Abstract—The fifth-generation mobile evolution enables Next-Generation Radio Access Networks (NG-RAN) transformations. The RAN protocol stack is split into eight disaggregated options combined in three network units, i.e., Central, Distributed, and Radio. Further advances allow the RAN functions to be virtualized on top of general-purpose hardware using the virtualized RAN (vRAN). The combination of NG-RAN and vRAN results in vNG-RAN, enabling the management of the disaggregated units and protocols as a set of radio functions. However, the orchestration-based placement of these radio functions is challenging since the best decision can be determined by multiple constraints involving RAN disaggregation, crosshaul network requirements, availability of computational resources, etc. This article proposes OPlaceRAN, a vNG-RAN deployment orchestrator framed within the NFV reference architecture and aligned with the Open RAN initiative. OPlaceRAN supports the dynamic placement of radio functions focusing on vNG-RAN planning and is designed to be agnostic to the placement optimization solution. We developed a prototype based on cloud-native tools to deploy RAN using containerized virtualization and the OpenAirInterface emulator. The evaluation is analyzed considering two different approaches as a proof-of-concept. First, we applied two placement solutions in a controlled real computing infrastructure with a crosshaul network. Second, we investigated the orchestrator’s scalability with a real and larger-scale topology. Our results show that OPlaceRAN is an effective cloud-native solution for containerized network function placement and agnostic to the placement solution, handling scale-out well. OPlaceRAN is up-to-date with the most advanced vNG-RAN design and development approaches, contributing to the evolution of fifth-generation networks.

Index Terms—NG-RAN, RAN disaggregation, NFV, CNF placement, crosshaul networks.

I. INTRODUCTION

The evolution of the fifth-generation mobile networks is guided by transforming technologies and architectures into centric-software concepts to meet the new service demands, e.g., ultra-low latency and high data bit rate applications. For mobile operators, the softwarization process brings significant advantages, especially concerning time-to-market, usage of commercial off-the-shelf (COTS) hardware, total cost of ownership (TCO) savings, adoption of open-source initiatives, continuous development and operations (DevOps), and network automation [1], [2].

The Radio Access Network (RAN) is transitioning from monolithic and highly distributed infrastructures to virtualization (i.e., vRAN) based on software concepts [3]. The most promising movement is led by the industry’s Open RAN (O-RAN) initiative, focusing on open and interoperable solutions [4]. Further advances are aligned with the proposed transformations toward the Next-Generation RAN (NG-RAN) architecture, allowing the functional split of the radio protocol stack into eight potential options. These options are combined into up to three network elements: (i) Central Unit (CU), (ii) Distributed Unit (DU), and (iii) Radio Unit (RU) [5], [6], [7], enabling radio features and improving cost efficiency compared to previous generations of mobile networks [6], [8].

The functional splitting of NG-RAN combined with the virtualization in vRAN enables the vNG-RAN architecture to provide flexibility when deploying mobile access networks [9], [10]. This flexibility allows the dynamic placement of radio functions following a fine-grained network management approach [11]. However, vNG-RAN introduces an unprecedented challenging problem since crosshaul transport networks (i.e., backhaul, midhaul, and fronthaul) and computational resources (hosting the virtualized radio functions) present
specific dimensioning, non-uniform resources consumption profiles and capacity constraints [5]. In this context, the Network Function Virtualization (NFV) architecture leads the development and standardization of Virtualized Network Functions (VNF) in mobile networks to cope with the deployment and dynamic placement of vNG-RAN. Moreover, Container Network Functions (CNF) are applied for radio functions based on the cloud-native approach and vRAN characteristics [4].

The NFV Orchestrator (NFVO) is an essential piece in the NFV reference model for the radio function chaining placement and orchestration. NFVO is responsible for two main controls: Resource Orchestration (RO) and Network Service Orchestration (NSO), to support the VNF placement decision-making and, consequently, Service Function Chaining (SFC) [12], [13]. Therefore, the RO control concerns are related to the capacity and resources of the NFV Infrastructure (NFVI), besides the capacity of the crosshaul networks (mainly latency and bit rate). Furthermore, the NSO manages the lifecycle of virtualized functions, particularly the chaining between them [14]. In this context, the O-RAN initiative is inspired by the NFV architecture, including the proposed Service Management and Orchestration (SMO) framework for the radio VNF placement decision-making [4], [10].

The state-of-the-art of vNG-RAN placement orchestration and SFC is diverse. On the one hand, the placement and SFC concepts are widely evaluated and massively developed based on exact and heuristic approaches [15], [16], [17], [18], [19], [20]. On the other hand, orchestration solutions are analyzed for generic functions, i.e., not specific to the RAN environment. In this context, the articles of Matoussi et al. [21] and Dalla-Costa et al. [22] are an exception in the literature. Matoussi et al. investigated the CNF orchestration of the predecessor architecture of NG-RAN (i.e., C-RAN) using of OpenAirInterface (OAI) tool, and Dalla-Costa et al. explored the load-balancing CNF orchestration of synthetic C-RAN oriented to fronthaul. In the industry context, open-source end-to-end orchestration platforms are being developed for VNFs and CNFs [23]. However, only Open Network Automation Platform (ONAP) works with O-RAN using an optimization framework [24] to the best of our knowledge. This framework is divided into the design time (network analysis) and run-time (working network) applications without a concrete placement orchestration solution [25], [26]. Open Source MANO (OSM) is another robust platform for orchestration and managing VNF/CNFs. However, this platform still lacks placement characteristics for vNG-RAN [27], [28], [29]. Furthermore, it is fundamental to emphasize that a significant tendency in the telecommunication industry standard applies the cloud-native approach [30] with the virtualization-based on containers, adopting Kubernentes (K8S) as the leading container orchestration tool [31], [32].

Investigations about orchestrators supporting vNG-RAN placement are lacking, despite the large number of studies covering the orchestration of RAN and the VNF placement problem presented in the literature and industry. The present article proposes the Orchestrator Placement RAN (OPlaceRAN), an NFVO solution for vNG-RAN deployment fitted within the NFV reference architecture and in line with the O-RAN SMO framework. OPlaceRAN is the unique orchestrator developed to deploy agnostic placement solutions of radio functions and its ecosystem, i.e., distributed topologies, computational resources, and crosshaul constraints. In this context, OPlaceRAN does not develop a placement solution since the main goal is to support different placement solutions, which would focus on network planning or validate a particular constraint. Therefore, the orchestrator design does not have run-time optimizations after the deployment. Moreover, OPlaceRAN is the first solution that connects the RAN placement problem with the real cloud-native environment to leverage the softwareization of the fifth generation of mobile networks and beyond. Furthermore, OPlaceRAN opens several opportunities for integration with other platforms, e.g., ONAP, once the proposed orchestrator is developed under up-to-date cloud-native tools.

OPlaceRAN is designed following the functional NFVO sub-blocks, considering a RO control named RANPlacer, a complementary optimization module named RANOptimizer, an NSO control called RANOptimizer, and a data repository referred to as RANCatalogs. RANPlacer handles the whole orchestration process, including external processing input from the Network Operator (quantity of radio units), crosshaul topology capacity, NFVI resources, network data input, and alternative placement solutions stored in the RANCatalogs. RANOptimizer works with exact and heuristics agnostic placement solutions, aware of the functional split requirements. In this case, the agnostic solution is a vNG-RAN placement strategy applied on the OPlaceRAN developed independently of the orchestrator. RANDeployer applies the virtualized radio functions addressed by the placement approaches and SFC planning according to the RANPlacer inputs. Moreover, RANPlacer inputs store the RAN CNFs in the RANCatalogs. All the configuration, initialization, and validation processes of the virtualized radio functions are performed and activated by the RANDeployer.

The results obtained in our evaluation show that OPlaceRAN is a powerful orchestrator to leverage the placement of virtualized functions of vNG-RAN in NFV architectures. Moreover, the whole orchestration and placement process is efficiently concluded in a few seconds, regardless of network scale. Our contributions can be summarized as follows:

- A conceptual orchestrator to deploy agnostic placement solutions for vNG-RAN aligned with the NFV architecture and the O-RAN SMO framework.
- A real prototype to design virtualized radio functions considering the three units of vNG-RAN, based on the cloud-native approach.
- A proof-of-concept considering two different approaches with real computational NFVI. The first provides the virtualized function placement analysis for vNG-RAN networks, and the second validates the scalability of the orchestrator.

Based on the contributions presented, the OPlaceRAN solution fills the gaps in the literature related to (i) an orchestrator of virtualized radio functions oriented to placement solutions, (ii) the use of NG-RAN architecture instead of C-RAN, and (iii) the application of the crosshaul transport network instead
of just the fronthaul. Furthermore, the complete orchestration solution is aligned with the cloud-native architecture.

The article is organized into the following sections. Initially, Section II introduces the concepts developed in the article. Next, Section III presents the OPlaceRAN architecture, and Section IV describes the prototype used in the experiment. Moreover, Section V shows the developed prototype and the proposed evaluation environment. Finally, Section VI brings the related work, and Section VII concludes the article.

II. VIRTUALIZED NG-RAN ORCHESTRATION

The decomposition of the radio software stack into non-monolithic components aggregated in elementary functions is vital to improving RAN flexibility and processing performance for 5G and beyond networks. The standardization initiatives such as 3GPP [9] and IMT-2020/5G from ITU-T [5] specify the number and types of functional splits. The NG-RAN architecture aims to disaggregate and distribute radio functionality into three units (CU, DU, and RU). Moreover, this architecture flexibility allows some other configurations with less than three units, named DU and RU integration (DU and RU), C-RAN (CU and DU), and D-RAN (CU, DU, and RU in a monolithic way) [5]. In a disaggregated NG-RAN, the crosshaul transport network provides communication paths among radio units composed of different segments such as backhaul (mobile core to CU), midhaul (CU to DU), and fronthaul (DU to RU). Each functional split depends on radio capacity (e.g., channel bandwidth, modulation, and others), imposing distinct latency and bit rate requirements for the crosshaul network [33], [34]. Therefore, the crosshaul must guarantee the latency and bit rate required according to functional radio splits [11], [33].

Adopting virtualization concepts is essential for RAN disaggregation, where each virtualized radio unit (vCU, vDU, and vRU) is a VNF, resulting in a vNG-RAN architecture [21]. In this context, the main reference virtualization architecture for telecommunication is the NFV architecture specified by the European Telecommunications Standards Institute (ETSI) [12]. In this architecture, the Management and Orchestration (MANO) part is responsible for managing and orchestrating VNFs, defining such processes as the automation, management, and operation of virtualized functions running on top of an infrastructure supporting virtualization and multi-tenancy. NFVO block orchestrates the NFVI resources across multiple Virtualized Infrastructures Managers (VIM) and manages the lifecycle of VNF services through the Virtual Network Functions Manager (VNFM). Moreover, NFVO receives data support from the Catalogs module to store and make available data related to the services, VNFs, SFCs, NFVI resources available on the network, etc. To this end, NFVO works with the RO control to provide abstract access to the NFVI resources, i.e., regardless of any VIM. Furthermore, NFVO performs the NSO control to create end-to-end services, including different VNFs, and manage network service for topology instances [35]. Fig. 1 presents the NFV architecture and the positioning of radio VNFs.

NFV architecture enables the execution of these radio VNFs over distributed Computational Resources (CR) interconnected through the crosshaul network. Moreover, it is essential to highlight that the O-RAN initiative follows the NFV architecture. In this sense, O-RAN proposes the SMO framework focusing on slicing concepts, programmable RAN network resources, and embedded artificial intelligence capabilities [4], [10]. O-RAN uses NFVO and VNFM MANO blocks to develop the orchestration and lifecycle of VNFs. Furthermore, the SMO framework defines the RAN Intelligent Controller (RIC) to manage the RAN network divided into two components: (i) Non-Real Time (centralized components) and (ii) Near-Real Time (closed to vNG-RAN). Therefore, the SMO framework can connect to the NFV architecture and provide a consolidated approach to the fifth-generation mobile networks [4], [34].

NFVO instantiates virtualized radio functions by leveraging hypervisor-based virtual machines or containers to deploy RAN VNFs on top of NFVIs. In this context, container virtualization is targeted by O-RAN since it shares the Operating System that hosts VNFs. Moreover, recent results show that the containerized solution tends to be the most appropriate for RAN virtualization due to the strict requirements of delay and scalability of resources. In this sense, each virtualized radio function is envisioned as a container [34], [36], [37], [38]. For the orchestration of containers in generic applications, four tools lead the implementations: (i) K8S, (ii) Docker Swarm, (iii) Mesosphere Marathon, and (iv) Cloudify. From the four options, K8S emerges in telecommunications virtualization as the main tool due to extensive Continuous Integration/Continuous Delivery (CI/CD) development, flexible configuration options (e.g., customized thresholds), and support for Docker-based containers (widespread). Moreover, K8S focuses on managing virtualized infrastructures and orchestration on clusters managed over a pool of computational resources, whether physical or virtual [31]. In this context, the vNG-RAN placement orchestration solution is explored in the next section.

III. OPlaceRAN: ORCHESTRATOR PLACEMENT RAN

This section introduces OPlaceRAN, i.e., an orchestrator with decision-making support for radio function placement.
We design OPlaceRAN to overcome the lack of placement orchestrators for vNG-RAN. OPlaceRAN seeks to be an agnostic placement tool considering exact and heuristic placement solutions for the orchestration of planning of vNG-RAN. According to the NFVO functional block of the NFV architecture, OPlaceRAN operates between RO and NSO. In this context, OPlaceRAN is aware of crosshaul topologies. Moreover, the defined virtualization technology is based on containers, following the O-RAN initiative to obtain the possibility of interoperability with tools that meet the evolution of the fifth-generation mobile networks.

Fig. 2 shows OPlaceRAN positioned into NFV reference architecture for the RAN planning and design in the SMO framework. The overall NFV architecture components are covered, including VNFM, VIM, CNFs, and NFVI blocks in the end-to-end orchestration structure of vNG-RAN. The OPlaceRAN design represents the intersection between the NFV architecture and the SMO framework since SMO adopts NFVO and VNFM blocks from the NFV architecture. Moreover, it is worth highlighting that Near-Real Time is positioned outside the SMO framework, i.e., close to vNG-RAN (vCU/vO-CU, vDU/vO-DU, and vRU/vO-RU). Lastly, it connects to the Core Network and Radio Antennas (including UEs) and receives external input data from Network Operator and Crosshaul Topology.

OPlaceRAN is responsible for the entire system orchestration to place virtualized radio functions into NFVI, playing as an NFVO. The main goals of OPlaceRAN are: (i) to receive input from Network Topology by Network Operator, (ii) to analyze the available CRs, (iii) to support any optimization solution for planning RAN networks, and (iv) to execute the deployment for positioning the virtualized radio functions. Moreover, OPlaceRAN interacts with the entire system. For example, OPlaceRAN communicates with VIM and VNFM blocks to collect infrastructure information and execute network scaling actions regarding MANO functions. In this context, OPlaceRAN adopts the indirect communication mode between VIM and VNFM blocks, as indicated in the NFV architecture [12]. For providing this indirect mode, OPlaceRAN is divided into four functional sub-blocks: RANPlacer, RANOptimiser, RANDeployer, and RANCatalogs. These functional sub-blocks are:

- **RANPlacer**: it is positioned as a RO functionality from the NFVO perspective, centralizing and determining the actions of OPlaceRAN for making decisions. RANPlacer operates on-demand from external inputs to begin the vNG-RAN planning process. Next, RANPlacer receives external vRUs from Network Operator and Crosshaul Topology and requests a VIM block to update the NFVI Resource Catalog based on these external inputs. Together with NFVI resources, these inputs are sent to the RANOptimiser sub-block to analyze the optimal placement of the virtualized radio functions. The regardless is analyzed by RANPlacer, which sends the VNF allocation and chaining plans to the RANDeployer sub-block.

- **RANOptimiser**: it is responsible for running the decision-making process for the placement of virtualized radio functions, including the chaining process. RANOptimiser receives requests from and returns solutions to RANPlacer. This request is made by inserting the RANCatalogs sub-block inputs and choosing a placement and chaining solution for runtime deployments. After that, RANOptimiser delivers the result of the radio VNF placement and chaining solution. In this case, RANPlacer receives from RANOptimiser a solution when feasible, including (i) the position of vRU, vDU, and vCU units, (ii) disaggregation RAN options, and (iii) CRs to be allocated in the crosshaul networks.

- **RANDeployer**: it allocates and deploys the virtualized radio functions according to placement results from RANOptimiser sent to RANPlacer. In this case, RANDeployer operates as an NSO in NFVO, interacting with VNFM and VIM blocks to deploy radio units. The interaction with the VNFM functional block assigns the deployment of CNFs placement and chaining of virtualized radio functions. Additionally, the interaction with VIM aims to allocate CNFs in the CRs. Therefore, RANDeployer processes the requests from RANPlacer, radio units CNFs from the RANCatalogs sub-block, and
sends the positions of CNFs to VNFM, deploying them in CRs of NFVI.

- **RANCatalogs**: It is a repository that stores data about the overall services, functions, and resources available on the network. This block is composed of four different sub-catalogs: (i) Topology Inputs, (ii) NFVI Resources, (iii) Placement Solutions, and (iv) RAN CNFs. First, the sub-catalog works with external data entries from the Crosshaul Topology inputs, e.g., network links, latency, and data rate capacities, updated by the RANPlacer. The NFVI resources catalog refers to the available and allocated computing resources in NFVI. Examples of data types found in NFVI resources are CPU, memory, and storage. In this case, the update is based on the communication between RANPlacer and VIM. The placement solutions sub-block supports a pool of the agnostic exact and heuristic approaches of placement solutions requested by RANOptimizer. Lastly, the RAN CNFs catalog is a repository of the container image registry for the radio units, i.e., vCU, vDU, and vRUs. The request for CNFs is triggered by the RANDeployer sending them to VNFM, aiming to access the radio units to allocate and deploy these CNFs.

Finally, Fig. 3 presents the sequence diagram of the OPlaceRAN workflow. The process starts with collecting the External Inputs event (step 01) of the Network Operator (NO) and Crosshaul Topology. Next, RANPlacer processes those inputs by considering the vRUs entry and the Crosshaul Topology (step 02). After that, RANPlacer sends a request to RANCatalog to update the VIM CRs (step 03), asking for an up-to-date view of NFVI (steps 04 and 05) and receiving the update notification (steps 06 and 07). In the following interaction, OPlaceRAN requires an update for the Topology Inputs and NFVI Resources to the RANCatalog (steps 08 and 09). Therefore, RANPlacer sends the updated information and the external data to RANOptimizer, selecting the placement solution of RANCatalog and providing a feasible radio function placement (steps 10, 11, 12, and 13). When the RANPlacer has all the information, it generates the manifests for executing the allocation plan (step 14). RANPlacer addresses the allocation plan to the RANDeployer function (step 15). Lastly, RANDeployer requires RANs CNFs Catalog images and sends them to VNFM for the CNFs position and chaining allocation and deploying (steps 16, 17, 18, and 19). Furthermore, the sequence diagram shows the interaction of the completion of the tasks between the NFVO module and the rest of the components of the whole NFV architecture through VIM, NFVI, and VNFM. The OPlaceRAN operation finishes with allocating VNFs in CRs and its notification (steps 20 and 21). The following section describes a prototype...
to validate the OPlaceRAN approach on K8S, considering this sequence diagram discussed.

IV. OPLACE RAN PROTOTYPE

We developed a K8S cloud-native tool prototype to validate the OPlaceRAN proposition. Fig. 4 shows the prototype components, which comprise the NFV and SMO blocks to support OPlaceRAN and their sub-blocks RANPlacer, RANDeployer, and RANCatalog data repository. The prototype covers all the aforementioned presented workflow, from the collection of external inputs (NO inputs and Crosshaul Topology) up to the CNFs chaining allocation process. In the prototype, RANCatalog supports the Topology Inputs and the NFVI resources implemented with the custom resources feature from Kubernetes and uses a Container Registry for the CNF’s repository. The current state of K8S, e.g., CPU and RAM utilization, is obtained from Kube State Metrics. Moreover, the K8S tools include both NFVI and VIM natively and support VNFM and NFVO of the NFV architecture.

The K8S cluster comprises nodes concerning CRs and runs either directly on top of the hardware (bare-metal approach) or abstracted as a virtual machine (VM). In this context, applications are deployed and managed by K8S into Pod resources, holding one or more containers, i.e., CNFs. In the K8S architecture, Master Nodes provide complete control of the cluster through a K8S Control Plane (K8S CP) deployed in specific nodes. However, CNFs are running in the Workload Nodes. The main components of this control plane are Controller Manager, Scheduler, Storage, and APIs. The NFVI cluster (K8S oriented) applied in the prototype is represented by VMs. The Calico plugin is used as a Container Network Interface (CNI) to provide an IP network connection between the Pods (i.e., radio functions). The prototype code is available on GitHub.

K8S is highly customizable, even considering its default components. For example, the control plane works with the Operators pattern to extend the Controller Manager and APIs. The operator pattern combines two extension points: the API server extension adding new resources to the K8S API and a controller monitoring and executing the defined resources. Therefore, an Operator can automate the whole lifecycle of the prototype, providing application-specific automation without changing the K8S default behavior. Moreover, Operators work under a Pod and are driven by the K8S object model. K8S uses additional layers of abstraction over the container interface to provide scale, resiliency, and lifecycle management capabilities. In this sense, RANPlacer and RANDeployer orchestration sub-blocks are developed under the K8S operator’s pattern in the Golang language. Operators manage the decision-making process of the solution. For example, RANPlacer builds the information base (i.e., the VIM data and placement result), which RANDeployer uses to allocate CNFs in Pods.

The Network Operator information includes the demand for vRUs and the Crosshaul Topology. The vRU demand is defined by the cluster of CR nodes, their vRU quantity, and the Crosshaul Input, one of three Catalog blocks defined as a ConfigMap. In the Catalog sub-block, NFVI Resources are acquired natively from K8S and matched with the nodes, i.e., the CPU and RAM capacity. The Placement Solution and the RAN CNFs Catalogs components are deployed in a Docker Registry server hosted in a virtual machine out of the NFVI cluster. The Docker Registry stores containerized images storing the placement solutions and the CNFs OAI images. These images are deployed on-demand by OPlaceRAN. We created images from available open-source projects: free5GC [39] for Core Network and OpenAirInterface (OAI) [40] for the RAN units. We only disabled the PHY layer to emulate RAN, considering all other layers to support the disaggregation on three units (vCU, vDU, and vRU), using the split options 2 (O2 - F1 interface) and 6 (O6 - nFAPI interface).

Fig. 4. OPlaceRAN prototype.
The RANOptimizer works based on three sub-blocks to deliver the placement result to the RANDeployer: (i) Optimization Control, (ii) Placement Jobs, and (iii) Placement Results. The Optimization Control receives requests from RANPlacer with the inputs and the Placement Solutions. The input is composed of the following pieces of information: (i) CR nodes, (ii) the number of chains that must be deployed, (iii) placement of vRUs in CRs, (iv) crosshaul topology, (v) the capacity of the CRs (CPU and RAM), and (vi) specific placement solution (either the exact or heuristic approaches). The Optimization Control works similarly to an HTTP server. Therefore, the Optimization Control asynchronously triggers the placement solution job execution and provides RANPlacer a token to get the placement solution status and the result. Moreover, the Placement Jobs are based on asynchronous tasks since they can take a considerable time to finish. In this case, a Job resource handles the execution of the placement solution, starting one or more Pods to provide a task-based workflow expected to run successfully after completing their goal. The chosen solution triggers a job according to the selected placement solution by the Network Operator. The Placement Solutions are deployed as container images accessible by the Optimization Control. Finally, the result of the placement solution is returned to RANPlacer to be suitable for sending to RANDeployer for the CNFs allocation process.

V. EVALUATION

This section evaluates OPlaceRAN in a proof-of-concept considering two different approaches with real computational NFVI. Section V-A describes the designed scenarios, detailing the placement solution applied, including topology and resources, by varying parameters to analyze the results of experiments. Section V-B presents the results obtained in the evaluations of OPlaceRAN concerning placement solution results, cluster, and the behavior of the nodes. Lastly, Section V-C discusses the general aspects and relevant characteristics of OPlaceRAN.

A. Scenarios Description

We detail the two scenarios we evaluated in this subsection. First, we describe the scenario used to test the ability of OPlaceRAN to be agnostic to placement solutions in a real and controlled network in Section V-A1. After, we analyze the scalability ability of the orchestrator to handle a wide network with a synthetic crosshaul input in Section V-A2.

1) Controlled Topology Scenario: We organized the assessment of the OPlaceRAN controlled topology experiment into four parts, as described in the following. The first part presents the topology used, while the second shows the computational capacity of the resources. In the third part, the parameters used in vNG-RAN are explored. Finally, the fourth part details the placement solutions used.

- **Topology**: we run the experiments in a K8S cluster in version 1.19.3, with six nodes. Each node of this cluster is hosted in a VM running on a VMware ESXi 6.7 hypervisor. On each host, we have two VMs running Ubuntu 18.04 with a low-latency kernel, working as Kubernetes Nodes. In this case, the testbed comprises six Kubernetes Nodes, i.e., two with the master control plane (no vNG-RAN was deployed at this stage) and four acting as workers. These VMs are hosted in three DELL PowerEdge M610 servers equipped with two Intel Xeon X5660 processors running at 2.80 GHz and 192 GB of RAM. The Crosshaul Topology applied in the evaluation includes a physical switch (pSwitch) connected with the workers through a virtual switch (vSwitch) in the border. Each crosshaul link has a 10 Gbps reserved bandwidth and 1ms of latency guaranteed, except the worker node six, which connects directly with the pSwitch and has 1 Gbps of bandwidth. We defined a 1Gbps link to bring asymmetry to the experiment. However, we did not control the latency of links between vSwitches and pSwitches due to the limitation of the virtualization solution utilized, i.e., VMware Vsphere 6.7. Figure 5 shows the topology used, which supports four NG-RAN scenarios considering the units and nodes distribution, as addressed in the RAN Units item (described below).

- **Computing Resources**: according to the OAI analysis, we focused on the processing capacity (i.e., CPU and RAM) because this metric is the most common bottleneck for computing devices in the context of vRAN evaluation [41], [42]. CR values in Table I indicate the number of available CPUs (Core unit) and RAM (GB unit) in worker K8S nodes when generating input data. We fixed these values to have control under the deployment scenario, but our operator can automatically collect these values in the real world during placement time. The heterogeneity of CRs capacity used adds complexity to our evaluation placement solutions. These capacities of CRs considered the CPU utilization profile of the RAN software as obtained from OAI implementation. The exact values may vary according to the

![Fig. 5. Experimental topology.](image)

| TABLE I | RESOURCES TO RAN CNFs ON K8S WORK NODES |
| --- | --- | --- | --- |
| Node | 1 | 2 | 3 | 4 |
| CPU (Core number) | 5 | 1 | 2 | 4 |
| Memory (GB) | 2 | 0.2 | 1.2 | 0.2 |
adopted software components and the computing device, but similar profiles are employed in different works [41], [42].

- **RAN Units:** we defined a vRU for each worker node. The parameters used in the two deployed placement models follow the definitions of 3GPP [9] and ITU-T standardizing organizations [5] and the OAI Long Term Evolution (LTE) tool support. The OAI LTE is developed based on Table II, applying the RU specifications: 20 MHz bandwidth, 1 antenna, 100 PRBs, and 15 kHz subcarrier spacing per macro base station [43]. We use four NG-RAN scenarios regarding the units and nodes distribution, i.e., (i) three units are divided into vCU, vDU, and vRU; (ii) two units co-located vCU and vDU, (iii) C-RAN scenario, using two units and integration of vDU and vRU; and (iv) D-RAN with all the units in a single node. Additional parameters shown in Table II are latency and bandwidth supported by the OAI emulator for RAN functional split. In this sense, the Core Network (CN) parameters represent the communication requirements between CN and vCU. We apply two RAN split options in this work, namely O2 and O6 [43], and the parameters of these splits are shown in Table II. For example, the O2 split parameters represent the vCU-vDU communication, and the O6 split parameters reflect the vDU-vRU communication. The analysis considered the peak capacity required by OAI LTE without differentiating workloads size and requirements, e.g., user quantity and throughput.

- **Placement Solutions:** we used two different placement solutions approaches to test the effectiveness of OPlaceRAN and the support for agnostic placement solutions. First PlaceRAN [20], an exact optimization approach focused on maximizing the aggregation level of RAN CNFs and minimizing the number of CRs used for such aggregation. The second is another exact optimization approach that combines a linearization technique with a cutting-planes method[15]. The focus is to apply the Multi-CUs vRAN concept to minimize the vRAN cost and overall routing based on CU’s positioning. Both approaches explore the optimal solution for the placement of radio functions in NG-RAN. We must highlight that our goal is not to investigate the performance evaluation of placement solutions but to the OPlaceRAN orchestrator that is agnostic support for these solutions.

2) **Scalability Scenario:** We organized the investigation of OPlaceRAN scalability into three parts, as described in the following. The first part presents the topology used, while the second shows the computational capacity of the resources. Finally, in the third part, we describe additional considerations about the scenario.

- **Topology:** we run the scalability experiment in the same K8S cluster. Virtual nodes are hosted in eight identical DELL PowerEdge M610, i.e., virtual nodes that play the role of the K8S cluster nodes are floating on these eight physical hosts. We increased the number of virtual nodes to 51 according to the real topology strategy applied in the PASSION project [44]. The 51 virtual nodes run Ubuntu 18.04 with a low latency kernel, in which two nodes have the master control plane (no vNG-RAN was deployed at this stage), and the other 49 nodes have the worker role. Moreover, crosshaul topology is inputted by synthetic data due to virtualized infrastructure, and the focus is on the behavior of OPlaceRAN in a larger-scale environment. In this sense, we do not use switch devices in the present evaluation (instead of controlled topology evaluation), and each crosshaul device corresponds to a virtual node. Fig. 6 shows the reference-tested topology, which is composed of four types of transport nodes considering the proximity to the Core and the number of neighbors: aggregation node 1 (AG1), aggregation node 2 (AG2), access node 1 (AC1), and access node 2 (AC2). Each transport node has a 10 Gbps connection with latency values of (i) 0.163ms from Core to AG1, (ii) 0.22ms from AG1 to AG2, and (iii) 0.174ms for the two links (AG2-AC1 and AC1-AC2). These latency values are also in line with the PASSION project requirements and applied in work [20]. The PASSION project defines the maximum, minimum, and average link distances, which we apply to latency parameters. Moreover, the link bandwidth is based on the PASSION project and the industry [45].

| NG-RAN Scenarios | Latency - one way (ms) | Bandwidth (Mbps) |
|------------------|-----------------------|------------------|
|                  | CN        | O2          | O6          | CN        | O2          | O6          |
| NG-RAN 3 units   | 30        | 30          | 2           | 151       | 151         | 152         |
| NG-RAN 2 units   | 30        | 30          | -           | -         | 151         | -           |
| C-RAN            | 30        | -           | 2           | 151       | -           | 152         |
| D-RAN            | 30        | -           | -           | 151       | -           | -           |

Fig. 6. Scalability experimental topology.
TABLE III
RESOURCES TO K8S NODES IN SCALABILITY EXPERIMENT

| Node Type | Role | Quantity | CPU (Core number) | Memory (GB) |
|-----------|------|----------|-------------------|-------------|
| K8S CP    | Master | 2        | 24                | 48          |
| AC1/AC2   | Worker  | 8        | 8                 | 16          |
| AC1/AC2   | Worker  | 41       | 6                 | 12          |

Table III indicate the total amount of CPU (Core unit) and RAM (GB unit) in K8S nodes. We configure three types of K8S nodes, one for the master and two for workers. For this experiment, the nodes of type master are running only K8S control Pods. The topology has more resources at the top and fewer at the bottom. The worker nodes with more resources focus on transport nodes AG1 and AG2, and the worker nodes with fewer resources focus on AC1 and AC2.

Additional Considerations: we used only PlaceRAN [20] as the placement solution to evaluate the scalability according to the focus of this experiment. We operated one vRU on the nodes AG2, AC1, and AC2 for RAN, adding up to the 49 vRU and 49 chains between vRU-vDU and vDU-vCU. Moreover, AC1 nodes follow the PASSION project and do not have vRU input. Furthermore, the configurations of the RAN units and specifications were the same as in Table II used in the previous subsection.

B. Results

In this subsection, we detail the two scenarios evaluated. First, in Section V-B1, we discuss the OPlaceRAN controlled topology assessment results with few nodes and real crosshaul. This scenario aims to validate the correctness of the OPlaceRAN solution. Therefore, we worked with a small and real topology. In Section V-B2, our goal is to analyze the results of the ability of the orchestrator to handle a much higher-scale network. In this second scenario, we considered a large topology with 51 nodes based on the PASSION project to validate the scalability of the OPlaceRAN solution.

1) Controlled Topology Results: we design a proof-of-concept to analyze and evaluate the OPlaceRAN orchestrator in the two scenarios described. Therefore, we first detail the RAN CNFs allocation plan based on the previously presented placement solutions. Second, we show the behavior of clusters and nodes when the OPlaceRAN orchestrator is working. Finally, we use just the PlaceRAN [20] placement solution to evaluate the cluster’s performance because this article focuses on the behavior of OPlaceRAN and not the implementation of the placement solutions.

Firstly, we investigated the capacity of OPlaceRAN to be agnostic of positioning solutions. We use two placement solutions, PlaceRAN [20] and Multi-CUs vRAN [15]. Both solutions are optimization models aiming to centralize vRAN functions subject to the constraints of crosshaul latency and bandwidth and nodes processing capacity in the worst case, i.e., without demand variation over time. However, PlaceRAN formulation allows a higher level of centralization since it considers more centralization options. Fig. 7 illustrates both placement solutions for the same instance. In this case, it is essential to note that the placement of vRAN nodes differs for each placement solution. We used such formulations to assess the abstraction power of OPlaceRAN since both models solve the vRAN placement problem. In the first point observed, OPlaceRAN effectively deals correctly with two different positioning solutions approaches. The second point is about the performance of these positioning solutions. The two optimal solutions define the same number of nodes. Each node operates with only one RU (in the present work, only vRU). PlaceRAN performs slightly better since it aggregates more CNFs than the Multi-CUs vRAN. Moreover, PlaceRAN concentrates on aggregating vCUs and vDUs CNFs only in nodes 1 and 3, while Multi-CUs vRAN aggregates in nodes 1, 3, and 4. This result shows PlaceRAN as better performing in centralizing CNFs than Multi-CUs vRAN. This centralization improvement is due to the flexible allocation of vDUs designed in PlaceRAN. The placement of DUs is predefined and inflexible in multi-CUs vRAN.

According to the placement solution results, the analysis of cluster computing behavior shows the relationship between the main events during the deployment of OPlaceRAN based on the use of CRs of the cluster. In this analysis, we only evaluate the PlaceRAN placement solution once the focus is on the behavior of OPlaceRAN and CRs. The events for a full deployment are highlighted seven times (t0 up to t7), as shown in Table IV. Initially, the time consumed by OPlaceRAN, from the external inputs (sent by the Network Operator) to the allocation of CNFs, is presented between t0 and t1s shown in Fig. 8. The time between t0 and t1 is approximately 70s, divided into seven main
TABLE IV
KEY OPERATOR’S ACTIVITIES IN FUNCTION OF TIME

| Time | Description                                                      |
|------|------------------------------------------------------------------|
| $t_0$ | Started OPlaceRAN                                               |
| $t_1$ | Ended OPlaceRAN, CNFs allocated                                 |
| $t_2$ | Configured OAI network parameters                               |
| $t_3$ | OAI layer is loaded                                             |
| $t_4$ | OAI functional and connected to Core Network                    |
| $t_5$ | Configured network tunnel between vRU and Core Network          |
| $t_6$ | Ping from simulated UE to Core Network started                  |
| $t_7$ | Ping from simulated UE to Core Network terminated               |

Fig. 8. Initial OPlaceRAN steps six nodes.

events. The first event occurs with OPlaceRAN Start, beginning with the Network Operator and immediately RANPlacer Start to manage the process. After that, the placement solution processes the inputs Network topology, vRUs locations, and node resources. The RANOptimizer Complete processing takes approximately 1.2s after starting the placement due to the low complexity of the topology composed of six nodes. In the next step, RANPlacer Process Allocation Plan takes around 31.5s. The RANPlacer Process Allocation Plan is responsible for generating the K8S manifest templates for the OAI emulator based on the placement models solutions and sends it for RANDeployer Start. In this sense, RANPlacer receives the placement solution and builds each OAI radio unit manifest (vCUs, vDUs, and vRUs) with the IP network and radio parameter filled. After that, it sends the whole data to RANDeployer Start for the deployment of the network chain. In possession of the manifests, RANDeployer sends the plan to the K8S with the configuration to be applied. Finally, the CPU and memory of vDU and vRU stabilize from $t_5$ (i.e., two vCPUs), shown at the bottom of Fig. 9. One of the reasons for this behavior is that the OAI version works with emulated UE (without PHY protocol), and these UEs are fixed, reducing the network’s signaling. This behavior was not considered trouble because of the focus of this article.

We must consider Table I to understand the CR consumption in each node of the K8S cluster. We did not separate the resources consumed by OPlaceRAN, as can be seen from Fig. 7 it is quite insignificant. Firstly, we show the amount of CPU and memory guaranteed for the CNFs placement, i.e., the input values to the PlaceRAN placement solution. The CRs diversity is necessary to produce placement solutions that have different behavior. In this case, the objective was to simulate an environment with scarce resources, with the placement solutions having the behavior of RAN units so diverse. Node 1 running in the network provided in Fig. 9 presents the highest average consumption of CRs during the experiment execution, 1053 CPU (Millicores) and 2135 MiBytes of Memory, as shown in Fig. 10. It makes sense because this node has higher CRs (5000 CPU Millicores and 2000 MiBytes of free memory), and all tested placement solutions prioritize allocating the CNFs in Node 1. Both Nodes 2 and 4 allocated just one vRU because they have lower free CRs (1000 CPU Millicores and 200 MiBytes of Memory). Moreover, PlaceRAN allocated just one vRU because these CNFs use lower CRs than others, as seen in Fig. 8. Finally, Node 3 has an intermediate capacity of free CRs (2000 CPU Millicores and 1200 MiBytes of...
Furthermore, this node has a medium average consumption of CRs 613 CPU (Millicores) and 1448 MiBytes of Memory, but it has the highest allocation of proportional usage of CRs. This behavior results from the PlaceRAN placement solution in CNFs allocation in Node 3, as is presented in Fig. 7. A relevant observation is the consumption of the Master nodes. Given the proof-of-concept scale, the six vRANs chains in the cluster did not generate a high workload for the K8S CP. Master 1 has higher CPU and Memory usage because it is the primary K8S CP.

2) Scalability Results: we also evaluated OPlaceRAN analyzing the behavior of the orchestrator scaling the topology with 51 virtual nodes. This analysis aims to evaluate the performance of the cluster and OPlaceRAN in an environment on a larger scale. We use a synthetic crosshaul data input due to the crosshaul limitation. Moreover, we only deploy the validation with the PlaceRAN [20] placement solution without a specific placement evaluation for scalability goals. The procedure steps follow the same seven $t$s events of Table IV, and the orchestrator’s actions are also concentrated between $t_0$ and $t_1$. In this sense, Fig. 11 shows the time steps of OPlaceRAN.

Initially, the time consumed by OPlaceRAN from the external inputs (sent by the Network Operator) to the allocation of CNFs is approximately 90s, shown between $t_0$ and $t_1$s in Fig. 11, divided into six main events. The RANOptimizer Complete processing takes about 7s after starting the placement due to the complexity of the topology composed of 51 virtual nodes.

Similarly, in the controlled topology, the next operation is the RANPlacer Process Allocation Plan which builds the OAI manifests and sends it for RANDeployer Start the chaining allocations. These steps take around the 40s. In the next step, RANDeployer sends the allocation plan to K8S with the configuration to be applied. Next, K8S allocates CNFs into Pods images (vCUs, vDUs, and vRUs) to run on K8S nodes. When K8S Received CNFs Allocation Plan occurs, nodes download OAI images, and CNFs Pods are started at 52s. Finally, the container layer loads OAI, and all CNFs become allocated (at 90s). These events are consolidated in Table V.

In the first step, the CR usage was 5000 CPU Millicores and 12000 MiBytes of memory in the entire cluster between $t_0$ and $t_1$, as shown in Fig. 12. The maximum CPU was used by vRUs CNFs between $t_1$ and $t_2$, near 25000 CPU Millicores (i.e., 25 vCPUs), as shown at the top of Fig. 12. Another relevant point observed in this step is the increase of 36% in CPU consumption by the K8S CP mainly due to the configuration of the CNFs deployments. Moreover, the memory used was 5000 MiBytes by vCUs during $t_2$ and $t_3$. In this step, $t_3$ and $t_4$, the memory consumed was 17000 MiBytes by vDUs, reaching 20000 MiBytes between $t_4$ and $t_5$. Finally, when the ping started in $t_5$ and $t_6$, vDU operated with 13000 CPU Millcores. After that, vDU reduced CPU usage, and the experiment ended. About the overhead of OPlaceRAN, even in this scenario with a larger scale, the Maximum CPU consumption was 28 Millicores against 6 Millicores in the controlled experiment, an increase of 366% in CPU consumption. The memory had an average usage of 105 MiBytes against 94 MiBytes in the controlled experiment, an increase of 11% in the scale experiment. It is essential to highlight that we are considering an increase of 1175% in the experiment scale.

C. Discussion

Initially, we highlight OPlaceRAN within its conceptual design, including the prototype description, and present the development of vNG-RAN. Three pillars are crucial for the solution to achieve great potential: (i) alignment and positioning with the up-to-date architecture and frameworks of vNG-RAN, (ii) agnostic placement of CNFs approach,
(iii) real prototype update with main tools and emulators and crosshaul topology analysis. We can emphasize that the first pillar opens opportunities for integrating OPlaceRAN with other solutions or platforms for orchestration, management, and network automation. Moreover, OPlaceRAN is in line with the NFV architecture and SMO framework and based on the cloud-native platforms within the scope of vNG-RAN itself or with the crosshaul transport networks. For example, OPlaceRAN can be integrated with ONAP Optimization Framework (OOF) [24] or with OSM placement optimization module (PLA) [27], which still lacks considering a vNG-RAN consolidate solution.

OPlaceRAN stands out as a unique solution concerning the possibility of operating in an agnostic way for placement solutions. Such an operation provides an opportunity to analyze different placement solutions, as performed in Section V-B1, which compares two different strategies for the vNG-RAN placement. Furthermore, the possibility of exploring different placement solutions brings near-real operational network results. For example, OPlaceRAN allows the deployment of any placement solutions in real NFVI infrastructures based on containerized RAN applications. In this sense, the study of the NFVI cluster and its respective nodes in Section V-B allows investigating the impact on CRs based on their hardware, operating system, functions, and consumption characteristics of native tools (e.g., K8S Master control and CNI Calico), and the deployment of the CNF allocation concurrently. Therefore, a real impact on the performance of NFVI and RAN CNFs demands deep investigation into the exact and heuristic solutions for the radio function placement.

Another relevant contribution of this article is the analysis of the scalability of 51 virtual nodes based on a real network. We see a small difference in execution in the seven stages evaluated in the time parameter compared to the evaluation of six nodes with a controlled infrastructure, as shown in Table V. Concerning OPlaceRAN actions, in events between $t0$ and $t1$, we can observe a variation of 20s considering the two topologies, denoting a small-time variation. For a deep time analysis, we show in Table V and expand the task sub-dividing on the main three OPlaceRAN blocks events from $t0.1$ to $t0.4$. In this sense, the difference of 5s in the $t0.2$ RANoptimizer events is worth mentioning, as it will be oriented to the complexity of the placement solution and the adopted network topology and infrastructure. Events $t0.3$ and $t0.4$ also require attention when RANDeployer customizes vNG-RAN nodes. K8S performs the processing between events $t0.4$ and $t1$ allocating Pods. The rest of the events add just 10s to the two evaluations since each OAI chain operates independently, and the free5GC tool can handle the signaling of the 51 virtual nodes in parallel.

Finally, the interconnection of vNG-RAN through crosshaul networks, currently one of the biggest challenges of disaggregated RAN, can be performed by OPlaceRAN. Due to the size of the topology investigated in the proof-of-concept, the chaining of CNFs vCUs, vDUs, and vRUs present very similar latency values. Overall near 18ms on average for the three scenarios presented by the placement in the end-to-end communication approaches between UE and CN (analysis performed within the mobile network, not evaluated in external connections). Two relevant aspects of the latency’s similarities are the number of hops (low processing latency) and the physical distance between nodes (all nodes in the same environment). Furthermore, since during the tests occurred, various latency peaks, to understand this, we have to consider certain aspects: (i) we calibrated and left some slack in the CRs available for the Core Network and RAN Pods, and (ii) we customized the OAI version as container-based with two splits disabling the PHY protocol. Finally, we have not investigated the bandwidth parameter in the evaluation because the maximum obtainable value is not relevant to generating an impact on the network (around 150 Mbps at maximum capacity, as shown in Table II). It is essential to highlight that OPlaceRAN supports bandwidth configuration.

**VI. RELATED WORK**

Recently, RAN faced an intense process of softwareization and virtualization, leading to a broad transformation. In this scenario, the orchestration process has great relevance for developing vNG-RAN. However, the research about virtualized RAN network functions orchestration has generated a divergent view between several works. Mainly concerning supporting the characteristics and requirements of vNG-RAN and the subsequent placement solutions strategies in the experimental orchestrators. In this context, experimental orchestration studies are classified into investigations by open-project organizations and academia. A strong alignment with the NFV architecture and the RAN disaggregation for open-source projects is observed. The projects Open Network Automation Platform (ONAP) [26], [46], Open Source MANO (OSM) [47], and Mosaic 5G [48], stand out in this scenario, led by The Linux Foundation, European Telecommunications Standards Institute (ETSI), and EURECOM, respectively. For example, the project under development by ONAP is natively oriented by the O-RAN initiative. However, open-source
projects lack considering the placement concepts and crosshaul network awareness, despite being design-oriented with an optimization framework and open to generic placement tools. OSM is developing the placement optimization module based on computing, networking, latency, and jitter cost, but without the RAN requirements [27], [28], [29]. Lastly, MOSAIC 5G does not have placement support. It is worth noting that these projects are complementary platforms to OPlaceRAN since there is synergy between the cloud-native tools applied.

For academic purposes, we developed Table VI to guide the analysis of works, showing the state-of-the-art orchestration, placement solutions, NFV architecture, CNF support, and RAN disaggregation. The orchestration characteristic is the most relevant when comparing the related work since there are investigations with different goals. In some studies, orchestration focuses on the fronthaul performance [49], [50], [51], while other works prove concepts related to the framework’s functioning without an orchestration strategy [25], [52], [53]. However, Dalla-Costa et al. [22] and Matoussi et al. [21] provide consistent orchestration goals. The first work introduces a load-balancing algorithm with an analysis of the performance of the fronthaul network and the computational resources. The authors use a synthetic C-RAN, with the behavior results closely regular and constant. The second work developed a particular heuristic approach for the placement of radio functions to reduce the consumption of computational resources. Based on the works presented, OPlaceRAN is the first orchestrator in the literature that supports agnostic placement solutions (widely studied on exact and heuristic models), contributing to the real evaluation of different placement strategies and resulting in the empowering of the vNG-RAN design. OPlaceRAN deals in an integrated way with CRs strategies and topologies features for crosshaul networks.

Two studies about virtualization are aligned with the NFV architecture and support for container technology. The NFV architecture is an essential guide to the virtualization process because its reference model integrates several parts of the fifth-generation of mobile networks, e.g., CN and RAN integration [2]. Moreover, container virtualization support is the leading solution for the granularity of virtualized radio functions [34]. In this sense, Rodriguez et al. [25] present a demonstration in line with the NFV architecture and meets container virtualization under the K8S tool. However, the authors only provide the RAN disaggregation for NG-RAN with three radio units guided by O-RAN concepts without considering crosshaul networks and do not consider any placement solutions. Therefore, despite all the ongoing development of virtualized architectures for the fifth-generation mobile networks, OPlaceRAN fills an open research gap regarding an orchestrator with vNG-RAN placement support aligned with NFV architecture, directives of the O-RAN projects, and evolution from RAN to vNG-RAN. Furthermore, OPlaceRAN is aligned with up-to-date virtualization tools and technologies based on CNFs, including functional RAN splitting requirements, awareness of the crosshaul topology and computational resources, and opportunities for integration with other platforms further than the mobile networks.

VII. Final Remarks

In this work, we have proposed and validated the OPlaceRAN solution to orchestrate placement optimization solutions of virtualized disaggregated radio functions guided by the standards defined for vNG-RAN and are aware of the requirements crosshaul networks and computational resources. Moreover, the OPlaceRAN concept has native support for generic and agnostic placement solutions approaches, providing opportunities for different solutions strategies to leverage the RAN planning evaluations. Furthermore, the concept aligned with the NFV architecture and set with the O-RAN framework brings an up-to-date position of the work in the RAN research. Finally, a prototype based on the cloud-native, CNFs concepts, and K8S tool are developed to validate the conceptual orchestrator.

OPlaceRAN was evaluated in two experimental scenarios. We validate two different placement solutions in the proof-of-concept in the first scenario. Moreover, we demonstrated a practical and agnostic optimization model in an end-to-end real testbed experiment, emulating a mobile network with open-source tools. In the second scenario, OPlaceRAN proved to be a solution to handle a real large topology with 51 nodes, even with a synthetic crosshaul input. In two scenarios, the conceptual solution and experimental results showed the effectiveness of OPlaceRAN in orchestrating control and deployment of placement solutions strategies for vNG-RAN. OPlaceRAN controls the orchestration steps with low computational overhead and low orchestrator block processing and deployment time. Finally, we can conclude that OPlaceRAN contributes to the vNG-RAN transformation and
opens opportunities for integration with other fifth-generation mobile network platforms. For future work, OPlaceRAN opens several opportunities. We envision the integration of our orchestrator with the ONAP and OSM platforms to leverage the development of vNG-RAN. Following the NFV architecture, another integration is to coordinate the crosshaul flow forwarding by a Software-defined Networking (SDN) controller to improve the performance and automatize the crosshaul network’s capacity in a topology with more nodes and complexity. Moreover, a relevant goal is to connect the core network standalone standard and slicing functions and integrate it with RAN placement strategies and O-RAN. Finally, we intend to apply a workload with different requests size and requirements variations (including throughput and latency requirements) based on the number of users within the previous three scenarios.

REFERENCES
[1] Y. Zhang, Network Function Virtualization: Concepts and Applicability in 5G Networks. Hoboken, NJ, USA: Wiley, 2018.
[2] Migration From Physical to Virtual Network Functions—Best Practices and Lessons Learned, GSMA, London, UK, 2018.
[3] M. Liyanage, A. Gurtov, and M. Ylianttila, Software Defined Mobile Networks (SDMN): Beyond LTE Network Architecture. Hoboken, NJ, USA: Wiley, 2015.
[4] “Cloud architecture and deployment scenarios for O-RAN virtualized RAN,” O-RAN Alliance, Alfter, Germany, White Paper, 2020.
[5] Transport Network Support of IMT-2020/5G, Int. Telecommun. Union, Geneva, Switzerland, 2018.
[6] L. J. V. Granville, “Orchestra: A customizable split-aware NFV orchestrator for dynamic cloud radio access networks,” IEEE J. Sel. Areas Commun., vol. 38, no. 6, pp. 1042–1054, Jun. 2020.
[7] M. Mamushiane, A. A. Lysko, T. Mukute, J. Mwangama, and Z. D. Toit, “Overview of 9 open-source resource orchestrating ETSI MANO compliant implementations: A brief survey,” in Proc. IEEE 2nd Wireless Aaf. Conf. (WAC), 2019, pp. 1–7.
[8] “Optimization service design framework.” ONAP. 2020. Accessed: Mar. 19, 2022. [Online]. Available: https://wiki.onap.org/
[9] V. Q. Rodriguez, R. Corbel, F. Guillemot, and A. Peri, “Cloud-RAN factory: Instantiating virtualized mobile networks with ONAP” in Proc. 11th Int. Conf. Netw. Future (NoF), 2020, pp. 120–122.
[10] “ONAP architecture overview.” ONAP. 2019. Accessed: Dec. 8, 2021. [Online]. Available: https://www.onap.org/architecture
[11] “OSM Release EIGHT Release Notes: Open Source MANO—Technical Steering Committee, ETSI, Sophia Antipolis, France, 2020.
[12] E. Eriksson, Placement of Services in Distributed Clouds, Open Source MANO, PoC Overview, ETSI, Sophia Antipolis, France, 2017.
[13] L.-G. Magnusson, Automatic Placement of VNFs, Open Source MANO, OSM Ecosystem Day, ETSI, Sophia Antipolis, France, 2020.
[14] “Network functions Virtualisation (NFV) release 3; architecture: report on the enhancements of the NFV architecture towards ‘cloud-native’ and ‘Paas’,” ETSI, Sophia Antipolis, France, document GR NFV-IFA 029, 2019.
[15] “Kubernetes documentation | Kubernetes.” Cloud Native Computing Foundation. 2020. [Online]. Available: https://kubernetes.io/docs/home/
[16] “Kubernetes documentation.” TL Foundation. 2020. [Online]. Available: https://kubernetes.io
[17] “OpenAirInterface.” O. S. Alliance. 2021. [Online]. Available: https://www.openairinterface.org/
[18] “ONAP architecture overview.” ONAP. 2019. Accessed: Mar. 19, 2022. [Online]. Available: https://wiki.onap.org/
[19] “Network functions Virtualisation (NFV) release 3; architecture: report on the enhancements of the NFV architecture towards ‘cloud-native’ and ‘Paas’,” ETSI, Sophia Antipolis, France, document GR NFV-IFA 029, 2019.
[20] “OpenAirInterface.” O. S. Alliance. 2021. [Online]. Available: https://www.openairinterface.org/
[21] “Free5G.” 2021. [Online]. Available: https://free5g.org/
[22] “OpenAirInterface.” O. S. Alliance. 2021. [Online]. Available: https://www.openairinterface.org/
[23] Y. Hu and J. Wang, “Architectural and cost implications of the 5G edge NFV systems,” in Proc. IEEE 37th Int. Conf. Comput. Design (ICCD), Sep 2019, pp. 594–603.
[24] “Small cell virtualization functional splits and use cases.” Small Cell Forum release 7.0. Jan. 2016. [Online]. Available: https://scf.io/en/documents/159_-_Small_Cell_Virtualization_Functional_Splits_and_Use_Cases.php
[25] PASSION Project. “Photicom technologies for programmable transmission switching on femto systems based on scalable spectrum aggregation for future agile high capacity metro networks.” 2020. [Online]. Available: http://www.passion-project.eu/project/
[26] “IEEE alliance.” 2021. [Online]. Available: https://ethernetalliance.org/
[27] “Open network automation platform.” L. Foundation. 2020. [Online]. Available: https://www.onap.org/
[28] “Open source MANO.” 2020. [Online]. Available: https://osm.etsi.org/
[29] “Mosaic5G: Agile and flexible service platforms for 5G research,” ACM SIGCOMM Comput. Commun. Rev., vol. 48, no. 3, pp. 29–34, 2018.
[49] N. Makris, C. Zarafetas, K. Choumas, P. Flegkas, and T. Korakis, “Virtualized heterogeneous 5G cloud-RAN deployment over redundant wireless links,” in Proc. IEEE Int. Symp. Local Metrop. Area Netw. (LANMAN), 2019, pp. 1–2.

[50] K. Kondepu, N. Sambo, F. Giannone, P. Castoldi, and L. Valcarenghi, “Orchestrating lightpath adaptation and flexible functional split to recover virtualized RAN connectivity,” in Proc. IEEE Opt. Fiber Commun. Conf. Expo., 2018, pp. 1–3.

[51] R. K. Saha, Y. Tsukamoto, S. Nanba, K. Nishimura, and K. Yamazaki, “Novel M-CORD based multi-functional split enabled virtualized cloud RAN testbed with ideal fronthaul,” in Proc. IEEE Globecom Workshops, 2018, pp. 1–7.

[52] S.-Q. Lee, J.-U. Kim, and H. Yoo, “Virtualization and orchestration of radio access network,” in Proc. IEEE Int. Conf. Commun. Syst., 2018, pp. 321–325.

[53] B. Dzogovic, B. Santos, V. T. Do, B. Feng, N. Jacot, and T. Van Do, “Connecting remote eNodeB with containerized 5G C-RANs in OpenStack cloud,” in Proc. 5th IEEE Int. Conf. Edge Comput. Scalable Cloud, 2019, pp. 14–19.

Fernando Zanferrari Morais received the bachelor’s degree from Centro Universitário Lasalle in 2009, and the M.Sc. degree from the University of Vale do Rio dos Sinos, Brazil. His research interests include telecommunication mobile networks, software-defined networking, virtualization, and cloud computing.

Gustavo Zanatta Bruno received the M.Sc. degree in computer science from Universidade Federal Fluminense, Brazil. He is currently pursuing the Ph.D. degree in computer science with the University of Vale do Rio dos Sinos, Brazil. He served as an Information Technology Infrastructure Analyst with MTI, a public company responsible for IT services, Mato Grosso, Brazil. His research interests include infrastructure automation, mobile telecommunication networks, and cloud computing.

Julio Renner graduated in computer science from the University of Vale do Rio dos Sinos, Brazil. His research interests include software containerization and orchestration, DevOps, and cloud computing.

Gabriel Matheus F. de Almeida has been a Computer Science Researcher and a member with the Laboratory Computer Networks and Distributed Systems, Federal University of Goiás, Brazil, since 2019. His research interests include wireless networks, software-defined networks, virtualization, resource allocation, and performance evaluation.

Luis M. Contreras received the first M.Sc. degree in telecom engineering from the Universidad Politécnica of Madrid in 1997, and the second M.Sc. degree in telematics from the Universidad Carlos III of Madrid in 2010. Since August 2011, he has been the part of Telefónica I+D/Telefónica CTIO Unit, working on 5G, SDN, virtualization, and transport networks. He is a part-time Lecturer with the Universidad Carlos III of Madrid, where is also pursuing a Ph.D. (expected for 2021).

Rodrigo da Rosa Righi received the M.S. and Ph.D. degrees in computer science from the Federal University of Rio Grande do Sul, Brazil, in 2005 and 2009, respectively. He is a Professor and a Researcher with the University of Vale do Rio dos Sinos, Brazil. He concluded his post-doctoral studies with the Korean Advanced Institute of Science and Technology, South Korea. He is a member of both IEEE Computer Society (senior member) and ACM.

Kleber Vieira Cardoso received the degree in computer science from the Universidade Federal de Goiás (UFG) in 1997, and the M.Sc. and Ph.D. degrees in electrical engineering from the COPPE—Universidade Federal do Rio de Janeiro in 2002 and 2009, respectively. He is an Associate Professor with the Institute of Informatics, UFG, where he has been a Professor and a Researcher since 2009. He spent his sabbatical with Virginia Tech, USA, in 2015, and Inria Saclay Research Centre, France, in 2020.

Cristiano Bonato Both is a Professor with the Applied Computing Graduate Program, University of Vale do Rio dos Sinos, Brazil. He coordinators research projects funded by H2020 EU-Brazil, CNPq, FAPERGS, and RNP. His research focuses on wireless networks, next-generation networks, softwarization, and virtualization technologies for telecommunication networks.