities into baseband unit (BBU) and remote radio heads (RRHs). The BBU concentrates baseband processing functions, whereas the RRHs process and convert baseband signals into radio signals (RF) to be sent to user equipment (UE) in the downlink direction, and the way back in the uplink direction. The eNodeB contains information about the UEs connected and RAN configurations.

Software-defined RAN (SD-RAN) enables changes to eNodeB functions in real-time by implementing a programmable RAN which allows to access information and apply modifications through APIs [2]. SD-RAN enables the implementation of RAN slicing which refers to the logical division of the RAN into several other independent parts that meet different requirements.

In this paper, we focus on a C-RAN architecture implementation using the OpenAirInterface platform and an SD-RAN implementation using FlexRAN for RAN slicing. The network elements are virtualized using Docker container implementation and orchestrated using Kubernetes. We demonstrate RAN slicing in the FlexRAN platform and its API communication in a containerized and orchestrated scenario.

This work was supported in part by the Innovation Center, Ericsson Telecomunicações S.A., Brazil, CNPq and the Capes Foundation, Ministry of Education of Brazil. L. Nóvoa, V. Tavares, C. Nahum and A. Klautau contributions include both development of the testbed and concepts, they are with LASSE - 5G Group, Federal University of Pará, Belém, Brazil (e-mails:{lucas.pinto, virginia.tavares}@itec.ufpa.br {cleversonahum, aldebaro}@ufpa.br). P. Batista contributions are theoretical concepts and he is with Ericsson Research, Ericsson AB, Sweden.

II. OAI, FlexRAN and Virtualization

The OpenAirInterface (OAI) software [3], developed by the EURECOM group, implements the real-time functions of the LTE core network (EPC) and RAN. The OAI has two main projects: *openair-cn* and *openairinterface5G*. The first is an implementation of the 3GPP specifications related to the EPC, containing the functions: home subscriber server (HSS), mobility management entity (MME), packet data network gateway (PGW) and serving gateway (SGW). The second project is the implementation of RAN (eNodeB and UE). Based on the OAI 4G/5G open-source platform, the non-profit initiative Mosaic5G launched FlexRAN [4] as a controller to RAN providing SD-RAN functionalities. The main module of FlexRAN is the RAN runtime which separates user and control plane. FlexRAN is composed of one or more Agents and a Master Controller. The Agent is always placed in an eNodeB, and the Master Controller is responsible for managing the Agents. Unlike OAI, which does not separate control and data plane, FlexRAN incorporates a module called Agent API for a clear separation. It simplifies the development of control applications on top of FlexRAN to enable RAN programming.

Virtualization at the operating system level using containers can optimize OAI and FlexRAN giving more flexibility to network deployment and facilitating network management. Docker software allows the construction of containers through pre-configured images, containing all necessary dependencies and configurations [5]. The orchestration of containers enables making the deployment and distribution of the network elements among the cluster machines according to specifications and demands; the Kubernetes software [6] works as an orchestrator, implementing the communication among containers, creating firewall rules to enable communication into the cluster and ensuring that all physical and virtual machines communicate.

III. Implemented System Description and Experimental Results

The BBU and RRH machines use a Core i5-7500 CPU@3.40GHz processor, they are configured with Ubuntu 18.04 LTS as the operating system (OS) and "4.8.0-52 low latency"as the Linux kernel. A USRP B210 is connected to the RRH machine through USB 3.0 to work as an RF transceiver. A different Linux version is required (Ubuntu 16.04 LTS) to execute the EPC. To this end, a virtual machine was created inside the BBU hardware. The EPC was implemented by an *openair-cn* snapshot, following [7]. The BBU, RRH and FlexRANMaster Controller have isolated Docker containers.

While the FlexRAN Agent is configured in the BBU container. Kubernetes is used for the container orchestration. The controller is entirely transparent to the end-user, which implies greater flexibility and reliability of the network. A Samsung Galaxy S4 smartphone is connected to the network to work as UE and generate the results.The testbed is illustrated in Fig. 1. The RAN slices are created according to the documentation in [8].

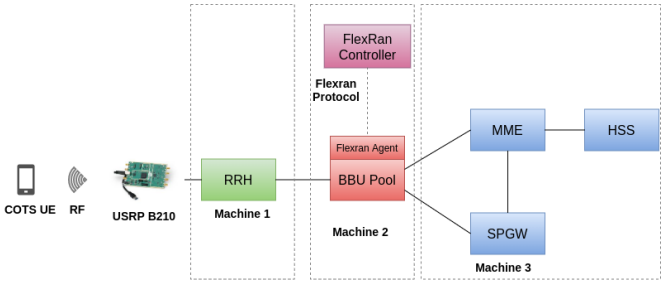

Fig. 1. System architecture.

The `Iperf3` tool is used for traffic generation in the UE and to perform measurements of the bandwidth. Fig. 2 shows the throughput of a UE, using different RAN slicing configurations where these values were set manually using the FlexRAN API itself, in the first scenario the UE uses 100% of the physical resources, while for the second and third scenarios it uses 88% and 76%, respectively. With the FlexRan we can set a RAN slicing of 100%, when the UE uses this quantity of physical resources, the bandwidth is such that the mean value of capacity of data transfer for the instant of time equal to zero is 16.58 Mbps. When a lower amount of RAN slicing resources are assigned to the UE, it perceives a decrease in bandwidth because it has fewer physical resources available to use. We change the values in the API of FlexRan, to get a RAN slicing of 88% the UE has fewer physical resources available and hence it presents a lower bandwidth than a RAN slicing using 100% of the resources. The same applies to the 76% RAN slicing. The RAN slicing values are defined through the FlexRAN API and any value can be set. UE throughput for each slice configuration varies because of the environment conditions for air transmission and the position variation of the UE during the tests.

Fig. 3 shows the FlexRAN API response which contains information about UEs connected and slices created in the FlexRAN Master Controller. In Fig. 3, the UE is connected to a slice in the eNodeB, labeled as xMBB (Extreme Mobile BroadBand) which requires extremely high transfer rates and low latency [9]. For the user associated with the slice id 0, the maximum modulation coding scheme (MCS) is 28, equivalent to a 64 QAM modulation, note that the MCS is given by eNB according to the condition of the UE channel. As a result, is possible to know about the data transfer rate of the network and the amount of the network being delivered for each slice. The dlSliceId/ulSliceId defines which slice the UE is associated with downlink and uplink, respectively. The proposed setup using the FlexRAN API can enable machine learning applications by providing both a data pipeline with RAN information and the ability to improve the network operation through optimized management.

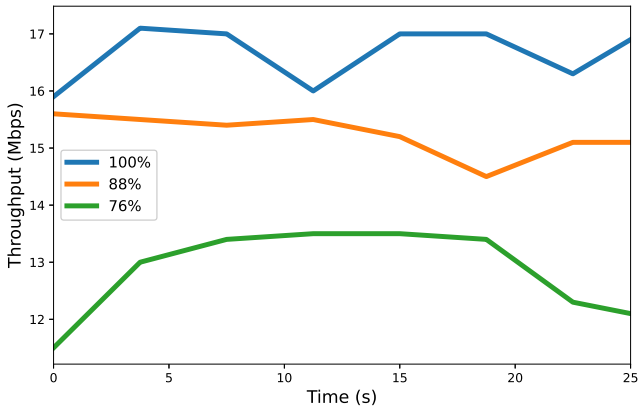

Fig. 2. UE throughput for different RAN slicing configurations. The percentage defines the amount of radio-resources assigned to the UE in the RAN slicing defined.


```
{
"UEs": [ { "rnti": 27002, "imsi": "208930000000001", "dlSliceId": 0,"ulSliceId": 0}],
"slices": {
        "dl": [{ "id": 0, "label": "xMBB", "percentage": 68, "maxmcs": 28 },
               { "id": 1, "label": "xMBB", "percentage": 32, "maxmcs": 28 }],
        "ul": [{ "id": 0, "label": "xMBB", "percentage": 68, "maxmcs": 20 },
               { "id": 1, "label": "xMBB", "percentage": 32, "maxmcs": 20 }]}
}
```


Fig. 3. FlexRAN API response showing UE, eNodeB and network slice configurations.

IV. Conclusion

This paper presented a deployment and implementation of a C-RAN architecture and SD-RAN using the OpenAirInteface and FlexRAN in a fully virtualized and automated way using Docker and Kubernetes. The RAN slicing function from FlexRAN was demonstrated with a comparison of the UE throughput variation for different RAN slicing scenarios, demonstrating how the testbed can be used to enable changes in the RAN.

Referências

[1] L. Gavrilovska, V. Rakovic, and D. Denkovski, "From Cloud RAN to Open RAN," *Wireless Personal Communications*, pp. 1–17, 2020.

[2] I. Afolabi, T. Taleb *et al.*, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2429–2453, 2018.

[3] OAI. OpenAirInterface 5G software alliance for democratising wireless innovation. [Online]. Available: http://www.openairinterface.org/

[4] X. Foukas, N. Nikaein, M. Kassem, M. Marina, and K. Kontovasilis, "FlexRAN: A Flexible and Programmable Platform for Software-Defined Radio Access Networks," 11 2016, pp. 427–441.

[5] B. B. Rad, H. J. Bhatti, and M. Ahmadi, "An introduction to Docker and analysis of its performance," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 17, no. 3, p. 228, 2017.

[6] E. Brewer, "Kubernetes and the New Cloud," in *Proceedings of the 2018 International Conference on Management of Data*, 2018, pp. 1–1.

[7] OAI. Leveraging an Ecosystem of 5G services. [Online]. Available: http://mosaic-5g.io/resources/mosaic5g-oai-snaps-tutorial.pdf

[8] FlexRAN. SliceConfiguration using FlexRAN. [Online]. Available: http://mosaic-5g.io/apidocs/flexran/#api-SliceConfiguration

[9] S. E. Elayoubi, M. Fallgren *et al.*, "5G service requirements and operational use cases: Analysis and METIS II vision," in *2016 European Conference on Networks and Communications (EuCNC)*, 2016, pp. 158–162.