Orion: RAN Slicing for a Flexible and Cost-Effective Multi-Service Mobile Network Architecture

Citation for published version:
Foukas, X, Marina, MK & Kontovasilis, K 2017, Orion: RAN Slicing for a Flexible and Cost-Effective Multi-Service Mobile Network Architecture. in MobiCom '17 Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking. ACM, pp. 127-140, 23rd Annual International Conference on Mobile Computing and Networking, Snowbird, United States, 16-20 October. DOI: 10.1145/3117811.3117831

Digital Object Identifier (DOI):
10.1145/3117811.3117831

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
MobiCom '17 Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking

Orion: RAN Slicing for a Flexible and Cost-Effective Multi-Service Mobile Network Architecture

Xenofon Foukas
The University of Edinburgh
x.foukas@ed.ac.uk

Mahesh K. Marina
The University of Edinburgh
mahesh@ed.ac.uk

Kimon Kontovasilis
NCSR "Demokritos"
kkont@iit.demokritos.gr

ABSTRACT

Emerging 5G mobile networks are envisioned to become multi-service environments, enabling the dynamic deployment of services with a diverse set of performance requirements, accommodating the needs of mobile network operators, verticals and over-the-top (OTT) service providers. Virtualizing the mobile network in a flexible way is of paramount importance for a cost-effective realization of this vision. While virtualization has been extensively studied in the case of the mobile core, virtualizing the radio access network (RAN) is still at its infancy. In this paper, we present Orion, a novel RAN slicing system that enables the dynamic on-the-fly virtualization of base stations, the flexible customization of slices to meet their respective service needs and which can be used in an end-to-end network slicing setting. Orion guarantees the functional and performance isolation of slices, while allowing for the efficient use of RAN resources among them. We present a concrete prototype implementation of Orion for LTE, with experimental results, considering alternative RAN slicing approaches, indicating its efficiency and highlighting its isolation capabilities. We also present an extension to Orion for accommodating the needs of OTT providers.

KEYWORDS

5G mobile networks; network architecture; RAN slicing; RAN virtualization; abstractions

1 INTRODUCTION

5G is on the horizon. There is an evolutionary dimension to 5G aimed at significantly scaling up and improving the efficiency of mobile networks to meet the demanding requirements such as 1000x increase in capacity via new radio access technologies and spectrum bands (e.g., massive MIMO, millimeter waves). On the other hand, an alternative and complementary *service-oriented* vision for 5G presents a significant opportunity to innovate on the architectural front with several resulting benefits (ease in service creation, network sharing, reduce capex/opex costs, enhance user experience, realize a more energy-efficient mobile network infrastructure, etc.). The idea underlying this service-oriented view is for the network to support a wide range of services, differing significantly in their service requirements and device types (including machine-type devices). For example, ITU [29] identifies three broad classes of 5G services (enhanced mobile broadband, massive machine type communications, ultra-reliable and low latency communications) and several specific services within these three service classes.

As a one-size fits all architecture is unlikely to be suitable for such diverse use cases, realizing the service-oriented 5G vision in a cost-effective manner necessitates a flexible mobile network architecture that can turn the physical infrastructure into multiple logical networks or *slices*, one per service instance. Each slice in such an architecture is an end-to-end virtualized network instance, spanning both the core and radio access network (RAN), and is tailored in terms of resources to meet the requirements of the service in question. Here resources comprise of different types including computing, network, storage, radio, access hardware and virtual network functions (VNFs). Unsurprisingly, most prominent 5G architectural visions embrace slicing [7, 16, 49, 52, 60, 61], building on its earlier success in multi-experiment testbed infrastructures such as PlanetLab [12] and multi-tenant data centers. As virtualization and softwarization (via software-defined networking (SDN) and network function virtualization (NFV)) are key slicing enablers, research prototypes and operational systems using virtualization technologies and SDN/NFV principles have started to appear, especially in the mobile core [9, 19, 30, 42, 46, 57, 58]. In fact, an early form of core slicing called dedicated core networks (DECOR) is already specified by the 3GPP standards [2].

In this paper, our focus is on RAN slicing which refers to the ability to dynamically create and manage virtual RANs, each customized to meet the requirements of an end-to-end service. RAN slicing is a challenging problem that is only starting to receive attention. The key difficulty is that the RAN virtualization and apportionment into different slices should satisfy two key objectives: (1) to ensure slice independence (functional isolation) so that tenants maintain full control of their slices to be able to tailor them to meet the respective service requirements; (2) to flexibly and adaptively share RAN resources (radio, processing, memory, networking), among different slice owners (or tenants), so that the RAN infrastructure is used as efficiently as possible, *without* violating objective (1).

Previous work on RAN slicing, as elaborated in the next section, represents extreme points in the design space and therefore has managed to only partially address the aforementioned objectives. One approach originating in RAN sharing focuses mainly on efficient sharing of radio resources with no support for functional isolation, giving the infrastructure provider full visibility and control over slices [18, 19, 50]. The other approach puts the isolation at the center stage without considering the efficient use of resources [47, 48].

In this paper, we present Orion, which to our knowledge is the first RAN slicing system that provides functional isolation among slices while facilitating efficient sharing of the RAN resources. The design of Orion makes an explicit distinction between the infrastructure

MobiCom '17, October 16–20, 2017, Snowbird, UT, USA.
© 2017 ACM. ISBN 978-1-4503-4916-1/17/10…$15.00
DOI: https://doi.org/10.1145/3117811.3117831

provider and service providers (slice owners), providing mechanisms that allow flexible and adaptive provisioning of resources to slices based on their requirements. Each slice can manipulate its allocated resources in a virtualized form completely independently, and similarly customize its control plane as per the service needs. The Base Station Hypervisor in Orion plays a key role in achieving this by virtualizing the radio resources via a novel set of abstractions introduced in this work and by having the control plane of each slice operating in an isolated container over it. Moreover, Orion's design allows the control planes of slices to be composed flexibly and independently, employing different levels of centralization, effectively leading to a more efficient utilization of the RAN resources and simplifying the coordination of base stations where and when required.

In summary, this paper makes the following contributions:

- (Sections 3 and 4) Present a realizable RAN slicing design in the form of Orion, that enables both the functional isolation among slices, as well as the efficient utilization of the underlying RAN resources via the novel Hypervisor component and by facilitating the flexible composition of the control planes of individual slices.
- (Section 4) Introduce a novel set of abstractions for the virtualization of the radio resources that is applicable to both current (LTE) and future (5G New Radio) RANs.
- (Sections 4, 5 and 6) Provide a concrete implementation of Orion over the OpenAirInterface (OAI) LTE platform [51] along with a detailed experimental evaluation of the various aspects of the system that highlight its performance and scalability as well as its various capabilities compared to the state-of-the-art solutions.
- (Section 7) Present an extended form of Orion that supports over-the-top (OTT) service providers and demonstrating its benefits.

2 RELATED WORK

RAN Slicing. State of the art on RAN slicing can be traced back to the earlier works on active RAN sharing [13, 22, 53, 63, 70]. Broadly speaking, two approaches under the names of Multi-Operator Core Networks (MOCN) and Multi-Operator RAN (MORAN) have been considered. While both approaches imply the use of separate core networks for each participating operator, MOCN allows for spectrum sharing among operators while MORAN requires dedicated spectrum for each. Relatively, more attention has been given in the literature to the MOCN approach (e.g., [26, 35, 41]), which has been standardized for LTE in Release 8. NVS [35] is a representative example. Note that the use of the term virtualization in some of these works is somewhat misleading as it refers only to the UE perceived performance isolation (i.e. throughput) among operators sharing the RAN radio resources and not on the functional isolation and corresponding performance isolation of the slices' virtual network functions in terms of the required computing resources (processing, memory, networking). As indicated at the outset, functional isolation is additionally essential in the RAN slicing context. The major focus of these works is on the design of efficient radio resource scheduling algorithms while considering certain guarantees for operators. The fact that radio resource sharing is also relevant for efficient RAN slicing is reflected in the more recent algorithmic work in this thread [11, 23, 32, 43]. This body of work is complementary to our focus on systems support for RAN slicing. In fact, we employ the NVS scheduling algorithm in our prototype to highlight the efficient radio resource use feature of Orion.

From a systems perspective, RAN sharing oriented slicing (with no functional isolation among slices) has been explored through the use of the FlexRAN software-defined RAN platform [18, 19, 36, 50], which we include in our comparative evaluations. FlexRAN decouples the control from the data plane of base stations using a custom-tailored southbound API and introduces a flexible and programmable control plane. Network slicing is enabled with FlexRAN by programmatically defining the way in which the radio resources need to be allocated among the connected UEs based on the requirements of the slice they belong to. A unified control plane, which is controlled by a single entity (usually the infrastructure provider), is responsible to perform the corresponding control operations. This approach can be limiting, since the capabilities of the slices are fully dependent on the types of control functions that are bundled in the control plane of FlexRAN. In contrast, Orion allows independent and fully customizable control planes for each slice so that slice owners can flexibly introduce their own functionality in the RAN and tailor their slice as per the needs of their service.

To accommodate this need for slice customizability, the other RAN slicing approach taken in the literature seeks full isolation (by running the virtual base station instance of a slice within a Docker container for example) but assumes dedicated radio hardware and spectrum per slice [47, 48], bearing some similarity to the MORAN form of RAN sharing in that resource sharing among slices is limited at best to computing, memory and storage resources. This has the downside of inefficient use of radio resources and foregoing potential statistical multiplexing gains. The work presented in [52] also takes the same approach, although the focus there is on the idea of a network store for VNFs to aid in dynamic network slicing. On a more general note, wireless virtualization overview and position papers like [5, 25, 39, 69] advocate functional isolation, which strengthen the motivation for the approach we take in Orion to have an isolated and customizable control plane for each slice.

In terms of radio resource virtualization, recent works in the domain of RAN slicing [24, 25, 36, 39] have advocated the need for abstractions that decouple the control plane decisions from the physical radio resource grid. However, the abstractions presented in these works are high-level and do not consider the constraints that can be imposed by the physical layer (e.g., frequency-dependencies in scheduling). The abstractions introduced in Orion for radio resource virtualization not only overcome these limitations but are also generic in that they are applicable to both current LTE and future 5G-NR air interface technologies.

Core Slicing. There has been significant progress on mobile core slicing to the point that it is fairly mature and also made its way into 3GPP standards in a basic form under the name of DECOR [2]. Virtualization of core network functions combined with the use of mature virtualization technologies (e.g., KVM [34], LXC [27], Docker [44], VLANs [8, 64]) have led to systems that realize core network slicing [33, 48, 52]. Even the possibility of EPC as a service over the cloud has been explored [66]. Several research proposals leveraging NFV in the core appeared in the recent past, aimed at its optimization for better scalability (e.g., [9]), customization for particular use cases (e.g., [67]) and in general making it more flexible (e.g., [58]).

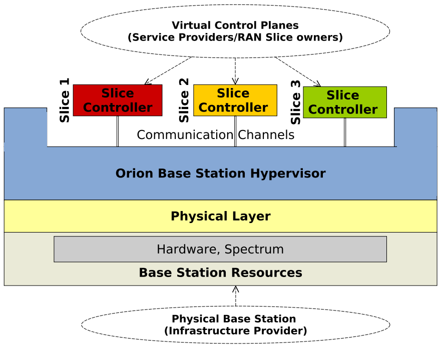


Figure 1: High-level architecture of Orion.

Management and Network Orchestration (MANO). The end-to-end nature of services create the need for a MANO entity responsible for realizing end-to-end slices spanning the core and the RAN as well as their life-cycle management to ensure compliance with their service requirements. This has led to MANO oriented research work [65] and several open-source MANO framework implementations (e.g., OSM [17], ONAP [40]). More pertinent to our work, the authors of Proteus [65] acknowledge the lack of a flexible RAN slicing solution and therefore consider use cases that focus on innovations around the mobile core. Orion fills this void. We present how Orion can be used in an end-to-end slicing context by interfacing with any of the MANO solutions mentioned above.

3 ORION OVERVIEW

The core contribution of this paper, Orion, is a novel RAN slicing system design that is in line with the spirit of network slicing and the needs of a flexible and cost-effective multi-service mobile network architecture – isolation among multiple virtual networks provides the necessary flexibility to customize and control a slice, whereas efficient sharing of the underlying physical infrastructure allows supporting diverse services in a cost-effective manner. Simultaneously being able to satisfy both these concerns of functional isolation between slices and efficient resource use in the context of the RAN is the main, as yet unresolved, challenge addressed by Orion.

Towards this end, Orion's design (Fig. 1) explicitly distinguishes the infrastructure provider from the service providers (the slice owners). The infrastructure provider is the owner of physical base stations, comprising of hardware resources (i.e., radio equipment, processing, memory, and network) and a chunk of spectrum. While it can be realized either via dedicated specialized hardware or in a cloud environment using re-programmable hardware (e.g., C-RAN baseband processing unit and remote radio heads), each physical base station in our model supports a single Radio Access Technology (RAT), meaning that all radio and spectrum resources available at the base station can be exploited through a *shared* physical layer. Note that, for concreteness in the description of Orion's design and implementation, we consider LTE as the underlying RAT and downlink scheduling as a running example throughout.

The Base Station Hypervisor that sits over the physical layer is the heart of Orion's design. It is the component used for managing RAN slices, for ensuring their full isolation – control logic for functional isolation and resources for performance isolation, as well as facilitating efficient sharing of underlying physical resources. Essentially, the Hypervisor binds the individual and isolated slices to the physical infrastructure, providing them with a virtual view of the underlying radio resources via a novel set of abstractions and the data plane state as well as applying their state changes over the physical data plane by mapping virtual to physical resources. The Hypervisor is part of the infrastructure provider's software infrastructure to support RAN slicing. The infrastructure provider is also responsible for admission control.

Service providers (e.g., MVNOs and verticals) in Orion realize their RAN slices through the creation of virtual base stations over the Hypervisor. Each virtual base station is a composition of a virtual control plane, responsible for managing data plane state that is revealed to it by the Hypervisor. The virtual control plane of a slice is effectively a local RAN-level slice controller running as a *separate process*, responsible to tailor the functionality and manage the allocation of resources to mobile devices (UEs) associated with the slice as if it was operating using its own dedicated infrastructure. The virtual control plane is also responsible for implementing the control protocols required for the communication and coordination of the virtual base station with the rest of the mobile infrastructure (e.g., S1 and X2 interfaces in LTE). This means that all operations defined for a given mobile network architecture can be supported by slices (including roaming) so long as the appropriate interfaces and messages are implemented as part of the respective virtual control planes. Note that although we discuss the data plane aspect both in our design and implementation, our primary focus expectedly is on the control plane. Following SDN principles, our design assumes control-data plane separation that is now widely accepted in the mobile networking domain [19, 37, 57, 68].

The communication of the slices' virtual control planes with the Hypervisor is message-based and happens through independent physical/virtual communication channels. This approach allows the deployment of slices either over the same physical machine as their Hypervisor or over separate physical machines. It also gives the flexibility to slice owners to compose their control planes using different levels of centralization to enable coordination based on their services' needs, independently of other slices.

The design of Orion provides strong isolation guarantees while allowing efficient resource sharing. First, since each of the slice controllers is running as a separate process, isolation among controllers in terms of memory and CPU can be achieved by employing well known OS and process virtualization techniques, like virtual machines (e.g., KVM) or containers (e.g., LXD and Docker). Second, the Hypervisor is the sole entity responsible for handling actual radio resources which it distributes among slices after virtualizing them, ensuring isolation from a radio resource perspective. Third, it can internally facilitate efficient resource use via a suitable allocation algorithm that also considers slices SLAs and underlying physical conditions. Additionally, from a UE perspective, the whole slicing operation is transparent, with each slice appearing as a different MVNO as in RAN sharing.

4 SYSTEM DESIGN & IMPLEMENTATION

This section details the key components of Orion: Base Station Hypervisor and Virtual Control Plane of a slice.

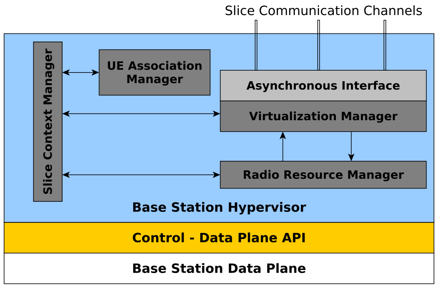


Figure 2: Architecture of the Orion Base Station Hypervisor.

4.1 Base Station Hypervisor

The Base Station Hypervisor (Fig. 2) acts as the intermediary between the data plane of the physical base station and the virtual control planes of slices. The Hypervisor communicates with the data plane via an API to access and modify the data plane state (e.g., obtain signal quality measurements and transmission queue sizes of UEs, apply scheduling decisions, etc.). The Hypervisor is responsible for allocating physical radio resources to slices with respect to their SLAs, transforms them into virtualized resources that are revealed to virtual control planes of slices in an isolated manner. On the other direction, slices use the provided virtual resources for their control plane operations and send their commands (e.g., scheduling decisions) back to the Hypervisor, which then translates them to a physical representation and applies them to the data plane. We now detail the internals of the Hypervisor.

4.1.1 Slice Context Manager. Prior to a slice's creation, the slice owner comes to an agreement on the required service type with the infrastructure provider, subject to admission control. This is formally translated into a *slice service description* that includes two elements:

A Service Level Agreement (SLA) that identifies the service requirements of the slice owner (e.g., average throughput, resource blocks). Orion does not restrict the SLA parameters and instead provides a flexible framework to implement suitably tailored mechanisms for admission control and fine-grained resource allocation.

A list of User Equipment (UE) identifiers required for the mapping of UEs to slices. Any unique identifier provided by the UE during the attachment process could be used.

The Slice Context Manager (Fig. 2) is responsible for the life-cycle management of slices at the base station. Each slice is associated with a context (Table 1) capturing the slice's capabilities and current state. Apart from the slice's SLA, this context keeps track of the active UEs on the slice, stores the configuration of the communication channel between the Hypervisor and the slice's virtual control plane, and all data structures for bookkeeping operations related to the use of physical resources by the slice in the past and present. The Slice Context Manager can obtain the identifiers of the UEs belonging to the slice from the management plane through a lookup function, further explained in Section 4.3.

Upon request for creation of a new slice, the Slice Context Manager does admission control. By checking the number of active slices, their SLAs and activity (through their context), and using an admission control policy adopted by the infrastructure provider (e.g., along the lines of [35]), either the slice is admitted with a corresponding context created, or rejected due to insufficient resources.

Slice Context
Slice id
UE identifier lookup function
Current slice SLA/policy settings
UEs currently connected to the slice
Communication channel configuration to slice local controller
Bookkeeping on usage of physical resources

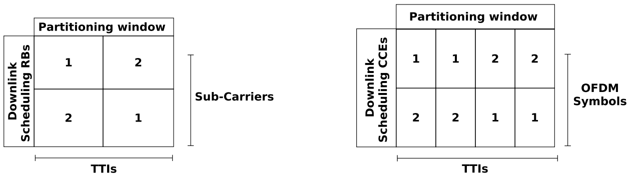
Table 1: Per slice context maintained by the Hypervisor.

4.1.2 Radio Resource Manager. The Radio Resource Manager is responsible for the allocation of physical radio resources among co-located slices in a flexible and efficient manner, while ensuring slice isolation. While this is dependent on the nature of each resource, Orion adopts the principle that any physical radio resource meant for individual UEs can also be allocated among slices in an isolated fashion, because, by definition, they can be quantized and assigned to specific UEs. Such resources include radio resources allocated for downlink/uplink user and control traffic as well as for paging.

As an example, consider downlink LTE scheduling, involving two types of radio resources: (i) Resource Blocks (RB) for transmission of user traffic and (ii) Control Channel Elements (CCE, essentially a set of OFDM symbols) for transmission of the corresponding scheduling decisions. Due to their nature, these resources can be dynamically allocated to individual slices based on various criteria, like their SLAs, the channel conditions experienced by UEs in the slice etc. On the other hand, physical resources used either for the transmission of cell-related information (e.g., broadcast) or in a contention-based manner (e.g., random access) should not be allocated to slices because concurrent modifications from multiple slice control planes could lead to conflicts. Such resources are exclusively managed by the Hypervisor as discussed later in this section.

Orion provides a generic framework for the dynamic allocation of radio resources, which allows the implementation of different radio resource allocation mechanisms to fulfill different types of SLAs. Any allocation algorithm implemented in Orion would have inputs in the form of the physical radio resource grid and the context information of the active slices. The radio resource allocation process occurs periodically once every *allocation window* of duration t. At the beginning of each window, the Radio Resource Manager obtains the bookkeeping information and the SLAs of all the active slices from the Slice Context Manager. Through this information, and with the current cell configuration and allocation mechanism implemented, the Radio Resource Manager decides a splitting of the available radio resources among slices for the upcoming window. For each type of partition-able radio resource, it fills a two dimensional array that expresses the slice assignments of resources through a generic representation of the resource in the time and frequency domain. As an example, Fig. 3 illustrates LTE downlink scheduling for 2 slices. Each column represents a 1ms LTE subframe in Transmission Time Interval (TTI) units and each row the corresponding resource in the frequency domain (RBs in Fig. 3a and OFDM symbols in Fig. 3b). The tabulated numbers correspond to the ids of the slices to which the resources were allocated.

The decision of the Radio Resource Manager is forwarded to the Slice Context Manager to update the context of slices with allocated resources. Note that the Slice Context Manager and Radio



(a) Resource Blocks **(b) Control Channel Elements**

Figure 3: Resource partitioning between 2 slices for the LTE downlink scheduling example.

Resource Manager can be seen to provide the functionality of the 5G network slice broker introduced in the context of the 3GPP network sharing management architecture in [63]. When the slices' control planes make the scheduling decisions based on a virtualized and isolated form of their allocated resources via the Virtualization Manager, the Radio Resource Manager translates and updates the base station data plane state through the control-data plane API, and keeps track of the physical resources used by slices (via context updates) to inform the allocation in upcoming windows.

4.1.3 Virtualization Manager. The Virtualization Manager (Fig. 2) directly interacts with the virtual control plane of slices, maintaining slice isolation in terms of physical network resources. The interaction occurs through dedicated slice communication channels managed by an asynchronous interface. Besides, the Virtualization Manager undertakes two main tasks: (i) presenting an abstract view of radio resources to slices by mapping physical to/from virtual resources at runtime; (ii) presenting a virtual/abstract view of the data plane state to slices. The challenge for this component is to realize these tasks in a manner that does not compromise slice isolation.

Abstracting radio resources To ensure radio resource isolation, the Virtualization Manager, creates a virtualized view of the radio resources tailored to each slice, by obtaining the map of resources assigned by the Radio Resource Manager from the slice contexts. This virtualized view omits all resources not dedicated to a slice, including resources for random access, broadcasting, resources used only by the physical layer (e.g., for reference signals) and those allocated to other slices. It also omits the exact placement of the resources in the resource grid along the frequency dimension and instead reveals an abstract representation to the slices' control planes. On one hand, this allows the control planes of slices to directly and independently view and control the radio resources allocated to them, which can be dynamically re-assigned to different slices over time. On the other hand, by withholding the frequency dimension, potential inference and manipulation of resources allocated to a slice by other slices is prevented, keeping in mind that different tenants could in fact be direct competitors. It should be noted that this inference issue is not present in RAN sharing or in a static radio resources allocation setup. In the RAN sharing case, a single entity (the infrastructure provider) performs all the control operations and thus the exact usage of the resources is 'hidden' from the slice owners, while in the other case, the radio resources are statically assigned to slices, providing no visibility to the resources of other co-located slices.

For the case of downlink/uplink UE data transmissions, one of the main challenges is how to reveal the virtualized radio resources to slices so that they can be allocated to the UEs in the most flexible way and keeping in mind that the slices must be completely unaware of the physical resource grid layout, while at the same time respecting the allocation constraints imposed by the physical layer. Such constraints include group-based allocations, where more than one physical resource block (PRB) must be used as the minimum allocation unit for a UE (e.g., as specified in some types of LTE allocations [4]). Other constraints include frequency dependencies that might arise in the radio resource allocation process, where the use of a certain PRB limits the scheduling of a UE to only a subset of all the available resources. Such frequency dependencies are very common in many radio resource allocation schemes, like for example in the type 1 downlink resource allocation of LTE [4] or in scheduling schemes where frequency hopping is employed for frequency diversity gains (e.g., in some types of LTE uplink allocation [4]).

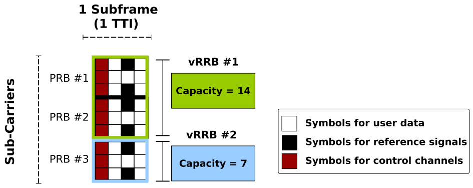


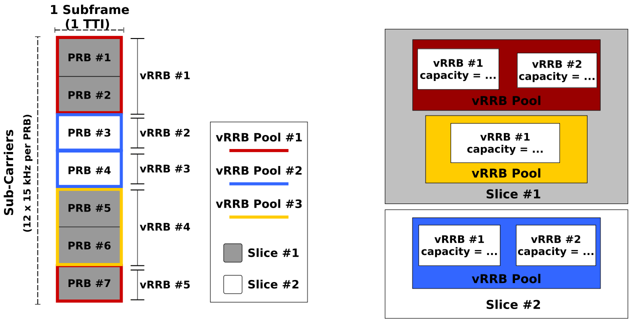
Figure 4: Mapping of PRBs to vRRBs in the context of LTE.

In order to deal with these issues, Orion introduces two abstractions for the virtualization of the radio resources, i.e., the *virtual Radio Resource Block (vRRB)* and the *vRRB pool*.

The vRRB is the fundamental radio resource abstraction employed by Orion. Each vRRB is characterized by a *capacity*, used to indicate the amount of data that it can hold. Capacity is expressed as the number of OFDM symbols contained in the vRRB and can be directly translated into a number of bits by the virtual control planes of slices based on the modulation and coding scheme (MCS) used. A vRRB can aggregate the OFDM symbols of one or more PRBs, after omitting symbols used for control channels and reference signals. This aggregation allows Orion to group together PRBs that must be used as a single unit, meaning that the capacity among different vRRBs can vary, as illustrated in Fig. 4 (vRRBs of one or two PRBs).

The Hypervisor aggregates and reveals the vRRBs to slices in the form of *vRRB pools*, where different sets of pools exist for the uplink and downlink. In each subframe, each slice can be assigned zero, one or more pools for downlink and correspondingly for uplink, each containing at least one vRRB. For example, in Fig. 5a, slice 1 has two vRRB pools, while slice 2 has only one. A slice can only view and allocate the vRRBs that are available in its pools (Fig. 5b), with the only constraint being that a UE can only be allocated vRRBs from the same pool. For example, in Fig. 5, a UE belonging to slice 1 can only be allocated resources from the red or yellow pool, but not both. The aggregation of vRRBs to pools captures the aforementioned frequency dependencies, since RBs that are mutually exclusive for a UE are assigned to different pools by the Hypervisor.

It should be noted that the abstractions of Orion are suitable both for current LTE as well as for future 5G New Radio (NR) networks, since both are based on OFDM and have the same fundamental frame structure [59]. The main difference is that NR is expected to support a more flexible resource grid, in which the number of TTIs and the



(a) vRRB and vRRB pool abstractions for LTE resource grid

(b) Abstractions of Fig. 5a from the slices' point of view

Figure 5: Illustration of Orion uplink/downlink radio resource virtualization abstractions in the context of LTE.

sub-carrier spacing of the RBs in a subframe can vary (Fig. 6) to support the diverse requirements of services and to provide support for mmWave communications [56]. In this context, the physical grid layout of the radio resources is one additional dimension that the Hypervisor should consider for mapping the physical resources to vRRBs. For example, for the FDD downlink grid presented in Fig. 6, a low data rate service without latency constraints should be allocated vRRBs 2 to 5 by the Hypervisor to maximize efficiency, while a low-latency service should be allocated vRRBs 6 and 7.

In contrast to the resources used for user traffic, control resources are provided in a simpler form, where the Hypervisor indicates to the virtual control planes of slices the role and amount of the assigned control resources through a message-based API. On the downlink, a virtual control plane can find out through this API how many CCEs are available for transmitting the scheduling decisions of its UEs, the resources available for sending uplink power control instructions as well as the resources available for uplink resource grants. On the uplink, this involves the TTIs in which a UE of the slice could report its signal measurements, so that it can be configured correspondingly by the higher layers of its virtual control plane (e.g., RRC).

The Virtualization Manager, besides presenting virtualized resources to slices, also performs the reverse mapping from slices' decisions over virtual resources to their physical counterparts. This involves consulting the context of the slices and performing all transformations required so that a command issued by the virtual control plane of a slice can be realized by the physical layer. Referring again to the example of LTE downlink scheduling, this mapping would involve a modification of the virtual control plane's scheduling decision to convert the vRRB allocation into a physical one, and the merging of the scheduling decisions from different slices into a single scheduling decision for the physical layer. Similarly, it would require the conversion of CCEs from the virtual to the physical form.

Virtualizing the data plane state To apply its control logic, a slice must be aware of its data plane state besides the resources allocated to it. Another task for the Virtualization Manager is to reveal the relevant data plane state to slices in an isolated manner, guaranteeing that any type of information related to specific UEs or their flows will be reported only to the corresponding slice. This virtualization in the data plane state ensures the isolation of data plane operations among slices, since the control plane of each slice is unaware of the data plane aspects that are relevant to the UEs of other co-located slices. Examples of relevant UE information include signal quality measurements, transmission queue sizes and number of flows (bearers in LTE) as well as event notifications like the (de)activation of a UE, the establishment of new flows, scheduling requests, etc. Given that the frequency domain is abstracted from the resource grid of slices, any UE-specific information related to the frequency domain is reported to the slice in an abstract form. As an example, relating to signal quality measurements and to the more fine-grained sub-band CQI reports that LTE can provide, the Virtualization Manager furnishes the measurements in the context of the vRRBs rather than the actual frequencies, through suitable mapping.

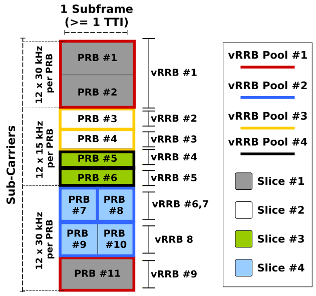


Figure 6: Orion uplink/downlink radio resource virtualization abstractions in the context of 5G NR – Flexible TTI and subcarrier spacing numerology.

Apart from UE specific information, the Virtualization Manager also informs slices about cell-related configuration information like resource block sizes, supported transmission modes, nominal transmission power of base station, etc. Since all slices need to be aware of this information and in order to ensure a conflict-free operation of the physical base station, all such cell-related information is provided to slices in a read-only form, while the infrastructure provider retains the exclusive privilege for making changes, when needed.

4.1.4 UE Association Manager.

The UE Association Manager (Fig. 2) associates UEs with slices in two steps: (i) the discovery of slices by UEs via the physical base station; (ii) the mapping of UEs to slices. To achieve this, the UE Association Manager interacts with the broadcast and random access processes of the base station, which are internal to the Hypervisor and not revealed to slices.

To aid in discovery, the UE Association Manager obtains the active slices from the Slice Context Manager. This information is then broadcasted by the base station. In the context of LTE, this can be done as in RAN sharing, where the base station broadcasts the list of PLMN ids, indicating all the MVNOs that are present. When a UE discovers a cell from its slice, it initiates the random access process. If successful, the attachment process begins and the UE is expected to send a unique identification (e.g., IMSI in LTE). Using this unique identifier, the UE Association Manager maps the UE to the appropriate slice by consulting the UE lookup function (Table 1) available for each slice through the Slice Context Manager. Once the correct slice is identified, the UE Association Manager updates the list of active UEs in the context of that slice. The corresponding virtual control plane is notified about the event through the

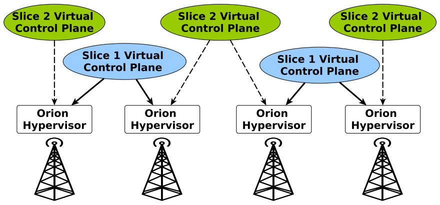

Slice 2 Virtual Control Plane
Slice 2 Virtual Control Plane
Slice 2 Virtual Control Plane
Slice 1 Virtual Control Plane
Slice 1 Virtual Control Plane
Orion Hypervisor
Orion Hypervisor
Orion Hypervisor
Orion Hypervisor

Figure 7: Flexible control plane composition enabled by Orion.

Virtualization Manager and from that point on takes up control of the UE (authentication, establishment of flows, scheduling, etc.). The attachment process is fully transparent to UEs in the sense that no changes are required to the protocols and signaling messages used for the interaction between the base station and the UEs.

4.2 Virtual Control Plane

As already alluded to in Section 3, Orion isolates memory and processing resources of slices by separating their virtual control planes from each other and the Hypervisor, thereby achieves functional isolation. Virtual control planes of slices in Orion are separate processes that exchange messages with the Hypervisor through dedicated communication channels. This separation allows slice deployment in a fully isolated manner using OS virtualization technologies, like containers and virtual machines, to enforce limits in terms of the allocated amount of memory and CPU.

A virtual control plane of a slice can be associated with multiple physical base stations (through their Hypervisors). Thus, the virtual control planes of individual slices can be composed independently and flexibly, like in the example of Fig. 7 for the case of two slices. As it can be seen, control plane centralization can be introduced into different regions of the RAN for each of the slices, based on the corresponding service's needs. This approach gives the flexibility to slice owners to enable RAN coordination (for load balancing, improved mobility management, etc.) when and where required. At the same time, it also enables an efficient utilization of the available computing resources for the placement of the slices' virtual control planes as demonstrated later through experiments in Section 5.2.

The communication between the virtual control plane of a slice and the Hypervisor is asynchronous (Fig. 8). To perform its operations, the control plane obtains the state of the virtual data plane and resources from the Hypervisor and stores it locally in a virtual RAN Information Base (vRIB). In its simplest form, the vRIB contains data structures with a raw representation of the virtual data plane state.

The vRIB can be both read and written by the slice's control plane, so changes due to control operations are first reflected in the vRIB. To maintain its consistency, the vRIB state is periodically synchronized with the Hypervisor via its Virtualization Manager and an asynchronous interface. Since the virtual control plane of slices and the Hypervisor can be deployed either on the same or on separate physical machines, depending on the setup and the available resources, the synchronization frequency of the vRIB can be tuned to the characteristics of the communication channel.

4.3 End-to-End Network Slicing

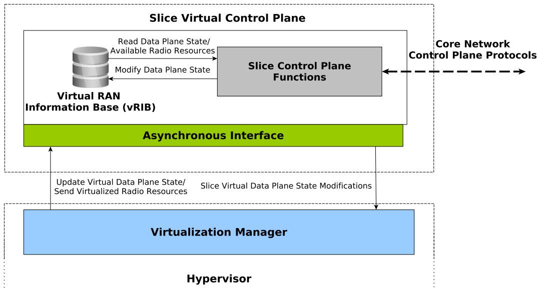

Slice Virtual Control Plane
Read Data Plane State/
Available Radio Resources
Modify Data Plane State
Slice Control Plane Functions
Virtual RAN
Information Base (vRIB)
Core Network
Control Plane Protocols
Asynchronous Interface
Update Virtual Data Plane State/
Send Virtualized Radio Resources
Slice Virtual Data Plane State Modifications
Virtualization Manager
Hypervisor

Figure 8: Virtual control plane of a slice in Orion.

We now discuss how Orion can be integrated in an end-to-end network slicing setting. The virtualization of various components making up the slice (including network functions) is key to enabling flexible component placement over the network infrastructure through a MANO entity, based on the needs of the service corresponding to the end-to-end slice. This is certainly not a hurdle for Orion as its components can be turned into VNFs. For a complete solution, it is also very important to investigate the interactions of the Orion components with the management plane and with the other components making up the slice. These interactions and the entities involved are depicted in Fig. 9, where each number-annotated arrow represents an interaction required for the right instantiation, operation and management of a slice throughout its life-cycle.

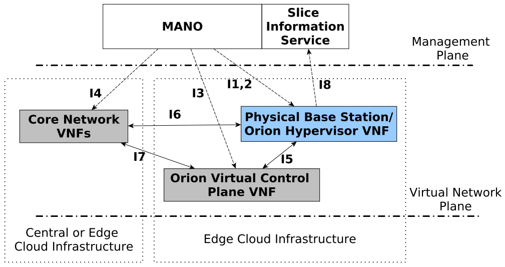

MANO
Slice Information Service
Management Plane
I4
I3
I1,2
I8
I6
Core Network VNFs
Physical Base Station/ Orion Hypervisor VNF
I7
I5
Orion Virtual Control Plane VNF
Virtual Network Plane
Central or Edge Cloud Infrastructure
Edge Cloud Infrastructure

Figure 9: Interactions among components involved in an end-to-end network slicing setting.

Starting from the RAN and before the deployment of any slice, the MANO must deploy the VNFs of the physical base stations and the Orion Hypervisor for the RAT of choice (I1). This VNF is expected to outlive the slices' VNFs. Management of slices should occur through an appropriate interface between the MANO and the Hypervisor (I2). The orchestration of a slice by the MANO involves the creation, migration and destruction of the Orion virtual control plane VNF (I3), as well as of the core network VNFs (I4) (e.g., LTE EPC components). While a slice is active, its core network VNFs interact with the Hypervisor for the user plane (I6) and with the virtual control plane for the control plane operations (I7), while the Hypervisor VNF also communicates with the slice's virtual control plane (I5). A *Slice Information Service* is required on the management plane, so that the Slice Context Manager in the Hypervisor can locate the information required for the mapping of UEs to slices (I8). This resembles the role of the HSS in LTE, but storing slice-related registration information instead.

Fig. 9 also illustrates that the Orion VNFs are deployed over an edge cloud near the RF front-end, while the core network VNFs can be deployed either at the edge or in a central cloud, depending on the slice's requirements. For example, control plane VNFs (e.g., MME) could be centralized to simplify control/management, while data plane VNFs (e.g., S-GW) could be placed closer to the user [15]. Similarly, different slices could employ different core network VNF setups, optimized for their particular use case (e.g., as in [67] for MTC traffic). The flexible placement of VNFs, coupled with the capabilities offered by Orion for the customization of the RAN, allow the creation of flexible slices adapted to the service needs.

4.4 Implementation

We developed a prototype implementation of Orion, following the design described so far in this section and considering LTE as the RAT. The Orion Hypervisor was implemented from scratch in C. For the interaction of the Hypervisor with the base station data plane, we leveraged the control–data plane API of the publicly available FlexRAN platform [1], which in turn is based on an open-source software implementation of LTE called OpenAirInterface (OAI) [54].

Our implementation provides full support for the virtualization of downlink radio resources, with the Hypervisor creating a single vRRB pool per slice in each subframe (equal to a TTI of 1ms in LTE), with each of its vRRBs corresponding to a group of physical resource blocks (2, 3 or 4 based on the bandwidth of the cell). This is because OAI currently supports resource allocation type 0 on the downlink, in which, based on the LTE specification [4], resource blocks can be assigned to UEs only in groups of 2, 3 or 4 blocks depending on the available spectrum.

The identifier used for mapping UEs to slices is the IMSI stored in the SIM card of UEs. This is sent to the MME of the network by UEs during their attachment using the Non-Access Stratum (NAS) protocols of LTE and is normally not visible to the eNodeB. Therefore, the RRC layer of OAI was modified to capture the relevant signaling messages, obtain these ids and pass them to the UE Association Manager of the Hypervisor. OAI was further modified to allow core networks (EPCs) of different slices to be connected over the same eNodeB. This required enabling the association of the MMEs, HSSs and S/P-GWs of different slices with the same eNodeB as well as the redirection of newly attached UEs to the correct core network by the UE Association Manager both for signaling and user traffic. Once a UE completes the random access process, its signaling messages are directed to the correct MME so that it can be authenticated by the corresponding slice's HSS and a bearer can be established with the slice's S/P-GW. From that point on, all the UE traffic goes through the core network of the hosting slice.

The current implementation allows expressing the SLA parameters stored in the context of active slices (Table 1) in three ways: (i) a fixed allocation of resource blocks; (ii) an average of the aggregate amount of resource blocks allocated to the slice over a window of 100ms; and (iii) an average target throughput for the slice over a window of 100ms. For the second and the third case, the current implementation uses the NVS radio resource allocation algorithm [35].

To realize the virtual control plane of slices, we implemented a modified version of the FlexRAN controller that operates over the abstract virtual resources presented by the Hypervisor and also with an enhanced RIB structure that supports write primitives. Given that the virtual control planes of slices can be deployed either locally or over a networked setting, we implemented two types of channels for message exchanges with the Hypervisor: one TCP-based channel optimized for a minimum message exchange delay in a network setting and one optimized for inter-process communication using ZeroMQ [28]. The appropriate type is specified in the slice's service description during the creation of the slice and the proper communication channel is deployed accordingly. Moreover, depending on the type of deployment, the synchronization of the Hypervisor with the vRIB must also be optimized. Towards this end, in the current implementation with inter-process communication the vRIB is fully synchronized with the Hypervisor every 1ms, while in a networked setting, only the time critical parts of the vRIB (virtual resource allocations, time synchronization in subframes) are synchronized at this rate, with the rest synchronized infrequently every 4ms.

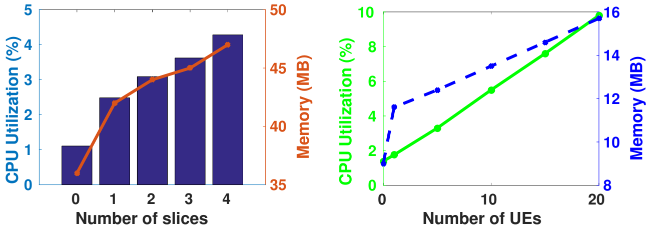
Finally, we created VNFs for the Hypervisor and the virtual control plane of Orion using Juju charms, enabling their deployment over an OpenStack-based environment and allowing usage of Orion with any OpenStack-compatible MANO framework (e.g., OSM).

5 EVALUATION

In this section, we quantitatively study Orion's resource consumption in different scenarios and benchmark it against other proposed RAN slicing solutions. We also examine the impact of the communication channel between the Hypervisor and the slice control plane. For the experiments, we used 2 Intel Xeon machines (E3-1245 @ 3.4GHz, 16GB RAM each) and 1 Intel i7 machine (5557U @ 3.10GHz, 8GB of RAM) for deploying physical base stations, the Orion Hypervisor and the virtual control planes of slices, depending on the specific experiment. All machines have Ubuntu 16.04 with a low-latency kernel and had support for Docker v1.13.1 [44], allowing the Orion components (Hypervisor, slice control planes) to be deployed in isolated containers. The core part of the network was deployed on an additional Intel-based machine (i7-4770R @ 3.2GHz, 8GB RAM), running the openair-cn open source EPC implementation [6]. For the RF front-end of physical base station, we used Ettus USRP B210 SDR boards and for UEs, up to 4 physical units (LG Nexus 5 and Samsung Galaxy Note 4 and 2 Huawei E3372 LTE dongles).

5.1 Scalability

Here we quantify how Orion scales in terms of the processing and memory requirements. To assess the overhead incurred for the virtualization operations performed by the Orion Hypervisor, we used a setup where each slice had 20 assigned (emulated) UEs and the available spectrum was saturated with TCP traffic. The results in Fig. 10a show that, apart from the initial overhead associated with the first slice, the addition of extra slices incurs a small and almost constant incremental overhead, both in terms of CPU and memory. This increase is mainly due to the Hypervisor's Virtualization Manager operations (construction of the virtual resource grids of slices, synchronization with slices' virtual control planes). The initial bigger increase observed after the creation of the first slice is due to the activation of the asynchronous interface as well as the periodic execution of the Radio Resource Manager for the allocation of resources to slices, which is inactive when there are no slices. The actual number of slices that can be supported by a physical base station depends both on the Hypervisor overhead presented here and



(a) Hypervisor overhead (4-core CPU and 20 UEs/slice)

(b) Virtual control plane overhead (2-core CPU)

Figure 10: Orion CPU and memory consumption scaling.

System # Slices	FLARE	FlexRAN	Orion
1	5MHz	5MHz	5MHz
2	5MHz/slice	10MHz	10MHz
3	5MHz for slices 1,2 10MHz for slice 3	20MHz	20MHz

Table 2: Allocation of channel bandwidth among slices. Both Orion and FlexRAN use shared spectrum for all slices.

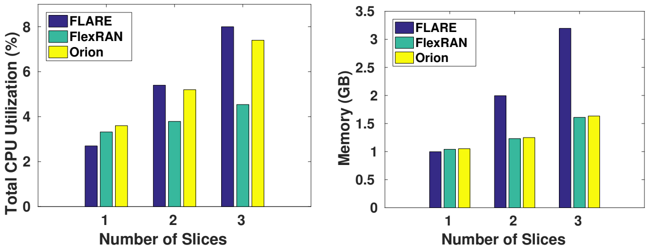
on the physical layer requirements, which can greatly vary depending on where the most demanding physical layer operations occur (e.g., at the baseband processing unit or at a remote radio head).

Next we turn our attention to the virtual control plane of a slice and quantify its overhead with a varying number of UEs (0 to 20) which increases linearly with the number attached UEs (Fig. 10b). The increases in the memory consumption are mainly due to the additional information stored in the vRIB as more UEs get attached and the memory required for the message exchanges with the Hypervisor. The initial sharp rise in the memory consumption is due to the creation of the UE-related part of the vRIB, once the first UE gets attached. The rise in CPU utilization is a result of the communication with the Hypervisor for the synchronization of the vRIB and from the scheduling of the attached UEs. However, the overhead is still fairly modest, meaning that the virtual control planes of multiple slices could be deployed side-by-side over the same physical machine (e.g., over an edge cloud data-center) without scaling issues. For example, in the commodity machine used for these experiments, at least eight slice control planes could be deployed without any issues.

5.2 Comparison with the State-of-the-Art

Here we compare the overall memory and processing requirements of Orion versus the state-of-the-art virtualization and RAN sharing solutions from the literature. One representative approach is *FLARE*, the RAN slicing solution proposed in [47, 48], which ensures isolation by deploying different instances of OAI over Docker containers and each slice is *statically* assigned a dedicated chunk of spectrum. The other alternative approach is represented by FlexRAN [18, 19, 50] that has been shown to provide RAN slicing capabilities following a RAN sharing approach but without functional isolation. It should be noted that, like Orion, both FLARE and FlexRAN are built using OAI as their basis. Therefore, any performance differences among them come mainly from the design choices of each system regarding how to perform RAN slicing; a fact that makes this comparison fair.

We design an experiment to compare Orion with these two alternatives for a varying number of slices, where each slice has TCP traffic (mimicking an aggregate demand from a set of UEs) to a single physical UE. Each slice was allocated spectrum according to the configuration of Table 2 for fairness. Three physical machines were used for this experiment as compute nodes for the various components required for each architecture. To capture the overall system performance, we measured the aggregate resources required for all the RAN components of each architecture across all physical machines. For the CPU utilization, this meant measuring the jiffies allocated to the RAN-related processes of each architecture over a constant number of 10 CPU cores (the total number of cores of all physical machines) and dividing this value with the total jiffies of all the CPUs for all system processes during the course of the experiment.



(a) CPU Utilization (10 cores)

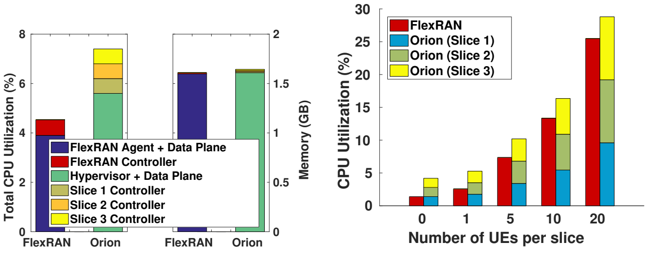
(b) Memory footprint

Figure 11: Comparison of Orion with FLARE and FlexRAN with varying number of slices.

We see from Fig. 11a that Orion has relatively higher CPU requirements compared to FlexRAN, which is expected due to the Hypervisor's operation and the unavoidable cost to pay for the additional control plane processes that must be deployed as new slices are added. On the other hand, Orion starts with higher CPU requirements than FLARE, but soon becomes more lightweight, due to the fact that each new FLARE slice requires the deployment of a complete protocol stack, including the very demanding (CPU-wise) physical layer. Since in Orion all slices share the same physical layer, its overhead comes mainly from the Hypervisor and the slices' virtual control planes, which, as shown earlier, are fairly lightweight. The same observations and for the same reasons can be made for the memory consumption (Fig. 11b), with Orion performing significantly better compared to FLARE and marginally worse compared to FlexRAN.

Since both Orion and FlexRAN are composed of a number of different network functions (Hypervisor/physical layer and virtual control planes in Orion; Agent/data plane and controller in FlexRAN), it is insightful to break down the results of Fig.11 in order to understand their differences. This is illustrated in Fig.12a for three slices. As it can be observed, the Orion virtual control planes and the FlexRAN controller consume a negligible amount of memory. On the other hand, in terms of CPU utilization, Orion's biggest impact comes from the physical layer and the Hypervisor, but, in contrast to memory, a significant contribution is also made from the virtual control planes of slices. In the case of FlexRAN, the base station network function (FlexRAN Agent and data plane) has the biggest CPU utilization, also with a significant contribution from the controller.

One important thing to notice is that while some components of both systems, namely the Orion Hypervisor and the FlexRAN Agent, are placed near the base station and are effectively monolithic components regardless of the number of deployed slices, this analogy is not carried over to the other network functions. While in the case



(a) Comparison of Orion and FlexRAN for 3 slices (1 UE/slice)

(b) Control plane CPU Utilization (2 cores)

Figure 12: Breakdown of resource requirements for Orion vs FlexRAN.

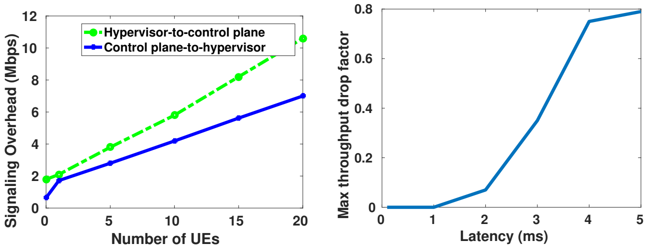
of FlexRAN the controller is also a monolithic component, Orion provides individual control planes to slices. Apart from the functional isolation that this ensures, it also enables a better utilization of the underlying resources through the flexible placement of the control plane functions over the available compute nodes. We illustrate this point through an experiment in which three slices are deployed using Orion and FlexRAN and the overall control plane CPU utilization is measured for a varying number of UEs per slice. Results shown in Fig.12b indicate that Orion always performs slightly worse than FlexRAN, however this overhead is compensated by the fact that the control plane can be broken down into three functions that can be flexibly placed over different CPU cores or even completely different compute nodes; something that FlexRAN is unable to do.

5.3 Impact of Communication Channel

We now assess Orion's requirements for the communication between a slice control plane and the Hypervisor (in terms of bandwidth and latency) for their deployment in a networked setting. Understanding these requirements is very important from an end-to-end perspective for deploying the components of Orion as VNFs, since the orchestration entity needs to be aware of the physical constraints regarding their placement as part of the overall network service description.

To measure the bandwidth requirements of the control channel, we varied the number of UEs for a slice over the physical base station with 5MHz spectrum. In this experiment, UEs continuously had traffic to keep the slice at full load. The highest overhead was incurred for messages sent by the Hypervisor towards the control plane of the slice (Fig. 13a), for the synchronization of the vRIB and for the provision of the slice with virtual radio resources. The overall bandwidth requirements for this setup did not exceed 20Mbps, implying that such a scenario is feasible in practice.

To study latency constraints, we used a similar setup but with a single COTS UE attached to the slice. We used *netem* [21] to set the latency between the Hypervisor and the slice's virtual control plane and measured the drop in the maximum achievable throughput for the UE compared to a co-located deployment. Fig. 13b shows that networked deployment of Orion is efficient only when the latency remains below 2ms. This constraint could be relaxed to an extent with some optimizations (e.g., HARQ reporting optimizations) but a co-located deployment is preferable if possible. The slice controller could also be made distributed and hierarchical, so that time-critical scheduling operations can be handled at or near the physical base station, while other operations can be placed flexibly at other locations.



(a) Signaling bandwidth requirements

(b) Effect of signaling latency on maximum throughput

Figure 13: Network requirements for the interaction of the Hypervisor with the virtual control plane when not co-located.

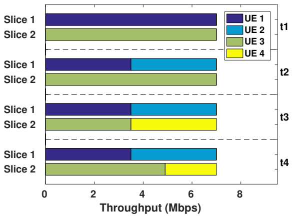


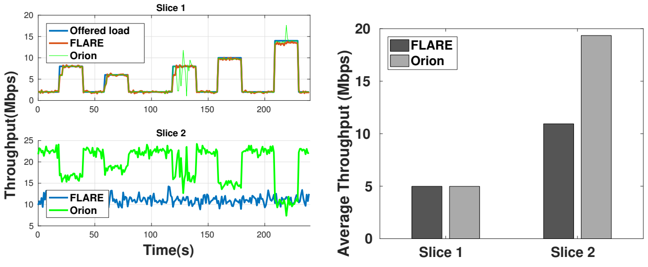
Figure 14: Isolation of RAN slices in terms of radio resources and control functions (scheduling).

6 CASE STUDIES

6.1 Isolation Capabilities

We demonstrate the ability of Orion to provide radio resource and functional isolation, which is not possible with RAN sharing oriented slicing approaches [18, 19]. In this experiment, 2 slices with proportional fair downlink schedulers were created and statically allocated 5MHz of spectrum in equal shares by the Hypervisor. Initially, we connected 2 physical UEs (one per slice) and performed a throughput measurement to both using iperf, ensuring ideal channel conditions. As shown in instance $t1$ of Fig. 14, the Hypervisor guarantees the radio resource isolation so that each slice can obtain only half of the available radio resources, leading to an equal throughput for the UEs. Further aspects of this isolation of radio resources are demonstrated in instances $t2$ and $t3$ of Fig. 14 where more UEs are added to the slices and where each UE is constrained by the Hypervisor to the radio resources allocated within its slice. For example, in instance $t2$ the performance of UE3 remains unaffected by the addition of UE2 since UE2 belongs to a different slice.

Finally, the inter-slice functional isolation is illustrated in instance $t4$ of Fig. 14. The scheduler in the virtual control plane of slice 2 was replaced with a class-based scheduler allocating 70% of the resources to UEs in class 1 and the rest to class 2. In the experiment, UE3 and UE4 were assigned to class 1 and class 2, respectively. As shown in Fig. 14, the change in the control plane of slice 2 only affects the throughput of the UEs belonging to that slice, leaving the control plane and the UEs of slice 1 completely oblivious.



(a) Instantaneous slice throughput in static (FLARE) vs dynamic resource allocation (Orion).

(b) Average throughput achieved through static and dynamic spectrum use.

Figure 15: Benefit of Orion's flexible radio resource allocation.

6.2 Flexible Radio Resource Allocation

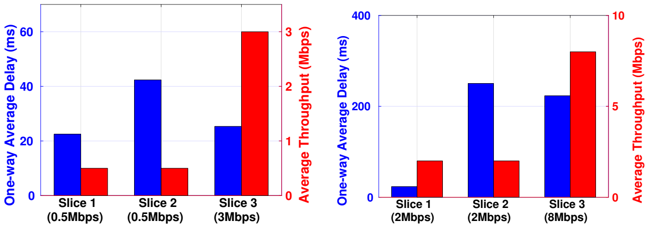
We now demonstrate the flexibility offered by Orion for dynamic radio resource allocation and how it can be used for efficiently utilizing the spectrum among co-existing slices. We created 2 slices, each with an average aggregate throughput requirement of 14Mbps. As a baseline, we considered FLARE, with each slice statically allocated 5MHz of spectrum. For the same scenario, we employed Orion with a pool of 10MHz for both slices and implemented the radio resource allocation using the algorithm of [35]. In the beginning of each allocation window, the resources were allocated to slices based on their effective traffic rate and on their average throughput requirement.

We connected 1 UE on slice 1 and 3 UEs on slice 2 and we used the D-ITG traffic generator [10] to generate TCP flows in slice 1. Specifically, we created a TCP flow with a constant rate of 2Mbps. On top of it, we sporadically created some short-lived flows (20s each), capable of attaining various rates (4-12Mbps). In slice 2, all the connected UEs accessed a DASH-based video streaming service to stream a video offering a wide range of bitrates. Using this setup, we measured the instantaneous (Fig. 15a) and the average (Fig. 15b) aggregate throughput achieved by each slice in FLARE and Orion for a period of 240s. As observed, for slice 1, both FLARE and Orion give similar results, fully covering the offered load and respecting the SLA of slice 1 for up to 14Mbps of *average* aggregate throughput.

For slice 2, however, Orion is seen to perform much better than FLARE, dynamically reallocating the idle resources from slice 1 and allowing the UEs in slice 2 to stream videos with a higher bitrate. FLARE, unlike Orion, is spectrum-limited and cannot go beyond 14Mbps at any point in time, irrespectively of what happens in slice 1. The flexibility of Orion comes without compromising slice SLAs. When slice 1 requires all of its expected resources (e.g., 210-230s in Fig. 15a), the Radio Resource Manager allocates fewer radio resources to slice 2, so the aggregate throughput of this slice drops.

6.3 Deployment in an End-to-End Setting

Here we highlight the capability of Orion to be deployed in an end-to-end network slicing setting and the effect of the slice configuration to overall performance in terms of throughput and delay. For this experiment we created 3 slices, each composed of VNFs for an EPC (HSS, MME, S/P-GW) and a virtual base station deployed over Orion. For slices 1 and 3, the S/P-GW VNFs of the EPC were placed on hosts 1 hop away from the eNodeB, to reduce the delay of the EPC–eNodeB communication as much as possible (less than 0.3ms). For slice 2, we deployed the S/P-GW VNF on a host connected to the eNodeB through a 20ms delay link (emulated with netem [21]) to reflect the placement of the mobile core functionality far from the eNodeB (e.g., in a central cloud). The eNodeB in this experiment was allocated 5MHz of spectrum, supporting a total rate of 14Mbps in ideal channel conditions. Finally, the Orion Hypervisor was configured so that slice 1 was guaranteed a static allocation of 4 resource blocks per subframe (maximum achievable throughput of 2Mbps) while slices 2 and 3 were guaranteed an average throughput of 2 and 8Mbps respectively, performing the allocation using the algorithm in [35].



(a) Low offered load

(b) High offered load

Figure 16: Performance impact of Core-eNodeB latency and radio resource allocation policy/guarantees for low and high offered loads (load values in parentheses on x-axis).

In this setup, we used the D-ITG traffic generator [10] to generate downlink UDP traffic with exponentially distributed packet inter-arrival times and measured slice performance in terms of average throughput and delay. Initially, the load offered to slices was kept low, compared to the slices' SLA guarantees, and all slices manage to support the traffic demand (Fig. 16a). Moreover, we can observe that the average packet delay for slice 2 was higher than that of slices 1 and 3 by about 20ms, something expected considering the (emulated) 'far from eNodeB' placement of the core functions for slice 2.

Next, we re-run the experiment, increasing the load offered to slices to reach their limit (Fig. 16b). As we can observe, all slices still achieve an average throughput close to their offered load, which was expected since their load remains within the guaranteed limits. However, while the average delay of slice 1 stays at the same level, the average delay of slices 2 and 3 is over 200ms. The reason is that the static allocation used for slice 1, although inflexible, ensures stricter delay guarantees, as arriving packets could always be served without queuing for long. On the other hand, the flexible dynamic allocation of radio resources to the other slices guarantees the average throughput, but this was done over 100 ms windows (as described in Section 4.4), rather than per subframe. In such almost saturating traffic conditions the packets are more likely to experience a longer waiting time in the slice queues, increasing the average delay.

7 MULTI-SERVICE SLICES EXTENSION

The design of Orion enables RAN slicing in cases where the UE has a 1:1 relationship with the slice (e.g., MVNOs and verticals). However, deployment of Over-The-Top (OTT) services as distinct slices breaks this assumption and control decisions among slices cannot be guaranteed to be conflict-free. To overcome this, we propose an extension to Orion in the form of *service containers*, which targets slices offering multi-service capabilities (e.g., an MVNO with OTT services).

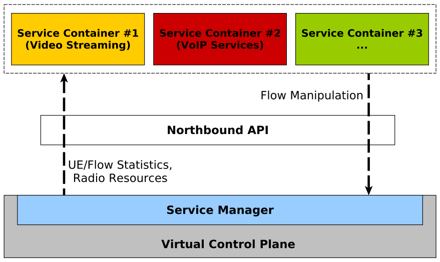

Service Container #1 (Video Streaming)
Service Container #2 (VoIP Services)
Service Container #3 ...
Flow Manipulation
Northbound API
UE/Flow Statistics, Radio Resources
Service Manager
Virtual Control Plane

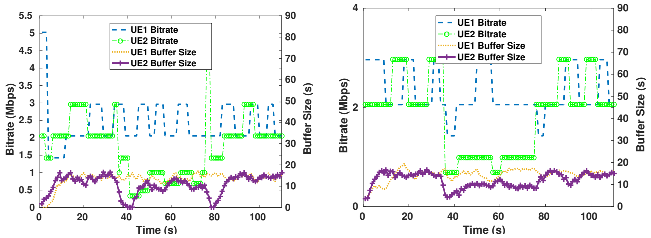
Figure 17: Extension of Orion to support multi-service slices.

The idea behind service containers is that an OTT service provider agrees with an MVNO to assume control of the flows corresponding to its service through its own isolated functions deployed over the virtual control plane of the slice. This concept is similar in spirit to the 3GPP Service Capability Exposure Function (SCEF) [3] and with research works like [62], where an API is defined for the interaction of applications with the mobile network to modify the service level or to obtain information about the network state. The deployment of service containers over the base stations of MVNOs can enhance this model with *real-time* control and monitoring capabilities, enabling the further optimization of OTT applications.

The design of this extension (Fig. 17) inherits the design principles discussed in Section 4. Service containers are deployed as processes over the virtual control plane of a slice, each with its own isolated memory and processing resources. The virtual control plane of the slice is extended with a Northbound API, enabling the exchange of information between the MVNO and the service provider. Using the API, a service can obtain information about the flows of UEs assigned to it (available UE flows, transmission queue sizes, signal quality etc.), as well as about the (abstracted) radio resources available for it. On the other direction, the service can issue scheduling decisions, indicating how its allocated resources should be distributed among its flows. A Service Manager sitting between the virtual control plane of the MVNO and the Northbound API is responsible to map flows to services, distribute the abstract radio resources to the containers and perform access control, ensuring that service containers can only access information and radio resources related to their serving flows.

In order to demonstrate the benefits of service containers, we consider the example of a DASH-based adaptive bitrate video streaming service provider for which a service container was developed and deployed over Orion for monitoring and scheduling the video flows of streaming UEs. The control function of this container is composed of a monitoring and a scheduling component. The monitoring component observes the CQI of the UEs that are streaming videos and provides real-time feedback to the video streaming client about the maximum sustainable bitrate that a UE can achieve given its CQI. Based on that feedback, the client readjusts the bitrate of the video to avoid freezes. The scheduling component makes the scheduling decisions (priority and MCS) for the video flows of the UEs by communicating with the video streaming clients every second and observing the buffer size of each video stream. Flows of video streams with lower buffer sizes are given a higher priority, on the premise that such flows need faster treatment to avoid buffer freezes.

For our evaluation, we considered a scenario with 2 UEs streaming a video [45] using the DASH reference client [14]. UE 1 had a fixed optimal signal quality and for UE 2 we introduced a big drop of the signal quality at $t = 35$s and then a big increase at $t = 75$s. Results obtained when the service container was active and managing the video flows (Fig. 18b) are compared against the regular non-assisted DASH service with the MVNO being responsible for scheduling all flows through a proportional-fair scheduler (Fig. 18a). When the signal quality of UE 2 changes suddenly, the non-assisted mechanism fails to adapt fast, leading to buffer freezes. In comparison, the cross-layer optimized control of flows enabled by the service containers allowed the bitrate to adjust more appropriately and to avoid stalls in the video stream, thus improving the overall quality of the service.



(a) MVNO managed **(b) OTT cross-layer optimized**

Figure 18: Video streaming performance in MVNO-managed vs. OTT-optimized control of flows.

This example is merely intended to demonstrate the capabilities enabled by service containers for OTTs, by allowing them to make control decisions using information unavailable to a non application-aware MVNO. More sophisticated control mechanisms could also be employed, like the cross-layer optimization mechanism for video delivery in [31] or the content-aware schedulers in [38] and [55].

8 CONCLUSIONS

We have presented Orion, a flexible RAN slicing architecture. Orion slices can be deployed dynamically over the network infrastructure, co-existing in a fully isolated manner in terms of radio resources and control functions. At the same time, Orion facilitates efficient radio resource sharing among slices. A prototype implementation of Orion was developed for LTE, with the performance evaluation indicating that it is a lightweight and flexible RAN virtualization solution. Orion's capabilites were highlighted through a number of use cases, revealing its suitability for future multi-service mobile networks.

The virtualization capabilities enabled by Orion can be seen as a concrete step towards realizing the so-called RAN-as-a-Service (RaaS) paradigm, where virtual RAN instances can be dynamically created over a cloud infrastructure. However, apart from virtualization, another significant aspect of the RaaS paradigm is the capability to perform a dynamic functional split of the RAN in terms of control and data plane operations, enabling a more flexible composition of RAN instances. Extending the design of Orion to accommodate this need, while retaining its virtualization capabilites is a natural next step. Another important aspect in the context of 5G is the efficient virtualization of radio resources in multi-RAT settings. Orion is currently designed for single-RAT settings, however investigating the feasibility of extending its capabilities for multi-RAT is a significant topic for future work. These and other future work issues are discussed further in [20].

ACKNOWLEDGEMENTS

We thank our shepherd Songwu Lu and the anonymous reviewers for the helpful suggestions on improving the paper.

REFERENCES

[1] 2017. FlexRAN. http://networks.inf.ed.ac.uk/flexran. (2017).

[2] 3GPP. 2017. Architecture enhancements for dedicated core networks. TS 23.707. (2017).

[3] 3GPP. 2017. Architecture Enhancements to Facilitate Communications with Packet Data Networks and Applications. TS 23.682. (2017).

[4] 3GPP. 2017. Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures. TS 36.213. (2017).

[5] Ian F Akyildiz, Pu Wang, and Shih-Chun Lin. 2015. SoftAir: A software defined networking architecture for 5G wireless systems. *Computer Networks* 85 (2015), 1–18.

[6] OpenAirInterface Software Alliance. 2017. Openair-cn repository. https://gitlab.eurecom.fr/oai/openair-cn. (2017).

[7] Xueli An et al. 2016. On end to end network slicing for 5G communication systems. *Transactions on Emerging Telecommunications Technologies* (2016).

[8] Unknown Author. 2005. IEEE 802.1q – IEEE standard for local and metropolitan area networks virtual bridged local area networks. *IEEE Computer Society* (2005).

[9] Arijit Banerjee et al. 2015. Scaling the LTE Control-Plane for Future Mobile Access. In *Proceedings of the 11th ACM CoNEXT*. ACM, 19.

[10] Alessio Botta, Alberto Dainotti, and Antonio Pescapé. 2012. A tool for the generation of realistic network workload for emerging networking scenarios. *Computer Networks* 56, 15 (2012), 3531–3547.

[11] Pablo Caballero et al. 2017. Network slicing games: Enabling customization in multi-tenent networks. In *Proceedings of IEEE INFOCOM 2017*. IEEE.

[12] Brent Chun et al. 2003. Planetlab: An Overlay Testbed for Broad-Coverage Services. *ACM SIGCOMM Computer Communication Review* 33, 3 (2003), 3–12.

[13] Xavier Costa-Pérez et al. 2013. Radio Access Network Virtualization for Future Mobile Carrier Networks. *IEEE Communications Magazine* 51, 7 (2013), 27–35.

[14] DASH Industry Forum. 2017. DASH Reference Client. http://dashif.org/reference/players/javascript/v2.4.1/samples/dash-if-reference-player/index.html. (2017).

[15] Ericsson. 2016. A Vision of the 5G Core: Flexibility for New Business Opportunities. (2016).

[16] Ericsson. 2017. 5G Systems - Enabling the Transformation of Industry and Society. (Jan 2017).

[17] ETSI. 2017. Open Source MANO. https://osm.etsi.org/. (2017).

[18] Eurecom. 2017. Demo at MWC 2017:5G Cloud RAN slice. https://insights.ubuntu.com/event/mobile-world-congress-2017/. (2017).

[19] Xenofon Foukas et al. 2016. FlexRAN: A Flexible and Programmable Platform for Software-Defined Radio Access Networks. In *Proceedings of the 12th ACM CoNEXT*. ACM, 427–441.

[20] Xenofon Foukas et al. 2017. Network Slicing in 5G: Survey and Challenges. *IEEE Communications Magazine* 55, 5 (2017), 94–100.

[21] Linux Foundation. 2017. NetEm Linux network emulator. https://wiki.linuxfoundation.org/networking/netem. (2017).

[22] Thomas Frisanco et al. 2008. Infrastructure Sharing and Shared Operations for Mobile Network Operators from a Deployment and Operations View. In *Proceedings of IEEE NOMS 2008*. IEEE, 129–136.

[23] Pablo Caballero Garcés et al. 2015. RMSC: A Cell Slicing Controller for Virtualized Multi-tenant Mobile Networks. In *Vehicular Technology Conference (VTC Spring), 2015 IEEE 81st*. IEEE, 1–6.

[24] Anteneh A Gebremariam et al. 2017. Resource pooling via dynamic spectrum-level slicing across heterogeneous networks. In *14th IEEE annual consumer communications and networking conference (CCNC)*.

[25] Aditya Gudipati, Li Erran Li, and Sachin Katti. 2014. RadioVisor: A Slicing Plane for Radio Access Networks. In *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 237–238.

[26] Jun He and Wei Song. 2015. AppRAN: Application-Oriented Radio Access Network Sharing in Mobile Networks. In *Proceedings of IEEE International Conference on Communications (ICC)*. IEEE, 3788–3794.

[27] Matt Helsley. 2009. LXC: Linux container tools. *IBM devloperWorks Technical Library* (2009), 11.

[28] Pieter Hintjens. 2013. *ZeroMQ: Messaging for Many Applications*. O'Reilly Media, Inc.

[29] ITU-R. 2015. IMT Vision – Framework and overall objectives of the future development of IMT for 2020 and beyond. (Sept 2015).

[30] Xin Jin et al. 2013. Softcell: Scalable and Flexible Cellular Core Network Architecture. In *Proceedings of the ninth ACM CoNEXT*. ACM, 163–174.

[31] Vinay Joseph and Gustavo de Veciana. 2014. NOVA: QoE-Driven Optimization of DASH-based Video Delivery in Networks. In *Proceedings of IEEE INFOCOM*. IEEE, 82–90.

[32] Mahmoud I Kamel, Long Bao Le, and André Girard. 2014. LTE Wireless Network Virtualization: Dynamic Slicing via Flexible Scheduling. In *Vehicular Technology Conference (VTC Fall), 2014 IEEE 80th*. IEEE, 1–5.

[33] Young Han Kim et al. 2014. Slicing the Next Mobile Packet Core Network. In *Wireless Communications Systems (ISWCS), 2014 11th International Symposium on*. IEEE, 901–904.

[34] Avi Kivity et al. 2007. kvm: the Linux virtual machine monitor. In *Proceedings of the Linux symposium*, Vol. 1. 225–230.

[35] Ravi Kokku et al. 2012. NVS: A Substrate for Virtualizing Wireless Resources in Cellular Networks. *IEEE/ACM Transactions on Networking (TON)* 20, 5 (2012), 1333–1346.

[36] Adlen Ksentini and Navid Nikaein. 2017. Toward Enforcing Network Slicing on RAN: Flexibility and Resources Abstraction. *IEEE Communications Magazine* 55, 6 (2017), 102–108.

[37] Li Erran Li, Z Morley Mao, and Jennifer Rexford. 2012. Toward Software-Defined Cellular Networks. In *Proceedings of 2012 European Workshop on Software Defined Networking (EWSDN)*. IEEE, 7–12.

[38] Yan Li et al. 2008. Content-Aware Playout and Packet Scheduling for Video Streaming over Wireless Links. *IEEE Transactions on Multimedia* 10, 5 (2008), 885–895.

[39] Chengchao Liang and F Richard Yu. 2015. Wireless Virtualization for Next Generation Mobile Cellular Networks. *IEEE Wireless Communications* 22, 1 (2015), 61–69.

[40] Linux Foundation. 2017. ONAP. https://www.onap.org/. (2017).

[41] Rajesh Mahindra et al. 2013. Radio Access Network Sharing in Cellular Networks. In *Proceedings of 21st IEEE International Conference on Network Protocols (ICNP)*. IEEE, 1–10.

[42] Toktam Mahmoodi and Srini Seetharaman. 2014. Traffic Jam: Handling the Increasing Volume of Mobile Data Traffic. *IEEE Vehicular Technology Magazine* 9, 3 (2014), 56–62.

[43] Ilaria Malanchini, Stefan Valentin, and Osman Aydin. 2014. Generalized Resource Sharing for Multiple Operators in Cellular Wireless Networks. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2014 International*. IEEE, 803–808.

[44] Dirk Merkel. 2014. Docker: lightweight linux containers for consistent development and deployment. *Linux Journal* 2014, 239 (2014), 2.

[45] Microsoft Test Content. 2017. DASH Sample Video. http://wams.edgesuite.net/media/MPTExpressionData02/BigBuckBunny_1080p24_IYUV_2ch.ism/manifest(format=mpd-time-csf). (2017).

[46] Mehrdad Moradi et al. 2014. SoftMoW: Recursive and Reconfigurable Cellular WAN Architecture. In *Proceedings of the 10th ACM CoNEXT*. ACM, 377–390.

[47] Akihiro Nakao et al. 2016. Demo at ITU FG IMT-2020 Workshop:Softwarized LTE in FLARE network slices. http://www.itu.int/en/ITU-T/Workshops-and-Seminars/201612/Pages/Programme.aspx. (Dec 2016).

[48] Akihiro Nakao, Ping Du, Yoshiaki Kiriha, Fabrizio Granelli, Anteneh Atumo Gebremariam, Tarik Taleb, and Miloud Bagaa. 2017. End-to-end Network Slicing for 5G Mobile Networks. *Journal of Information Processing* 25 (2017), 153–163.

[49] NGMN-Alliance. 2015. 5G White Paper. (Feb 2015).

[50] Navid Nikaein. 2017. FlexRAN tutorial on RAN sharing. https://gitlab.eurecom.fr/mosaic-5g/mosaic-5g/wikis/ran-sharing. (2017).

[51] Navid Nikaein et al. 2014. OpenAirInterface: A flexible platform for 5G research. *ACM SIGCOMM Computer Communication Review* 44, 5 (2014), 33–38.

[52] Navid Nikaein et al. 2015. Network store: Exploring slicing in future 5G networks. In *Proc. 10th ACM International Workshop on Mobility in the Evolving Internet Architecture (MobiArch'15)*. 8–13.

[53] Nokia. 2014. Network Sharing: Delivering mobile broadband more efficiently and at lower cost. http://resources.alcatel-lucent.com/asset/200192. (2014).

[54] OpenAirInterface Software Alliance. 2017. OpenAirInterface repository. https://gitlab.eurecom.fr/oai/openairinterface5g. (2017).

[55] Peshala Pahalawatta et al. 2007. Content-Aware Resource Allocation and Packet Scheduling for Video Transmission over Wireless Networks. *IEEE Journal on Selected Areas in Communications* 25, 4 (2007).

[56] Klaus I Pedersen et al. 2016. A Flexible 5G Frame Structure Design for Frequency-Division Duplex Cases. *IEEE Communications Magazine* 54, 3 (2016), 53–59.

[57] Kostas Pentikousis, Yan Wang, and Weihua Hu. 2013. Mobileflow: Toward Software-Defined Mobile Networks. *IEEE Communications magazine* 51, 7 (2013), 44–53.

[58] Zafar Ayyub Qazi et al. 2016. KLEIN: A Minimally Disruptive Design for an Elastic Cellular Core. In *Proceedings of the Symposium on SDN Research*. ACM, 2.

[59] Qualcomm. 2016. Making 5G NR a reality. (Dec 2016).

[60] Peter Rost et al. 2016. Mobile Network Architecture Evolution Toward 5G. *IEEE Communications* 54, 5 (2016).

[61] Peter Rost et al. 2017. Network Slicing to Enable Scalability and Flexibility in 5G Mobile Networks. *IEEE Communications magazine* (2017).

[62] Konstantinos Samdanis et al. 2013. Service Boost: Towards on-demand QoS enhancements for OTT apps in LTE. In *Network Protocols (ICNP), 2013 21st IEEE International Conference on*. IEEE, 1–6.

[63] Konstantinos Samdanis, Xavier Costa-Perez, and Vincenzo Sciancalepore. 2016. From Network Sharing to Multi-Tenancy: The 5G Network Slice Broker. *IEEE*

Communications 54, 7 (2016).

[64] Rob Sherwood et al. 2009. FlowVisor: A Network Virtualization Layer. *OpenFlow Switch Consortium, Tech. Rep* (2009), 1–13.

[65] Aisha Syed and Jacobus Van der Merwe. 2016. Proteus: A Network Service Control Platform for Service Evolution in a Mobile Software Defined Infrastructure. In *Proceedings of the 22nd ACM MobiCom*. ACM, 257–270.

[66] Tarik Taleb et al. 2015. EASE: EPC as a Service to Ease Mobile Core Network Deployment over Cloud. *IEEE Network* 29, 2 (2015), 78–88.

[67] Tarik Taleb, Adlen Ksentini, and Abdellatif Kobbane. 2014. Lightweight Mobile Core Networks for Machine Type Communications. *IEEE Access* 2 (2014), 1128–1137.

[68] Volkan Yazıcı, Ulas C Kozat, and M Oguz Sunay. 2014. A New Control Plane for 5G Network Architecture with a Case Study on Unified Handoff, Mobility, and Routing Management. *IEEE Communications Magazine* 52, 11 (2014), 76–85.

[69] Yasir Zaki et al. 2011. LTE Mobile Network Virtualization. *Mobile Networks and Applications* 16, 4 (2011), 424–432.

[70] Liang Zhao et al. 2011. LTE virtualization: From Theoretical Gain to Practical Solution. In *Proceedings of the 23rd International Teletraffic Congress*. International Teletraffic Congress, 71–78.