Toward Enforcing Network Slicing on RAN: Flexibility and Resources Abstraction

Adlen Ksentini and Navid Nikaein

The authors propose a new framework to enforce network slices, featuring radio resources abstraction. The proposed framework is complementary to the ongoing solutions of network slicing, and fully compliant with the 3GPP vision.

ABSTRACT

Knowing the variety of services and applications to be supported in the upcoming 5G systems, the current "one size fits all" network architecture is no more efficient. Indeed, each 5G service may have different needs in terms of latency, bandwidth, and reliability, which cannot be sustained by the same physical network infrastructure. In this context, network virtualization represents a viable way to provide a network slice tailored to each service. Several 5G initiatives (from industry and academia) have been pushing for solutions to enable network slicing in mobile networks, mainly based on SDN, NFV, and cloud computing as key enablers. The proposed architectures focus principally on the process of instantiating and deploying network slices, while ignoring how they are enforced in the mobile network. While several techniques of slicing the network infrastructure exist, slicing the RAN is still challenging. In this article, we propose a new framework to enforce network slices, featuring radio resources abstraction. The proposed framework is complementary to the ongoing solutions of network slicing, and fully compliant with the 3GPP vision. Indeed, our contributions are twofold: a fully programmable network slicing architecture based on the 3GPP DCN and a flexible RAN (i.e., programmable RAN) to enforce network slicing; a two-level MAC scheduler to abstract and share the physical resources among slices. Finally, a proof of concept on RAN slicing has been developed on top of OAI to derive key performance results, focusing on the flexibility and dynamicity of the proposed architecture to share the RAN resources among slices.

INTRODUCTION

Network slicing is definitely one of the key enablers of the upcoming fifth generation (5G) systems, where the objective is to build a novel network architecture that should support not only classical mobile broadband applications and services, but also vertical industry (e.g., automotive systems, smart grid, public safety) and Internet of Things (IoT) services. Besides human oriented devices (i.e., smartphones and tablets), 5G systems will include sensors, actuators, and vehicles, allowing the support of more than 50 use cases and scenarios [1].

To enable network slicing in the future mobile network generation, Third Generation Partnership Project (3GPP) SA2 (http://www.3gpp.org/Specifications-groups/sa-plenary/53-sa2-architecture, accessed 7 March 2017) and RAN3 (http://www.3gpp.org/specifications-groups/ran-plenary, accessed 7 March 2017) groups are building technical specifications to integrate network slicing in the upcoming 3GPP standards. Other standardization bodies, like the International Telecommunication Union Telecommunication Standardization Sector (ITU-T) through the IMT 2020 group (http://www.itu.int/en/ITU-T/focusgroups/imt-2020/Pages/default.aspx, accessed 7 March 2017), and the Next Generation Mobile Network Alliance (NGMN) (https://www.ngmn.org/home.html, accessed 7 March 2017) are studying the requirements and architectures that will enable network slicing in 5G. Meanwhile, many 5G initiatives and projects, such as the Fifth Generation Public Private Partnership (5GPPP) (http://5g-ppp.eu, accessed 7 March 2017) European program, have crossed the border by including network slicing in their first outputs. Particularly, a global commitment has been made to the definition of network slice categories, wherein each 5G service may fall:

- Extreme mobile broadband (xMBB) type, which requires both high data rates and low latency in some areas, and reliable broadband access over large areas
- Massive machine-type communication (mMTC) type, which needs wireless connectivity for massive deployment of devices
- Ultra-reliable and low-latency communications (uRLLC) or ultra-reliable MTC, which covers all services requiring ultra-low latency connections with a certain level of reliability

Stemming from the fact that these three types of services cannot be sustained by the same physical infrastructure, agile and programmable network architecture is envisioned; each service should have a tailored network instance to satisfy its requirements. Using software defined networking (SDN), network functions virtualization (NFV), and cloud computing will enable building a programmable and flexible network instance (i.e., virtual network) tailored to services' needs.

So far, most of the devised network architectures [2–5] that enable network slicing is based on SDN, NFV, and cloud computing. These proposals share the same principle, with some difference in the way to instantiate and deploy a

Digital Object Identifier:
10.1109/MCOM.2017.1601119

The authors are with EURECOM Sophia Antipolis.

network slice. Mainly, a global slice orchestrator is proposed on top of an NFV-like architecture, which translates the slice provider requests by selecting the appropriate virtual network functions (VNFs) (e.g., core network — CN — functions, firewall, deep packet inspection — DPI) along with their service graph, which specifies how logically these VNFs are connected. Then the VNFs are deployed over the distributed cloud using a virtual infrastructure manager (VIM) and SDN rules to interconnect them. Each slice might include its own SDN controller to manage the intra slice traffic. Resources (i.e., infrastructure, radio spectrum, and transport network) may belong to the same or different administrative domains; the latter case requires a multi-domain orchestrator. Note that an administrative domain corresponds to a domain managed by one entity (e.g., network operator).

Clearly, these propositions cover a high-level view of how to dynamically build and manage a slice. They assume that the infrastructure and the RAN are easily virtualized and sliced. While several techniques on slicing/virtualizing the infrastructure exist, slicing the RAN is still very challenging.

Our contributions in this article are:

- A network slicing architecture based on eDECOR [6], a solution introduced by the 3GPP to implement the principle of dedicated core network (DCN), which enables enforcing slices in the mobile network (particularly at the core network level)
- A two-level MAC scheduler; the first level of scheduling is done by a slice-specific scheduler, while the second level is done by a common scheduler that maps the first level scheduling propositions to physical radio resources allocation
- Programmable and flexible RAN following SDN principles
- A proof of concept (PoC) based on OpenAirInterface (OAI), (http://www.openairinterface.org/, accessed 7 Mar. 2017) which is an open source tool implementation of 3GPP RAN and CN functions

The remainder of this article is organized as follows. The next section presents the concepts of network slicing, focusing on the challenges and solutions for RAN slicing. Then we detail our proposed architecture for enforcing slices in mobile networks. Following that, we introduce the two-level medium access control (MAC) scheduler, featuring RAN resources abstraction. Then we give some results obtained from the PoC of the architecture built on OAI. Finally, we conclude this article.

Related Work

Network slicing in a mobile network is highly related to network sharing, particularly to RAN sharing in the case of mobile networks. Indeed, 3GPP has defined and ratified different kinds of architecture with varying degrees of sharing [7]: multi-operator RAN (MORAN): only equipment is shared; multi-operator core network (MOCN): both spectrum and equipment are shared; and gateway core network (GWCN), in which both the RAN and some elements of the CN are shared.

Focusing on RAN slicing, there are different envisioned models to implement it. Depending on the level of resource isolation, we may mention dedicated resources and shared resources models. In the dedicated resource model, the RAN slice is built by separating and isolating slices in terms of control and user plane traffic, MAC scheduler and physical resources. Each slice has access to its own remote radio control (RRC)/radio link control (RLC)/packet data control protocol (PDCP)/MAC[1] instances, and the physical resources are strictly dedicated to a specific slice, for example, a percentage of physical resource blocks (PRBs) is dedicated to each slice, or a subset of the channel is dedicated to each slice. Although dedicated resource model ensures committed elementary resources to the slice, it reduces the slice elasticity as well as scalability, and limits the multiplexing gain. Indeed, using the dedicated resource model does not allow a slice owner to easily modify the amount of resource (i.e., PRB) committed to a slice during its life-cycle. Furthermore, the dedicated resources model may lead to a waste of resources, as the PRBs are strictly dedicated to a slice, even if they are not used. The second approach, that is, the shared resources model, allows the slice to share the same control plane, MAC scheduler, and physical resources. In this solution, the PRBs are managed by a common scheduler that distributes the PRB to slices' users according to different criteria, including service level agreement (SLA), priority, and so on. While this solution exploits statistical scheduling of physical resources, which ensures more scalability and elasticity by reporting to the dedicated resources model, it may lack the support of strict quality of service (QoS) guarantee for slices and traffic isolation.

Regarding the literature, many works have addressed the challenge of RAN sharing from two main perspectives: resource sharing among mobile virtual network operators (MVNOs) by modifying the MAC scheduler; and radio resources isolation. For resources sharing, the authors in [8] introduce a network virtualization substrate (NVS), which operates on top of the MAC scheduler. Its objective is to flexibly allocate shared resources modifying the MAC scheduler to reflect an MVNO's traffic need and SLA. NVS was adapted to the case of RAN sharing in LTE [9], with the aim to virtualize the RAN resources. Arguing that most of the MAC schedulers for RAN sharing are less flexible and consider only SLA-based resource sharing, the authors propose AppRAN [10], an application-oriented RAN sharing solution. The aim is to adapt the RAN sharing mechanism to the applications' need in terms of QoS. Looking to radio resource isolation, RadioVisor [11] represents one of the major works that addresses this issue. RadioVisor aims to share RAN resources, which are represented in a three-dimensional grid (radio element index, time slots, and frequency slots). The radio resources (in the grid) are sliced by RadioVisor to enable resource sharing for different controllers, which provides wireless access to applications. Each controller is allowed to independently use the allocated radio resources without referring to the other controllers. It is worth noting that these works mainly focused on RAN sharing issues without considering the flexibility and dynamicity required to enable network slicing.

RadioVisor aims to share RAN resources, which are represented in a three-dimensional grid. The radio resources (in the grid) are sliced by RadioVisor to enable resources sharing for different controllers, which provide wireless access to applications. Each controller is allowed to independently use the allocated radio resources without referring to the other controllers.

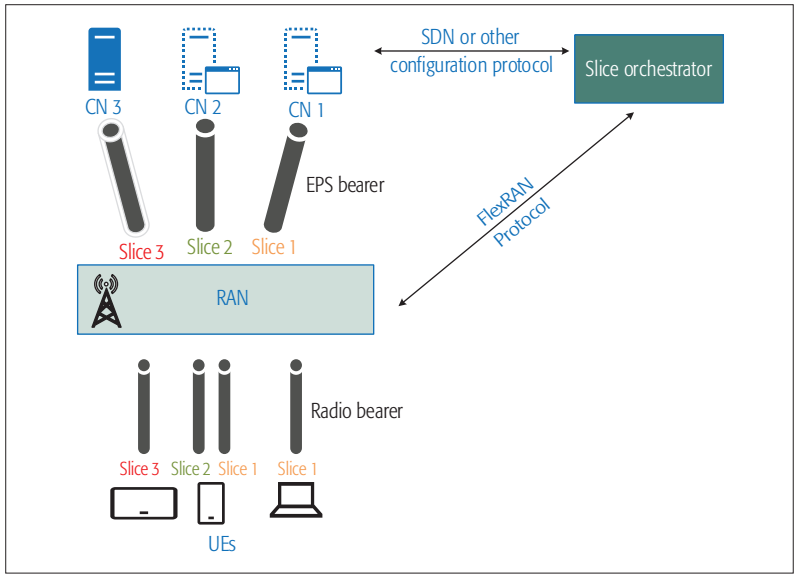

Figure 1. Global overview of the envisioned network slicing architecture

Network Slicing Architecture: Enforcing the Slices

In this section, we assume that three types of slice exist: xMBB, uRLLC, and mMTC; that is, each slice should belong to one of these types. Although we are referring to only three slices, the proposed solution and particularly the two-level scheduler can manage an arbitrary number of slices.

A global overview of the envisioned architecture is depicted in Fig. 1. In this figure, we present the network architecture after the slice orchestrator (SO) has instantiated the CN elements, using a physical network function (PNF) or a VNF. Each CN instance includes a set of VNFs or PNFs representing CN functions, such mobility management, authentication as well as data plane forwarding functions (i.e., gateways to the Internet). The CN instances are connected to the shared RAN using the classical S1 interfaces or the new interfaces (i.e., Gx, Gy) proposed by the 3GPP SA 2 group [12]. The RAN (i.e., eNodeB) is able to steer the slice traffic to the correct CN instances using the concept of eDECOR, in which the UE indicates a slice ID that allows the eNodeB to select the appropriate CN elements for the UE traffic. The slice ID could be hard encoded in the UE (i.e., USIM) or encoded through the public land mobile network (PLMN). Moreover, the UE communicates the slice ID during the RRC connection procedure as well as in the non-access stratum (NAS) procedure, which allows the eNodeB to contain the UE within the requested slice(s) and treat it according to the agreed SLA. For instance, use the appropriate MAC scheduler instance that will handle the slice resources to satisfy the required QoS. It is important to note that the slice ID should indicate the slice type (e.g., xMBB, uRLLC, or mMTC) in addition to the tenant ID. The tenant ID is the entity that is in charge of the slice. Finally, the eNodeB maintains a mapping between the slice ID and the CN elements (i.e., IP addresses), which is communicated by the SO during the slice instantiation process.

Figure 2 depicts the architecture of the eNodeB when using our proposed network slicing solution. The proposed architecture shares many concepts with the legacy LTE architecture, particularly the usage of logical channel sand their mapping to Evolved Packet System (EPS) bearers. The main difference is related to the abstraction (i.e., virtualization) of the PRBs, where an abstraction layer (the resource mapper — RM) is added. The latter acts as an interface between the shared PRB and the slice resource manager (SRM). The SRM is in charge of scheduling resources for UEs belonging to its slice. Any popular scheduling algorithm, such as Proportional Fair (PF), Round-Robin (RR), Priority-Based, and Delay-Based, could be used. Each SRM may use a different scheduler, as configured by the SO. The RM will expose the information to each SRM regarding:

- The number of packets, per UE and per logical channel, waiting for transmissions, in both the uplink and downlink directions
- The channel quality indicator (CQI) of each UE
- Optionally, the latency of the oldest packet in the UE's queue as well as the history and statistics of UE traffic

This information will be used by the SRM to schedule UEs over the virtual resource blocks (vRBs). The number of available vRBs is not limited and could be infinite; that is, the SRM may schedule all UEs during one cycle. However, the vRBs should be mapped to PRBs, and given that the number of PRBs is limited and dependent on the physical characteristics (i.e., throughput), not all the UEs will be served during the transmission time iInterval (TTI). The RM will be in charge of accommodating the vRBs to PRBs according to the amount of resources (noted slice dedicated bandwidth [SDB]) that should be allowed to each slice. The SDB is the policy enforced by the SO to the RM when the slice is first created. It is worth noting that the SDB is dynamic; it could be adapted as a function of workload demand and slice requirements upon a request from the slice. Moreover, the SDB policy can be expressed in terms of percentage of PRB or the bandwidth to be allocated to a slice among the others.

Another important function block is the agent, which takes charge of the communication with the remote eNodeB controller. Indeed, the envisioned architecture assumes that the SO is using a northbound application programming interface (API) exposed by an eNodeB controller to (re)configure, in real time, the eNodeBs. The eNodeB controller translates the SO requests to configuration messages (southbound API) communicated to the eNodeB agent. For instance, the SO uses a northbound API to communicate the SDB policy of a specific slice as well as the scheduler algorithm to be used by the SRM to the eNodeB controller, which will translate these requests to MAC-layer-related configuration messages to be sent to the eNodeB agent. The latter will configure the SRM and the RM. For more details on the eNodeB controller, agent, and configuration protocol, readers may refer to the FlexRAN work [13].

The remaining functional blocks are similar to the 4G eNodeB architecture. The user plane is in charge of handling per UE and bearer traffic, including UL and DL. The routing function is in

charge of steering the user traffic to the appropriate CN instance, and steering the downlink traffic to the appropriate logical channel queue. As mentioned earlier, the eNodeB uses the slice ID communicated by the UE during the RRC connection procedure to know to which slice, SRM, and CN instances the UE belongs. Hence, the eNodeB needs to maintain a mapping between the UE ID (i.e., cell radio network temporary identifier [C-RNTI], tunnel ID [TEID]) and the SRM and CN instances. Every time a packet is received from the CN, the eNodeB enqueues the packets to the logical channel associated with a UE for a specific slice; we may see the logical channels as a two-dimensional matrix, where the horizontal axis represents the slice ID, while the vertical one represents the UE ID. Therefore, a UE may belong to more than one slice, where each logical channel is managed by a specific SRM and CN, and one SRM and CN instance may handle several logical channels.

It is important to note that we omitted virtualizing the physical channel (i.e., RAN isolation) for the reasons stated above. In addition, we argue that creating a specific physical channel for a slice will require that each physical layer processes the common in-phase quadrature phase (I/Q) flow (e.g., orthogonal frequency-division multiplexing [OFDM] modulation/demodulation in LTE) before being able to decode the traffic dedicated to UE from one slice, which is very resources consuming and inefficient. However, user-specific physical layer processing (e.g., turbo coding/decoding in LTE) can be virtualized (i.e., shared) depending on the required level of isolation. Finally, we believe that in the absence of beamforming operation, affecting one PRB per UE is enough to ensure RAN traffic isolation.

Two-Level Scheduling Process

As stated earlier, we propose to virtualize the PRB by introducing the RM layer, which is in charge of the vRB/UE mapping to a PRB, using the SRM scheduling output as much as possible. Accordingly, we partitioned the MAC operation into two levels. The SRM performs the first level by ensuring intra-slice traffic scheduling, while the RM assigns PRBs to UEs according to the mapping provided by each active SRM, the SDB policy, the actual channel state (i.e., the available PRB), and slice priority. The proposed two-level scheduling is preferred over the joint scheduling (all in one pass), as the latter is very complex and requires multi-dimensional scheduling. Indeed, this type of scheduling algorithm formulates a multi-objective function that should satisfy heterogeneous slice requirements (e.g., latency and bandwidth), where the optimal solution is usually NP-hard.

Clearly, using a different SRM will ensure: (i) more scalability, as the SRM is created when the slice is instantiated; (ii) dynamic resources management, since using the SDB policy will allow the resources to be adapted to the workload demand (service elasticity and scalability) when instructed by the SO.

SRM Functions

The SRM's main function consists of scheduling the intra slice UEs' traffic by assigning vRBs to UEs. The vRBs are virtual and do not have any link to the available PRB. Nevertheless, they share the same size in order to ease the alignment of the vRB/UE mapping to PRB/UE. It is worth noting that the size depends on the common physical channel characteristics (i.e., bandwidth, frequency, etc.).

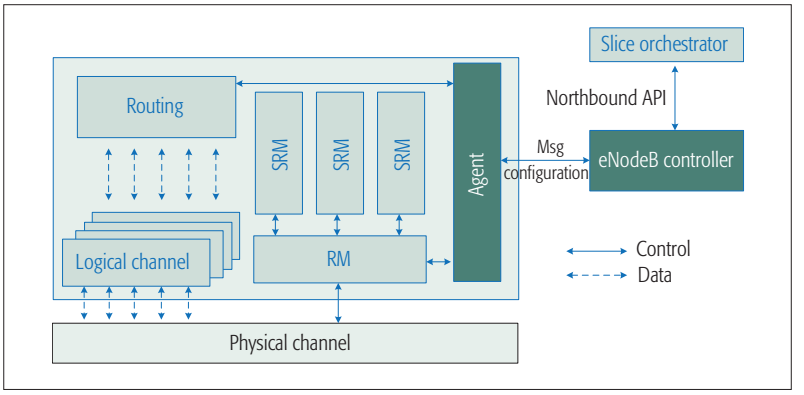

Figure 2. An overview of the eNodeB functions enforcing network slices at the RAN.

The scheduling algorithm is configured by the SO when the slice is instantiated. Depending on the slice type, the SRM functions and the needed inputs (from the RM) will be different.

xMMB: For this type of slice, the used scheduler could be the popular PF algorithm. In this case, the scheduling algorithm requires as inputs the list of UEs with their workloads waiting to be scheduled and the UE's CQI. The algorithm will produce, per UE, the number of needed vRBs along with the modulation and coding scheme (MCS) to be used per vRB. The proposed scheduling list should be sorted according to UE priority. Indeed, due to resource limitations (caused by physical or SDB limitation), the RM may not schedule the entire list of UEs provided by the SRM. For instance, the priority may be based on the difference in the target throughput for a UE. The higher this value, the higher is the priority of the UE.

uRLLC: For this kind of slice, the used scheduler should consider two important criteria. The first one is the latency, which should be minimized (i.e., use a delay-based scheduler). The second one is the service reliability. To maximize the latter, the MCS to be used by UEs should be very robust to channel errors; that is, robust modulations are favored over high data rate modulations. Therefore, the SRM requires the list of UEs, and for each UE, the latency experienced by the headline packet of the logical channel. The outputs will be the mapping vRB/UE to be scheduled as well the MCS. Note that the SRM also provides the priority of UEs according to the remaining time before reaching the deadline. The lower this value, the higher is the priority of the UE.

mMTC: This type of slice is very special, as it involves more UL traffic than DL traffic. Indeed, since the DL traffic is not very important, a simple scheduler like RR or PF may be used. But for the UL traffic, we may distinguish between periodic update (i.e., an MTC is activated during a predefined time interval) and an event-driven MTC traffic pattern. For the periodic update, we propose to use pre-fixed scheduling (e.g., semi persistent scheduling [SPS]). In this case, the SO should indicate when the MTC will be triggered

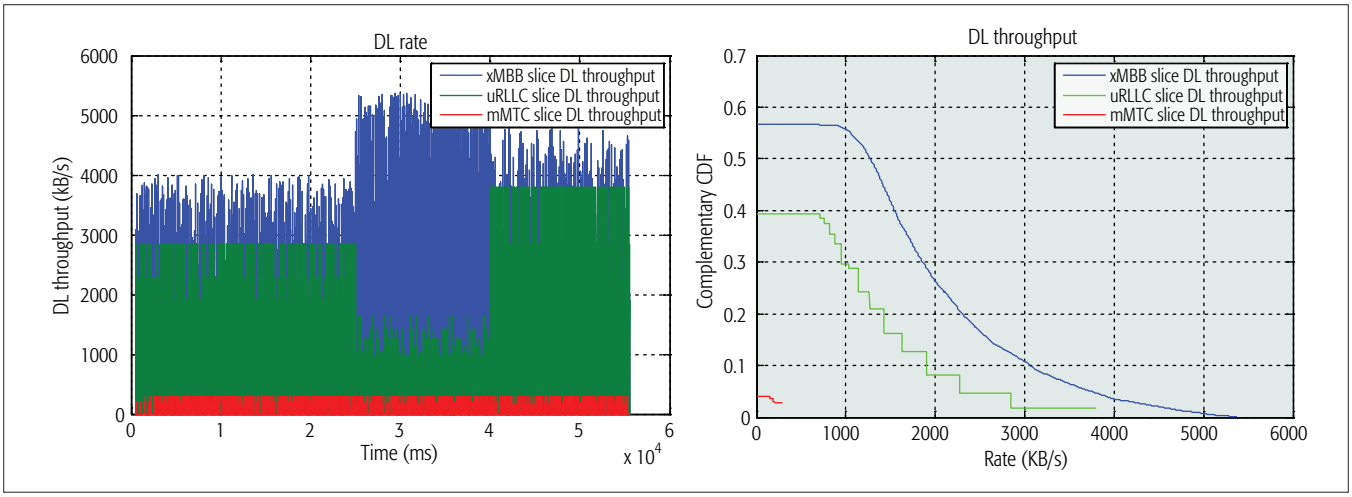

Figure 3. Throughput performance: a) Aggregated throughput per slice; b) Complementary Cumulative Distribution Function (CCDF) of the throughput per slice.

by the application to send reports. Consequently, the RM will dedicate specific resources to the MTC devices during the activation period, avoiding high contention on the channel, particularly during the physical random access channel (PRACH) procedure. It is important to note that a new RRC state needs to be introduced, which allows keeping the radio resources dedicated to an active UE (i.e., C-RNTI, radio bearer, etc.) even if the UE is inactive (i.e., RRC connected, UE inactive). The PRB schedule will be included directly by the RM in the physical downlink control channel (PDCCH). However, in the case of the event-driven MTC traffic pattern, the UL should also be handled by the SRM. In this case, the scheduling algorithm depends on the type of channel access (e.g., regular, random, or contention). For regular channel access, any simple algorithm like RR is sufficient, since this type of application requires neither ultra-low latency communications nor high bandwidth, whereas for content-based access, a group-based PF may be applied.

Thanks to the eNodeB programmability, the scheduler algorithm can be updated on the fly during the slice lifetime. Indeed, the SO may decide to change the scheduling algorithm and the SDB policy if one or both of them are not efficient or not appropriate for the application on top of the slice.

RM Functions

The RM scheduling process begins by sequentially treating each SRM mapping. Two cases may happen when mapping the vRB/UE to PRB/UE. First, if the scheduled PRB for a slice reaches the SDB, the SRM will move to the second slice, and the current slice state will be marked as "paused." Second, if the SDB is not reached and no UE is waiting to be scheduled, the remaining PRB will be kept for other slices (to maximize the multiplexing gain), and the current slice state will be marked as "ok." Indeed, the RM will do a second iteration over the slice marked as "paused" to allocate to them the remaining PRBs that are not used by the slices treated in the first round. Here, a second level of priority among slices could be applied. For instance, uRLLC slices will be favored over xMBB as the former requires low-latency communication. The process will end when all the slices are marked as "end" or no more PRB are available. In any case, the loop is ended after the second iteration in order to ensure that the SRM and RM scheduling can be done during a TTI interval, that is, 1 ms. Indeed, the two-level scheduler's complexity is kept low, as the SRM can be run in parallel, and the RM complexity is proportional to the number of slices.

By having a loop on the resource assignment at the RM level, the proposed algorithm will maximize the resources usage compared to static scheduling. Using the SDB policy will ensure that the scheduling of resources is not fixed throughout the slice life cycle, allowing the SO to manage resources for a slice when needed. Indeed, this would happen if a quality degradation is monitored at the application level, or the slice operator needs more resources to accommodate more UEs (i.e., in the case of an event). The RM is also allowed to preempt resources for an emergency slice. In this condition, the SO indicates that the SRM corresponds to a high-priority slice; hence, the RM will first schedule all UEs of this slice according to the SRM output, without considering the constraints of the SDB policy (i.e., no limit on the used resources). We recall that the scheduling information will be available to all UEs in the PDCCH.

Proof of Concept and Results

The proposed architecture has been deployed using OAI in emulation mode, meaning that the physical channels and UEs are emulated, while the remaining protocol stack operates as specified in 3GPP. In particular, we implemented the two-level scheduler in the eNodeB. The eNodeB is controlled and configured using the FlexRAN protocol introduced in [13]. The SO is run as an application on top of the eNodeB controller in order to configure the SRM instances and the RM. Three slices have been created: xMBB, uRLLC, and mMTC. The xMBB slice deploys a video streaming application, with a bursty and high traffic rate. The uRLLC slice runs medium rate traffic with high variability. The mMTC slice runs a periodic update application using a constant low

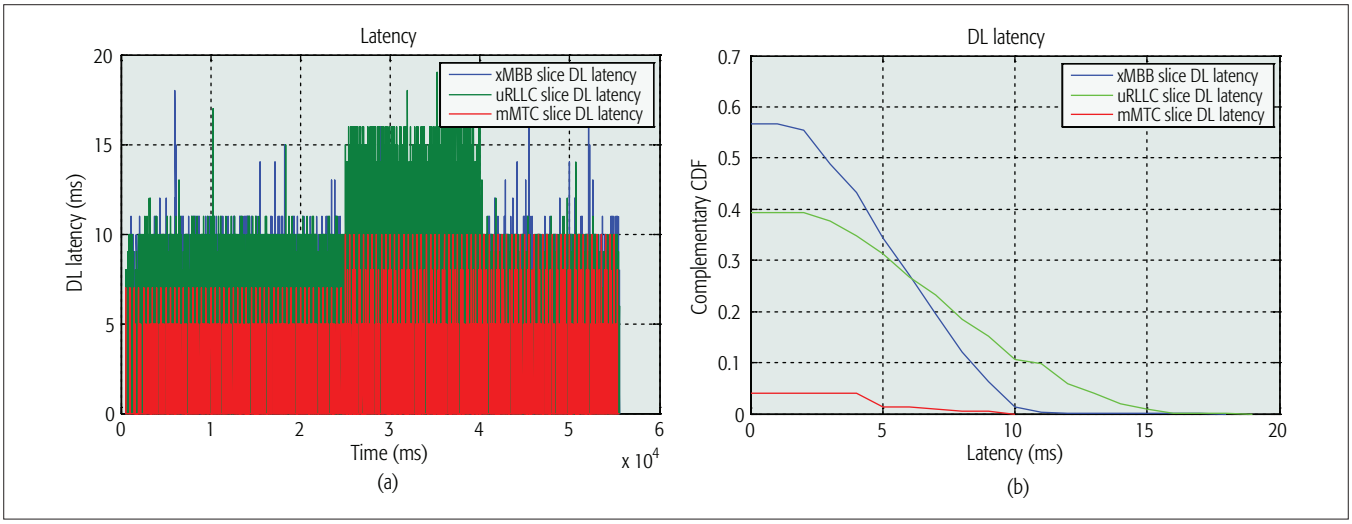

Figure 4. Latency performance: a) DL Latency per slice; b) CCDF of the latency per slice.

bit rate. The SRM of both xMBB and uRLLC slices use a PF scheduler, while SPS is employed for the mMTC slice.

For our experiment, each slice is assigned five UEs, while the percentage of radio resource blocks allocated (i.e., SDB policy) to each slice was dynamically adjusted by the SO. The simulated scenario represents the case where the SO tunes the resource sharing according to the application need, that is, trying three different configurations. To this aim, we divided the simulated scenario into three phases. The first phase, "fair scheduling," is between t = 0 s to t = 25 s, where the SDB of each slice represents 33 percent of the total resources. Between t = 25 s and t = 40 s ("greedy scheduling for xMBB"), the SO changes the configuration by increasing the SDB of xMBB slice to 60 percent, while reducing the SDB of uRLLC and mMTC to 20 percent each. The last phase ("proportional scheduling between uRLLC and xMBB") is between t = 40 s and t = 55 s, wherein the SO changes the SDB for xMBB and uRLLC to 40 and 50 percent, respectively, leaving the SDB of mMTC at 10 percent. As stated before, the aim of this experiment is to study the flexibility and dynamicity of the proposed architecture rather than focusing on the two-level scheduler performance, which is left for future study.

Figure 3 represents the total throughput (DL) obtained by each slice. While Fig. 3a illustrates the aggregated throughput, Fig. 3b represents the CCDF of the throughput. The CCDF plot, for a given throughput value, displays the fraction of throughput greater than that value. From these figures, we remark clearly the adaptability of the proposed scheduler to share the resources. Indeed, when the SO modifies the SDB (at t = 25 s) to give more resources to the xMBB slice, we directly observe the impact on the performance; the xMBB will see an increase of its throughput, while the uRLLC slice a reduction of its throughput. The same behavior is seen when the SDB is changed at t = 40 s. Further, we observe that the xMBB slice always obtains the highest throughput, thus satisfying its requirement. Last, we argue that the uRLLC slice achieves less throughput by the fact that the used MCSs are rather more robust than throughput-efficient.

Knowing the importance of the latency performance, particularly for the uRLLC slice, we draw in Fig. 4 the latency experienced by each slice in the DL. Figure 4a indicates the instantaneous latency, and Fig. 4b shows the CCDF of the latency for each slices. Like the throughput case, we observe without ambiguity the impact of the SDB policy on the slices' performance; particularly on the uRLLC slice, which experiences higher latency when the SO considerably reduces its SDB (i.e., from 33 to 20 percent) and lower latency otherwise (i.e., for 20 to 50 percent). Indeed, reducing the SDB value impacts the UEs' latency as they experience fewer transmission opportunities. Obviously, reducing the SDB will lead to reducing the number of PRBs affected to a slice.

Conclusion

In this article, we have proposed a network slicing architecture featuring RAN abstraction. The proposed architecture relies on the dedicated core network principle to separate the traffic toward the appropriate CN, and uses a two-level scheduler to abstract and share the network resources among slices. Moreover, the proposed architecture enables flexibility and dynamicity, using the FlexRAN concept, to enforce network slicing in the RAN, and adapt the resources allocation policy according to the slice needs. According to the obtained results via the PoC, two important facts should be highlighted:

- Correctly tuning the SDB will help to support the heterogeneous needs of a slice, while ensuring efficient and fair resources sharing among slices.
- Using admission control at the SO will allow regulation of the number of admitted slices, hence enforcing the negotiated SLA for each slice.

Our future work will focus on studying the two-level scheduling process in more detail, in terms of scalability, stability, and performance.

Acknowledgment

Research and development leading to these results have been partially funded by the European Framework Programme under H2020 grant agreement no. 723172 for the 5G!Pagoda project

and grant agreement no. 671639 for the COHERENT project

References

[1] 3GPP, TR 22.891, "Feasibility Study on New Services and Markets Technology Enablers Stage 1," Release 14.
[2] K. Samdanis, X. Costa-Perez, and V. Sciancalepore, "From Network Sharing to Multi-Tenancy: The 5G Network Slice Broker," *IEEE Commun. Mag.*, vol. 54, no. 7, July 2016.
[3] B. Sayadi *et al.*, "SDN for 5G Mobile Networks: NORMA Perspective," *Proc. CrownCom 2016*, *Lecture Notes of the Inst. for Comp. Sci., Social Informatics, and Telecommun. Eng.*, vol. 172, Springer.
[4] X. Zhou *et al.*, "Network Slicing as a Service: Enabling Enterprises' Own Software-Defined Cellular Networks," *IEEE Commun. Mag.*, vol. 54, no. 7, July 2016.
[5] "5G PPP 5G Architecture," 5G PPP Architecture Working Group, white paper, https://5g-ppp.eu/wp-content/uploads/2014/02/5G-PPP-5G-Architecture-WP-July-2016.pdf, accessed 7 Mar. 2017.
[6] 3GPP, TR 23.711, "Enhancement of Dedicated Core Networks Selection Mechanism," Release 14.
[7] 3GPP, TS 23.251, "Network Sharing: Architecture and Functional Description," Release 10.
[8] T. Guo and R. Arnott, "Active LTE RAN Sharing with Partial Resource Reservation," *Proc. IEEE VTC-Fall*, Las Vegas, NV, Sept. 2013 pp. 1–5.
[9] X. Costa-Perez *et al.*, "Radio Access Network Virtualization for Future Mobile Carrier Networks," *IEEE Commun. Mag.*, vol. 51, no. 7, July 2013.
[10] J. He and W. Song, "AppRAN: Application-Oriented Radio Access Network Sharing in Mobile Networks," *Proc. IEEE ICC*, London, U.K., June 2015, pp. 3788–94.
[11] S. Katti and L. Li, "RadioVisor: A Slicing Plane for Radio Access Networks," *Proc. ACM 3rd Wksp. Hot Topics in Software Defined Networking*, Chicago, IL, Aug. 2014, pp. 237–38.
[12] 3GPP, TR 23.799, "Study on Architecture for Next Generation System," Release 14.
[13] X. Foukas *et al.*, "FlexRAN: A Flexible and Programmable Platform for Software-Defined Radio Access Networks," *Proc. 12th ACM Int'l. Conf. Emerging Networking Experiments and Technologies*, Irvine, CA, Dec. 2016, pp. 427–41.

Biographies

ADLEN KSENTINI received his M.Sc. degree in telecommunication and multimedia networking from the University of Versailles Saint-Quentin-en-Yvelines, and his Ph.D. degree in computer science from the University of Cergy-Pontoise in 2005, with a dissertation on QoS provisioning in IEEE 802.11-based networks. From 2006 to 2016, he worked at the University of Rennes 1 as an assistant professor. During this period, he was a member of the Dionysos Team with INRIA, Rennes. Since March 2016, he has been working as an assistant professor in the Communication Systems Department of Eurecom. He has been involved in several national and European projects on QoS and QoE support in future wireless, network virtualization, cloud networking, and mobile networks. He has co-authored over 100 technical journal and international conference papers. He received the best paper award from IEEE IWCMC 2016, IEEE ICC 2012, and ACM MSWiM 2005.

NAVID NIKAEIN is an assistant professor in the Communication System Department at Eurecom since 2009. He received his Ph.D. degree in communication systems from the Swiss Federal Institute of Technology in 2003. Currently, he is leading a research group focusing on experimental system research related to wireless systems and networking. Broadly, his research interests include wireless access and networking protocols (4G/5G), cloud-native and programmable mobile networks (SDN, NFV, MEC), and real-time RF prototyping and emulation/simulation.