A Network Slicing Prototype for a Flexible Cloud Radio Access Network

Salvatore Costanzo*, Ilhem Fajjari†, Nadjib Aitsaadi‡ and Rami Langar§
* Sorbonne Universites, UPMC Univ. Paris 06, CNRS, LIP6 UMR 7606, F-75005 Paris, France
† Orange-Labs, F-92320 Châtillon, France
‡ University Paris-Est, LIGM-CNRS UMR 8049, ESIEE Paris: F-93162, Noisy-le-Grand, France
§ University Paris-Est, LIGM-CNRS UMR 8049, UPEM, F-77420, Marne-la-Vallée, France
Emails: Salvatore.Costanzo@lip6.fr, ilhem.fajjari@orange.com, nadjib.aitsaadi@esiee.fr, rami.langar@u-pem.fr

Abstract—**The next 5G infrastructure is expected to serve a multitude of services with heterogeneous requirements, which might be potentially managed by multiple Mobile Virtual Network Operators (MVNOs) that share the same network infrastructure. The new emerging technologies, such as i) Software Defined Networking (SDN), ii) Network Function Virtualization (NFV) and iii) Network Slicing, where physical resources are partitioned and allocated in an isolated manner to a set of services or to MVNOs according to a specific Service Level Agreement (SLA), are seen as the key enabling approaches to fulfill the diversity of requirements of 5G services in a cost-effective manner.**

In this paper, we design and prototype a network slicing solution, which we have developed in a Cloud-RAN (C-RAN) infrastructure based on the Open Air Interface (OAI) platform and FlexRAN SDN controller. The aim of our work is to validate the feasibility of the prototype in handling the creation and configuration of network slices on-demand, taking into account some requirements that are elaborated from SDN-based slicing applications. By means of emulations, we show that our prototype reacts well to the inputs coming from the SDN application and is finally capable of providing isolation among multiple slices in a dynamic fashion.

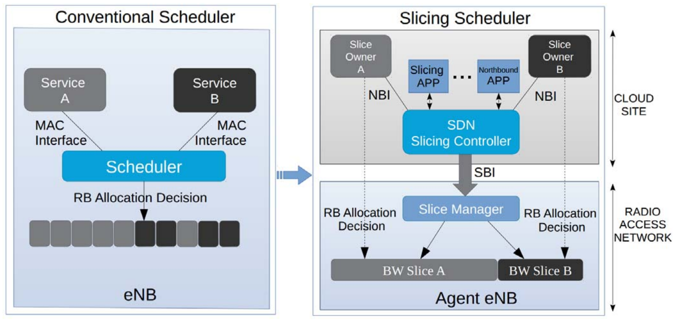
Keywords: 5G, Network Slicing, Cloud RAN, SDN, OAI.

I. INTRODUCTION

Thanks to the advent of the long-awaited fifth generation (5G) mobile networks, mobile data and online services are becoming widely accessible. Discussions of this new standard have taken place in both industry and academia to design this emerging architecture. The main objective is to ensure, by 2020, the capability to respond to the different application needs such as videos, games, web searching, etc., while ensuring higher data rate and enhanced Quality of Service (QoS). To this end, 5G has been designed in such a way that offers a programmable and a flexible infrastructure, allowing different networks (e.g., IoT, cellular, vehicular, etc.) to share the same access network. A way to efficiently match the diverse service requirements of 5G is to adopt cloud computing-based architectures and employ the concept of network slicing. The latter is a pillar of 5G architectures enabling the split of operator infrastructure into multiple logical networks, referred to as slices, to flexibly deploy diverse 5G services over a common shared network infrastructure, while guaranteeing QoS and Service Level Agreement (SLA). Besides, Cloud Radio Access Network (C-RAN) [1] is considered as the reference architecture for 5G, wherein the baseband operations are centralized, introducing several opportunities for performing a centralized computation of the radio resources, while reducing Capital Expenditure (CAPEX) and Operational Expenditure (OPEX). Moreover, C-RAN can enable the virtualization of a set of RAN functions, opening the road for the so-called virtualized RAN. In doing so, the resources can be abstracted to create multiple virtual networks or slices.

According to the NGMN's vision for 5G [2], slices are defined as end-to-end self-contained logical networks, which can be controlled and managed in an independent way by the slices' owners, such as Over-The-Top (OTT) service providers and Virtual Mobile Network Operators (VMNOs). Indeed, slices consist of a number of network resources, which are selected from a common network infrastructure. The network resources are virtualized and allocated to the logical slices making use of the principles of Software Defined Networking (SDN) and Network Function Virtualization (NFV) [3]. In fact, the latter provides the appropriate tools to orchestrate the underlying resources and enables the coexistence of independent slices within the same physical infrastructure. To this end, the Open Networking Foundation (ONF) puts forward in [4], an SDN-based architecture for network slicing, where a logically centralized unit, named controller, acts as an intermediate entity between the slice's owner and the shared physical network resources. The controller provides an abstracted view of the network resources to the slices' owners and let them communicate their service demands through an appropriate northbound interface.

In this paper, our main objective is to deal with the slicing problematic in a practical and concrete manner, while taking as a starting point the ONF architecture. Our contribution is twofold. Firstly, we design and prototype a network slicing solution in a C-RAN infrastructure, which instantiates the SDN logic architecture proposed by ONF, with the aim to efficiently share spectrum between different slices while considering their requirements. Secondly, we design and implement a northbound SDN application, which enables the creation and configuration of slices on demand. To do so, we put forward a first algorithm to evaluate the effective capability of the prototype to configure slices on-demand according to predefined set



(a) Conventional Scheduler (b) Proposed Slicing Scheduler

Fig. 1: Network Slicing Solution

of inputs. To validate our design and evaluate its performance, we make use of both OpenAirInterface (OAI) [5] platform and FlexRAN SDN controller [6]. It is worth noting that OAI Software Alliance is an open-source Software Defined Radio (SDR) implementation of both the Long Term Evolution (LTE) Radio Access Network (RAN) and the Evolved Packet Core (EPC), while FlexRAN is an SDN controller compliant with the OAI platform. Interestingly, OAI and FlexRAN have been successfully employed in the framework proposed in [7], wherein authors provide an initial performance evaluation of a scenario with multiple slices sharing a single RAN and with multiple core networks. Differently from [7], in this paper we mainly focus on the resource allocation issue and we propose a dynamic slicing approach with the aim to validate the isolation property which is a key enabler for an efficient slicing. By means of emulations, we show that our prototype is capable of handling multiples slices in real-time, while providing isolation among them.

The reminder of this paper proceeds as follows. In Section II, we present the proposed network slicing solution, followed by a description of our prototype architecture in Section III. In Section IV, we evaluate the performances of the proposed network slicing solution, while Section V concludes the paper.

II. Proposed slicing solution for C-RAN

In this section, we detail our SDN-based network slicing solution, which enables a flexible spectrum sharing between multiple slices owners, such as OTT service providers or VMNOs, while ensuring the required isolation between slices.

Our solution makes use of an SDN approach to manage i) the creation of each slice and ii) the dynamic configuration of its size on demand, in accordance to the agreed SLA constraints. More specifically, we ensure a real-time allocation of the bandwidth resources thanks to an efficient slicing scheduling algorithm that is driven by a centralized SDN controller. To highlight the added value of our slicing approach, depicted in Fig. 1(b), we compare it with the conventional LTE scheduler solution architecture depicted in Fig. 1(a). Indeed, it is straightforward to see that, in the conventional LTE architecture, the control part of the Medium Access Control (MAC) layer, dealing with scheduling decisions, and the action part, which is responsible for the execution of the decision, are merged together. We recall that MAC scheduling decisions are handled by the scheduler, which allocates for each service an amount of Resource Blocks (RBs), i.e., the minimum resource allocation unit in LTE. It is worth noting that the total bandwidth corresponding to the total number of RBs is shared among the different services according to a specific scheduling policy. The latter takes into account some priority weights, that are related to the type of service. In this context, several scheduling algorithms have been proposed in the related work. Some of them give priority to real-time services, by penalizing best effort services, while others aim at achieving more fairness among all the services, by penalizing the system aggregate throughput (e.g., proportional fair scheduler). Unfortunately, in both cases, the finite and limited nature of the spectrum may considerably impact the performances of services which concurrently share it.

The main idea behind our slicing approach is to separate the control plane from the data plane. Hence, the scheduling logic is detached from the action logic and then instantiated as a northbound application of the SDN controller. The latter is hosted in the cloud site. By doing so, the LTE eNodeB (eNB) is freed from the control responsibilities and only handles the RB allocation based on the scheduling decision made by the SDN controller. The latter, referred to as SDN Slicing Controller, is interfaced with the eNB via an appropriate SouthBound Interface (SBI) and provides a NorthBound Interface (NBI) to the slices' owners as depicted in Fig. 1 (b). On the top of the SDN Slicing Controller, we have developed a northbound application (Slicing APP in Fig. 1(b)) to manage the slicing scheduling process in real-time.

According to the SDN architecture, illustrated in Fig. 1(b), a slice's owner can communicate a service request to the SDN Slicing Controller via the NBI interface. For instance, it may request an amount of resources for a number of users which are located in a specific area, while guaranteeing a specific QoS target. Accordingly, the SDN Slicing Controller verifies whether such a service request can be satisfied according to the real-time network state, and communicates the so called "slicing decision" to the Slice Manager function, through the SBI interface. Finally, the Slice Manager instantiates the slices, by reserving an appropriate portion of the bandwidth (i.e., an amount of RBs), to each slice's owner in accordance to the slicing decision carried out by the SDN Slicing Controller.

III. Experimental platform

In this section, we present the technical details of the experimental platform, that we implemented to evaluate the performance of our SDN-based slicing solution.

Our prototype, illustrated in Fig. 2, makes use of OAI software [5] and FlexRAN SDN controller [6]. It is in accordance with the NGFI C-RAN architecture [8], wherein the baseband processing functions are carried out at the Radio Cloud Center (RCC) node, which in turns sends I/Q samples to the Radio Remote Unit (RRU) via a fronthaul interface (FH). Note that the prototype relies on an Ethernet-based FH to interconnect

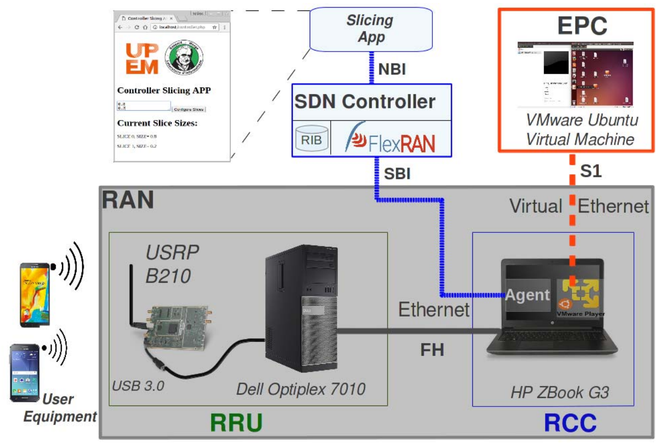


Fig. 2: C-RAN testbed

RCC and RRU. The C-RAN architecture adopts the IF4p5 splitting [8] provided by OAI. It is worth noting that IF4p5 refers to the split-point at the input (TX) and output (RX) of the OFDM symbol generator. More details about the C-RAN splitting solutions, made available by OAI, are provided in [8].

In our prototype, the RRU consists of Ettus USRP B210 card [9] which is connected to a server via USB 3.0 interface. The server has a processor power (CPU) Intel i7-3770 8-core (@3GHz), and a Random Access Memory (RAM) of 16 GB performances. It is running with "Ubuntu 14.04" operating system characterized by "Linux kernel" release 3.19.0-61-*lowlatency SMP PREEMPT* and has 1 Gigabit Ethernet port.

The RCC, which implements the remaining levels of the LTE protocol stack in accordance with IF4p5, consists in a laptop, characterized by Intel $i7-6500U$ 4-core (@2.50GHz) CPU and RAM of 16 GB. The latter is running with the same operating system as the RRU. Upon the aforementioned operating system, we run a VMware Ubuntu 14.04 Virtual Machine where all the functionalities of the EPC are implemented by the OAI software. Moreover, this laptop runs an SDN controller, which is based on FlexRAN, and a specific Agent [6], that interconnects the RCC and the controller through an SBI interface, which makes use of Google Protobuf [6].

On the top of SDN controller, we have implemented a northbound Slicing APP, which enables any user of our prototype to configure the slicing process in real-time via a web GUI, as illustrated in Fig. 2. By means of the Slicing APP, the users of our prototype can experiment our slicing solution and manually set the size of each slice for testing purposes. It is worth noting that, the configuration of the slicing process can be also done automatically making use of a dynamic slicing algorithm. In fact, to put forward the interest of our SDN-based slicing solution, we have developed a dynamic allocation algorithm for the Slicing APP, which exploits the unused RBs of a specific slice and distribute them to a slice having worst performance. The pseudo-code of the proposed dynamic slicing algorithm is detailed in Algorithm 1. At each transmission time interval (TTI), the Slicing APP makes an estimation of the resource allocation satisfaction of each slice (steps 3-4). If that value, averaged in a time interval period, is less than a specific threshold, then the algorithm exploits the unused RBs of a specific slice and distribute them to the slices having worst performance (steps 9-10).

IV. Performance evaluation

We have performed diverse emulations to assess the feasibility and gauge the efficiency of employing our SDN-based slicing solution in the prototype described in Section III. In our emulations, the C-RAN network has been configured with a 5 MHz bandwidth, using the LTE Frequency Division Duplex (FDD) scheme, while it is assumed the presence of two smartphones, which act as Commercial Off-the-Shelf (COTS) user equipment (UEs). The first UE, denoted as UE_0, streams an H.264 encoded test video, with an overall bit rate of 250 Mbps, a 3840x2160 resolution and total size of 897 MB. The second UE, denoted as UE_1, generates Best Effort (BE) traffic, by downloading one or more files from an FTP server. For testing purposes, we assume that UE_1 downloads one or more $4K$ videos with a file size of 990.8 MB.

In our emulations, we define three scenarios. The first one, named Baseline (BL), corresponds to the conventional scenario (see Fig. 1(a)) with a single Proportional Fair (PF) scheduler. This means that the total bandwidth is shared among the 2 UEs and no slices are instantiated. In the second scenario, named *S*_80_20, a static partition of the bandwidth is performed. More specifically, 80% of the bandwidth is allocated to one slice, named S_0, while the remaining portion of the bandwidth is allocated to the second slice, named S_1. We assume that the UE_0 traffic is associated to S_0, while the UE_1 traffic is handled by S_1. The third scenario, named *Dyn_S*, employs the slicing algorithm described in section III, wherein the configuration of the slices is dynamically performed, starting from the initial configuration of the static slicing scenario. Note that a video of our demonstration has been made available at [10].

Fig. 3 depicts the average throughput, in downlink (DL), measured in both slices. Note that the results are obtained by varying the number of BE flows, i.e., the number of running FTP downloads at the UE_1, while keeping the UE_0, which belongs to the slice S_0, streaming 1 video with the profile described previously.

Algorithm 1 Dynamic Slicing

```
1:  for $TTI \in T$ do
2:      for $s$=1:$S$ do
3:          $RB\_R(TTI,s)=\frac{\#RBs\_alloc(TTI,s)}{\#RBs\_required(TTI,s)}$
4:          $RB\_R_{\Delta}(s) = AVG(RB\_R(TTI,s), \Delta T)$
5:          if $RB\_R_{\Delta}(s) \leq Thr(s)$ then
6:              SET C = $\{S \setminus s\}$
7:              $c\_s$=$search\_slice\_with\_Max\_RB\_R\Delta(C)$
8:              if $RB\_R_{\Delta}(c\_s) \geq Thr(c\_s)$ then
9:                  $setSize(s)[TTI_{+1}]$=$Size(s)[TTI]$+%$Size(c\_s)$
10:                 $setSize(c\_s)[TTI_{+1}]$=$Size(c\_s)[TTI]$-%$Size(c\_s)$
11:             end if
12:         end if
13:     end for
14: end for
```

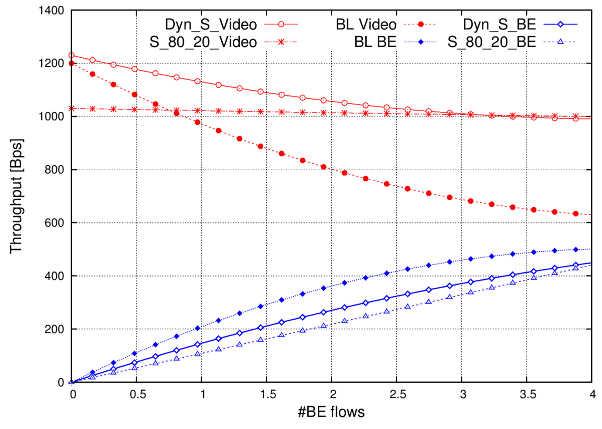


Fig. 3: Average DL Throughput vs #BE flows

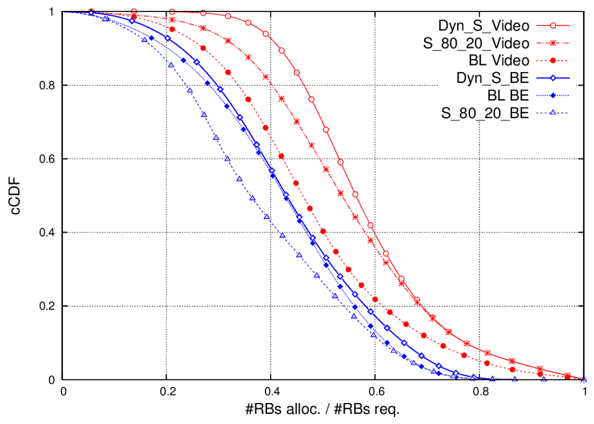


Fig. 4: cCDF of RB_R

Fig. 3 shows that the performance of the video flow is strongly impacted by the load of the BE flows in the BL scenario (i.e., conventional scheduler), while a different trend in the other two scenarios is observed. In particular, in the *S*_80_20 scenario, an up to double throughput gain is observed for high load of BE traffic, as compared to the BL scenario. This behavior can be explained as follows. In the BL scenario, a PF scheduler is employed and the traffic is not isolated in both slices, as opposed to the other scenarios. Consequently, the increasing load of BE traffic impacts the scheduling operation, which, in turn, tries to allocate more RBs to the BE UE to achieve a fairness among all the traffic flows.

In the other two scenarios, wherein the slicing solution is employed, the final scheduler decision is highly influenced by the size of each slice. Indeed, the slice S_0 has at its disposal up to 80% of the total number of RBs, i.e., 25 for a 5 MHz bandwidth. Through the emulation results, it can be seen that the size of the slice S_0 is sufficient to accommodate the traffic requests in S_0, while the slice S_1 struggles to ensure the required performances due to the limited amount of its allocated RBs. This results in lower performances for S_1, compared to the BL case. The deterioration of the performances of the BE traffic flows derives from the lack of multiplexing gain for the BE traffic in the RB allocation process. In fact, the BE flows cannot be scheduled in the portion of the spectrum that is exclusively dedicated to the slice S_0, due to the static slice configuration policy adopted in the scenario *S*_80_20. Besides, it can be observed that in the *Dyn_S* scenario, the performance of the BE traffic is slightly improved as compared to the static slicing scenario.

Interestingly, Fig. 3 provides an important result in terms of validation of the *Isolation* property of network slicing. In fact, as it can be seen, differently from the BL scenario, the throughput of S_0 is not affected anymore by the traffic load in S_1 in both the *S_80_20* and *Dyn_S* scenarios. S_0 and S_1 are indeed isolated from each other.

Fig. 4 shows the complementary CDF (cCDF) of the $RB\_R(TTI, s)$ value defined in Algorithm 1, which provides an estimation of the resource allocation satisfaction of each slice, in the three different aforementioned scenarios. Interestingly, we can observe that the probability for the slice S_0 to get a 50% satisfaction in the resource allocation process is 20% and 30% higher than in the BL case for the *S_80_20* and *Dyn_S* scenarios, respectively.

V. Conclusion

This paper has proposed and validated a network slicing prototype for a flexible 5G infrastructure based on the C-RAN software provided by OAI. We expect our SDN-based prototype design to provide worthwhile insights into developing efficient slicing solutions for future 5G systems with an in-depth consideration of practical implementation.

Future work is needed to design new slicing algorithms, aiming at achieving an optimized allocation of resources towards a complete RAN virtualized solution. The instantiation process of slices shall be enhanced to take into account multiple RAN parameters, such as different fronthaul splitting options for each slice.

Acknowledgment

This work has been funded by the FUI ELASTIC Network project (Grant no. C16/0287).

References

[1] China Mobile Research Institute, "*C-RAN The Road Towards Green RAN*", White Paper, Oct. 2011.
[2] N. G. M. N. Alliance, "*5G white paper*", Next generation mobile networks, White paper, vol. 1.0, Feb. 2015.
[3] ETSI's Network Functions Virtualisation (NFV) Industry Specification Group, "*Network Operator Perspectives on NFV priorities for 5G*", ETSI White Paper, Feb. 2017.
[4] ONF TR-526, "*Applying SDN Architecture to 5G Slicing*", Issue 1, Apr. 2016.
[5] N. Nikaien, "*OpenAirInterface Simulator/Emulator*", available on line at http://www.openairinterface.org/, Jul. 2015.
[6] X. Foukas, N. Nikaein, M. M. Kassem, M. K. Marina and K. Kontovasilis. "*FlexRAN: A Flexible and Programmable Platform for Software-Defined Radio Access Networks*", ACM CoNEXT, California, USA, Dec. 2016.
[7] A. Ksentini and N. Nikaein, "*Toward Enforcing Network Slicing on RAN: Flexibility and Resources Abstraction",* IEEE Communications Magazine, vol. 55, no. 6, pp. 102-108, Jun. 2017.
[8] *http://gitlab.eurecom.fr/oai/openairinterface5G/wikis/how-to-connect-cots-ue-to-oai-enb-via-ngfi-rru*
[9] "*USRP B200/B210 Specification Sheet*", avalaible online at https://www.ettus.com/product/details/UB200-KIT
[10] "*DEMO of Network Slicing Prototype for C-RAN*", avalaible online at https://sdr-lab.lip6.fr/slicing-C-RAN.html