Orchestra: A Customizable Split-Aware NFV Orchestrator for Dynamic Cloud Radio Access Networks

Ariel Galante Dalla-Costa, Lucas Bondan, Juliano Araújo Wickboldt, Cristiano Bonato Both, and Lisandro Zambenedetti Granville

***Abstract*— Dynamic Cloud Radio Access Networks (C-RAN) is an emerging wireless architecture that splits radio functions in a hierarchical cloud-based infrastructure. In Dynamic C-RAN, wireless functionalities can be divided into smaller components and distributed along with a hierarchical cloud infrastructure. Network Functions Virtualization (NFV) concepts have been recently investigated to facilitate management-related operations of these wireless functionalities. Despite the advances in providing flexible NFV orchestration for Virtualized Network Functions (VNF)s, there is still a lack of solutions aware of VNFs internal composition, a key characteristic of split virtualized radio functions. In this article, we advance NFV orchestration in a Dynamic C-RAN scenario presenting Orchestra, a split-aware NFVO for Dynamic C-RAN with support to customizable orchestration algorithms for different radio function splits. Orchestra is validated through instantiates and migrates VNFCs following the decisions of personalized orchestration algorithms, with a petty impact in the VNFC's deployment and migration time.**

***Index Terms*— NFV, orchestration, dynamic C-RAN, 5G.**

I. Introduction

CLOUD Radio Access Network (C-RAN) is a wireless mobile architecture that decouples the Base Station (BS) component into radio and processing unities called Remote Radio Head (RRH) and Baseband Unit (BBU) [1]. RRH interfaces with fiber and other functionalities, such as digital processing, digital and analog conversion, power amplification, and filtering. BBU is responsible for signal processing, often referred to as the Data Unit (DU). In C-RAN, RRHs are placed at the RAN, while BBUs can be placed at a centralized cloud-based infrastructure. This architecture provides resource sharing among sites in a virtualized BBU pool, resulting in reduction of network operational costs, since power and energy consumption are reduced compared to traditional RANs [2]. Dynamic C-RAN has emerged as an evolution of C-RAN [3]–[5]. While in C-RAN all BBU functions are moved to a centralized cloud, in Dynamic C-RAN radio functions are distributed in a hierarchical cloud infrastructure along the network [6], [7]. Such distribution results in benefits like adaptability, load balancing, service deployment flexibility, and improved energy efficiency, which are some of the characteristics of 5G [8]–[10].

In Dynamic C-RAN, while some functions (*e.g.,* digital and analog conversion) requires support of dedicated hardware because of performance limitations, others can execute across commodity computer hardware, enabling software-based implementation of functions (*e.g.*, coding and modulation schemes). For this reason, radio functions can be dynamically moved along with the network according to specific service requirements, such as, low latency and high fronthaul data rate. This dynamics takes advantage of cloud computing concepts to provide flexibility, organized in hierarchical data centers [6]. Thus, Dynamic C-RAN also takes advantage of the hierarchical cloud infrastructure to achieve effective placement of radio functions, enabling flexible service provisioning [11]. Additionally, hierarchical cloud infrastructures allows splitting radio functions processing over cloud data centers, reducing the fronthaul data rate requirements [12], [13].

Despite Dynamic C-RAN's benefits, coordinating radio functions over cloud data centers is not a trivial task. Each radio function may have strict requirements, such as, retransmission in the subframe layer that requires latency lower than 1ms, and Hybrid Automatic Repeat Request (HARQ) time in the MAC layer must take less than/equal to 8ms [14]. Academia and industry have been exploiting concepts of Network Functions Virtualization (NFV) to allocate and orchestrate virtualized radio functionalities designed as Virtualized Network Functions (VNF) [15]. As NFV concepts evolved, the importance of orchestrating VNF has increased, leading the NFV ISG to create the NFV Management and Orchestration (NFV MANO) group [16]. NFV MANO envisages the creation of a standard architecture to provide industry support for the management and orchestration of VNFs.

NFV functionalities, such as on-demand services and resources orchestration, may be used in Dynamic C-RAN contexts [17], allowing to adapt, manage, and orchestrate virtualized radio functions [15]. By employing NFV concepts in Dynamic C-RANs, BBUs can be implemented as VNFs [6], which are composed of Virtualized Network Functions Components (VNFC) representing each radio function

Manuscript received February 1, 2019; revised December 3, 2019; accepted January 28, 2020. Date of publication April 8, 2020; date of current version May 21, 2020. *(Corresponding author: Ariel Galante Dalla-Costa.)*

Ariel Galante Dalla-Costa, Lucas Bondan, Juliano Araújo Wickboldt, and Lisandro Zambenedetti Granville are with the Applied Computing Graduate Program, Federal University of Rio Grande do Sul (UFRGS), Porto Alegre 91509-900, Brazil (e-mail: agdcosta@inf.ufrgs.br; lbondan@inf.ufrgs.br; jwickboldt@inf.ufrgs.br; granville@inf.ufrgs.br).

Cristiano Bonato Both is with the Applied Computing Graduate Program, University of Vale do Rio dos Sinos (UNISINOS), São Leopoldo 93022-750, Brazil (e-mail: cbboth@unisinos.br).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSAC.2020.2986689

of the BBU and deployed in Virtualized Network Deployment Units (VDU) [18]. VDUs can be implemented using different virtualization technologies, *i.e.*, hard virtualization and container virtualization. Therefore, an NFV Orchestrator (NFVO) has two primary responsibilities: (*i*) orchestration of resources across the multiple virtualized infrastructures and (*ii*) life-cycle management of VNFs and network services, including the proper placement of VNFCs and VDUs, considering the restrictions of the radio functions.

Orchestration platforms, such as Cloud4NFV [19], T-NOVA [20], OpenBaton [21], and KORA [22] handle VNFs as atomic unities, *i.e.,* indivisible elements. However, such platforms do not take full advantage of NFV orchestration concepts. A virtualized BBU may be deployed in two different ways: implemented in a radio front-end to save network resources (*e.g.,* data rate), as proposed by Mishra *et al.* [22], or split over the Dynamic C-RAN infrastructure to balance the processing load over the available clouds [3]. To satisfy all constraints of Dynamic C-RAN, NFVO should consider the allocation and life-cycle management of virtualized radio functions with each resource constraint.

Despite the large number of works covering the orchestration of radio functions and the VNF placement problem [3]–[5], there are no investigations about the orchestration and VNF placement considering VNFs split into VNFCs. In this article, we cover the lack of NFV Orchestrators aware of VNF composition to deploy VNFCs in hierarchical cloud infrastructures, enabled by an experimental Dynamic C-RAN environment. We advance the research in NFV orchestration for Dynamic C-RAN proposing Orchestra, a split-aware NFVO for Dynamic C-RAN with support to customizable orchestration algorithms for different radio function splits. In this way, Orchestra enables Network Operators to write customizable algorithms to cover all decision-making involved by the virtualized infrastructure scenario. Orchestra is analyzed in a real experimental network scenario, considering the use of virtualized radio functions using the FUTEBOL's Container and Orchestration Provisioning Architecture (COPA) [23]. A conceptual algorithm for inter-cloud load balancing is used to evaluate Orchestra operation, as a case study. We show Orchestra correctly instantiates and migrates VNFCs following the decision-making written in the customizable orchestration algorithms, with an petty impact on the VNFC's deployment and migration time.

The remaining of this article is organized as follows. In Section II, we present background and related work regarding Dynamic C-RAN and the applicability of NFV concepts in such environments. In Section III, we propose the Orchestra architecture with a conceptual orchestration algorithm for demonstrating how Orchestra properly performs VNFC orchestration following the algorithm defined. In Section IV, the results obtained through experimental evaluations are presented and discussed in detail. Finally, in Section V, we address final remarks and future work.

II. Background and Related Work

The main benefit of the hierarchical nature of Dynamic C-RAN is that it allows changing the placement of radio functions, enabling flexible service provisioning [11]. The

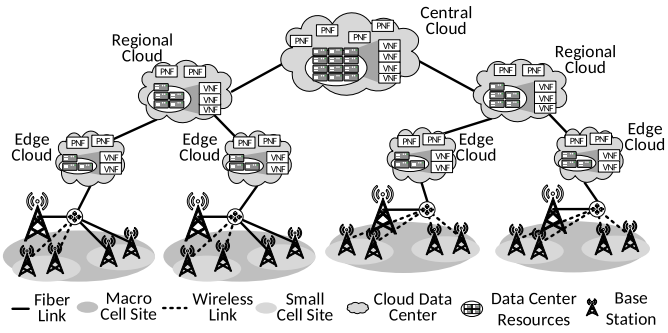

Fig. 1. Example of dynamic C-RAN hierarchical infrastructure.

Dynamic C-RAN scenario depicted in Fig. 1 is composed of hierarchically organized cloud data centers [6]. These clouds process both legacy Physical Network Functions (PNF) and generic virtualized functions known as VNFs. The central cloud provides the highest resource volume in the infrastructure, while the regional cloud and edge cloud (which have fewer computational resources than the central cloud) are closer to the radio front-end. Considering such a hierarchical setup, a virtualized BBU can be deployed at the regional or central cloud for load balancing purposes, considering an increasing number of active users on a specific edge [11]. Furthermore, different policies can be quickly deployed, such as customized load balancing strategies, and energy-saving policies, given the ability to distribute radio function services among the clouds [6].

Bartelt *et al.* and Liu *et al.* [13] discuss the advantages of splitting radio functions in local and remote processing, showing that splitting radio functions can reduce fronthaul's data rate requirements. Likewise, Wubben *et al.* [24] analyze the impact of fronthaul data rate for each split option using Radio Access Point (RAP) by showing the data rate consumed in the fronthaul by each radio functions, such as MAC, Soft-Bit, RX Data, Subframe, and In-phase & Quadrature (I/Q). When compared to the centralized approach of C-RAN, Dynamic C-RAN presents many advantages, such as energy efficiency, Capital and Operational Expenditure (CAPEX and OPEX) savings, execution-time adaptability, and service provisioning flexibility. For this reason, 5G mobile networks are being deployed in Dynamic C-RAN [6], [8]. Although the studies mentioned earlier highlight the importance of splitting radio functions, their authors do not consider the orchestration of split radio functions over the cloud infrastructure.

Coordinating VNFs in Dynamic C-RAN is not an easy task, mainly because of strict constraints, such as the mobility of User Equipment (UE) throughout the infrastructure, requiring the network to be adaptable according to variations of user demands [14]. For this reason, both academia and industry have been exploiting concepts of NFV to manage and orchestrate virtualized radio functions [13], [25]. Abdelwahab *et al.* [17], Liu *et al.* [13], Marotta *et al.* [26], and Riggio *et al.* [27] advocate that NFV can have an essential role in managing and orchestrating virtualized wireless radio functions [15]. In this way, Dynamic C-RAN can consider virtualized BBUs as VNFs, with each radio function becoming a VNFC [18]. VNFs are the central elements of the ETSI's NFV architectural framework, which are deployed

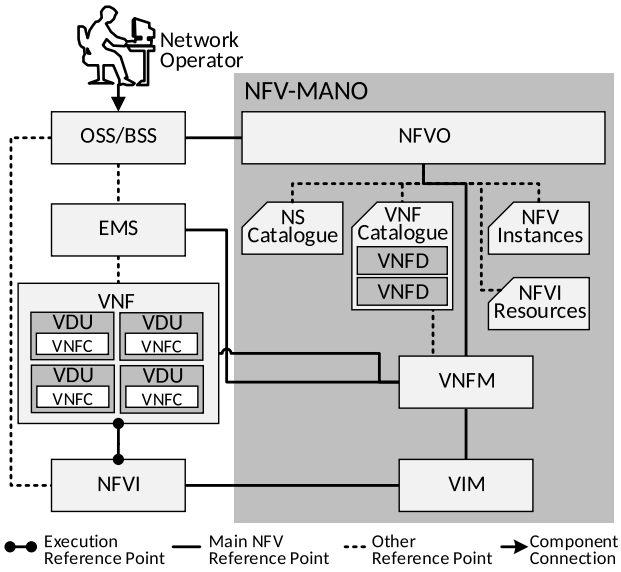

Fig. 2. NFV architectural with detailed MANO blocks as defined by ETSI.

at NFV Infrastructure (NFVI), as depicted in Fig. 2. VNFs can be split into VNFCs, which are the smallest elements of VNFs. VNFCs, in turn, are deployed in virtualization elements known as VDUs, which are directly mapped into a dedicated virtualized node (*e.g.*, virtual machine or container) [28]. To deploy VNFs, each one is described through a VNF Descriptor (VNFD) that details how each VDU is connected to compose each VNF. In this way, each VNFC can represent a virtualized radio function, as discussed by Abdelwhab *et al.* [17], that proposes an architecture for NFV in 5G networks, considering the employment of virtualized radio functions following the ETSI's NFV MANO architecture. Moreover, Abdelwhab *et al.* address an NFVO for 5G environments to orchestrate virtualized resources and manage information flows among VNFs, VNFCs, and Physical Network Functions (PNFs).

All NFV MANO blocks are connected to the NFV architectural framework. For example, the Operations/Business Support System (OSS/BSS) is responsible for controlling functions and managing register rules. NFVO is responsible for bringing intelligence to service delivery and the VNF composition process, handling information about VNF composition, instantiation, and operation. In this case, the employment of NFVO in Dynamic C-RAN leads to two main challenges: (*i*) how to deploy, distribute, and orchestrate VNFs in a Dynamic C-RAN infrastructure, and (*ii*) how to guarantee satisfactory performance in the placement and life-cycle management of VNFs, supporting all requirements of Dynamic C-RAN. These challenges are covered by the state-of-the-art considering formal evaluations and experimental proposals.

A formal evaluation of VNF placement was proposed by Moens and Turck [29], considering both VNFs and legacy physical functions. Bhamare *et al.* [30] propose a combinatorial optimization model based on a heuristic, for a 5G C-RAN, but without considering Dynamic C-RAN splits. In the same way, Bagaa *et al.* [31] propose an algorithm for the placement of virtual instances over federated clouds, based on Mixed Integer Linear Programming (MILP). However, the authors also do not consider the Dynamic C-RAN splits. Moreover, proposals focused on SFC are also found in the literature, including the autonomic placement of virtual nodes and service allocation [32]–[36], [36]–[38]. In the same way, Nam *et al.* [39] propose a clustered NFV service chaining (cNSC) scheme that computes the optimal number of clusters to minimize the end-to-end time of Mobile Edge Computing (MEC) services. In the context of wireless networks, Riggio *et al.* [27] proposed an orchestrator for WLAN, without the division of functions in components.

TABLE I
SUMMARY OF VNF AND SFC ORCHESTRATION SOLUTIONS CAPABILITIES

References	End-to-end service delivery	Marketplace	Life-cycle management	Optimizing module
[20] T-NOVA	✓	✓	VNF and SFC	✓
[40] FENDE		✓	VNF and SFC	
[41] OPNFV	✓		VNF	
[19] Cloud4NFV	✓	✓	VNF and SFC	✓
[19] Tacker	✓		VNF	
[21] Open Baton	✓		VNF	
[22] KORA			VNF	✓

Based on ESTI NFV MANO specifications, some experimental proposals have emerged, focusing on the management and orchestration of VNFs. T-NOVA [20] presents an NFVO aimed to provide a VNF marketplace, supporting end-to-end services. Similarly, the FENDE project [40] provides a marketplace and ecosystem for the distribution and execution of VNFs and the composition of Service Function Chains (SFCs). OPNFV [41] is a MANO NFV platform that aims to integrate with other open-source projects to meet different NFV requirements. OPNFV works closely with ETSI to encourage the consistent implementation of open standards. OPNFV promotes an open-source network that brings companies together to accelerate innovation, as well as market new technologies. In the same way, Soares *et al.* [19] present an NFVO for end-to-end services orchestration, proposing Cloud4NFV, a platform that manages and orchestrates VNFs following standard ETSI MANO specifications. Similarly, OpenStack Tacker [42] is an NFV platform for generic VNF orchestration, providing a functional stack to orchestrate end-to-end network services using VNFs. Likewise, Open Baton is an open-source NFVO solution focusing on VNF performance and portability using the generic Element Manager System (EMS) and generic VNFM [21]. In a 5G scenario, KORA [22] was proposed as an NFV dynamic resource management framework for 5G C-RANs, focused on reducing energy consumption and maximizing service quality to end-users. We summarize some of the most relevant VNF and SFC placement solutions found in the literature in Table I, highlighting their main capabilities.

T-NOVA and Cloud4NFV provide support to end-to-end services and provide life-cycle management for VNFs and SFC together, as can be seen in Table I. Moreover, T-NOVA, Cloud4NFV, and FENDE offer Marketplaces of network functions. Considering the optimization perspective, T-NOVA, Cloud4NFV, and KORA have modules with optimization algorithms. However, these works do not consider VNFCs

to represent split radio functions in C-RAN scenarios, *i.e.*, a split-aware orchestration. Even though the split of radio functions brings substantial gains for wireless networks [17], NFV MANO research efforts do not focus on the composition of split VNFs in the decision-making process of Dynamic C-RAN environments and the benefits of orchestrating split VNFs. In this context, our experimental proposal focuses on two-folds (*i*) the advantages of dividing VNFs into VNFCs for Dynamic C-RANs and (*ii*) the lack of NFVO with support to the orchestration of VNFCs. Therefore, we propose Orchestra: a split-aware NFV Orchestrator with support to customizable orchestration algorithms designed by network operators, supporting different radio function splits. Orchestration algorithms can consider various VNF compositions (*i.e.*, VNFCs) during the placement decision, optimizing resource allocation, or energy consumption.

III. Orchestra NFVO

To fulfill the lack of NFVO solutions for a customized orchestration of VNFs, in this section, we describe Orchestra, a split-aware NFV orchestrator for Dynamic C-RAN. In Subsection III-A, the main functional blocks of the Orchestra are described in detail. Moreover, an orchestration algorithm designed as a proof-of-concept for Orchestra is presented in Subsection III-B.

A. Orchestra Architecture

Orchestra was designed to fulfill the lack of NFVOs that take into account VNF composition in the decision mechanism. In Orchestra, network operators can write different algorithms for Dynamic C-RAN environments, focusing on employing personalized orchestration algorithms based on network operators' requirements. Orchestra differs itself from other NFV orchestrators by dealing with the granularity required to orchestrate VNFs with multiple components (VNFCs). In our previous work, we proposed and evaluated the fine-grained orchestration of VNFs in a simulated environment [18]. In contrast, in this work, Orchestra is not only employed in an experimental environment, orchestrating realistic virtualized radio functions, but it also provides customized algorithms-based orchestration for network operators.

Fig. 3 shows the functional blocks of Orchestra and their relationships with the NFV blocks defined by ETSI. Network operators interact with Orchestra using OSS/BSS support, which allows operators to develop, create, and register VNFs and network services. Network operators can write custom algorithms for the Optimizing Manager block to optimize provisioning and positioning of VNFs and network services, according to specific requirements of the underlying infrastructure. For instance, functions with low latency requirements that need to be moved to the edge when bottlenecks are detected at the central, and cost-expensive VNFs that may cause overhead in the edge of the network, which have to be moved to the regional or central in Dynamic C-RANs. All functional blocks of Orchestra are described below.

1) Catalogs: in this block, storage and provisioning of data related to VNFs, services, VNFCs chaining of each VNF, and resources available in the NFVI are performed. Furthermore,

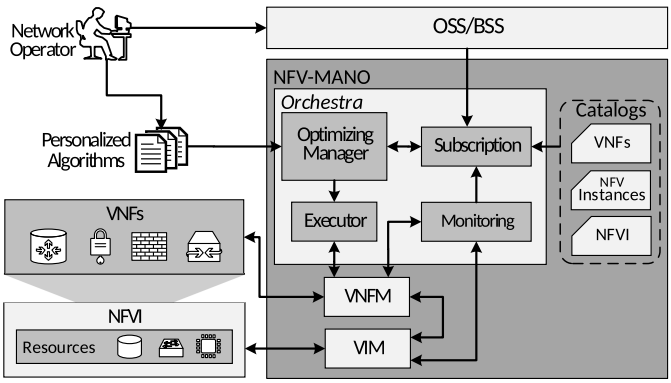

Fig. 3. Orchestra architecture.

data from VNF instances are handled in the Catalogs, which can help network operators re-instantiate VNFs in case of failures. The Catalogs block connects to the Subscription block, which provides access to the information in the catalogs through an API.

2) Subscription: this block is responsible for NFV orchestration subscription and registration, maintaining the state of infrastructure's resources, VNFs, and network services. The Subscription block access catalogs of services and network functions, handling information related to resources, registered VNFs, and their instances are executing across the NFVI. This block implements a database that stores VNFs instances and states. Moreover, it also stores information about monitoring, providing an interface between data and collectors (*e.g.,* sFlow *collector*). This information is used to deploy new functions and services, as well as to manage existing network functions and services.

3) Optimizing Manager: this block enforces the decisions about VNFCs and VNFs taken in the algorithms written by the network operator. Additionally, this block provides information to the network operator's algorithms (*e.g.,* CPU, memory and disk usage, latency, throughput, and network delay time), which are provided by VNFMs and VIMs. The Optimizing Manager block receives input algorithms written by the network operator or software vendors to cover network objectives and can be adapted as needed.

4) Executor: this block is responsible for translating orchestration commands for specific VNF platforms, applying in the VNFM and VIM the operations generated in the Optimizing Manager block. Moreover, this block also configures VIMs and VNFMs (*e.g.,* network setup, CPU, and memory usage thresholds). Once the operations are defined in the Optimizing Manager block, the Executor block receives such actions and translates them into native API calls of each VIM and VNFM platform so that these elements can execute the operations over VNFs and the NFVI.

5) Monitoring: this block is responsible for monitoring and collecting information regarding VNFs operation and NFVI resources, forwarding all collected data to the Subscription block. The information monitoring is performed by polling, periodically requesting VNFMs and VIM operational states of VNFs and NFVI. Keeping VNF and NFVI operational information updated is crucial to achieving results in the

Optimizing Manager block, so using polling intervals is recommended in the Monitoring block.

Both Executor and Monitoring blocks connect to VIM and VNFM, which are responsible for VNFs life-cycle management and NFV resource management, respectively. Orchestra can operate with any VIM and VNFM through an abstraction layer, present in both Executor and Monitoring blocks. Such an abstraction layer translates actions into specific VIM and VNFM functions, abstracting all delegation or optimization processes generated in the optimization algorithm. To do so, Orchestra uses standard calls of VNFM and VIM communication APIs. Moreover, both VNFM and VIM must support such a feature in their communication APIs to implement a new feature in Orchestra. For example, to copy a running VNFC from a cloud to another one while keeping its internal state, VNFM must support such a feature in its API so that Orchestra can request this operation to the VNFM.

Orchestra provides an API for the delegation of actions to be taken based on the execution of network operator's algorithms. For example, after an algorithm finding a CPU usage bottleneck in some cloud (*e.g.,* edge, regional, or central), the algorithm requests the Orchestra API to migrate some VNFCs. In this way, the orchestrator is in charge of migrating and re-positioning the VNFC, re-allocating and re-establishing the service, and the communication between VDUs. Each algorithm can consider different parameters to make its decision, so the network operator decides which parameters are significant to be analyzed by the algorithm. Orchestration algorithms can be focused on load balancing strategies, resource scaling, and change the composition of VNFs.

Orchestra has two operational modes. In the first one, algorithms are dynamically loaded, *i.e.,* the network operator registers custom algorithms in Orchestra, which are loaded automatically. Orchestra does not trigger algorithms but loads the algorithms at the same time. Actions defined in an algorithm are executed when monitored information collected by Orchestra triggers such actions in the algorithm (*e.g.,* when some cloud reaches 100% of CPU usage). Orchestra loads the algorithms during its startup and updates its list of available algorithms every 15 minutes. We chose 15 minutes based on tests where this value covered a full VNF migration time with a small safety margin. Such value can be changed by network operators whenever it is needed.

In the second operational mode, the algorithms are directly moved to the local machine of the network operator or software vendor. Then, the algorithms are imported into the Orchestra library, which contains all orchestration methods (*e.g.,* migrate, deploy, delete). This operational mode allows the network operator to execute algorithms directly from its local machine by remotely calling the functions of the Orchestra library. In this case, Orchestra becomes an abstraction layer between VIM and VNFM and, from the calls made by the algorithm of the network operator, makes the application in the infrastructure managed by Orchestra, abstracting every process of the network operator. However, this operational mode does not provide automatic algorithm loading.

TABLE II
FUNCTIONS OF THE ORCHESTRATOR TO THE NETWORK OPERATOR

Function Name	Functionality
VNF_DEPLOY(vnf_name, vnfcs_list)	Implements VNF over multiple clouds
VNF_DELETE(vnf_name)	Remove a VNF and all VNFCs
VNFC_MIGRATE(origin_cloud, vnfc_name, destinity_cloud)	Migrates VNFC from source to destination cloud
CLOUD_LIST()	Returns the list of clouds registered in the infrastructure
CLOUD_INFORMATION()	Returns information about each one of the clouds (e.g. memory, CPU)
VNFCS_INFORMATION()	Returns information about each one VNFC
FLUX_INFORMATION()	Returns information about traffic and data rate between VFC and clouds

Orchestra is designed to orchestrate fine-grained radio functions split into VNFCs over clouds (*e.g.,* edge, regional, and central) [43]. For each infrastructure and network, the operator can write algorithms or obtain others from software vendors.

B. Orchestration Algorithm

One of Orchestra's input is the algorithm customized to achieve the goals of network operators or software vendors. The algorithm can adapt the VNF deployment, removing, and reallocating VNFCs. Table II shows the methods and functions provided by Orchestra to network operators to write their customized algorithms. In the left column, the methods and parameters of each function can be seen. In the right column, a description of the functionality of each method is presented. For example, in the first row, the VNF_DEPLOY function is designed to utilize a VNF with VNFCs in cloud infrastructure, and can deploy each VNFC in a different cloud or all VNFCs of a VNF at the same cloud, depending on the parameters passed to the function. Following the life-cycle of VNFs, the VNF_DELETE function is designed to exclude a VNF and all VNFCs that compose such VNF. The VNFC_MIGRATE function is designed to migrate VNFCs, *i.e.,* from one cloud to another. Functions CLOUD_LIST, CLOUD_INFORMATION, VNFC_INFORMATION, and FLUX_INFORMATION are designed to return collected data from the NFVI and VNFCs, provided by the Subscription and Monitoring blocks.

Algorithm 1 exemplifies a conceptual algorithm focused on migrating VNFCs when a cloud is overloaded in terms of CPU usage, with a threshold-based decision, set by the network operator. N contains the list of clouds and CPU variable stores information about the CPU usage in the clouds, both provided by Orchestra function. Moreover, $VNFCs$ variable contains information regarding VNFCs, such as CPU and memory usage. The $threshold$ defined is designed to support the decision process and to choose whether a migration action is necessary. This threshold represents the maximum amount of CPU occupancy, which is supported in each cloud. Furthermore, it defines whether a server is high CPU occupying or not, and hence triggers the execution of subsequent actions. Similarly, to continue maintaining the service, the algorithm could observe flow data and consequently make different decisions, such as stopping a VNFC or the whole VNF.

Algorithm 1 Conceptual Load Balancing Algorithm Between Clouds

```
N ← CLOUD_LIST();
CPU ← CLOUD_INFORMATION(cpu);
VNFCs ← VNFCS_INFORMATION();
threshold ← 95;
discovery all clouds overloaded than the threshold in cpu usage;
find all VNFC that needs to be migrated;
search next available cloud to migrate each VNFC;
foreach necessary VNFC migration do
VNFC_MIGRATE(cloud_origin, vnfc, cloud_destiny);
if no available clouds then
    shows a message "no available clouds to migration"
end
```

IV. Case Study and Evaluation Model

Orchestra was evaluated in an experimental environment through the design and analysis of a proof-of-concept. The details of the proof-of-concept and the evaluation scenario used to evaluate Orchestra are presented in Subsection IV-A. Next, the obtained results are presented and discussed in Subsection IV-B.

A. Proof-of-Concept and Evaluation Scenario

To orchestrate fine-grained radio functions split over Dynamic C-RAN infrastructures, Orchestra uses different tools. Network operators or software vendors can write algorithms to orchestrate functions over their infrastructures. Therefore, to implement personalized algorithms, Orchestra natively provides an API with support to *Python* programming language. To support additional programming languages, Orchestra presents an integrated JSON API to interact with shell-based scripts. For instance, an algorithm can call the function *shellexecute* to execute specific shell commands using Python. Through the JSON API, the function *Runtime.exec* has the same functionality using JAVA language.

Orchestra's blocks use different tools to provide information as input to personalized orchestration algorithms, as described in Section III. The Monitoring block is implemented using the Simple Network Management Protocol (SNMP), monitoring cloud statuses, such as CPU, memory, and disk usage. For network monitoring, Simple Flow (sFlow) and OpenFlow are used to send flows that are generated between VNFCs in the cloud. Finally, VNFs information is monitored using VNFMs' monitoring API.

Our case study is based on radio functions provided by the H2020 Wishful project,[1] enforced on Dynamic C-RAN scenarios [43], [44]. These radio functions enable the deployment of a Dynamic C-RAN experimental environment, where each radio function becomes a VNFC deployed using Linux Containers (LXD).[2] These containers offer a user experience similar to VMs, with the instantiation of LXD images based on predefined templates available in a repository. Fig. 4 (A) demonstrates how VNFCs splits are implemented in our scenario, and the part (B) shows how radio functions were implemented as VNFCs. We highlight that not all radio functions were implemented, but only the main functions to validate our case study, following the approach of Kist *et al.* [45]. For example, the MAC layer was not implemented, since Orchestra was tested in an end-to-end scenario.

In our experimental scenario, 12 BSs are used, requiring 24 USRP devices to handle all radio communication (downlink and uplink traffic). In the Base Station (BS) downlink, radio functions processing (BBU) are split into 3 VNFCs: (*i*) *Frame Generation*: responsible for *CRC Calculation*, *Header Generation*, and *Payload Generation*; (*ii*) *Modulation*: responsible for *Modulation*; and (*iii*) *OFDM*: responsible for *OFDM Carrier Allocation*, *FFT* and *CP Adder*. For BS uplink, the processing is made by an atomic VNF, *i.e.,* just one VNFC is allocated to process *CP Decision*, *Header/Payload Demux*, *Payload Processing*, *Header Processing*, *Payload Decoding*, and *CRC Verification*. Based on the implementation described previously, a full BS end-to-end communication is deployed with both uplink and downlink. The implemented radio functions are based on those defined by Wubben *et al.* [24], *i.e.,* I/Q, Subframe, TX/RX Data, and Soft-Bit, as depicted in Fig. 4.

An additional VNFC is attached to the end of the VNFC chain, responsible for emulating a Universal Software Radio Peripheral (USRP), which receives signal samples generated in previous VNFCs in the chain and forwards these samples. The USRP emulation was performed using signal processing blocks provided by GNU Radio [46], a framework that enables the deployment of specialized software for wireless signal processing. USRP receives a stream of digital signal samples from the VNFC connected to it. Therefore, USRP converts these samples into radio-frequency signals, working as a Radio Remote Head (RRH). At the receiver side, a USRP receiver is configured to receive wireless signals on the same frequency as the transmitter, capturing analog signals and digitizing them, originating a digital signal stream. This stream is sent to the GNU Radio software that implements the receiver equipment. The original digital signal (generated by the VNFC) and the signal generated by the USRP receiver must be identical in perfect communication, *i.e.,* without any degradation effect.

Given the high number of USRP devices, we used a USRP emulator, *i.e.,* GNU Radio software, without a real USRP device attached to the server, since GNU Radio performs the signal processing as a USRP device. The differences between the emulated scenario and the one using actual USRP devices are shown in Fig. 5. In the second column, a real USRP scenario is illustrated, with the flow of signal samples performed by a real USRP between the RRH and the receiver previously described. In the emulated scenario (third column), digital signal samples are sent directly through a wired connection. As such, the USRP emulator forwards digital signal samples directly to the software that implements the receiver equipment, *i.e.,* digital-to-analog conversion at the sender, and analog-to-digital conversion at the receiver are both removed.

[1] http://www.wishful-project.eu/
[2] https://linuxcontainers.org/lxd/

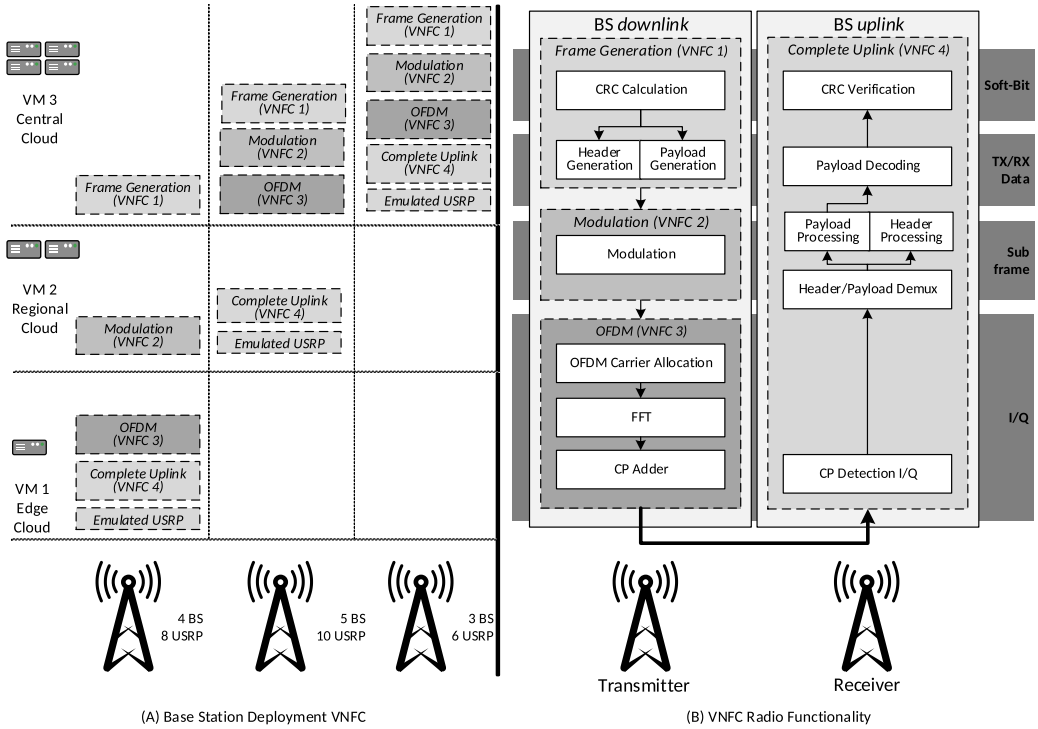

Fig. 4. BS's radio functions split (downlink and uplink).

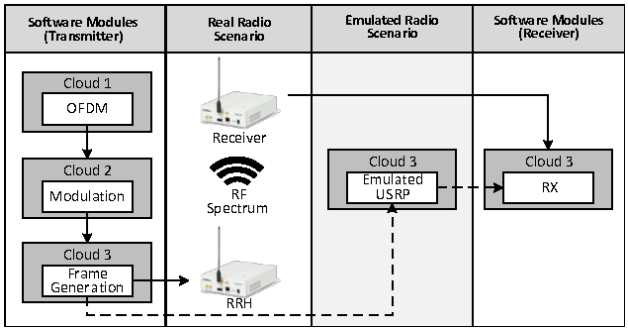

Fig. 5. VNFCs in the real and emulated USRP scenarios.

Fig. 6 presents the tools used in the experimental scenario. For virtualized infrastructure management and control, the *Aurora Cloud Manager* [47] was used as VIM. Aurora is a cloud platform of Infrastructure as Service (IaaS) that enables flexible resources managing. In this article, Aurora provides Virtual Machines (VM)s management, as well as OpenFlow-based network connectivity between VMs. Using Aurora, 3 VMs were deployed, each one representing a hierarchical level of Dynamic C-RAN clouds. Moreover, Fig. 6 shows $VM1$, $VM2$, and $VM3$ representing respectively edge, regional, and central clouds. These VMs have LXD to support VNFC/VNF deployment using containers.

Table III describes the resource distribution over the clouds used in the experimental evaluation. The edge cloud, *i.e.,* VM 1, has fewer resources than other clouds, providing 1 CPU core and 4 Gb of RAM. The regional cloud, *i.e.,* VM 2, has 2 CPU cores and 8 Gb of RAM. The central cloud, *i.e.,* VM 3, with largest capacity, has 4 CPU cores and 16 Gb of RAM. All VMs are virtualized in an Intel(R) Xeon

TABLE III
AVAILABLE RESOURCES IN EACH VM (CLOUD)

VM Configuration	VM 1 (Edge)	VM 2 (Regional)	VM 3 (Central)	VM 4 (Orchestrator)
CPU (cores)	1	2	4	2
RAM memory (Gb)	4	8	16	2
Disk (Gb)	200	200	200	50

$1.90GHz/6$ processor, with 6 cores, and each VM was allocated with 200 Gb of available storage. VNFCs are deployed in LXD Linux containers, which are chained together to create a VNF representing a BS with both downlink and uplink capabilities. As can be seen in Fig. 4, radio functions are placed in VM 1, VM 2, and VM 3, according to the network operator's algorithms. Furthermore, VMs have SNMP agents used to monitor the available resources on them.

As illustrated in the center of Fig. 6, an OpenFlow-based SDN is used to provide connectivity to VMs, using the FloodLight OpenFlow controller hosted in VM 4. Moreover, sFlow is used to collect network flows between containers and VMs, acquiring information regarding data rate consumption of the fronthaul, and providing such information to the orchestration algorithm. VM 4 is responsible for managing and orchestrating virtualized radio functions, with 2 Gb of RAM, 2 CPU cores, and 50 Gb of storage.

For container management, the *Container Orchestration and Provisioning Architecture* (COPA) is used, hosted in VM 4. COPA is an LXD container management engine designed in the context of the FUTEBOL project [23] that provides an abstraction layer between LXD containers and their management APIs. In this article, COPA is used as VNFM, providing life-cycle management of VNFCs and VNFs and supporting

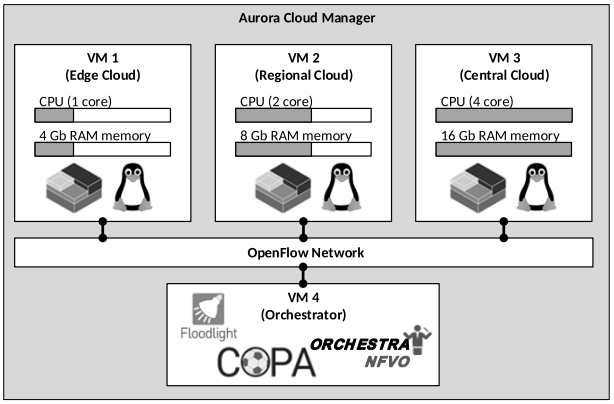

Fig. 6. Experimental scenario using wireless network functions.

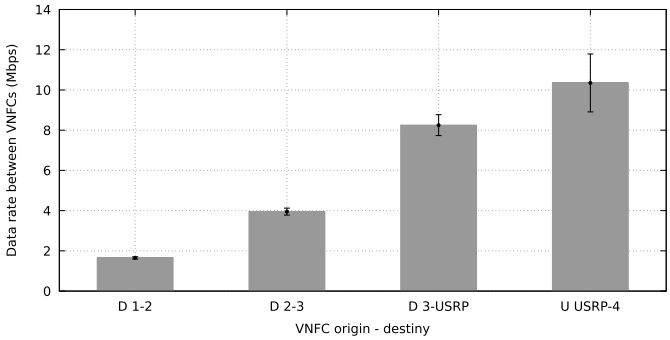

Fig. 7. Fronthaul data rate between VNFCs downlink (D) and uplink (U).

TABLE IV
PARAMETERS AND TOOLS USED FOR THE EXPERIMENTAL EVALUATION

Parameter/Tool	**Value**
Base Stations (VNF)	12
VNFCs (Containers)	5-60
Container Engine	LXD
VMs	4
Network Flow Collector	sFlow
Traffic analysis Tool	NFSen
Monitoring Tools	COPA REST API and SNMP agent
OpenFlow Controller	FloodLight
Host Configuration	Intel(R) Xeon 1.90GHz 6 processor/6 cores; 32 GB; Linux 64 bits
Emulated USRPs	12 Receiver/Transmitter
VNFM	COPA
VIM	Aurora
Confidence Interval	95%

VNF orchestration. Therefore, when the orchestration algorithm defines actions, Orchestra (also hosted on $VM\ 4$) sends such actions to COPA, which performs such operations over VNFCs and VNFs (*e.g.,* migration of one or more VNFCs from one cloud to another).

B. Evaluation Results and Discussion

The orchestration algorithm to maximize the use of computational resources in the clouds was used for VNFC placement. During the experiments, the algorithm has deployed: 4 VNFs considering 3 clouds (*i.e.,* VNFCs on the edge, regional, and central clouds, using first on the edge cloud until all resources being allocated); 5 VNFs considering 2 clouds (*i.e.,* VNFCs on the regional and central clouds until resources exhaustion on the regional cloud); and 3 VNFs on the central cloud (*i.e.,* all VNFCs deployed at central cloud until its processing reach 100%), totalizing 12 BSs and 60 Linux containers over the infrastructure. The experiments described in the following were repeated 20 times, achieving a confidence interval of 95%. The data rate was measured using sFlow together with NFSen for graphics generation. Floodlight was used as the OpenFlow controller for network forwarding control, and VMs resource monitoring was performed using SNMP and COPA VNFM API. All parameters and tools used to evaluate Orchestra are summarized in Table IV.

The first experiment was designed to analyze the data rate between VNFCs, aiming to identify which are the connection points between VNFCs that present higher and lower data rates. Fig. 7 summarizes the obtained results, with the y-axis representing the average consumption in Mbps between VNFCs, and the x-axis showing the VNFCs division. Letters 'D' and 'U' in the bars' labels represent downlink and uplink traffic, respectively.

The downlink data rate between VNFC 1 and VNFC 2 was 1.16 Mbps on average, while the data rate between VNFC 2 and VNFC 3 showed an average of 3.95 Mbps, and 8.25 Mbps between VNFC 3 and USRP. In the uplink analysis, an average of 10.35 Mbps was obtained between USRP and VNFC 4. Note that the evaluation does not consider data used in the TCP responses, *i.e.,* ACK messages, considering only uplink and downlink flows. Such results show that higher data rates are present between USRP and VNFC 4, where only one VNFC is used to process all uplink traffic, different from the downlink traffic, where all radio functions are split into three different VNFCs. Moreover, the data rate between VNFCs increases gradually in downlink flows (almost two times more significant from VNFC 1 and VNFC 2 and from VNFC 2 to VNFC 3). Such growth is directly related to the communication type between radio functions. This growth occurs due to the transformation of bits during their processing in each container. At the beginning of the communication, transmitted bits represent only end-user information. At the end of the communication, end-users bits are represented by IQ samples, containing more information regarding data transferred in the fronthaul.

The next experiment was performed to evaluate the end-to-end communication among VNFCs. Such an analysis is essential to observe the impact of BSs instances over the overall communication. We measured the latency behavior in the face of an increasing number of BSs instances, *i.e.,*, VNFCs performing the BSs functions. Fig. 8 shows the results of the end-to-end VNFC communication evaluation, with the latency in milliseconds represented on the y-axis and x-axis showing the number of VNFCs instantiated to perform BSs functions.

When considering 5 VNFCs, latency was 32ms on average, increasing to 113ms when 60 VNFCs were instantiated, showing that the latency increases as the amount of available processing capacity decreases. A fundamental factor directly related to the latency growth is the amount of traffic passing through the fronthaul, which requires more processing in the

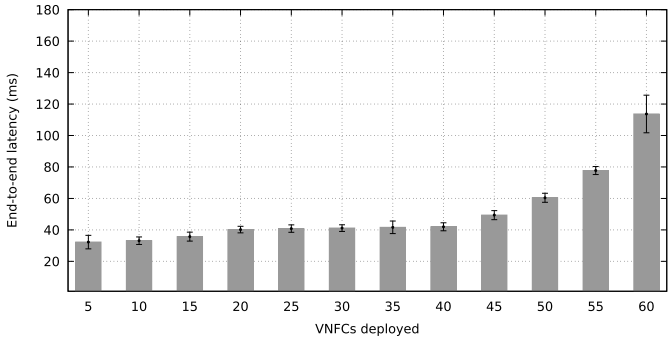

Fig. 8. VNF End-to-end latency.

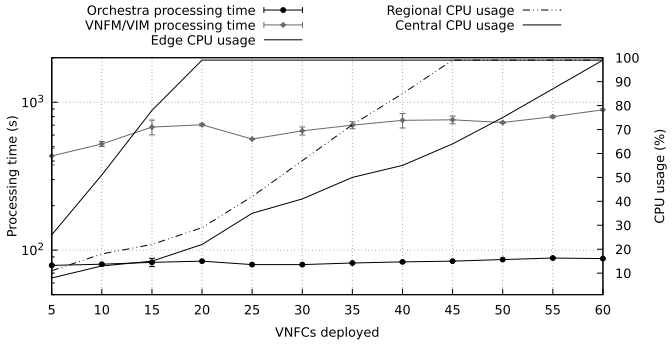

Fig. 9. VNF deployment time.

switches and clouds. Another factor directly related to the high latency increase was that 60 VNFCs were hosted in the same physical machine, with limited computational resources. In our evaluation, we aimed to demonstrate the viability of Dynamic C-RAN scenarios with split VNFs. Therefore, when implemented in a production environment, with higher computational resources, the latency tends to decrease.

Next, we evaluated the time required by Orchestra to deploy VNFCs in the cloud infrastructure in the face of an increasing number of VNFCs, which is an essential metric because BSs must be operational as soon as possible to avoid losses in traffic. The results are shown in Fig. 9, detaching the deployment time of the VNFM used (*i.e.,* COPA and LXD) from the orchestration time of Orchestra. Fig. 9 also shows the CPU usage growth on edge, regional, and central clouds as the number of deployed VNFCs increases. The CPU usage is a key parameter in the deployment process, considering a deployment strategy where a cloud does not receive more VNFCs when it reaches 100% of CPU usage. The VNFM deployment time refers to the amount of time to deploy VNFs and their respective VNFCs (*i.e.,* the LXD containers) on the clouds and start their operation, *i.e.,* VNFCs be ready for Orchestra management, without any configuration. The orchestration time is the time required by Orchestra to establish the connection with VNFCs, configure them, and start traffic among functions. The total deployment time of VNFs is the sum of VNFC deployment performed by the VNFM and the orchestration time performed by Orchestra.

In Fig. 9, the left y-axis shows the average time (in logarithmic scale) to deploy a VNF, *i.e.,* 5 VNFCs, while the right y-axis shows the CPU usage in each cloud, and the x-axis

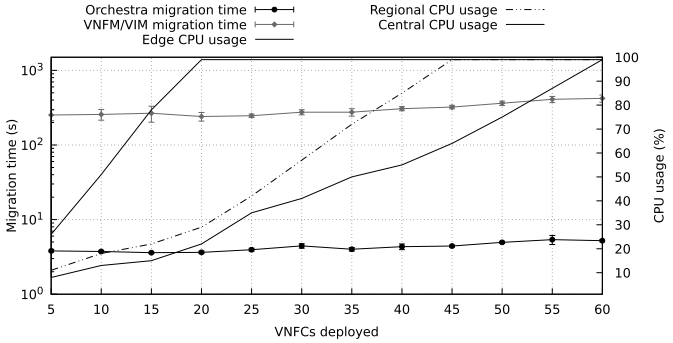

Fig. 10. Migration time of VNFC from origin to destiny clouds.

represents the number of VNFCs used in the infrastructure. To deploy 5 VNFCs, the VNFM needed 433.56 seconds, and 888.96 seconds to deploy of 60 VNFCs, presenting an exponential growth of the deployment time. Regarding configuration time, Orchestra needed 78 and 88 seconds to configure 5 and 60 VNFCs, respectively. Observing the results, we can conclude that, as the CPU usage in the cloud being used for deployment reaches 100%, the VNFC deployment time increases, as can be seen with 20 and 45 VNFCs in Fig. 9. Right after another cloud is selected to deploy VNFCs, the deployment time drops, as can be seen in the cases with 25 and 50 VNFCs. The growth in configuration time was smaller than in deployment time, showing that the time requested by Orchestra to fully configure a VNFC did not impact the VNF deployment process.

The last evaluation was designed to analyze the migration time of VNFCs across the cloud infrastructure. First, VNFCs are instantiated in the whole infrastructure, *i.e.,* considering edge, regional, and central clouds, giving priority to clouds closer to the USRP (*i.e.,* higher priority to the edge than to the central cloud). As CPU usage in the edge reaches a threshold (100%, for instance), the deployment priority is then given to the next cloud in the C-RAN hierarchy, *i.e.,* regional cloud. Once the orchestration algorithm defines a migration, the migrated VNFC must be available as soon as possible in its new location, avoiding both packet loss in communication and resource exhaustion in the original VNFC location. To analyze the migration time, a VNFC was moved from its source cloud to another one and then moved back to the source cloud. Results are presented in Fig. 10, with the migration time in seconds represented in the left y-axis (in logarithmic scale), the CPU usage of each cloud described in the right y-axis, and the number of VNFCs deployed in the fronthaul are represented in the x-axis. The migration time was divided into the time needed by the VNFM and Orchestra to move and the timed needed to reconfigure each VNFC, respectively.

The average migration time of VNFM/VIM varied from 42 to 252 seconds on average, as the number of VNFCs deployed increased from 5 to 60 VNFCs, with Orchestra using 3.8 seconds to configure 5 VNFCs and 5.2 seconds to configure 60 VNFCs. The results showed that as the number of VNFCs instantiated increases, the time required to migrate a VNFC also increases for both deployment and configuration

performed by the VNFM and Orchestra. Such growth in migration time is directly related to the CPU usage in the clouds, since the higher the CPU usage in the cloud, the longer the migration time on that cloud. However, when a cloud reaches 100% of CPU usage, the deployment priority is given to the next cloud, reducing the migration time. This behavior can be observed with the increasing migration time when up to 30 VNFCs are deployed, which is a situation where the edge reaches 100%. Moreover, VNFC deployment priority is given to the regional cloud, dropping the migration time even with an increased number of VNFCs deployed (35 VNFCs).

In summary, the experiments performed showed that both migration time and latency are directly related to the number of VNFCs deployed over the cloud infrastructure. The higher the number of VNFCs, the longer the migration time and latency in the end-to-end communication. Complementary, the deployment time increases as the CPU usage gets close to 100%, with changes in the VNFs split potentially decreasing the deployment time overall.

V. Conclusion and Future Work

Considering the advantages of dividing VNFs into VNFCs for Dynamic C-RANs and the lack of NFVO capable of orchestrating VNFCs, in this article, we proposed Orchestra: a split aware NFV orchestrator for Dynamic C-RAN environments. Orchestra's modular architecture enables network operators to design algorithms to improve flexibility, elasticity, resource consumption optimization, among other benefits for VNFs' life-cycle in Dynamic C-RAN.

Orchestra was evaluated in an experimental scenario, based on the employment of an algorithm designed to migrate VNFCs considering the CPU usage in the C-RAN infrastructure. The results obtained showed the effectiveness of Orchestra in the life-cycle management of VNFCs that compose a VNF with both downlink and uplink capabilities of virtualized wireless Base Station functions. Orchestra can fully instantiate and migrate VNFCs according to the predefined algorithm, with small impact in the overall VNFC's deployment and migration time, corresponding to only 15% of the deployment time for 5 VNFCs and 9% considering 60 VNFC, as well as 11.5% of the migration time considering 5 VNFCs and 1.94% considering 60 VNFCs.

There are several opportunities for future research. First, we plan to fully employ Orchestra in experimental Dynamic C-RAN environments, evaluating the impact of real-time changes in the fronthaul infrastructure (*e.g.,* increase/decrease CPU and memory capacity on edge and regional clouds). Moreover, Orchestra can be employed in different network scenarios, such as optical networks, evaluating Orchestra's capability of fine-grained NFV orchestration with multiple types of VNFs and VNFCs. Additionally, extending Orchestra to operate with legacy network functions is also a challenge. For instance, scenarios where network operators are gradually moving from a traditional network environment with physical network functions and functions executing in atomic virtual machines to an NFV split-aware scenario represent an excellent opportunity for Orchestra employment.

References

[1] Z. Tan, C. Yang, and Z. Wang, "Energy consume analysis for ring-topology TWDM-PON front-haul enabled cloud RAN," *J. Lightw. Technol.*, vol. 35, no. 20, pp. 4526–4534, Oct. 15, 2017.

[2] A. Checko *et al.*, "Cloud RAN for mobile Networks—A technology overview," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 405–426, 1st Quart., 2015.

[3] W.-C. Chien, C.-F. Lai, and H.-C. Chao, "Dynamic resource prediction and allocation in C-RAN with edge artificial intelligence," *IEEE Trans. Ind. Informat.*, vol. 15, no. 7, pp. 4306–4314, Jul. 2019.

[4] Y. Zhou, J. Li, Y. Shi, and V. W. S. Wong, "Flexible functional split design for downlink C-RAN with capacity-constrained fronthaul," *IEEE Trans. Veh. Technol.*, vol. 68, no. 6, pp. 6050–6063, Jun. 2019.

[5] L. Li, N. Deng, and W. Zhou, "Environment-aware dynamic management for energy saving in MIMO-based C-RAN," *IEEE Access*, vol. 7, pp. 77514–77523, 2019.

[6] 5G PPP Architecture Working Group, "View on 5G architecture," 5G PPP, Tech. Rep. 1, 2016.

[7] K. Chen and R. Duan, "C-RAN the road towards green RAN. White Paper," China Mobile Res. Inst., Beijing, China, Tech. Rep. 1, 2011.

[8] 5G PPP Architecture Working Group, "View on 5G architecture," 5G PPP, Tech. Rep. 2, 2017.

[9] I. Afolabi, T. Taleb, P. A. Frangoudis, M. Bagaa, and A. Ksentini, "Network slicing-based customization of 5G mobile services," *IEEE Netw.*, vol. 33, no. 5, pp. 134–141, Sep. 2019.

[10] A. Laghrissi, T. Taleb, M. Bagaa, and H. Flinck, "Towards edge slicing: VNF placement algorithms for a dynamic & realistic edge cloud environment," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2017, pp. 1–6.

[11] M. Peng, Y. Li, J. Jiang, J. Li, and C. Wang, "Heterogeneous cloud radio access networks: A new perspective for enhancing spectral and energy efficiencies," *IEEE Wireless Commun.*, vol. 21, no. 6, pp. 126–135, Dec. 2014.

[12] J. Bartelt, P. Rost, D. Wubben, J. Lessmann, B. Melis, and G. Fettweis, "Fronthaul and backhaul requirements of flexibly centralized radio access networks," *IEEE Wireless Commun.*, vol. 22, no. 5, pp. 105–111, Oct. 2015.

[13] J. Liu, S. Zhou, J. Gong, Z. Niu, and S. Xu, "Graph-based framework for flexible baseband function splitting and placement in C-RAN," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2015, pp. 1958–1963.

[14] J. Ortin, P. Caballero, and R. Imdea, "D5.2 final definition of iJOIN requirements and scenarios," iJOIN Project, Tech. Rep. 1, 2014.

[15] R. Mijumbi, J. Serrat, J.-L. Gorricho, S. Latre, M. Charalambides, and D. Lopez, "Management and orchestration challenges in network functions virtualization," *IEEE Commun. Mag.*, vol. 54, no. 1, pp. 98–105, Jan. 2016.

[16] J. Quittek *et al.*, "Network functions virtualisation (NFV)—Management and orchestration," ETSI NFV ISG, White Paper, Sophia Antipolis, France, 2014. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf

[17] S. Abdelwahab, B. Hamdaoui, M. Guizani, and T. Znati, "Network function virtualization in 5G," *IEEE Commun. Mag.*, vol. 54, no. 4, pp. 84–91, Apr. 2016.

[18] A. G. Dalla-Costa, L. Bondan, J. A. Wickboldt, C. B. Both, and L. Z. Granville, "Maestro: An NFV orchestrator for wireless environments aware of VNF internal compositions," in *Proc. IEEE 31st Int. Conf. Adv. Inf. Netw. Appl. (AINA)*, Mar. 2017, pp. 484–491.

[19] J. Soares, M. Dias, J. Carapinha, B. Parreira, and S. Sargento, "Cloud4NFV: A platform for virtual network functions," in *Proc. IEEE 3rd Int. Conf. Cloud Netw. (CloudNet)*, Oct. 2014, pp. 288–293.

[20] G. Xilouris *et al.*, "T-NOVA: A marketplace for virtualized network functions," in *Proc. Eur. Conf. Netw. Commun. (EuCNC)*, Jun. 2014, pp. 1–5.

[21] A. Wantamanee, S. Watarakitpaisarn, G. Carella, C. Aswakul, and T. Magedanz, "Virtualising machine to machine (M2M) application using open baton as NFV-compliant framework for building energy management system," in *Proc. 11th Int. Conf. Comput. Sci. Edu. (ICCSE)*, Aug. 2016, pp. 199–204.

[22] D. Mishra, H. Gupta, B. R. Tamma, and A. A. Franklin, "KORA: A framework for dynamic consolidation & relocation of control units in virtualized 5G RAN," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–7.

[23] C. Both *et al.*, "FUTEBOL control framework: Enabling experimentation in convergent optical, wireless, and cloud infrastructures," *IEEE Commun. Mag.*, vol. 57, no. 10, pp. 56–62, Oct. 2019.

[24] D. Wubben *et al.*, "Benefits and impact of cloud computing on 5G signal processing: Flexible centralization through cloud-RAN," *IEEE Signal Process. Mag.*, vol. 31, no. 6, pp. 35–44, Nov. 2014.
[25] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 2429–2453, 3rd Quart., 2018.
[26] A. Marotta *et al.*, "Impact of CoMP VNF placement on 5G coordinated scheduling performance," in *Proc. Eur. Conf. Netw. Commun. (EuCNC)*, Jun. 2017, pp. 1–6.
[27] R. Riggio, T. Rasheed, and R. Narayanan, "Virtual network functions orchestration in enterprise WLANs," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM)*, May 2015, pp. 1220–1225.
[28] M. Chiosi *et al.*, "Network functions virtualisation (NFV)," ETSI NFV ISG, Sophia Antipolis, France, White Paper 1, 2012.
[29] H. Moens and F. D. Turck, "VNF-P: A model for efficient placement of virtualized network functions," in *Proc. 10th Int. Conf. Netw. Service Manage. (CNSM) Workshop*, Nov. 2014, pp. 418–423.
[30] D. Bhamare, A. Erbad, R. Jain, M. Zolanvari, and M. Samaka, "Efficient virtual network function placement strategies for cloud radio access networks," *Comput. Commun.*, vol. 127, pp. 50–60, Sep. 2018.
[31] M. Bagaa, T. Taleb, A. Laghrissi, A. Ksentini, and H. Flinck, "Coalitional game for the creation of efficient virtual core network slices in 5G mobile systems," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 469–484, Mar. 2018.
[32] S. Clayman, E. Maini, A. Galis, A. Manzalini, and N. Mazzocca, "The dynamic placement of virtual network functions," in *Proc. IEEE Netw. Operations Manage. Symp. (NOMS)*, May 2014, pp. 1–9.
[33] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, "Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM)*, May 2015, pp. 98–106.
[34] X. Li and C. Qian, "A survey of network function placement," in *Proc. 13th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2016, pp. 948–953.
[35] F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and O. C. M. B. Duarte, "Orchestrating virtualized network functions," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 4, pp. 725–739, Dec. 2016.
[36] M. Mechtri, C. Ghribi, O. Soualah, and D. Zeghlache, "NFV orchestration framework addressing SFC challenges," *IEEE Commun. Mag.*, vol. 55, no. 6, pp. 16–23, Jun. 2017.
[37] Z. Xu, W. Liang, A. Galis, and Y. Ma, "Throughput maximization and resource optimization in NFV-enabled networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–7.
[38] I. Jang, D. Suh, S. Pack, and G. Dan, "Joint optimization of service function placement and flow distribution for service function chaining," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2532–2541, Nov. 2017.
[39] Y. Nam, S. Song, and J.-M. Chung, "Clustered NFV service chaining optimization in mobile edge clouds," *IEEE Commun. Lett.*, vol. 21, no. 2, pp. 350–353, Feb. 2017.
[40] L. Bondan *et al.*, "FENDE: Marketplace-based distribution, execution, and life cycle management of VNFs," *IEEE Commun. Mag.*, vol. 57, no. 1, pp. 13–19, Jan. 2019.
[41] Open Platform for NFV (OPNFV). *The Linux Foundation*. Accessed: Jan. 2019. [Online]. Available: https://www.opnfv.org/
[42] J. Chen, Y. Chen, S.-C. Tsai, and Y.-B. Lin, "Implementing NFV system with OpenStack," in *Proc. IEEE Conf. Dependable Secure Comput.*, Aug. 2017, pp. 188–194.
[43] M. Kist, J. A. Wickboldt, L. Z. Granville, J. Rochol, L. A. DaSilva, and C. B. Both, "Flexible fine-grained baseband processing with network functions virtualization: Benefits and impacts," *Comput. Netw.*, vol. 151, pp. 158–165, Mar. 2019.
[44] M. Kist, J. Rochol, L. A. DaSilva, and C. B. Both, "SDR virtualization in future mobile networks: Enabling multi-programmable air-interfaces," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.
[45] M. Kist, J. Rochol, L. A. DaSilva, and C. B. Both, "HyDRA: A hypervisor for software defined radios to enable radio virtualization in mobile networks," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, May 2017, pp. 960–961.
[46] T. Vilches and D. Dujovne, "GNUradio and 802.11: Performance evaluation and limitations," *IEEE Netw.*, vol. 28, no. 5, pp. 27–31, Sep. 2014.
[47] J. A. Wickboldt, R. P. Esteves, M. B. de Carvalho, and L. Z. Granville, "Resource management in IaaS cloud platforms made flexible through programmability," *Comput. Netw.*, vol. 68, pp. 54–70, Aug. 2014.

Ariel Galante Dalla-Costa received the B.Sc. degree in computer science from the Federal University of Fronteira Sul in 2015 and the master's degree in computer science from the Federal University of Rio Grande do Sul (UFRGS), Brazil, in 2018. He is currently an NFV Researcher with the Institute of Informatics, UFRGS. His research interests include NFV, network management and orchestration, and VNF placement.

Lucas Bondan received the Computer Engineering degree from the Pontifícia Universidade Católica do Rio Grande do Sul and the master's degree in computer science from the Federal University of Rio Grande do Sul (UFRGS), Brazil, where he is currently pursuing the Ph.D. degree in computer science with the Institute of Informatics. His research interests include network functions virtualization, network management and orchestration, and service function chains.

Juliano Araújo Wickboldt received the B.Sc. degree in computer science from the Pontifical Catholic University of Rio Grande do Sul in 2006, the M.Sc. degree from the Federal University of Rio Grande do Sul (UFRGS), Brazil, conducted in a joint project with HP Labs Bristol and Palo Alto, and the Ph.D. degree from UFRGS. His current research interests include cloud resource management and software-defined networking.

Cristiano Bonato Both is currently a Professor with the Applied Computing Graduate Program, University of Vale do Rio dos Sinos (UNISINOS), Brazil. His research focuses on wireless networks, mobile technologies (RAN and 5GC), softwarization and virtualization technologies for telecommunication networks, and SDN-like solutions for IoT low-power wide area network (LPWAN).

Lisandro Zambenedetti Granville received the M.Sc. and Ph.D. degrees in computer science from the Federal University of Rio Grande do Sul (UFRGS), Brazil, in 1998 and 2001, respectively. He is currently a Full Professor with the Institute of Informatics, UFRGS. He has served as a TPC member for many events in the area of computer networks, such as IM, NOMS, and CNSM. He has served as the TPC Co-Chair of DSOM 2007 and NOMS 2010.