POSENS: A Practical Open Source Solution for End-to-End Network Slicing

Gines Garcia-Aviles, Marco Gramaglia, Pablo Serrano, and Albert Banchs

Abstract

Network slicing represents a new paradigm to operate mobile networks. With network slicing, the underlying infrastructure is "sliced" into logically separate networks that can be customized to the specific needs of their tenant. Hands-on experiments on this technology are essential to understand its benefits and limits, and to validate the design and deployment choices. While some network slicing prototypes have been built for RANs, leveraging on the wide availability of radio hardware and open source software, there is currently no open source suite for end-to-end network slicing available to the research community. In this article we fill this gap by developing an end-to-end network slicing protocol stack, `POSENS`, which relies on a slice-aware shared RAN solution. We design the required algorithms and protocols, and provide a full implementation leveraging on state-of-the-art software components. We validate the effectiveness of `POSENS` in achieving tenant isolation and network slices customization, showing that no price in performance is paid toward this end. We believe that our tool will prove very useful to researchers and practitioners working on this novel architectural paradigm.

Introduction

Fifth generation (5G) networks will change the way in which cellular connectivity is provided. High data rates (50+ Mb/s), extensive coverage (10+ Tb/s/km^2), and low latencies (< 5 ms) are just a few of the target key performance indicators (KPIs) to be fulfilled by the next generation mobile networks [1]. However, not all services are going to require these KPIs, as different applications will have different requirements. To efficiently provide services that meet these requirements, one key enabling technology is *network slicing* [2].

A *network slice* consists of a set of resources assigned to a *tenant* to provide a specific *service*.[1] Those resources are both network resources (e.g., spectrum, link capacities) and cloud resources (i.e., the infrastructure required to run the virtual network functions, VNFs). Tenants could be mobile network operators providing enhanced mobile broadband (eMBB), or third party *verticals* [3] that use a slice specifically tailored to their needs (e.g., ultra-low latency). To satisfy the service requirements of each tenant, a different network slice will be instantiated to provide the corresponding service. This ability to provide highly customizable services over the same shared infrastructure will increase the revenue opportunities and drastically reduce the costs of 5G networking due to the improvements in efficiency.

The advantages of network slicing are clear [4], and there is a wide consensus among the industrial and standardization communities on the need to adopt this technology. However, we lack a thorough experimental validation of its effectiveness, for example, on the gains when using different mechanisms for orchestrations, or under different traffic scenarios. While there are implementations for some of the the enablers for network slicing, to the best of our knowledge there is no solution that implements end-to-end network slicing. More specifically, virtualization is a mature technology that has been extensively used for the wired elements, with technologies such as OpenStack (https://www.openstack.org) and Kubernetes (https://kubernetes.io) for virtual machine and container management, respectively. However, the situation is less mature for the wireless access part, Orion [5] being among the few proposals to implement slicing at the radio access network (RAN) that have been tested in practice [6].

In this article, we fill this gap with the design of `POSENS`, a practical open source solution for end-to-end network slicing that comprises all the elements of an end-to-end mobile network: the user equipment, the RAN, and the core network (CN). `POSENS` implements a "slice-aware shared RAN" solution, enabling effective and efficient sharing of the network resources between different tenants that can independently provide different services.

While `POSENS` is based on state-of-the-art open source solutions for mobile networks, these are substantially extended with the following additional implementations: a multi-slice UE, a slice-aware shared RAN solution, and specific multi-slice management and orchestration (MANO) capabilities, all of which are needed to provide an end-to-end solution for network slicing.

`POSENS` provides a complete solution to instantiate end-to-end slices, using commodity hardware and software defined radio (SDR) boards for development. Our results show that

[1] In this article, we use the terms slice, tenant, and service interchangeably.

Digital Object Identifier:
10.1109/MWC.2018.1800050

Gines Garcia-Aviles, Marco Gramaglia, and Pablo Serrano are with Universidad Carlos III; Albert Banchs is with Universidad Carlos III de Madrid and Institute IMDEA Networks.

1536-1284/18/$25.00 © 2018 IEEE

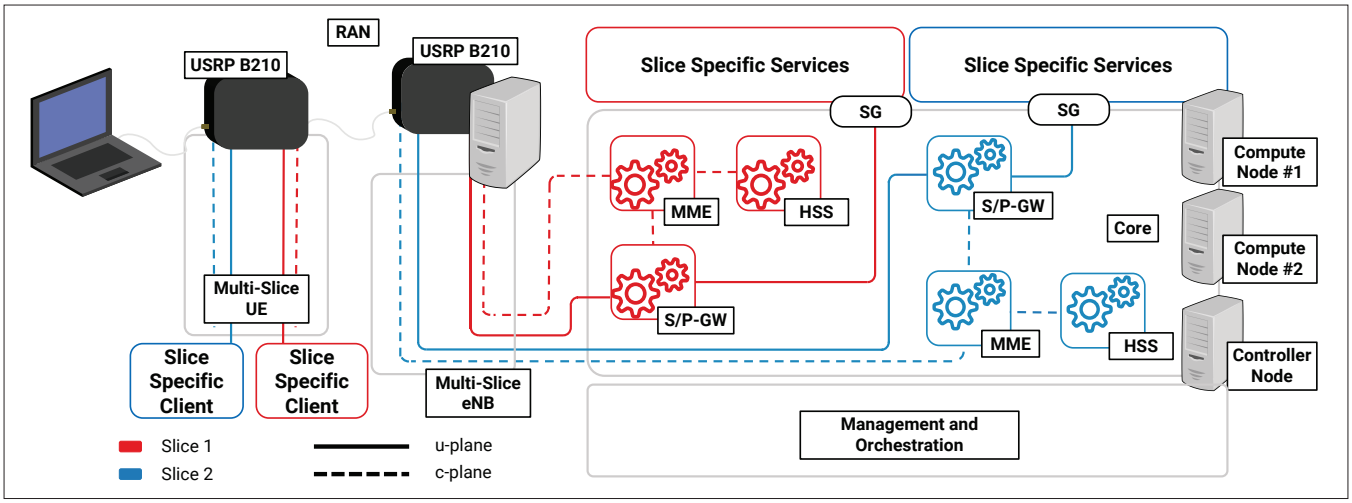

FIGURE 1. A multi-slice network architecture. We also used this blueprint for the evaluation of POSENS.

it supports efficient instantiation of independent, customizable network slices. This open source solution is available (the source code and detailed installation guidelines are available at https://github.com/wnlUC3M/) for developers to put their slicing ideas in practice. This tool will thus support researchers and practitioners experimenting with different algorithms and mechanisms for network slicing. The codebase includes the most important network elements and the MANO part. POSENS can run on any compliant physical hardware, independent of the deployed transport network.

The rest of this article is organized as follows. The following section provides a discussion of the building blocks needed to implement the network slicing concept, including a review of open source software projects in the field of virtualized wireless mobile networking. We then describe the design of POSENS and validate its efficiency in providing isolation across slices. Finally, we conclude the article.

The Path toward End-to-End Network Slicing

Network slicing can be seen as a consequence of the *softwarization* of the protocol stack. We next review the path toward this softwarization, highlighting the complexity involved in sharing RAN resource across tenants due to the tight synchronization required. Then we review different approaches to share the RAN, each one imposing a different trade-off between efficiency and isolation. Finally, we review the most promising software solutions to instantiate a mobile network, identifying the building blocks for the design and implementation of POSENS.

Softwarization of Networks

4G and previous networks are usually composed of monolithic physical boxes, each one providing a very specific functionality and running specialized software on specialized hardware. 5G and future networks will be based on network functions virtualization (NFV) and software defined networking (SDN), which enable flexible network deployments thanks to *network programmability*.[2]

In this new approach, a network is decomposed into three layers:

- Infrastructure, which consists of general-purpose hardware (e.g., cloud computing servers)
- Network, composed of all the networking functions, virtual (VNFs) or physical (PNFs)
- Management and orchestration, that extends the legacy management layer (e.g., the element managers defined by the Third Generation Partnership Project, 3GPP) to support the instantiation and orchestration of network functions.

This approach is represented in Fig. 1, which illustrates one configuration of the testbed that we use to validate POSENS, where the same user equipment (UE, in POSENS a "multi-slice UE") runs two independent slices (blue and red) over the same set of physical resources. The infrastructure layer is composed of a laptop, two SDR cards (USRP B210 boards) that provide the RF front-end, and a small set of server nodes. The network layer runs over this infrastructure, and is composed of the required network functions such as the RAN, home subscriber server (HSS), and serving/packet gateway (S/P-GW).[3] Finally, the MANO also runs over the same infrastructure, and is in charge of instantiating and connecting the networking functions composing the slices.

To support the above vision, data from different slices (and not necessarily from different UEs) has to be (de)multiplexed over a set of shared resources. That is, the mobile network protocol stack has to be divided into VNFs that explicitly belong to one tenant (i.e., usually the CN) and functions that are shared across them (i.e, usually the access network, to lower the deployment costs). This imposes some novel requirements on various elements, particularly on the RAN functions to support, for example, the existence of multiple CNs, or for the UE to attach to multiple slices at the same time. Allowing UEs to simultaneously access different slices is essential for many scenarios envisioned in 5G, including simultaneous access to services supported by different slices as well as provisioning a service that employs multiple slices. For instance, for Industry 4.0 scenarios, augmented reality devices could connect to an "industrial" slice and to an eMBB slice; for vehicular scenarios, different slices could be used for automated driving and infotainment services. This is in line

[2] In fact, one of the most relevant features of future 5G networks is to reduce the time needed to deploy a new service from 90 minutes to 90 seconds.

[3] In POSENS we use LTE/EPC VNFs as they are currently the only open source option available. However, POSENS may easily integrate other 5G VNFs, especially the ones related to the CN.

This multi-slice support requires that traffic from different tenants has to be handled over the same spectrum, which makes it more complex to have dedicated/customized RANs for different slices. In what follows, we discuss how to perform RAN slicing from an architectural point of view.

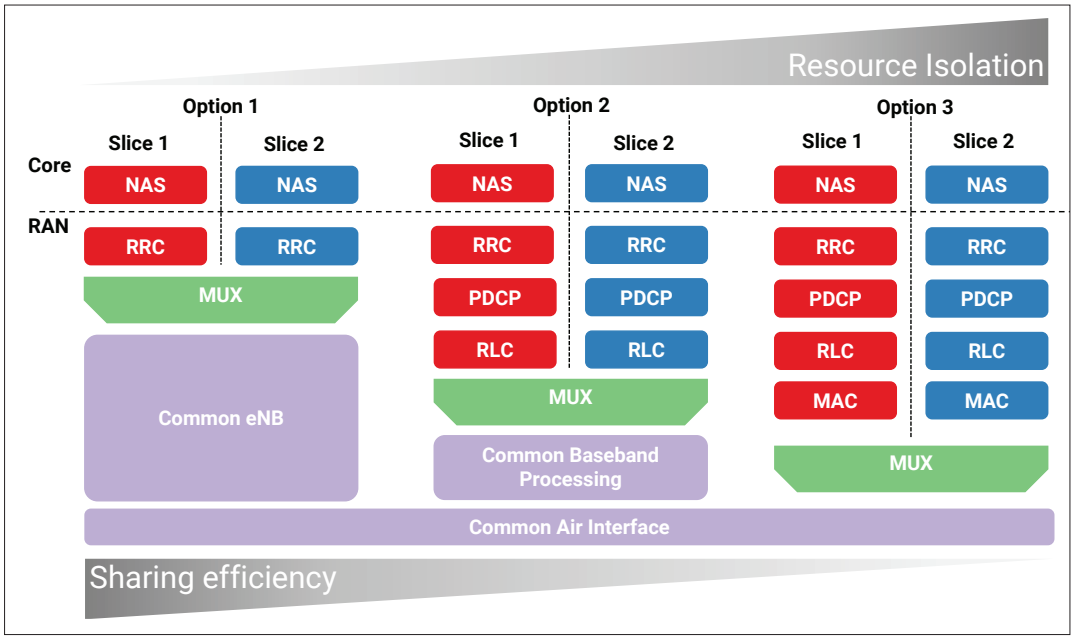

FIGURE 2. Different RAN slicing options.

with the 3GPP SA2 standardization work, which envisions a 5G CN that can attach to up to eight network slice instances at the same time. This multi-slice support requires that traffic from different tenants has to be handled over the same spectrum, which makes it more complex to have dedicated/customized RANs for different slices. In what follows, we discuss how to perform RAN slicing from an architectural point of view.

Addressing RAN Slicing

We next present the three architectural options that have been proposed in the literature [4] for RAN network slicing. These options are presented in order of increasing depth in Fig. 2, where the deeper the slicing (the MUX block represents this depth), the fewer the functions shared by different tenants.

The leftmost option (Option 1) is the so-called *slice-aware shared RAN*, which basically consists of sharing the complete RAN, and then each tenant is responsible for its CN. With this option, the same UE can use different slices, and therefore connect to different CNs. This solution, which can be considered as the "basic" solution to support network slicing, provides relatively little isolation across tenants, but also leads to the highest potential gains in terms of efficiency. This solution can be related to some current proposals such as 3GPP LTE eDECOR [7], introduced to support the instantiation of dedicated CNs. However, eDECOR requires introducing changes to the CN and new signaling messages for the connection setup (something that `POSENS` does not). We also remark that the 3GPP RAN3 Working Group [8] is considering a functional split performed at this level. This approach nicely fits with the shared RAN slicing option, in which multiple network slices are handled by a centralized unit. With this option the RAN is shared to a large extent by different slices, and the core instances are completely independent among tenants, allowing per-tenant configuration, orchestration and (cloud) resource assignment.

The central option in the figure (Option 2) is the *slice-specific radio bearer* configuration. With this option, the slicing goes deeper in the network stack, and basically only cell-specific functionality is shared, that is, the physical (PHY) and medium access control (MAC) layers in the user plane, and the radio resource control (RRC) in the control plane. This configuration increases the resource isolation between tenants, at the price of higher complexity at the MAC layer (e.g., to fully exploit this resource isolation, slice-aware scheduling algorithms are required).

Finally, the last option (Option 3) is the so-called *slice-specific RAN*. In this case, only the air interface is shared among network slices, while all the other functionality is instantiated specifically for each tenant. This configuration provides the maximum degree of freedom, given that each network slice can be customized down to the PHY layer. However, this option also requires tight synchronization between the multi-tenancy policies implemented by a common part and the per-slice (dedicated) implementation.

This option could be particularly useful in scenarios where different radio access technologies coexist within the same shared spectrum (e.g., 4G and 5G). Since it may be very challenging to dynamically reallocate spectrum resources at a fine time granularity, this option may potentially harm resource utilization and limit the potential multiplexing gains.

The above options can be regarded as a roadmap to enable a fully configurable protocol stack to support any network slicing option, where each option presents a different trade-off between efficiency, isolation, and complexity. For the first release of `POSENS`, we decided to implement Option 1, which can bring the maximum efficiency gains and provides end-to-end slicing, thus providing researchers with a tool to experiment with different algorithms and mechanisms. Although Options 2 and 3 provide a higher degree of isolation between slices, Option 1 already enables key features without requiring

[4] While Options 2 and 3 require very tight synchronization among slices, this is not an issue for Option 1 since it employs a conventional RAN stack that already provides the required synchronization.

Work	Base software	Main purpose	Main feature	Limitations	Open source
Mendes *et al.* [12]	srsLTE	RAN slicing	Multiple, per-tenant, eNB virtualization	Implementation only up to MAC layer.	No
Chang *et al.* [13]	OAI	RAN slicing	Thorough evaluation of slices utilization	RAN slicing only.	No
Foukas *et al.* [14]	OAI	RAN slicing	SDN-based RAN slicing	It does not include a core.	Download upon request
Foukas *et al.* [5]	OAI	End-to-end slicing	Core network handling multiple slices	Single slice UE only.	No
POSENS	srsLTE	End-to-end slicing	Slice-aware shared RAN.	One RAN split available.	Yes

TABLE 1. Recent software contributions in network slicing.

the complexity of more advanced RAN schema.[4] More specifically, this option readily supports experimentation on fundamental research items in 5G, such as per-tenant service function chaining (SFC), as the network slices flows go through chains that contain instantiations of different VNFs, or per-tenant orchestration, as different tenants can implement their own MANO using their preferred software on their cloud, thus enforcing service-specific MANO policies regardless of other tenants' ones.

Software Building Blocks

There are several recent initiatives to prototype mobile networks in software, with most solutions building on the GNU Radio development suite and the Ettus Research USRP SDR platforms, and running on standard Linux-based computing equipment (Intel x86 PC architectures).[5] We next provide a short review of the current ecosystem of open solutions.

Concerning the RAN part, three of the most popular software solutions to run LTE over SDR are Eurecom's OpenAirInterface (OAI) (http://www.openairinterface.org/), openLTE (http://openlte.sourceforge.net/), and srsLTE (https://github.com/srsLTE/srsLTE). OAI [9] provides an implementation of a subset of LTE Release 10 elements, including the UE and the eNB. Its performance is considered relatively good, although is also acknowledged that the code structure is complex and difficult to customize. It is also worth mentioning that the eNB and UE RAN are licensed under a specific OAI Public License.

openLTE is an open source project providing an implementation of LTE specifications, which includes a C library, Octave code for testing downlink (DL) and uplink (UL) physical random access channel (PRACH) functionalities, GNU Radio applications for DL functionalities, both simulated and using hardware platforms, and a simple implementation of an eNB using USRP. While its code is considered relatively well organized and documented, resulting in it being easier to modify, it does not provide a UE, and many features are still unstable or under development.

Finally, the srsLTE [10] open source project provides a platform for LTE Release 8 experimentation, designed for maximum modularity. The RAN part provides a complete UE application and a complete eNodeB application. The project is more recent than OAI, and in general the source code is considered easy to customize, although it also consumes more CPU resources than the other alternatives. The code is provided under an AGPL v3.0 license.

Given that our aim is to develop a solution to validate end-to-end slicing, and not efficient software to put into production, code modularity and reusability are determinant factors when selecting a platform, and therefore we decided to design our solution as an extension of the srsUE and the srsENB (the applications for the UE and eNodeB, respectively).

We were convinced by the srsUE source code availability, as having a stable UE software implementation is beneficial for several reasons; for example, it supports the development of multi-slice inside the UE, and speeds up the deployment and testing of new orchestration solutions.

Concerning the CN, apart from commercial solutions such as OpenEPC (https://www.openepc.com) (also supporting shared source code licensing), two of the most relevant solutions are the ones associated with the initiatives mentioned above. First, srsLTE has very recently released srsEPC, a lightweight CN implementation, including the mobility management entity (MME), HSS), P-GW, and S-GW, under the same license. Second, OAI also provides the same elements for a basic EPC solution, in this case released under a standard Apache v2.0 license. Given that when we started our work only the latter was available, we used OAI CN as the software solution for the CN.

One additional advantage of our POSENS, in contrast with eDECOR, is interoperability: our solution works with any implementation supporting the S1AP protocol (we confirmed that POSENS is compatible with different different commercial EPCs, whose names we cannot disclose due to confidentiality agreements).

Finally, MANO has received a lot of attention from both the open source community and enterprises [11]. There is a wide range of full-fledged MANO tools, such as Open Baton (https://openbaton.github.io/), Open-O (https://wiki.open-o.org/), and OSM (https://osm.etsi.org/), that provide the required functionalities related to the VNF life cycle management, including their scaling on a virtual infrastructure. They rely on a virtual infrastructure manager (VIM), software that is more mature, as it has already been employed in production cloud computing environments for many years. Among VIMs, we can list solutions such as OpenStack (https://www.openstack.org). However, as key required features such as per-tenant orchestration are not available with existing open source solutions, we decided to implement POSENS MANO using dedicated software that directly leverages on the VIM APIs.

We finish this section by reviewing state-of-the-art solutions on network slicing that have

[5] We note that there are complete commercial products such as the Amari LTE 100 (a fully software-based LTE BS solution), but their closed licenses makes them unsuitable for research activities.

The UE plays a fundamental role in the network slice selection procedure. One slice performs a full radio configuration of all the RAN layers (including PHY and MAC), while the other one relies on the RRC configuration parameters set by the first slice and prepares the PDCP entities and the RLC channel managers.

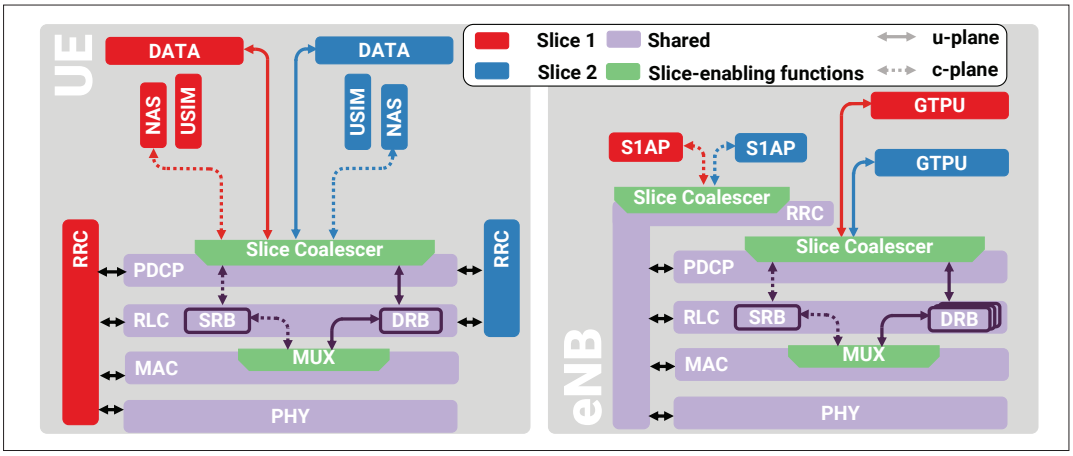

FIGURE 3. Design of POSENS: changes introduced at the UE and the eNodeB.

recently appeared, which we list in Table 1. To the best of our knowledge, POSENS is the most complete solution as it includes an open source, end-to-end, network-slicing-aware mobile network stack that also includes a MANO framework. Other solutions either consider the RAN only [12, 13, 14] or neglect the UE role [5]. Furthermore, POSENS is the only completely open source solution that is readily available in a public repository (GitHub).

Design of POSENS

POSENS provides an implementation of all the modules needed for an end-to-end network slicing-aware mobile network. This includes elements belonging to all the realms of a mobile network (UE, RAN, and CN), plus an orchestration framework. Still, the most important enabler of an end-to-end network slicing setup is RAN slicing.

In the following, we describe the design of our solution to support RAN slicing. This solution consists of introducing a number of changes and new modules to the srsLTE UE and eNB implementations. The resulting software architecture, for the case of two slices, is illustrated in Fig. 3, where each slice is depicted with a different color (we consider the case of two slices for simplicity, but the software can easily be extended to support more slices). We also assume for simplicity that each slice is associated with a different tenant.

As discussed above, in our first release of POSENS we decided to implement Option 1 for RAN slicing, where slices are multiplexed and demultiplexed at the PDCP layer. This option has the additional advantage of requiring fewer changes in the eNB software implementation, which is the main cause of instabilities in an SDR-based testbed. The cornerstones of the solution are the "slice coalescer" modules, located at the PDCP layer and above. These modules forward the control and data layer information for each slice over the common communication channel. Another key feature of our implementation is that each slice at the UE has its own RRC module, and does not require any additional functionality inside the CN. Conversely, at the eNB there is only one RRC module, as with its default behavior it is capable of managing multiple non-access stratum (NAS) from various users simultaneously. In what follows, we provide a more detailed description of the enhancements required by our solution by describing the behavior of the UE and the eNB.

User Equipment

The UE plays a fundamental role in the network slice selection procedure. As depicted in Fig. 3, one slice performs a full radio configuration of all the RAN layers (including PHY and MAC), while the other one relies on the RRC configuration parameters set by the first slice, and prepares the PDCP entities and the RLC channel managers (Acknowledged mode for the u-plane and Transparent mode for signaling messages).

Once the UE has been powered on, the (unmodified) PHY performs the usual cell search (following the configuration provided within the MIB, SIB0, SIB1, and SIB2 messages) and synchronization. Then the RRC module corresponding to the first slice sets up the initial connection with the eNB by performing the random access (RA) procedure to get an initial UL grant, that is, a valid configuration for PDCP, RLC, MAC, and PHY. This configuration is shared across slices, and therefore subsequent RRC modules (corresponding to other slices) will not request it. This means that the `RRCConnectionSetup` message which arrives during the RA process has to be stored within the PDCP module for subsequent slices to be able to establish their session with the CN.

Following the initial UL grant, the NAS protocol of the first slice establishes a session with the CN, generating an `RRCConnectionSetupComplete` message nesting the initial NAS messages in the same packet. The selection of different slices happens in a sequential fashion: after the first slice RRC has configured the wireless link, the subsequent slices are configured using the reception of an `RRCConnectionSetupComplete` as the triggering event.

That is, upon a `RRCConnectionSetupComplete`, the PDCP sends to the next slice a previously stored `RRCConnectionSetup` message containing the details of the RRC channel. This, in turn, triggers the NAS authentication procedure in the new slice. Each time a slice finishes its NAS configuration, the RRC calls a `slice_configured` function within the PDCP, including the (`slice_id, IP address`) tuple of the slice,

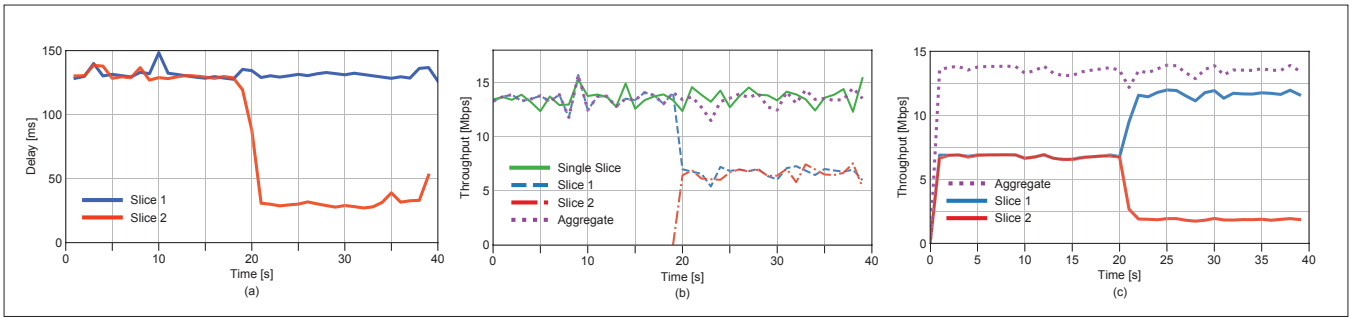

FIGURE 4. POSENS evaluation experiments: a) independence between slices; b) total and per-slice throughput performance; c) independent service function chaining.

which will support the proper forwarding of information within the module (this is only needed for receiving information).

Besides coordinating the c-plane, the slice coalescer in the UE also has to multiplex and demultiplex the u-plane. This is achieved by exploiting the data multiplexer available at the MAC for the uplink: this function demultiplexes the data coming from the lower layers and forwards to the appropriate slice instance at the PDCP on a per-destination IP prefix basis.

eNodeB

The changes in the eNB are the counterpart of the ones introduced in the UE. That is, the slice coalescer handles the multiplexing and demultiplexing of the c- and u-planes. The multi-slice updates in the eNB are less elaborated than the ones in the UE, as the eNB is already capable of handling parallel authentications coming from different UEs. We remark that multiple authentications coming from the same multi-slice UE (e.g., the one described above) can be considered as atomic operations, as they happen sequentially.

This simplifies the required enhancements at the eNB, as there are no concurrent NAS procedures for the same UE running simultaneously, and therefore each one can use the same inner data structure available at the RRC. In this way, we use a flag to mark the slice under configuration, which enables forwarding the control traffic to the corresponding CN via the appropriate S1AP interface.

As with the UE implementation, the u-plane multiplexing happens in the PDCP, following an IP-prefix matching approach, that is, data traffic is forwarded to the right CN by considering the source address of IP packets.

Core and MANO

The main enabler of our slicing solution is RAN slicing. Therefore, to allow easier experimentation with unmodified software solutions, we did not tackle CN VNFs, leveraging on a vanilla implementation such as the one provided in the OAI suite. Similar considerations hold for the MANO part: one of our objectives is to allow the open experimentation of MANO algorithms on top of the POSENS stack.

The MANO of VNFs, a fundamental part of the future 5G networks, is being standardized by the 3GPP SA5 and will leverage on the already consolidated elements of the ETSI NFV MANO architecture [15]. We include in POSENS a baseline implementation of this MANO functionality, which builds on top of an open source VIM (OpenStack), and provides a per-slice orchestration (which is the functional role played by the VNF manager and the NFV orchestrator in the ETSI architecture) through an ad hoc Java software. This implementation leverages directly on the Nova and Neutron APIs to provide a lightweight version of the VNFM-Vi and Or-Vi reference points defined by ETSI.

Evaluation

We next validate and evaluate our solution by deploying a small testbed consisting of one UE implementing two slices, and one eNB connected to two different CNs (one per slice). The testbed architecture is depicted in Fig. 1. The UE runs over an Ettus USRP B210 board connected to an HP OMEN laptop, running Ubuntu Linux 16.04. The eNB runs over another B210 board, connected to a Intel NUC running the same Linux distribution. The TX and RX ports of one B210 board are connected to the RX and TX ports, respectively, of the other board, using coaxial cables with SMC connectors to prevent any interference. To implement the CN, we run two instances of the OAI CN implementation, which contains the MME, HSS, S-GW, and P-GW. The OAI-CN VNFs are packaged in Ubuntu 16.04 VM, running in an OpenStack managed cloud composed of three compute nodes and one controller node.

Before performing the actual validation of POSENS, we first conduct an extensive evaluation of the best RAN (i.e., srsLTE) parameters that lead to the most reliable configuration. To find a good trade-off between RAN performance (in terms of throughput) and stability, we set the channel bandwidth to 10 MHz, and an RX gain of 60 dB for the UE and 60 dB for the eNB. We used the LTE channel 7 (centered around 2600 MHz).

Independence between Slices

We first validate that the slices can run simultaneously and independently, in this way supporting experimentation in scenarios with multiple slices, each one potentially reconfigured in real time. To this aim, the experiment starts with two configured slices, each one implementing a periodic request- response service between the UE and a server. We emulate these servers being relatively far away by introducing an extra delay of

We validated POSENS in a lab deployment, showing how it can obtain per-slice customization without paying a price in terms of performance. Our ultimate goal is to provide a tool that allows researchers and practitioners to experiment with per-tenant MANO algorithms, using the now widely available SDR and cloud hardware commodities.

100 ms via the tc command. Then, after 20 s, the server of the second slice is moved to the eNB, simulating, for example, the use of a multi-access edge computing (MEC)-like solution. We represent the obtained performance in terms of average round-trip times (RTTs) across 10 repetitions in Fig. 4a.

As the figure illustrates, at the beginning of the experiment both slices experience the same RTT of approximately 120 ms, with a few outliers across experiments. The re-allocation of the server in the second slice has an obvious impact on performance, with the RTTs immediately reduced to approximately 20 ms, while the performance with the first slice remains unaltered. With this experiment, we thus confirm that researchers could prototype scenarios where different services are provided with different slices, and each service could be independently modified without altering the others.

Throughput Performance

We next quantitatively assess the performance of our solution to analyze if the overall efficiency is degraded because of the use of slicing, and if the slices are fairly sharing the available resources. To this aim, we start our experiment with both slices configured, but only one (Slice 1) performing a TCP download from a server. Then, after 20~s, another download is performed on the second slice (Slice 2), from a different server. We illustrate the per-slice throughput and the total throughput (Aggregated) averaged over windows of 1 s in Fig. 4b. We also represent in the figure the throughput performance when no slicing is done, that is, both the UE and the eNB use the vanilla version of srsLTE (Single Slice).

The figure illustrates two main results. First, there is practically no difference in total throughput between our implementation and the use of the vanilla version of srsLTE, which confirms the efficiency of the developed solution. Second, when both slices are active, they fairly share the medium, each obtaining approximately 50 percent of the total throughput (we repeated the experiment several times and in all cases the performance was very similar).

Slice Customization and Orchestration

We next show how our solution supports per-slice orchestration and customization of services, as well as the adjustment of the resources that a slice consumes. We demonstrate this capability by modifying in real time the chain of VNFs that build a service. In particular, we insert two additional user plane functions into an operating slice: a traffic shaper and a firewall. Our experiment works as follows. We start with two slices serving downlink traffic to the UE, fairly sharing the channel as illustrated in Fig. 4c. Then, after 20 s, we add into Slice 2 a firewall function to block incoming connections and a traffic shaper function to limit the bandwidth to 2 Mb/s. As the figure illustrates, the effect is immediate, and Slice 1 receives higher throughput. We also confirmed that connections were blocked immediately. This shows that even though the our slicing solution cannot allocate RAN resources directly, it can control the overall resource consumption (including RAN) as long as terminals employ a congestion-aware sending mechanism.

Compatibility with Commercial Equipment

In this section, we confirm that our solution is compatible with commercial equipment. To this end, we perform a connectivity test using a Nexus-5 phone equipped with a Sysmocom programmable SIM card (http://shop.sysmocom.de/products/sysmousim-sjs1). To support this test, we slightly modified the hardware setup, attaching an antenna to the eNB SDR card.

We confirmed that POSENS supports both modified UEs and commercial UEs, namely, slice-aware and slice-unaware UEs. In this way, we support scenarios where several UEs can be attached to the same slice (e.g., eMBB), and only a few UEs, in need of specific services, require the instantiation of a different slice (e.g., an ultra-reliable low-latency communications, URLLC, service). This further extends the applicability of our solution, opening it to a very wide range of testing scenarios.

Conclusions

We have presented POSENS, an open source solution for practical end-to-end network slicing based on slice-aware shared RAN. This tool includes all the software components needed to deploy a multi-slice network setup. POSENS enables the slicing of the RAN as well as the core, which are fundamental building blocks for achieving end-to-end network slicing. We validated POSENS in a lab deployment, showing how it can obtain per-slice customization without paying a price in terms of performance. Our ultimate goal is to provide a tool that allows researchers and practitioners to experiment with per-tenant MANO algorithms using the now widely available SDR and cloud hardware commodities.

Acknowledgments

This work has been performed in the framework of the H2020 projects 5G NORMA (Grant Agreement No. 671584) and 5G-MoNArch (Grant Agreement No. 761445), part of the Fifth Generation Public Private Partnership (5G-PPP) program partially funded by the European Commission within the Horizon 2020 Framework Program.

References

[1] D. Bega *et al.*,"Toward the Network of the Future: From Enabling Technologies to 5G Concepts," *Trans. Emerging Telecommun. Tech.*, 2017.
[2] I. Afolabi *et al.*, "Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies and Solutions," *IEEE Commun. Surveys & Tutorials*, 2018.
[3] 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, pp. 32–39.
[4] P. Rost *et al.*, "Network Slicing to Enable Scalability and Flexibility in 5G Mobile Networks," *IEEE Commun. Mag.*, vol. 55, no. 5, May 2017, pp. 72–79.
[5] X. Foukas, M. K. Marina, and K. Kontovasilis, "Orion: RANSlicing for a Flexible and Cost-Effective Multi-Service Mobile NetworkArchitecture," *Proc. MobiCom '17*, Oct. 2017.
[6] C. Hoymann *et al.*, "LTE Release 14 Outlook," *IEEE Commun. Mag.*, vol. 54, no. 6, June 2016, pp. 44–49.
[7] 3GPP TR 23.711 V14.0.0, Technical Spec. Group Services and System Aspects, "Enhancements of Dedicated Core Networks Selection Mechanism," Rel-14, Sept. 2016.

[8] 3GPP TS 38.300 V15.0.0, "NR; Overall Description; Stage-2,", Rel-15, Jan. 2018.
[9] N. Nikaein *et al.*, "OpenAirInterface: A Flexible Platform for 5G Research," *Computer Commun. Review*, vol 44, 2014, pp. 33–38.
[10] I. Gomez-Miguelez *et al.*, "srsLTE: An Open-Source Platform for LTE Evolution andExperimentation," *ACM WiNTECH 2016*, NY, Oct. 2016.
[11] ETSI Plugtests Report, "1st ETSI NFV Plugtests," Madrid, Spain, Mar. 2017.
[12] J. Mendes *et al.*," Access Multi-Tenancy Through Small Cell Virtualization and Common RF Front-End Sharing," *Proc. ACM WiNTECH*, Oct. 2017.
[13] C. Chang, N. Nikaein, and T. Spyropoulos,"Radio Access Network Resource Slicing for Flexible Service Execution," *Proc. IEEE INFOCOM Wksp. RS-FCN*, Apr. 2018
[14] X. Foukas *et al.*, "FlexRAN: A Flexible and Programmable Platform for Software-Defined Radio Access Networks," *Proc. ACM CoNEXT*, 2016.
[15] ETSI, "Network Functions Virtualisation (NFV); Management and Orchestration," GS NFV-MAN 001, v1.1.1, Dec. 2014.

Biographies

Gines Garcia-Aviles is a Ph.D. candidate at IMDEA Networks Institute, and he is pursuing his Ph.D. at Universidad Carlos III de Madrid (UC3M). His research interests are SDN and NFV technologies for future 5G networks.

Marco Gramaglia is a postdoctoral researcher at UC3M. He received an M.Sc (2009) and a Ph.D (2012) in telematics engineering from the same university and an M.Sc. degree (2009) in computer science engineering from Politecnico di Torino. Before joining UC3M, he held postdoctoral research positions at Istituto Superiore Mario Boella, Torino, Italy, the Institute of Electronics, Computer, and Telecommunications Engineering of the National Research Council of Italy, Torino, and the IMDEA Networks institute, Madrid, Spain. He likes researching on several aspects of mobile networks, ranging from vehicular networking to future 5G networks. He is also interested in big data analytics and end user privacy.

Pablo Serrano [M'09, SM'16] received his degree in telecommunication engineering and his Ph.D. from UC3M in 2002 and 2006, respectively. He has been with the Telematics Department of UC3M since 2002, where he currently holds the position of associate professor. He has over 80 scientific papers in peer-reviewed international journals and conferences. He has served as a Guest Editor for *Computer Networks*, and on the TPCs of a number of conferences and workshops including IEEE INFOCOM and IEEE WoWMoM.

Albert Banchs [M'04-SM'12] received his M.Sc. and Ph.D. degrees from the Polytechnic University of Catalonia in 1997 and 2002, respectively. He is currently a full professor at UC3M, and has a double affiliation as deputy director of the IMDEA Networks institute. Before joining UC3M, he was at ICSI Berkeley in 1997, at Telefonica I+D in 1998, and at NEC Europe Ltd. from 1998 to 2003. He is an Editor of *IEEE Transactions on Wireless Communications* and *IEEE/ACM Transactions on Networking*. His research interests include the performance evaluation and algorithm design in wireless and wired networks.