Towards enforcing Network Slicing on RAN: Flexibility and Resources abstraction

Adlen Ksentini, Navid Nikaein

Communication systems department
EURECOM Sophia Antipolis, Biot, France
firstname.lastname@eurecom.fr

Abstract – **Knowing the variety of services and applications to be supported in the upcoming 5G systems, the current "one fits all" network architecture is no more efficient. Indeed, each 5G service may require different needs in term 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 Software Defined Networking (SDN), Network Function Virtualization (NFV) and cloud computing as key enablers. The proposed architectures focus principally on the process of instantiating and deploying the Network Slices, while ignoring how they are enforced in the mobile network. Whilst several techniques of slicing the network infrastructure exist, slicing the Radio Access Network (RAN) is still challenging. In this paper, 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 Dedicated Core Network (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 OpenAirInterface (OAI) to derive key performance results; focusing on the flexibility and dynamicity of the proposed architecture to share the RAN resources among Slices.**

Keywords: Network Slicing, 5G, Flexible RAN, Scheduling, Programmable Networks

1. Introduction

Network Slicing is definitely one of the key enablers of the upcoming 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, etc.) and Internet of Things (IoT) services. Besides human oriented devices (i.e. smart-phone and tablet), 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, 3GPP SA2[1] and RAN3[2] groups are building technical specifications to integrate Network Slicing in the upcoming 3GPP standards. Other standardization bodies, like ITU-T through the IMT 2020[3] group and NGMN[4], are studying the requirements and architectures that will enable Network Slicing in 5G. Meanwhile, many 5G initiatives and projects (e.g. 5GPPP[5] European program) have crossed the border by including Network Slicing in their first outputs. Particularly, a global commitment has been done on the definition of Network Slice categories, wherein each 5G service may fall: (i) Extreme Mobile Broadband (xMBB) type, which requires both high data rates and low latency in some area, while reliable broadband access over large areas; (ii) Massive Machine Type Communication (mMTC) type, which needs wireless connectivity for massive deployment of devices; (iii) Ultra-reliable and low-latency communications (uRLLC) or ultra-reliable MTC, which covers all services requiring ultra-low latency connections with 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 Function Virtualization (NFV) and Cloud Computing, will enable building a programmable and flexible network instances (i.e. virtual network) tailored to services' needs.

So far, most of the devised network architectures [2][3][4][5] that enable Network Slicing is based on SDN, NFV and Cloud Computing. These proposals share the same principle, with some difference on the way to instantiate and deploy a network Slice. Mainly, a Global Slice Orchestrator is proposed on top of NFV-like architecture, which translates the Slice provider requests by selecting the appropriate Virtual Network Functions (VNF) (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 administrative or to different 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 on 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 paper are: (i) 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); (ii) 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; (iii) Programmable and Flexible RAN following SDN principles; (iv) a proof of concept (PoC) based on OpenAirInterface (OAI)[6], which is an open-source tool implementation of 3GPP RAN and CN functions.

[1] http://www.3gpp.org/Specifications-groups/sa-plenary/53-sa2-architecture, accessed on: 7/3/2017
[2] http://www.3gpp.org/specifications-groups/ran-plenary, accessed on: 7/3/2017
[3] http://www.itu.int/en/ITU-T/focusgroups/imt-2020/Pages/default.aspx, accessed on: 7/3/2017
[4] https://www.ngmn.org/home.html, accessed on: 7/3/2017
[5] http://5g-ppp.eu, accessed on: 7/3/2017
[6] http://www.openairinterface.org/, accessed on: 7/3/2017

The remaining of this paper is organized as follows. Section 2 presents the concepts of Network Slicing, focusing on the challenges and solutions for RAN slicing. Section 3 details our proposed architecture for enforcing Slices in mobile networks. Section 4 introduces the two-level MAC scheduler, featuring RAN resources abstraction. Section 5 gives some results obtained from the PoC of the architecture built on OAI. Finally, section 6 concludes this paper.

2. Related Work

Network Slicing in mobile network is highly related to Network sharing, particularly to RAN Sharing in 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 Gateway Core Network (GWCN) both the RAN and some elements of the core network 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 RRC/RLC/PDCP/MAC[7] instances, and the physical resources are strictly dedicated to a specific slice, e.g. a percentage of Physical Resource Blocks (PRB)s 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, 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, i.e. 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, like Service Level Agreement (SLA), priority, etc. Whilst this solution exploits statistical scheduling of physical resources, which ensures more scalability and elasticity by report to the dedicated resources model, it may lack the support of strict QoS guarantee for Slices and traffic isolation.

Regarding the literature, many works have addressed the challenge of RAN sharing from two main perspectives: (i) resource sharing among Mobile Virtual Network Operators (MVNO), by modifying the MAC scheduler; (ii) radio resources isolation. For resources sharing, authors in [8] introduce 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 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 term of QoS. Looking to radio resource isolation, RadioVisor [11] represents one of the major works that addressed this issue. RadioVisor aims to share RAN resources, which are represented in a

[7] RRC: Radio Resources Control, RLC: Radio Link Control, MAC: Medium Access Control

three-dimensional grid (radio element index, time slots and frequency slots). 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. It worth noting that these works mainly focused on RAN sharing issues without considering flexibility and dynamicity as required to enable Network Slicing.

3. Network Slicing architecture: enforcing the Slices

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

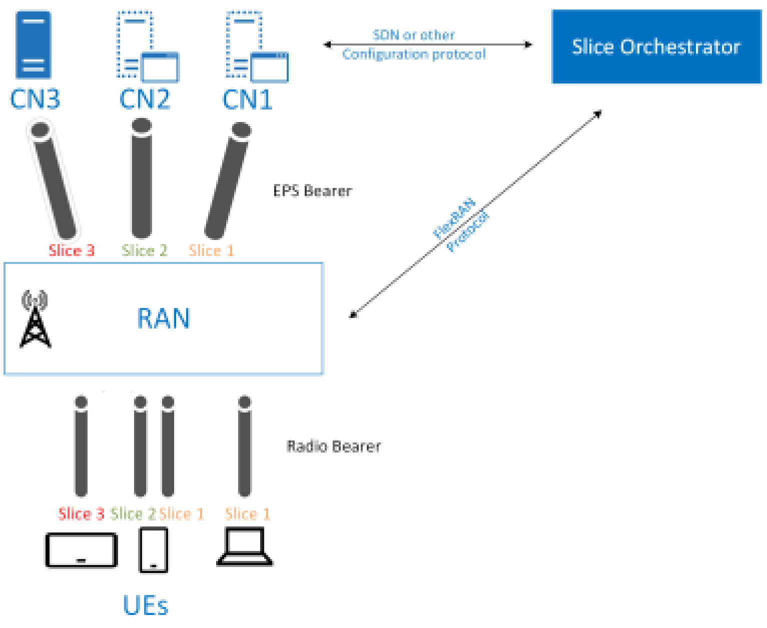


Fig 1. Global overview of the envisioned Network Slicing architecture

A global overview of the envisioned architecture is depicted in Fig. 1. In this figure, we present the network architecture after that the Slice Orchestrator (SO) has instantiated the CN elements, using Physical Network Function (PNF) or VNF. Each CN instance includes a set of VNF or PNFs representing core network functions, such mobility management, authentication as well as data plane forwarding functions (i.e. Gateways to Internet). The CN instances are connected to the shared RAN using the classical S1 interfaces or the newly 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 which will handle the Slice resources to satisfy the required QoS. It is important to note that the Slice ID should indicate the slice Type (for instance 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.

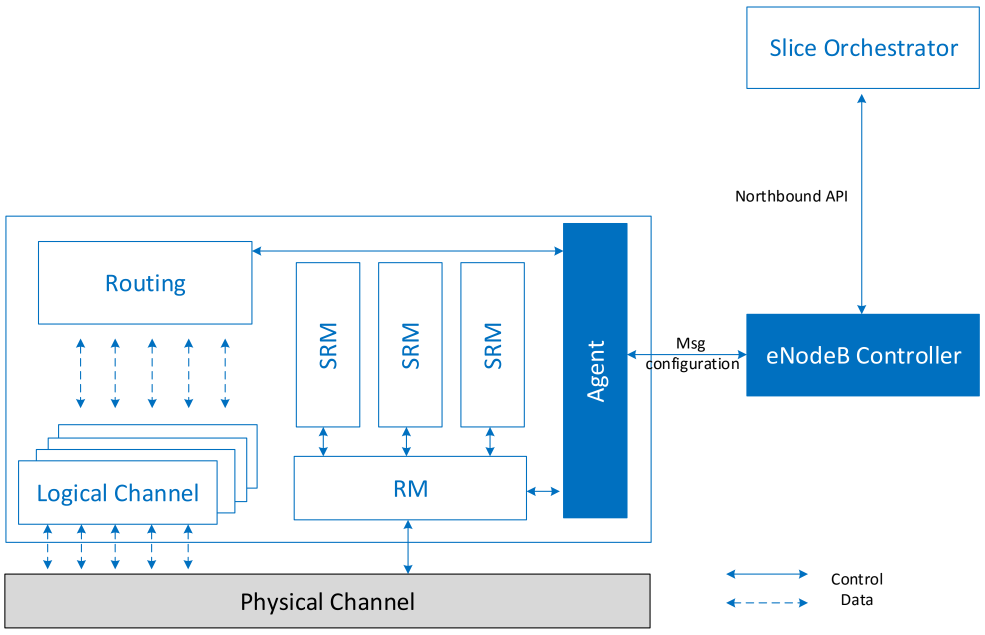


Fig 2. An overview of the eNodeB functions enforcing network slices at the RAN.

Fig. 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 and their mapping to Evolved Packet System (EPS) bearers. The main difference is related to the abstraction (i.e. virtualization) of the physical resource blocks, where an abstraction layer (named 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 (ex. Proportional Fair - PF, Round Robin – RR, Priority-based, 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: (i) the number of packets, per UE and per logical channel, waiting for transmissions, in both directions Uplink and Downlink; (ii) the Channel Quality Indicator (CQI) of each UE; and 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 (vRB). The number of the available vRBs is not limited and could be infinite; i.e. the SRM may schedule all UEs during one cycle. However, the vRBs should be mapped to PRBs, and giving that the number of PRBs is limited and dependent of the physical characteristics (i.e. throughput), not all the UEs will be served during the Transmission Time Interval (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 in charge the communication with the remote eNodeB controller. Indeed, the envisioned architecture assumes that the SO is using Northbound API exposed by a 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 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 send to the eNodeB Agent. The latter will configure the SRM and the RM. For more details on the eNodeB controller, agent, 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, CN instances. Every time a packet is received from the CN, the eNodeB enqueues the packets to the logical channel associated to a UE for a specific Slice; we may see the logical channels as a two dimensions' matrix, where the horizontal axe 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 to virtualize the physical channel (i.e. RAN isolation) for the reasons stated in Section 2. In addition, we argue that creating a specific physical channel for a Slice will require that each Physical layer processes the common I/Q[8] flow (e.g. OFDM[9] 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 absence of beamforming operation, affecting one PRB per UE is enough to ensure RAN traffic isolation.

4. 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 PRB, using as much as possible the SRM scheduling output. 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 jointly scheduling (all in one pass), as the latter is very complex and requires multi-dimensional scheduling. Indeed, this type of scheduling algorithms formulates a multi-objective function that

[8] I/Q: In-phase - Quadrature phase
[9] OFDM: Orthogonal Frequency-Division Multiplexing

should satisfy heterogeneous Slice requirements (e.g. latency and bandwidth), where the optimal solution is usually NP-hard.

Clearly, using 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 to adapt the resources to the workload demand (service elasticity and scalability) when instructed by the SO.

4.1. SRM functions

The SRM main function consists in 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.).

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 UE with their workload waiting to be scheduled and the UE's CQI. The algorithm will produce, per UE, the number of the 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 resources limitation (caused by physical or SDB limitation), the RM may not schedule the entire list of UE provided by the SRM. For instance, the priority may be based on the difference with the target throughput for a UE. Higher is this value, 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; i.e. robust modulations are favored over high data rate modulations. Therefore, the SRM requires the list of UE, and for each UE, the latency experienced by the head-line packet of the logical channel. The outputs will be the mapping vRB/UE to be scheduled as well the MCS. Note that, the SRM provides also the priority of UEs according to the remaining time before reaching the deadline. Lower is this value, higher is the priority of UE.
- mMTC: This type of slice is very special, as it involves more Uplink traffic than Downlink traffic. Indeed, since the DL traffic is not very important, a simple scheduler like RR or PF may be used. But for the uplink traffic, we may distinguish between periodic update (i.e. MTC are activated during a predefined time interval) and event-driven MTC traffic pattern. For the periodic update, we propose to use a pre-fixed scheduling (e.g. Semi Persistent Scheduling – SPS). In this case, the SO should indicate when the MTC will be triggered by the application to send reports. By consequence, 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 a UE active (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 case of event-driven MTC traffic pattern, the UL

also should be handled by the SRM. In this case, the scheduling algorithm depends on the type of channel access (e.g. regular, random or contention access). For regular channel access, any simple algorithm like RR is sufficient, since this type of applications neither requires 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.

4.2. RM functions

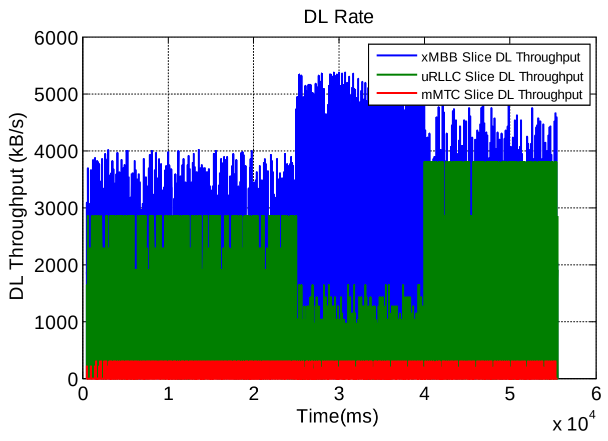
The RM scheduling process begins by sequentially treating each SRM mapping. Two cases may happen when mapping the vRB/UE to PRB/UE. Firstly, if the scheduled PRB for a Slice reaches the SDB, then the SRM will move to the second slice, and the current Slice state will be marked as “paused”. Secondly, if the SDB is not reached and no UEs 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 them the remaining PRB 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 cases, 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, i.e. 1ms. Indeed, the two-level scheduler’s complexity is kept low, as the SRM can be run on 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 maximum the resources usage, compared to a static scheduling. Using the SDB will ensure that the scheduling of resources is not fix throughout the Slice lifecycle; 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. case of an event). The RM is also allowed to pre-empt resources for an Emergency Slice. In this condition, the SO indicates that the SRM corresponds to a high priority Slice, hence the RM will schedule first 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.

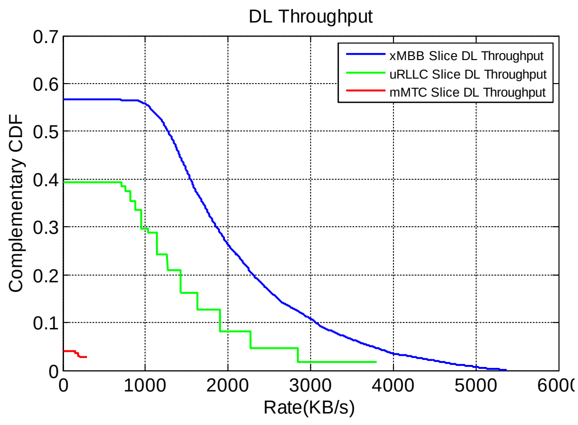
5. Proof of concept and Results

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. Particularly, 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 bit rate. The SRM of both xMBB and uRLLC Slices are using a PF scheduler, while semi persistent scheduling is employed for mMTC Slice.

For our experiment, each Slice is assigned with 5 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 resources sharing according to the application need, i.e. trying three different configurations. To this aim, we divided the simulated scenario into three phases. The first phase "fair scheduling" is between t=0s to t=25s, where the SDB of each Slice represents 33% of the total resources. Between t=25s and t=40s ("greedy scheduling for xMBB"), the SO changes the configuration, by increasing the SDB of xMBB Slice to 60%, while reducing the SDB of uRLLC and mMTC to 20% and 20%, respectively. The last phase ("proportional scheduling between uRLLC and xMBB") is between t=40s and t=55s, wherein the SO changes the SDB for xMBB and uRLLC to 40% and 50%, respectively; leaving the SDB of mMTC to 10%. 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 performances, which is left for future works.



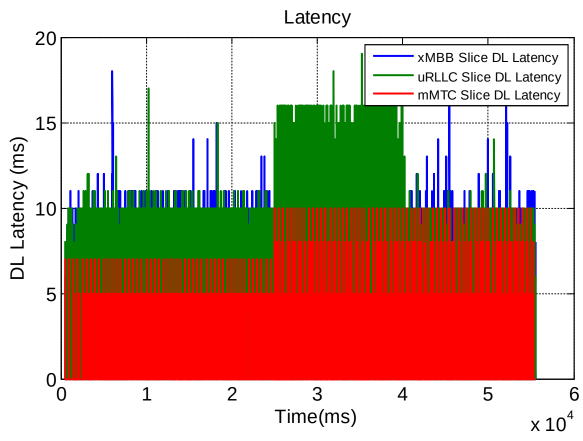
(a) Aggregated throughput per slice

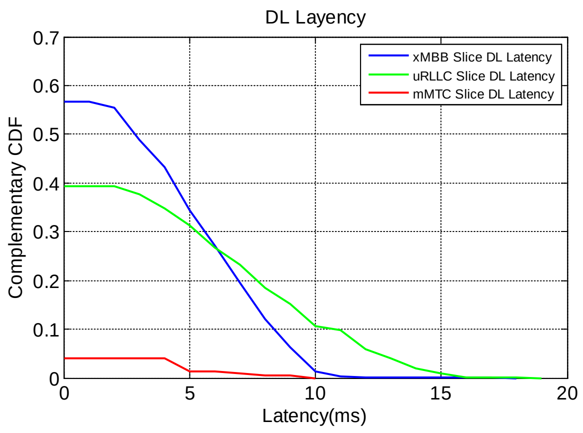


(b) Complementary Cumulative Distribution Function (CCDF) of the throughput per slice

Fig. 3. Throughput performance

Fig.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=25s), to give more resources to the xMBB Slice, we observe directly the impact on the performances; 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=40s. Further, we observe that the xMBB Slice obtains always the highest throughput, satisfying thus its requirement. Lastly, we argue that uRLLC Slice achieves less throughput by the fact that the used MCSs are rather robust than throughput efficient.





(a) DL Latency per slice (b) CCDF of the latency per slice

Fig. 4. Latency performance

Knowing the importance of the latency performances, particularly for uRLLC Slice, we draw in Fig. 4 the latency experienced by each Slice in the DL. Fig. 4(a) indicates the instantaneous latency, and Fig. 4(b) 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 performances; particularly on the uRLLC Slice, which experiences higher latency when the SO reduces considerably its SDB (i.e. from 33% to 20%) and lower latency otherwise (i.e. for 20% to 50%). Indeed, reducing the SDB value impacts the UEs latency as they experience less transmission opportunities. Obviously, reducing the SDB will lead to reduce the number of PRBs affected to a Slice.

6. Conclusion

In this paper, we 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: (i) tuning correctly the SDB will help to support the heterogeneous needs of a Slice, while ensuring an efficient and fair resources sharing among Slices; (ii) using an admission control at the SO will allow to regulate the number of admitted Slices, hence enforcing the negotiated SLA for each Slice. Our future works will focus on studying in more details the two-level scheduling process, in terms of scalability, stability and performances.

7. 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, V. Sciancalepore, "From Network Sharing to Multi-tenancy: The 5G Network Slice Broker", in IEEE Communications Magazine, Vol. 54, Issue 7, July 2016.

[3] B. Sayadi et al., "SDN for 5G Mobile Networks: NORMA perspective", in Proc. of CrownCom 2016. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Vol. 172. Springer, Cham

[4] X. Zhou et al. "Network Slicing as a Service: Enabling Enterprises' Own Software-Defined Cellular Networks", in IEEE Communications Magazine, Vol. 54, Issue. 7, July 2016.

[5] “5G PPP 5G Architecture”, the 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 on: 7 march 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”, in Proc. of the 78th IEEE Vehicular Technology Conference (VTC Fall), Pages 1-5, Las Vegas, Sep. 2013.

[9] X. Costa-Perez et al. “Radio Access Network Virtualization for Future Mobile Carrier Networks”, in IEEE Communications Magazine, Vol. 51, Issue 7, July 2013.

[10] J. He, W. Song, “AppRAN: Application-oriented radio access network sharing in mobile networks”, in Proc. of IEEE International Conference on Communications (ICC), Pages 3788 - 3794, London, UK, June 2015.

[11] S. Katti and L. Li, “RadioVisor: A slicing plane for radio access networks”, in Proc. of ACM of the third workshop on Hot topics in software defined networking (HotSDN), Pages 237-238, Chicago, Illinois, USA, Aug. 2014.

[12] 3GPP, TR 23.799, “Study on Architecture for Next Generation System”, release 14.

[13] X. Foukas, N. Nikaein et al. “FlexRAN: A flexible and programmable Platform for Software-Defined Radio Access Networks”, in Proc. of the 12th ACM International on Conference on emerging Networking EXperiments and Technologies (CoNext), Pages 427-441, Irvine, California, USA, Dec. 2016.