IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 5, 2020, accepted August 14, 2020, date of publication August 19, 2020, date of current version August 31, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3017917

eXP-RAN–An Emulator for Gaining Experience With Radio Access Networks, Edge Computing, and Slicing

JOÃO PAULO ESPER[1], ABDALLAH S. ABDALLAH[2], (Member, IEEE), STUART CLAYMAN[3], (Member, IEEE), WALDIR MOREIRA[4], ANTONIO OLIVEIRA-JR[1,4], (Member, IEEE), SAND LUZ CORREA[1], AND KLEBER VIEIRA CARDOSO[1], (Member, IEEE)

[1]Institute of Informatics (INF), Federal University of Goiás (UFG), Goiânia 74690-900, Brazil
[2]Penn State Erie-The Behrend College, Erie, PA 16563, USA
[3]University College London, London WC1E 6BT, U.K.
[4]Fraunhofer Portugal AICOS, 4200-135 Porto, Portugal

Corresponding author: João Paulo Esper (joaopauloesper@inf.ufg.br)

This work was supported in part by the Horizon 2020 EU-Brazil Joint Call (NECOS-Novel Enablers for Cloud Slicing) funded by the European Commission and the Brazilian Ministry of Science, Technology, Innovation, and Communication (MCTIC) through Rede Nacional de Ensino e Pesquisa (RNP) and Ciência Tecnologias Inovação e Comunicação (CTIC) under Grant 777067, in part by the Fundação de Amparoà Pesquisa do Estado de Goiás (FAPEG) in the Call 05/2016 under Grant 201610267001193, and in part by the Fundação para a Ciência e a Tecnologia (FCT), Portugal, in 2020.

ABSTRACT Radio Access Networks (RAN), Edge Computing (EC), and Network Slicing are some of the critical components in 5G networks used to provide different transport services over the same infrastructure and to enable new features, e.g., low latency and context awareness. It is important to have tools for evaluating present and future applications over these components; however, there is a gap between theoretical work on resource allocation and its deployment in simulated, emulated, or real-world experiments. This paper addresses these issues through a new emulation software named eXP-RAN, which enables experimenting with network slicing in virtualized RAN nodes and EC scenarios. In this version, we have focused on the computing part of RANs. Experiments can be created manually or imported from an optimization model. We describe the software architecture and its advantages over traditional tools such as ns-3 and Mininet, as well as more recent tool targets to EC simulations such as EdgeCloudSim. We also illustrate how two use-cases, a virtual RAN (vRAN) / Multi-Access Edge Computing (MEC) orchestration and a network slicing for video service providers can be emulated and explored using eXP-RAN.

INDEX TERMS Radio access networks, edge computing, network slicing, emulation tool.

I. INTRODUCTION

Since mobile networks became important tools for human daily lives, users have been creating high expectations about the increased capacity and the new services offered by the next generation of these networks. For 5G networks, this means gigabits per second and support for the 3GPP standardized types of scenarios [1], [2], i.e., enhanced mobile broadband (eMBB, e.g., UHD/4K/8K videos); ultra-reliable low-latency communications (URLLC, e.g., autonomous cars); massive machine-type communications (mMTC, e.g., smart cities, sensor networks, IoT). However, these new services do not depend only on more bandwidth, as in the previous generations of mobile networks, but they also need low latency, high reliability, edge computing capabilities, resource partitioning, among other features. New technologies, such as Network Function Virtualization (NFV) [3], Multi-Access Edge Computing (MEC) [4], and Network Slicing [5], have been introduced as part of the 5G networks standards. A large part of the success of the 5G networks depends on these new technologies and its proper integration with the Radio Access Network (RAN) in

The associate editor coordinating the review of this manuscript and approving it for publication was Petros Nicopolitidis.

such a way that meets user expectations. Thus, there is great interest in experimenting with these multiple technologies in order to anticipate problems and identify opportunities.

Nowadays, tests involving RAN, Edge Computing (EC), and network slicing primarily rely on simulations or small testbed environments. The simulation offers scalability, repeatability and allows the users to evaluate several aspects such as network throughput, packet loss, power and CPU consumption [6]. However, for a service or application to be evaluated, it must be recreated within the simulator, employing the specific abstractions of that simulator. This is generally time-consuming and may not accurately represent real-world systems. Simulation tools also lack abstractions to offer truly isolated views of physical and virtual resources, which is a relevant characteristic for network slicing [7], [8]. This happens because simulation tools work with models that simulate the behavior of a physical or logical resource, while emulation tools try to mimic the resource itself. This is a point which gives emulation tools advantage over simulation tools to create isolated views of physical and virtual resources. On the other hand, the smaller test environments [9], [10] fail to provide scalability, limiting the scope of the evaluations, especially regarding RAN topology and service coverage. Even setting up a small real-world environment involving RAN, EC, and network slicing is also a complex, error-prone, and time-consuming activity. Repeatability is also a common issue, mainly for third-parties trying to compare with previous work. There are a few emulation tools (e.g., [11]) that allow users to evaluate isolated aspects of a service, such as the network performance of a specific virtualized function. However, these tools do not have the abstractions and features for implementing a complete evaluation environment.

In this work, we introduce eXP-RAN, an emulation tool designed to lower the barrier for gaining experience with RAN and EC. eXP-RAN, in its current version, is focused on the computing part of RANs and provides a modular architecture that allows the user to experiment with a variety of functionalities related to RAN and EC such as desegregated RAN modeling, network slice modeling, service representation through lightweight container-based virtualization, and the monitoring of the entire emulated environment. eXP-RAN by design supports the emulation of multiple scenarios where the base stations, MEC hosts and cloud servers are running in coordination. In the case of emulation involving desegregated RAN, the model to convert radio processing functions (generally, virtualized) into computational resources have to be provided by the user either as the output of an optimization model or as a service file. In addition, eXP-RAN provides two useful functionalities: (1) the ability of converting the output of an optimization model into an infrastructure and service description that can be processed and emulated by eXP-RAN; and (2) the capability of auto-tuning. The first capability allows users to perform sensitive analysis of optimization models, which can be time consuming when performed using the solver. The second capability allows users to scale the emulation scenarios while keeping the same hardware capacity. eXP-RAN is publicly available as an open-source project.[1]

To demonstrate the capabilities of eXP-RAN, we designed two use-cases. In the first use-case, we use eXP-RAN to emulate a vRAN/MEC orchestration model. In the second use-case, we use eXP-RAN to emulate the deployment of two network slices with different Quality of Service (QoS) demands and analyse how both slices are affected as workload increases.

This paper is organized as follows. Section II introduces important requirements that must be satisfied by a tool such as eXP-RAN and provides an overview of existing tools and to which extent such tools support the desired requirements. Section III presents the system architecture, design layout, and implementation details of eXP-RAN. Section IV presents the auto-tuning feature of eXP-RAN. In Section V use-cases are emulated with eXP-RAN and the results of the experiments are discussed. Finally, Section VI presents final considerations and future work.

II. REQUIREMENTS AND RELATED TOOLS

In this section we briefly discuss the relevant requirements for a tool which goal is to allow users to analyze different engineering problems in the RAN, EC, and network slicing domain. We also summarize the extent to which currently existing tools are able to fulfill several requirements. The list of requirements is presented below.

Slicing abstraction – the need to create subsets of virtualized and isolated resources is required for supporting new applications (e.g., distributed AR/VR) and business models (e.g., wholesale sales for vertical industries). Thus, an emulation (or simulation) tool must offer a proper slicing abstraction in order to assess the deployment of concurrent and new services over a single shared substrate. From the service providers' perspective, a slice must appear as a traditional set of resources that they are able to monitor, control, and manage. From the infrastructure providers' perspective, it should be possible to create or destroy independent and isolated slices.

Network modeling – some characteristics of the backhaul from a RAN (and, eventually, of the fronthaul) must be represented. For example, link capacity and propagation delay need to be represented in order to evaluate optimization models involving the RAN. Also, these characteristics must be available, not only for the physical substrate, but also for every virtual network slice created over this substrate. The ability to emulate real-world network technologies, such as the TCP/IP stack and Software-Defined Networking (SDN) switches, can also increase the accuracy level in the evaluations.

Computing modeling – EC allows to push applications, data, and computing power to the edge of the network, at the proximity of data sources or destinations. Thus, in addition to network modeling, an emulation (or simulation) tool for

[1] https://github.com/LABORA-INF-UFG/eXP-RAN

EC scenarios must offer a way to represent virtualized computing resources (e.g., CPU and RAM). Additionally, the computing modeling must be integrated with the network modeling in order to allow, for example, the representation of virtual network functions.

Service representation – 5G networks are creating the need for service providers to verify the behavior of the set of applications that compose their services in a given set of virtualized resources. Nowadays, it is a common practice to represent services as collections of virtual functions, implemented, for example, as containers. Also important is a way to formally model or represent the entire service, and in this respect, some common approaches are optimization models and service descriptions in a formal language.

Service reconfigurability – the ability to scale up, scale down, and elastically redirect loads or replace resources are extremely important features to support running services over virtualized resources. In order to implement these features, we must be able to reconfigure the service by increasing / decreasing the number of virtual functions, migrating virtual functions, (re) chaining them, and other operations involving the service elements during run time, both seamlessly and without service interruption.

Predictable performance – in the context of RAN and EC, there are several metrics that depend on fine-tuning the resource allocation of the computing resources among the software components. In an event-driven simulator, this is not a concern, but predictable performance is critical for emulators. For example, throughput, delay, and CPU consumption are affected by the amount of computing resources allocated to a software component. Thus, an emulation tool needs to identify the limitations of the hardware and properly adjust the allocation of the resources.

Monitoring – a key feature of any evaluation tool is the ability to monitor the test environment. Traditional metrics such as CPU and RAM consumption, bandwidth, packet delay and loss are very common in almost every evaluation tool. However, the ability to monitor independently each network slice is rare. It is also important to have flexibility in the monitoring so that the experimenter can: choose which metrics need to be followed online, choose which metrics need to be persisted, compute statistics over the metrics, and add new metrics.

Ease of use – tools such as eXP-RAN may have a wide range of potential users with diverse background knowledge. Thus, it is important to provide enough abstractions to allow beginners to start experimenting with the tool without difficulties. Afterwards, documentation, basic examples, and full use-cases can pave the way for users to master a tool and achieve their goals.

Reusability – many emulation and simulation tools have reached large adoption and a long life due to, at least in part, high reusability. A modular architecture and properly designed components are key aspects for reusability, mainly in large and complex software systems. In the context of 5G networks, service representations and slicing abstractions should be easily composed, reused, and extended.

Table 1 summarizes to which extent some available tools fulfill the requirements discussed above. The adhering level is directly related to our description of the requirements, as explained below.

TABLE 1. Some relevant tools comparable to eXP-RAN.

Requirement	VLSP/ Slice Controller	Mininet	Fogbed	EdgeCloudSim	ns-3
Slicing abstraction	M	L	L	–	–
Network modeling	L	H	H	M	H
Computing modeling	H	M	H	H	L
Service representation	M	L	H	M	–
Service reconfigurability	H	L	M	L	L
Predictable performance	–	–	–	H	H
Monitoring	M	M	M	M	M
Ease of use	H	H	H	H	M
Reusability	H	H	H	H	H

(H) High (M) Medium (L) Low (–) Not Available

Among the reviewed tools, none of them covers all the requirements (as shown in Table 1). The Very Lightweight Network & Service Platform (VLSP) [12] is a tool that can emulate / simulate different aspects in SDN and virtualized environments. Slice Controller [13] was introduced as a companion software for slicing resources of a virtual data center, where an isolated instance of VLSP can act as a Virtual Infrastructure Manager (VIM) of each slice. Since this software is focused on data centers, network slicing is simplified, while network characteristics, such as generic topologies, link capacity, and propagation delay are not properly represented. Service representation involves the creation of applications in Java, which is the programming language used to develop this software. Monitoring is quite flexible, but was not designed with slicing support as a goal.

Mininet [14] is a popular emulation tool used to create realistic virtual networks, and to experiment with OpenFlow and SDN systems. Indeed, some aspects of eXP-RAN were inspired by Mininet, mainly those related to the network representation and ease of use. The native support for slicing in Mininet is based on third-party tools such as FlowVisor [15] and OpenVirteX [16], which consist on creating virtual SDNs by using an OpenFlow controller proxy. Thus, the slicing abstraction is tied to OpenFlow rules and does not provide a representation for a physical substrate. Even though Mininet is based on containers, natively, the service representation depends on the user creating and composing her application with containers.

Fogbed [17] is an emulation framework that focuses on fog computing environments. Fogbed has several tools in common with eXP-RAN, but the software has different goals. Fogbed also has adopted a design strategy with potential

advantages, but with some serious risks. This software is built on top of other large software sets, such as Containernet [18] and MaxiNet [19], which are forks from Mininet. Thus, Fogbed is highly dependent on multiple non-mature software tools that are managed by communities of developers with their own roadmaps. Additionally, slicing abstraction is an inheritance from Mininet and predictable performance is not a concern in the design of Fogbed.

EdgeCloudSim [6] is a simulator focused on edge computing to allow experiments that involve modeling of networking and computing resources. EdgeCloudSim was based on the known simulator CloudSim [20] and was improved with features to support sophisticated network models, mobile devices, and more realistic load generation. EdgeCloudSim allows the application properties to be defined in a specification file, but the application itself must be developed in Java, i.e., the programming language of the simulator. Additionally, the simulator was not built with slicing abstraction support as a goal and this change has a large impact in its code base.

The ns-3 [21] is one of the most widely used discrete-event network simulators. To introduce general support for a slicing abstraction in ns-3 would imply a large number of modifications to the simulator code. Thus, those interested in using ns-3 have been making specific changes in order to attend to their needs [22], [23], for example, associating IDs to flows, or grouping resources in a customized way. Similar to VLSP, in ns-3 the service representation requires creating an application inside the simulator, but using C++ or Python as the programming language. However, differently from VLSP, ns-3 does not properly deal with virtualized environments, despite some non-scalable initiatives [24], [25]. This means ns-3 lacks the proper support for service representation, plus its support for service reconfigurability is low. Although the monitoring requirement is traditionally well satisfied by simulation tools, ns-3 does not support isolated multi-perspective monitoring. This explains, for example, why it scored as `Medium` with respect to the monitoring support.

In summary, eXP-RAN was designed by taking into consideration all the previous requirements. The native support for slicing abstraction, service representation, and predictable performance are some of its key characteristics. Its slicing abstraction can be easily described as a specification listing a subset of the network and computing resources. eXP-RAN can represent services as containers. Additionally, eXP-RAN also introduces the self-tuning functionality, which aims to avoid inaccuracy in evaluations due to hardware limitations.

III. eXP-RAN: DESIGN AND IMPLEMENTATION

This section presents the software architecture, system modules, and implementation details of eXP-RAN.

A. SOFTWARE ARCHITECTURE

The eXP-RAN architecture is divided into three layers as depicted in Fig. 1. The *Infrastructure layer* is responsible for creating virtual resources (e.g., machines, links, switches), which emulate physical ones. The *Slicing layer* manages sets of virtual resources by partitioning and isolating them, consequently representing slices in the infrastructure. The *Service layer* deploys a service abstraction over a given slice. These three layers are discussed in detail.

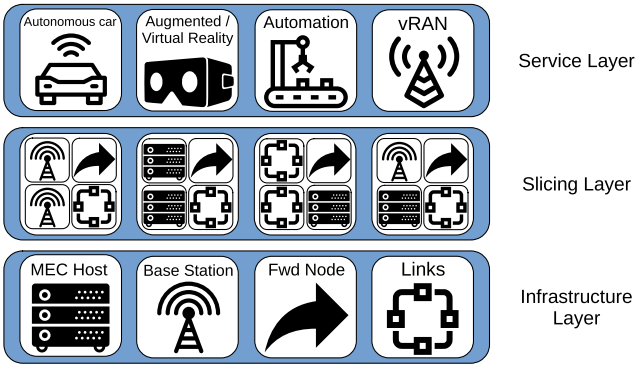

FIGURE 1. eXP-RAN layers.

1) INFRASTRUCTURE LAYER

The Infrastructure layer was designed following the standards terminology defined by ETSI – European Telecommunications Standards Institute [26]. This layer is responsible for the emulation of the four major network elements in a 5G RAN, which are: (i) edge computing resources (e.g., MEC hosts), (ii) base stations (e.g., 4G/eNodeB, 5G/gNB), (iii) forwarding nodes, and (iv) links. Fig. 2 shows an example of a topology that can be emulated in eXP-RAN with all four types of network elements. Thus, in eXP-RAN, a network topology is emulated by nodes (MEC host, base station, and forwarding nodes) and links.

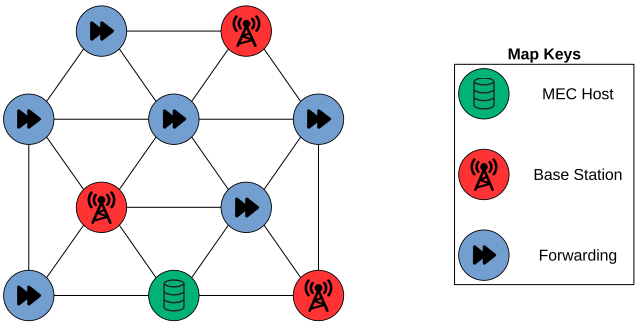

FIGURE 2. Example of an eXP-RAN topology.

Forwarding nodes are only used to forward traffic; therefore they have no computing capacity. Base station nodes represent the cell sites of the RAN. They need to emulate signal processing for the RAN, and thus they have a limited computing capacity. MEC hosts, on the other hand, are the processing power of the RAN, thus they possess higher computing capacity compared to base station nodes. In real-world, computing capacity is represented by physical servers.

In eXP-RAN, we emulate physical servers using Virtual Machines (VMs).

2) SLICING LAYER

This layer emulates the slices created over the infrastructure, i.e., the set of virtual resources and services owned and managed by a certain tenant. In eXP-RAN, virtual resources are emulated by containers and links. Services, on the other hand, are emulated by Virtualized Network Functions (VNFs) running as containers. Thus, a slice includes a combination of containers, links and VNFs running inside the containers.

3) SERVICE LAYER

This layer emulates the services being offered within the slices. Services are offered as a set of VNFs. The service layer is mainly responsible for configuring the VNFs inside the containers, so that they can properly run as an entire service.

B. WORKFLOW AND SYSTEM MODULES

Fig. 3 shows the main steps to go through an emulation with eXP-RAN. The system currently supports three different ways of user interactions. In the first method (step 1a in Fig. 3), the user provides the output of an optimization model. This method is useful to perform sensitive analysis of optimization models, which can be time consuming when performed using the solver. eXP-RAN, on the other hand, can be used in this situation to provide quick insights on different parameters of interest.

The second method of interaction with eXP-RAN is by using the Topology Generator (step 1b in Fig. 3). This is a module of the system that generates random network topologies for experimenting with RAN and EC. This method is particularly interesting for users with limited knowledge of RAN/EC scenarios as well as to support educational purposes.

The third method of interaction allows the user to specify her own infrastructure and services (step 1c in Fig. 3). In this case, the user describes the infrastructure and services by writing JSON files following the eXP-RAN notation. Listing 1 and 2 are part of an infrastructure description example according to this notation: Listing 1 describes infrastructure nodes of different types (forwarding and base station nodes), VMs, and containers; while Listing 2 defines the infrastructure links connecting these nodes. This is the most flexible method of interaction with the system.

Alternatively, users can interact with eXP-RAN using the benchmarker module (step 7 in Fig. 3). Note that step 7 has double meaning as it is an optional flow and an user input.

In the current version of eXP-RAN, there are six main modules available. These modules are described below.

The **Model Adapter module** is responsible for converting the output of an optimization model into an Infrastructure and Service Description file (step 2a in Fig. 3) for eXP-RAN. As of today, the Model Adapter module is a proof of concept

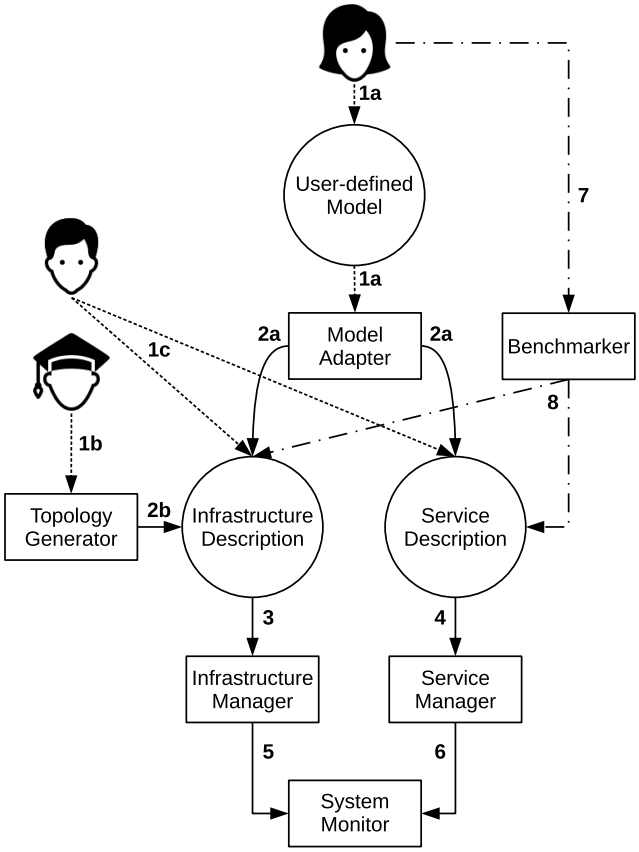

FIGURE 3. eXP-RAN workflow and system modules. Single arrows represent mandatory flow; arrows only with dashed lines identify user inputs; arrows composed by dots and dashed lines represent optional flow; rectangles identify the system modules and circles represent input files.

```
{
    "nodes": [
        {
            "nodeNumber": 1,
            "nodeType": "Forwarding"
        },
        {
            "nodeNumber": 2,
            "nodeType": "BaseStation",
            "vms": [
                {
                    "vmNumber": 1,
                    "cpu": 1,
                    "ram": 1024,
                    "containers": [
                        {
                            "ctnNumber": 1,
                            "cpu": 0.2,
                            "ram": 512
                        }
                        ...
}
```

Listing 1. Nodes described in an Infrastructure Description file in JSON format.

using the CPLEX[2] solver and must be customized by the user for each different model, including the ones imported

[2]https://www.ibm.com/analytics/cplex-optimizer

TABLE 2. Ethernet-based link profiles (proc. delay = 5 (μs), packet size = 1518 Bytes).

Technology	Throughput (Gbps)	Prop. delay (μs)
mmWave (60-80 GHz)	0.9, 1.25, 1.5, 2, 3, 4, 8	1-20
μWave (6-60GHz)	0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1, 1.25, 1.5, 2	1-100
Copper (1000/10G/40GBASE-T)	1, 10, 40	0.05-0.5, 0.275, 0.15
SMF fiber @ 1310 nm (1000, 10G, 40, 100GBASE-EX, LR, LR-4)	1, 10, 40, 100	1-200, 50, 50, 50
SMF fiber @ 1550 nm (1000, 10G, 40, 100GBASE-ZX, ER, ER-4)	1, 10, 40, 100	1-350, 200, 200, 200
TbE (*under development)	200, 400	1-50

```
{
    "links": [
        {
            "linkType": "Nodes",
            "Connections": [
                {
                    "linkNumber": 1,
                    "fromNode": 1,
                    "toNode": 2,
                    "delay": 0.122,
                    "capacity": 50.0
                }
                ...
}
```

Listing 2. Links described in an Infrastructure Description file in JSON format.

from CPLEX. Models implemented in other solvers can also be used as long as the user modifies the Model Adapter module. However, customizing the existing Model Adapter module for CPLEX or other solvers models is much easier than implementing the whole solution from scratch, since eXP-RAN specifies well defined interfaces and generalizations to support such conversions.

The **Topology Generator module** is designed to generate random RAN topologies. In order to generate realistic topologies, this module takes into account real-world topology metrics such as graph topology, type of nodes (forwarding, MEC host, and base station) and their proportions, and the characteristics of the links. Random topologies are generated from a Waxman graph [27], a type of graph commonly used to represent RAN topologies in the literature [10], [28]. Once the topology is created, nodes are labeled according to the following proportions: 5% of the nodes are labed MEC host nodes; 30% are base stations; and 65% are forwarding nodes. These percentages are derived from studies conducted in [28]. Finally, links are characterized by randomly choosing one of the profiles described in Table 2. These profiles are derived from studies conducted in [10]. The generated topology is then converted into an Infrastructure Description file (step 2b in Fig. 3). It is important to mention that the Waxman graph does not label the vertex of the graph. The labeling approach occurs after the graph being generated and was proposed in [28] to represent RAN topologies. Thus, it is possible that, after labeling, the resulting topology does not match a physical one, although it keeps its properties. Indeed this is a known problem and despite of that, the methodology proposed in [28] has been used to represent the topologies of RANs, e.g., in [10]. Alternatively, as mentioned before, the user can provide her own topology by describing it as a JSON file. In the JSON file, the user can specify the percentage of each node, node configuration, link profile, connectivity, etc.

The **Infrastructure Manager module** takes as input an Infrastructure Description file (step 3 in Fig. 3) – generated either by the Model Adapter, or the Topology Generator, or provided by the user – and instantiates an environment that emulates the infrastructure description accordingly. Since services may have constraints regarding how traffic between two service components should be routed (e.g., load balancing constraints), eXP-RAN also has a Network Controller that allows the users to specify paths in the network to route the traffic of the services, and to apply network rules accordingly to ensure these constraints. The network rules are in the form of OpenFlow rules that are applied to Open vSwitch (OvS) [29] switches in the network as needed.

A slice abstraction is implemented in eXP-RAN by a combination of tools such as Docker, OvS, Linux Traffic Control (LTC) and OpenFlow. The isolation among the slices is achieved because each Docker container is a process isolated from other containers by default and we create an overlay network for each slice using OvS. Furthermore, LTC is used to set different bandwidth and latency constraints on each slice and OpenFlow rules are responsible for managing the network traffic of each slice in the emulation. OpenFlow manages the network traffic by applying network rules on each virtual switch to properly forward each packet. The user does not need to know OpenFlow in order to specify the rules to route the traffic of her service. For example, in the vRAN use-case, OpenFlow is used to represent vRAN splits and the necessary OpenFlow rules are specified in an abstract way by the user. The user is able to do that simply by setting the same first and last node in the Service Description file. Other OpenFlow rules to manage the network traffic include ARP rules that are also set in an abstract way once the user specifies the source and destination of each service. All OpenFlow rules are created and managed in the emulation in an abstract way by converting the description files provided by the user to each rule.

```
{
    "services": [
        {
            "flows": [
                {
                    "flowIdentifier": 1,
                    "bandwidth": 125,
                    "nodes": [3, 1, 2],
                    "ctnSourceNum": 1,
                    "ctnTargetNum": 3
                },
                {
                    "flowIdentifier": 2,
                    "bandwidth": 100,
                    "nodes": [3, 1, 2],
                    "ctnSourceNum": 2,
                    "ctnTargetNum": 4
                }
                ...
}
```

Listing 3. Example of a Service Description file in JSON format in eXP-RAN.

Given the Infrastructure Description file, the Infrastructure Manager first creates each type of node and link, then it applies the limitations to the links. After that, the Infrastructure Manager starts the creation and configuration of the VMs (representing the computing capacity of the nodes) inside the nodes followed by the creation of the containers (representing the virtual resources) inside the VMs.

As mentioned before, eXP-RAN emulates a service as a set of applications (VNFs) running inside containers. Containers, in turn, are deployed as part of the infrastructure. However, some services may need specific configurations in order to work properly. In eXP-RAN, configurations related to services are described in the Service Description file and are applied by the **Service Manager module** (step 4 in Fig. 3). The Service Description file can be generated by the Model Adapter or provided by the user. Listing 3 shows an example of this type of file. The service configuration takes place after the infrastructure being instantiated by the Infrastructure Manager.

The **System Monitor module** is designed for collecting specific monitoring metrics related to the infrastructure and the services (steps 5 and 6 in Fig. 3). The collected metrics are saved in text files and can be accessed by the user in order to understand the results of the emulation.

Finally, the **Benchmarker module** is a powerful built-in feature in eXP-RAN that is not available in any of the tools analyzed in Section II. The Benchmarker ensures the predictable performance requirement discussed in Section II by configuring the eXP-RAN tool according to the hardware capacity where it is running. In this context, this module helps the user make proper choices during the process of configuring an emulator (steps 7 and 8 in Fig. 3). More details on how the Benchmarker can be used are given in Section IV.

C. IMPLEMENTATION DETAILS

This subsection details how we implemented the three types of nodes, available in eXP-RAN, to emulate a network topology. Fig. 4 illustrates the main elements of each node.

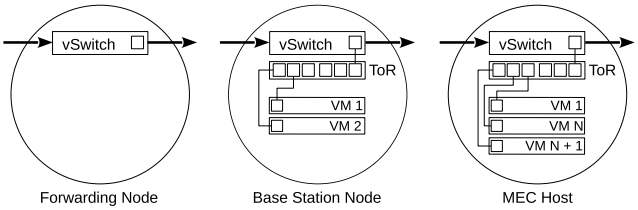

FIGURE 4. eXP-RAN type of nodes.

1) FORWARDING NODE

Fig. 5 shows the technology used to implement a forwarding node. Since forwarding nodes are only used to forward traffic, they have a single virtual switch. This virtual switch is implemented using OvS and configured using LTC and OpenFlow rules. LTC is needed in order to emulate the link restrictions, e.g., capacity (or bandwidth) and latency, while OpenFlow is used to route the traffic of the service.

2) BASE STATION NODE

Fig. 5 shows the technology used to implement a base station node. This node has some VMs to represent the limited computing capacity and a couple of virtual switches. The first switch works like the switch mentioned on the forwarding nodes, i.e., it does packet forwarding. The second one is used to connect the VMs inside the node, i.e., it works as a Top of Rack (ToR) switch. OvS, LTC and OpenFlow are used with the same purpose as in the forwarding nodes. We use Xen VMs to represent the physical servers and Docker containers to represent virtual resources.

3) MEC HOST

Fig. 5 shows the technology used to implement a MEC host node. MEC hosts are nodes with high computing capacity (i.e., multiple VMs) and two virtual switches that have the same functions as the switches of a base station node. Indeed the main difference between base station nodes and MEC hosts is the amount of (physical and virtual) computing resources in the node, i.e., the amount of VMs and Docker containers.

Fig. 6 summarizes how some real-world components are emulated by eXP-RAN.

IV. BENCHMARKING A SERVER BEFORE EMULATION

Many of the potential users of eXP-RAN may use hardware whose computer resources are not sufficient to emulate certain types of services, e.g., a network load generation with demanding throughput rates. Usually, such limitations are caused by the CPU capacity. For example, in both use-cases that will be presented in Section V, we use a CPU configuration of 2x Intel Xeon Silver 4114 Processor. Even with

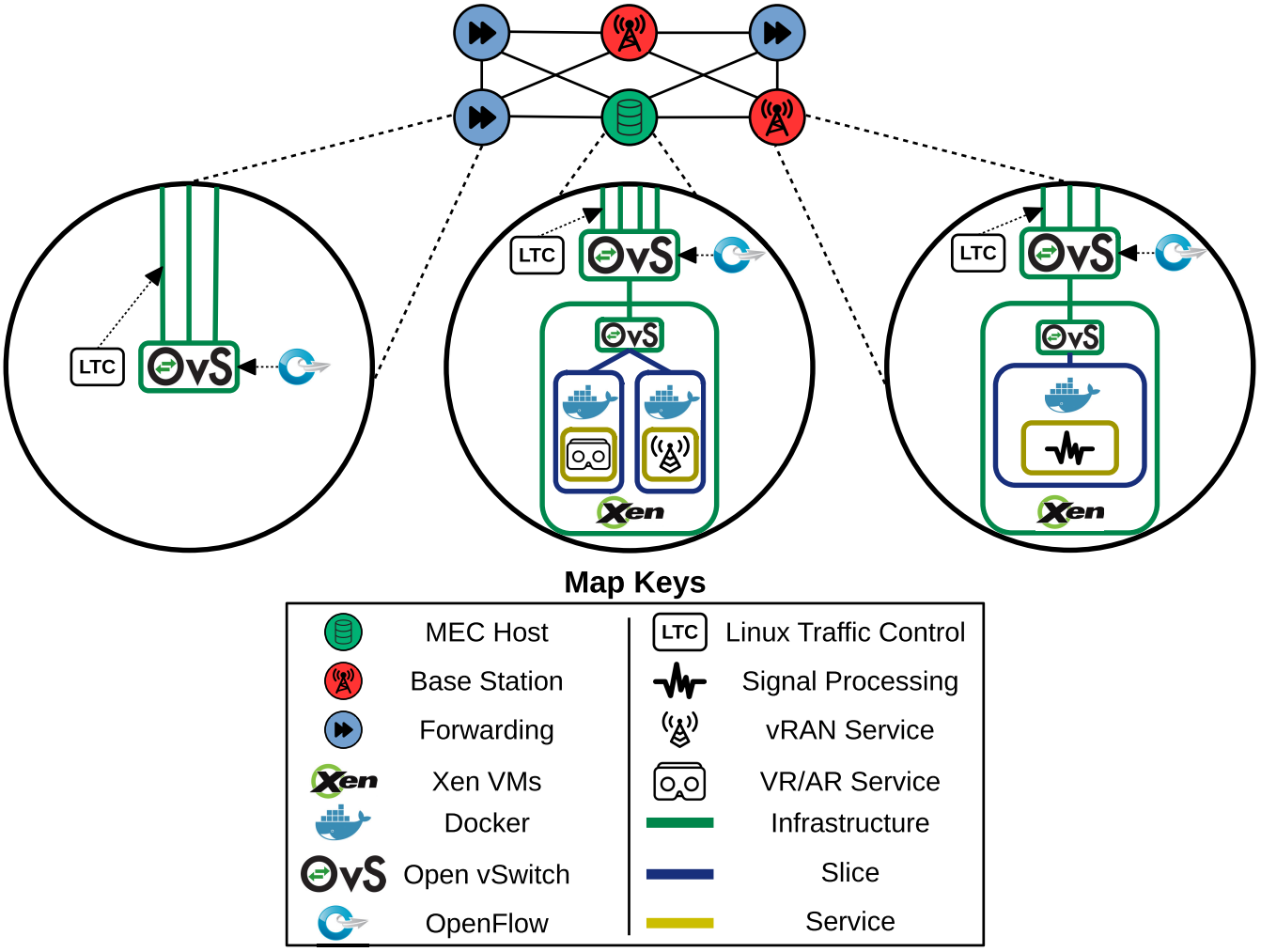

FIGURE 5. Forwarding, base station and MEC host.

Real world	Emulation	Key
Physical server	Xen virtual machine	1
Virtual machine	Docker container	2
Physical switch	OvS virtual switch	3
RAN service	Emulated service	4
Physical network interface card (NIC)	Xen virtual network interface	5
Virtual network interface (VIF)	Docker virtual network interface	6

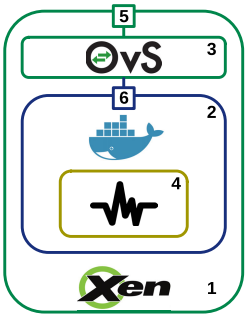

FIGURE 6. Real-world components and how they are represented in eXP-RAN.

its combined resource of 20 cores, 40 threads, and a boost clock up to 3 GHz, the server could not fulfil the scenarios we wanted to emulate. Given that the real scenarios can easily scale up to more than one hundred nodes (varying from MEC hosts, base stations, and forwarding nodes), each one requiring one CPU core to either generate or process 2.5 Gbps of throughput, our hardware was not able to fulfil these requirements.

The Benchmarker module in eXP-RAN was designed with this limitation in mind. The idea is that the Benchmarker configures the tool, so that eXP-RAN can only use the amount of computer resources provided by the user, while generating an emulation very close to real-world performance. To do that, the Benchmarker needs the following information: the throughput that should be generated, and the accepted packet loss (i.e., how much

of packet loss the user considers reasonable for her use-case).

The benchmark is performed in two phases. In the first phase, the Benchmarker starts with the 'best-case scenario', i.e., it tries to generate and process the exact throughput provided by the user, using one vCPU to generate and another one to consume the throughput. After this first attempt, if the packet loss is equal to or less than the one accepted by the user, the Benchmarker goes to the second phase of the benchmark test. Otherwise, the Benchmarker scales down the throughput, i.e., it reduces the throughput rate in order to comply with the specified packet loss. This is done by dividing the throughput by two and running the test again. If the test result does not have the acceptable packet loss, the Benchmarker divides the throughput again, but now by ten. This process is repeated, dividing the throughput rate by 2 and by 10, alternately, until the Benchmarker finds a throughput rate that can be processed within the accepted packet loss. Alternatively, the user can provide the series of numbers to be used to scale down the throughput rate.

Once the Benchmarker finds a throughput that complies with the packet loss specified by the user, it proceeds to the second phase of the benchmark test. The goal of this phase is to ensure that the emulation uses the most of the hardware. Given that, in most cases, the number of flows the user wants to generate is higher than the amount of cores and threads available, it is not reasonable to use a full vCPU to either process or generate a single flow. Thus, the second phase estimates the percentages of vCPU required for generating and processing the throughput rate defined in the first phase.

V. EVALUATION

In this section, we discuss two use-cases that illustrate some of the benefits of using eXP-RAN, and how our tool contributes to experimentation with RAN and EC. In all of the experiments, we employed as the reference hardware a server with the following configuration: 2x CPUs Intel Xeon Silver 4114@2.20 GHz and hyper-threading, 20 cores and 40 threads, 128 GB (8x 16 GB DDR4@2666 MHz), and 12 TB in RAID5 (8x HDs of 2 TB, 7200 RPM, SATA 6 Gbps).

A. VIRTUALIZED RAN

In a virtualized RAN (vRAN), the functionalities of a base station (e.g., eNodeB or gNodeB) may be implemented as VNFs where their instantiation place depends on the functional split. A functional split defines which VNFs run in a Remote Unit (RU) and which VNFs run in the Central Unit (CU). However, each pair of VNFs has specific demands for capacity and latency, as summarized in Table 3. As described in [30], industry and academic researchers are focused on three options for real scenarios, namely PDCP-RLC, PHY split I and PHY split IV. The proper positioning of the VNFs may bring benefits, for example, related to interference control and energy efficiency, but the choice involves several base stations with different network characteristics (i.e., capacity and latency). Therefore, there is a complex resource allocation [31], [32] problem that must be solved before properly deploying a vRAN solution.

TABLE 3. Capacity and latency requirements for each split, considering 150 Mbps DL and 50 Mbps UL demand.

Split	One-way latency	DL capacity	UL capacity
RRC-PDCP	30ms	151Mbps	48Mbps
PDCP-RLC	30ms	151Mbps	48Mbps
RLC-MAC	6ms	151Mbps	48Mbps
Split MAC	6ms	151Mbps	49Mbps
MAC-PHY	250μs	152Mbps	49Mbps
PHY split I	250μs	173Mbps	452Mbps
PHY split II	250μs	933Mbps	903Mbps
PHY split III	250μs	1075Mbps	922Mbps
PHY split IIIb	250μs	1966Mbps	1966Mbps
PHY split IV	250μs	2457.6Mbps	2457.6Mbps

Several works [9], [10], [33] have investigated the allocation problem associated with the functional split and the positioning of VNFs in the vRAN context. In order to illustrate eXP-RAN capabilities, we selected a third-party work [9] from which we only had access to the text. From the text we extracted the optimization model, solved it, and expanded its evaluation using eXP-RAN. In the following, initially, we present how eXP-RAN employs its Benchmarker module to identify the proper scale to the values employed in the optimization model and, thus, becomes able to provide coherent performance results. Later, we describe the process of emulating the FluidRAN model, proposed in [9], using eXP-RAN.

1) AUTO TUNING

A common problem in emulating a RAN is the unintentionally generated packet loss by the transmitter, when the aggregated throughput is higher than a certain threshold. Thus, the first step is defining a packet loss that could be considered negligible to the evaluation. For example, we chose 0.1% as the maximum acceptable packet loss, which is provided as an input to the Benchmarker.

According to Table 3, eXP-RAN should be prepared to emulate links with capacity up to near 2.5 Gbps, which works as another input to the Benchmarker. It is hard to achieve this performance with a single vCPU, but the packet loss is unknown a priori and the correct value to scale down the capacity in order to not exceed the maximum acceptable packet loss is also unknown. As illustrated in Fig. 7, the Benchmarker provides this information through several automatic tests. In addition to the reference hardware (Intel Xeon Silver 4114), Fig. 7 also presents the results for another hardware with the following configuration: 1x CPU Xeon E31270 V2@3.50 GHz and hyper-threading, 4 cores and 8 threads, 16 GB (2x 8 GB DDR3@1600 MHz), and 1 TB in RAID2 (2x HDs of 1 TB, 7200 RPM, SATA 6 Gbps). Since the traffic generation is a CPU intensive task, both machines have a similar performance and the higher clock poses a

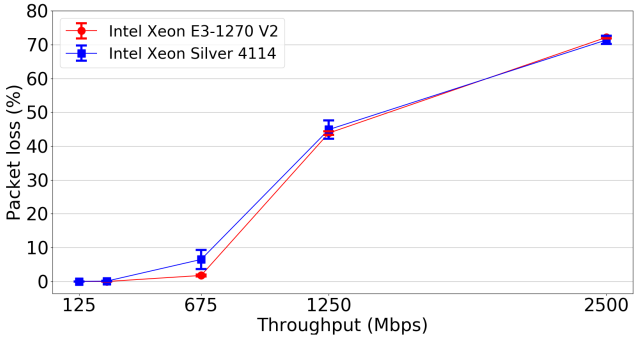

FIGURE 7. Packet loss as a function of the generated throughput.

small advantage to the older hardware that has a more modest general configuration. The emulation becomes more realistic as the processing power of the hardware running the emulator increases. In this case, as the CPU clock speed increases, the packet loss decreases and the aggregated throughput recommended by the Benchmarker for the emulation also increases.

By default, the Benchmarker runs 30 tests of 120 seconds and decreases the throughput from the initial value (e.g., 2.5 Gbps) until the one that satisfies, on average, the packet loss threshold (e.g., 0.1%). In order to speed up the tests, also by default, the Benchmarker decreases the throughput by dividing it by 2 and by 10, alternately, until it reaches the threshold. In some scenarios, it may be useful to choose a scale factor lower than the value recommended by the Benchmarker. For example, the topology to be evaluated may have a number of nodes that demands the reservation of much less than one vCPU per node.

Once the Benchmarker finds the scale down value, i.e., 20:1 (2500/125) in our example, then the tool starts verifying if less than 1 vCPU is also able to transmit the throughput (e.g., 125 Mbps) constrained to the maximum packet loss (e.g., 0.1%). By default, the Benchmarker runs 30 tests of 120 seconds and decreases the amount of vCPU by a step of 10% while the average packet does not exceed the maximum threshold (e.g., 0.1%). In our reference hardware, we found that 0.5 vCPU was able to transmit 125 Mbps in a sustainable way. This evaluation focused on the demand of a transmitter node, but it is necessary to identify the demand for the receiver, which has initially a dedicated vCPU. In general, the receiver demands fewer resources than the transmitter, so the Benchmarker can start from the transmitter configuration (e.g., 0.5 vCPU). Again, by default, the Benchmarker runs 30 tests of 120 seconds and decreases the amount of vCPU by a step of 10% while the average packet loss does not exceed the maximum threshold (e.g., 0.1%). We found that 0.2 vCPU was able to receive 125 Mbps in a sustainable way.

2) MODEL EMULATION

As previously described, in [9], the authors present the FluidRAN Design Problem (FRD) that has as objective function the minimization of the network operator costs while satisfying the users' demand. The costs are related to data transfer and the computing resources necessary for the VNFs of the vRAN, in order to satisfy the throughput demanded by the users. The FRD problem considers a simplified version of the functional splits in which there are only four possible functions and three splits. The splits are:

1) CU: **f3** – RU: f2, f1, **f0** (PDCP-RLC).
2) CU: **f3**, f2 – RU: f1, **f0** (MAC-PHY).
3) CU: **f3**, f2, f1 – RU: **f0** (PHY).

In [9], the authors also explore how FluidRAN can be adapted to represent MEC services deployed in the network. This is achieved by simply adding a new VNF **f4** to the service chain. The capacity and the delay of this VNF can be specified for each service it represents. The main issue is to evaluate the impact of MEC services in the vRAN. The authors identify that FRD is NP-hard to solve and so they employed Benders' decomposition method to approach the problem. Analyzing the robustness of a solution generated by the FluidRAN model or making a sensitivity analysis of the model are very time-consuming. In this context, eXP-RAN becomes an useful tool, since it makes it possible to test the model varying different parameters of interest and collecting several metrics.

Using the Model Adapter module, described in Section III-B, we created six FluidRAN emulated scenarios for evaluation - labelled from *A* to *F*. All scenarios are based on the topology shown in Fig. 8. The blue node is the Central Unit (CU), the (eleven) black nodes represent the Remote Units (RUs) / Base Stations (BSs), and the red nodes are the forwarding nodes. Based on the optimization model output, vRAN consists of different functional splits of the BSs indicated in the Fig. 8. Part of the VNFs are run in each RU and the rest are run in the CU. Since each pair RU-CU may have a different split, we labeled each one as a different vRAN service. There are eleven vRAN services because there are eleven pairs CU-RU, i.e., eleven BSs, which are independently defined by the optimization model. However, many vRAN services have the same functional split because the FluidRAN model compacts Table 3 to only three options, as described above.

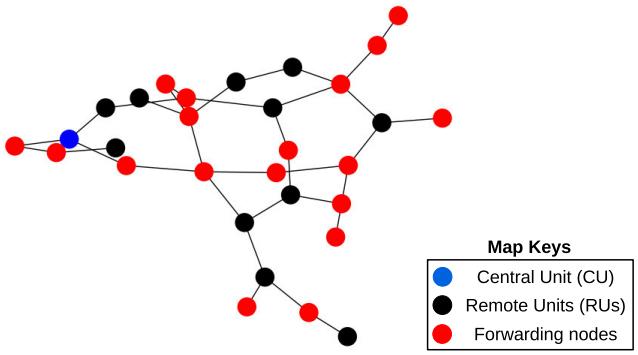

FIGURE 8. Emulated topology randomly generated.

Each scenario emulates the same eleven vRAN services, while scenarios from *B* to *F* also emulate one additional MEC service deployed in the CU. In scenario *A*, there are only the eleven vRAN services that were optimally deployed. This scenario confirmed the expected performance derived from the Benchmarker, i.e., the packet loss was neglected in all vRAN services by applying the proper scale factor. Within the scenarios from *B* to *F*, we increased by one the number of shared links between the MEC and the vRAN services, i.e., scenario *B* has one shared link, scenario *C* has two shared links, and so on. Table 4 compares each FluidRAN scenario described above.

TABLE 4. Emulated FluidRAN scenarios comparison.

	Shared links	MEC service emulation
Scenario A	0 links	No
Scenario B	1 link	Yes
Scenario C	2 links	Yes
Scenario D	3 links	Yes
Scenario E	4 links	Yes
Scenario F	5 links	Yes

Scenarios from *B* to *F* do not necessarily have the same shared links in common, since the randomly generated RAN topology and the flows distribution of the vRAN services do not favor this sort of comparison. These scenarios are also differentiated from one another by the place where the MEC service is consumed, i.e., where the mobile users demanding the MEC service are positioned. The positioning of the users affects the MEC service routing and so does the amount of links that are shared between the MEC and the vRAN services.

For the sake of clarity, we randomly chose three vRAN services for analyzing, in addition to the MEC service. The emulated MEC service provides to each of its users the equivalent to 20 Mbps in the real-world (considering the scale factor of 20:1, recommended by the Benchmarker). This could correspond to a 4K-video streaming or an early stage VR service [34]. Fig. 9 presents the results when the MEC service is accessed by 10 users. 'vRAN 1' represents the MAC-PHY split, 'vRAN 2' represents the PDCP-RLC split and 'Split vRAN' corresponds to a vRAN service that had its flow split as a consequence of the optimal solution. For every scenario, we performed 30 tests of 20 minutes each. The mean values are presented with confidence interval bounds at a confidence level of 95%.

From Fig. 9, we can observe that the vRAN services are noticeably impacted only in scenario *D*. In order to improve its performance, this result suggests that the focus should be on the shared links of this scenario. The solution could involve rerouting some flows of the vRAN services or changing some functional splits. However, the low and localized impact suggests that a simple approach, e.g., a basic heuristic or a greedy solution could solve the problem. Probably, it would not be necessary to recompute the FRD problem again.

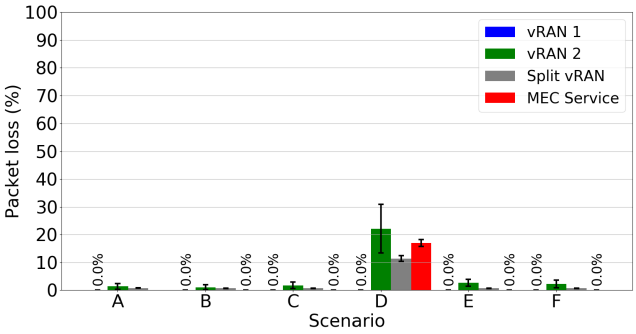

FIGURE 9. Impact of the MEC service consumed by 10 users.

When the number of MEC users increases significantly, it is expected that the impact tends to be notably higher in the vRAN services. However, the RAN topology and the flow distribution of the vRAN services make it difficult to predict where and the extent of this impact. Again, eXP-RAN provides valuable information through an easy manner. Fig. 10 presents the results when the MEC service is accessed by 100 users, while the remaining configuration is kept identical, as with the previous tests. All the scenarios with the MEC service emulation exhibit severe packet loss and all the vRAN services are affected in at least one scenario, but it is not trivial the way that the MEC service impacted the vRAN services. For example, the 'vRAN 2' service is not the most affected service as previously, while the scenarios *B* and *C* present the highest packet loss. Differently from the result of the previous tests, under this load, it would be useful to recompute the FRD problem again, including the MEC service as a new VNF.

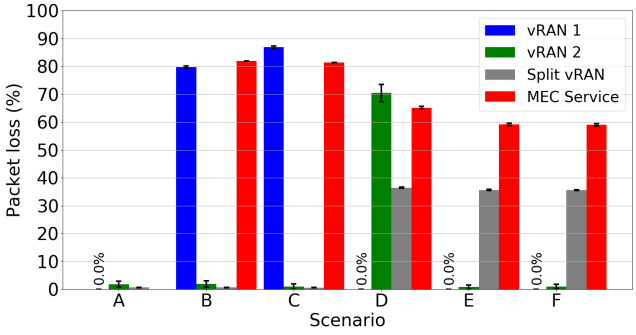

FIGURE 10. Impact of the MEC service consumed by 100 users.

In this use-case, we evaluated the impact of a MEC service introduced after the optimal allocation of the vRAN services. The evaluation was focused on the packet loss generated by the overload in the RAN links. However, the evaluation could be easily expanded to include multiple services, users changing from one BS to another, reconfiguration of the nodes in terms of computing capacity, reconfiguration of the links in terms of capacity or delay, among several other changes in the experimentation environment. Additionally, this use-case would be hard to represent in tools such as ns-3 and Mininet. The ns-3 is focused on discrete event

simulation and cannot represent computing resources, while Mininet service representation depends on the user creating and composing her application inside the tool. Furthermore, VLSP/Slice Controller would depend on the user to properly adjust the application/service flows since this software tool does not have an auto-tune feature.

B. VIDEO SERVICE DEPLOYMENT

According to Cisco [35], by 2023, more than 80% of all IP traffic will be video [36], with a large part of this traffic generated by mobile devices. The use of caching at the edge of the Internet is an old approach, introduced by Content Delivery Networks (CDNs) at the end of the last century, i.e., more than 20 years ago. To date, companies like Akamai, Cloudflare, Limelight, and several others offer content distribution services on a global scale. Given the relevance of this market and the impact on communication networks, several telecommunications companies have also started offering content distribution services, such as AT&T, China Telecom, and Telefonica. In addition, companies like Netflix and Google have created their own unique CDNs, known as Netflix Open Connect and Google Global Cache, respectively. The introduction of ETSI MEC contributes to standardizing the offer of caching services in a much simpler, faster, and flexible way through VNFs. The benefits of MEC (for caching) can be considered clear for users, service providers, and infrastructure operators. However, the effective advantage of Network Slicing may not be so clear, mainly for users and service providers who will be charged in a different manner in order to gain from it. Network Slicing promises to offer virtualized infrastructures to customers (tenants), e.g., a video service provider, that will be able to easily deploy their services. Additionally, each customer will be able to manage her virtualized infrastructure as needed, for example, configuring the virtual (computing and network) resources or allocating to her users a certain amount of network capacity. In this use-case, we illustrate how eXP-RAN can emulate two network slices, from different tenants with different QoS demands, and analyze how they are affected as workload increases. Scenarios with two network slices are commonly used in the literature to illustrate the benefits of using network slicing [37], [38].

We considered two slices. One, denoted by 'Premium' slice, is hired by the video service provider A (or tenant A), while the other, denoted 'Best-Effort' slice, is provided to the video service provider B (or tenant B). Tenant A offers a video service to clients willing to pay a higher fee to receive their video content with assured quality, while tenant B provides video services to clients that paid a lower fee to receive their video content and can have their video quality affected by other clients' demand on the network. Thus, tenant A provides a premium video service, while tenant B offers a best-effort video service.

We assume that both tenants have leased the network slices from the same infrastructure provider of the previous use-case, but covering only the subset of resources illustrated in Fig. 11. Thus, both network slices share the same physical infrastructure shown in Fig. 11, which involves: a MEC host, three base stations, and three forwarding nodes. This topology could be built by modifying the Infrastructure Description file generated previously and providing it as input to the Infrastructure Manager module. The Service Description file must be created for each tenant and then provided to the Service Manager module.

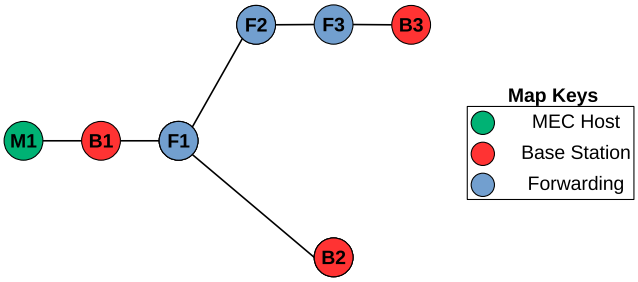

FIGURE 11. Topology for the emulation of the video service.

In the 'Premium' slice, the video service deployed is a containerized version of the FFmpeg software [39]. The server side of the service is deployed in the MEC host, while the clients run in the base station B2. The service deployed in the 'Best-Effort' slice follows a similar approach with the containerized version of the FFmpeg running in the MEC host and the clients running in the base station B3. In both slices, the clients receive 1080p video at a Constant Bit Rate (CBR). For sake of simplicity, we consider that each tenant has five clients. In each slice, one client is started every 30 seconds until all the five clients started.

Fig. 12 presents the result of this experiment. The x-axis indicates the number of the time window, where each time window has a length of 10 seconds. Thus 1 is the first time window of 10 seconds, 5 is the fifth time window of 10 seconds, and so one. Each point in the y-axis represents the quality of the service received by each client during the 10-second time window. The QoS is the ratio between the average expected throughput and the average provided throughput to the video flow. In Fig. 12 this ratio is computed every 10 seconds. In the figures, only one client of the "Premium" slice is illustrated because all of them show the same performance.

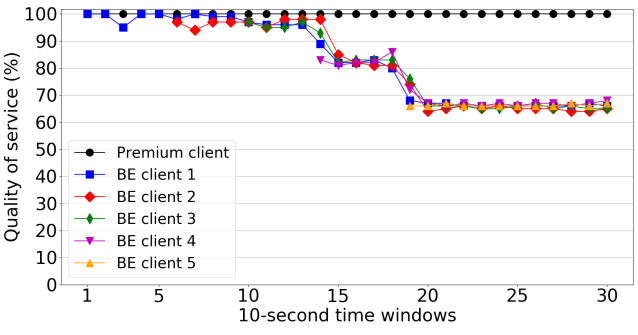

FIGURE 12. Video stream results.

We can observe that, as the workload increases, the QoS provided to the client of the "Premium" slice does not change. On the contrary, in the "Best-Effort" slice, as the workload increases, the QoS provided to each client decreases. Thus, each network slice meets their requirements.

Finally, it is worth to mention that this use-case would be hard to represent in tools such as Mininet, Fogbed, EdgeCloudSim, and ns-3 due to lack or poor slice abstraction support.

VI. CONCLUSION

In this paper, we presented eXP-RAN – an emulator for gaining experience with Radio Access Network, Edge Computing, and Slicing. eXP-RAN is an open-source tool used to emulate RANs and EC infrastructures with the ability to slice the network resources and evaluate the performance of running services. eXP-RAN is distinguished from other tools by its ability to abstract and represent RANs, virtualized services and network slicing.

We described how the application empowers researchers to run more realistic evaluation studies to support the emerging telecommunication applications. Two use-cases were analyzed, one focusing on a vRAN design and the other one on a slice running a video streaming service.

As future work, we plan 1) to add the radio part of the RAN to the tool, i.e., radio resource allocation, loss propagation, shadowing, fading, etc.; 2) to address the labeling problem in the topology generator; 3) to add new use-cases, exploring other aspects such as large number of network slices and combination of network slicing and placement of vRAN network functions; 4) to improve the existing support for automating the service deployment and the system monitoring. Creating a fully service agnostic approach is challenging, but there is a lot of effort in literature and tools such as Open Networking Automation Platform (ONAP) and Open Source Mano (OSM) [40] are already following this direction. Among the improvements in system monitoring, there are the integration with analytics and interactive visualization tools such as Grafana.

ACKNOWLEDGMENT

The authors thank Felipe F. Fonseca for his contributions to the source code of eXP-RAN.

REFERENCES

[1] A. Osseiran, F. Boccardi, V. Braun, K. Kusume, P. Marsch, M. Maternia, O. Queseth, M. Schellmann, H. Schotten, H. Taoka, H. Tullberg, M. A. Uusitalo, B. Timus, and M. Fallgren, "Scenarios for 5G mobile and wireless communications: The vision of the METIS project," *IEEE Commun. Mag.*, vol. 52, no. 5, pp. 26–35, May 2014.

[2] K. Vieira Cardoso, C. Bonato Both, L. Rene Prade, C. J. A. Macedo, and V. Hugo L. Lopes, "A softwarized perspective of the 5G networks," 2020, *arXiv:2006.10409*. [Online]. Available: http://arxiv.org/abs/2006.10409

[3] European Telecommunications Standards Institute. (2015). *Network Functions Virtualisation (NFV); Infrastructure Overview*. [Online]. Available: https://www.etsi.org/

[4] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G," Eur. Telecommun. Standards Inst., France, Tech. Rep., 2015.

[5] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5G: Survey and challenges," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 94–100, May 2017.

[6] C. Sonmez, A. Ozgovde, and C. Ersoy, "EdgeCloudSim: An environment for performance evaluation of edge computing systems," *Trans. Emerg. Telecommun. Technol.*, vol. 29, no. 11, p. e3493, Nov. 2018.

[7] 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.

[8] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca, and J. Folgueira, "Network slicing for 5G with SDN/NFV: Concepts, architectures, and challenges," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 80–87, May 2017.

[9] A. Garcia-Saavedra, X. Costa-Perez, D. J. Leith, and G. Iosifidis, "FluidRAN: Optimized vRAN/MEC orchestration," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2018, pp. 2366–2374.

[10] A. Garcia-Saavedra, J. X. Salvat, X. Li, and X. Costa-Perez, "WizHaul: On the centralization degree of cloud RAN next generation fronthaul," *IEEE Trans. Mobile Comput.*, vol. 17, no. 10, pp. 2452–2466, Oct. 2018.

[11] L. Mamatas, S. Clayman, and A. Galis, "A service-aware virtualized software-defined infrastructure," *IEEE Commun. Mag.*, vol. 53, no. 4, pp. 166–174, Apr. 2015.

[12] S. Clayman, L. Mamatas, and A. Galis, "Experimenting with control operations in software-defined infrastructures," in *Proc. IEEE NetSoft Conf. Workshops (NetSoft)*, Jun. 2016, pp. 390–396.

[13] S. Clayman, F. Tusa, and A. Galis, "Extending slices into data centers: The VIM on-demand model," in *Proc. 9th Int. Conf. Netw. Future (NOF)*, Nov. 2018, pp. 31–38.

[14] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proc. 9th ACM SIGCOMM Workshop Hot Topics Netw.*, Oct. 2010, p. 19.

[15] R. Sherwood, G. Gibb, K. kiong Yap, M. Casado, N. Mckeown, and G. Parulkar, "Flowvisor: A network virtualization layer," OpenFlow Switch Consortium, Tech. Rep. OPENFLOW-TR-2009-1, 2009, p. 132, vol. 1.

[16] A. Al-Shabibi, M. De Leenheer, M. Gerola, A. Koshibe, G. Parulkar, E. Salvadori, and B. Snow, "OpenVirteX: Make your virtual SDNs programmable," in *Proc. 3rd Workshop Hot Topics Softw. Defined Netw.*, 2014, pp. 25–30.

[17] A. Coutinho, F. Greve, C. Prazeres, and J. Cardoso, "Fogbed: A rapid-prototyping emulation environment for fog computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–7.

[18] M. Peuster, H. Karl, and S. van Rossem, "MeDICINE: Rapid prototyping of production-ready network services in multi-PoP environments," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN)*, Nov. 2016, pp. 148–153.

[19] P. Wette, M. Draxler, and A. Schwabe, "MaxiNet: Distributed emulation of software-defined networks," in *Proc. IFIP Netw. Conf.*, Jun. 2014, pp. 1–9.

[20] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw., Pract. Exper.*, vol. 41, no. 1, pp. 23–50, Jan. 2011.

[21] G. F. Riley and T. R. Henderson, *The NS-3 Network Simulator*. Berlin, Germany: Springer 2010, pp. 15–34.

[22] P. H. A. Rezende and E. R. M. Madeira, "An adaptive network slicing for LTE radio access networks," in *Proc. Wireless Days (WD)*, Apr. 2018, pp. 68–73.

[23] M. Richart, J. Baliosian, J. Serrati, J.-L. Gorricho, R. Aguero, and N. Agoulmine, "Resource allocation for network slicing in WiFi access points," in *Proc. 13th Int. Conf. Netw. Service Manage. (CNSM)*, Nov. 2017, pp. 1–4.

[24] M. Miozzo, N. Bartzoudis, M. Requena, O. Font-Bach, P. Harbanau, D. Lopez-Bueno, M. Payaro, and J. Mangues, "SDR and NFV extensions in the ns-3 LTE module for 5G rapid prototyping," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2018, pp. 1–6.

[25] *Emulation Overview*. Accessed: May 22, 2020. [Online]. Available: https://www.nsnam.org/docs/release/ 3.11/models/html/emulation-overview.html

[26] European Telecommunications Standards Institute. (2019). *Multi-access Edge Computing (MEC); Terminology*. [Online]. Available: https://www.etsi.org/

[27] B. M. Waxman, "Routing of multipoint connections," *IEEE J. Sel. Areas Commun.*, vol. 6, no. 9, pp. 1617–1622, Dec. 1988.

[28] J. Lessmann, "Resource optimization in realistic mobile backhaul networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2015, pp. 3861–3866.
[29] *Open Virtual Switch*. Accessed: May 22, 2020. [Online]. Available: https://www.openvswitch.org/
[30] S. Shew, "Transport network support of IMT-2020/5G," ITU-T, Geneva, Switzerland, Tech. Rep., I. GSTR-TN5G, Feb. 2018.
[31] M. Peng and K. Zhang, "Recent advances in fog radio access networks: Performance analysis and radio resource allocation," *IEEE Access*, vol. 4, pp. 5003–5009, 2016.
[32] G. Sun, Z. T. Gebrekidan, G. O. Boateng, D. Ayepah-Mensah, and W. Jiang, "Dynamic reservation and deep reinforcement learning based autonomous resource slicing for virtualized radio access networks," *IEEE Access*, vol. 7, pp. 45758–45772, 2019.
[33] F. Fonseca, S. Correa, and K. Cardoso, "Optimizing allocation and positioning in a disaggregated radio access network," in *Anais Principais do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. Porto Alegre, Brazil: SBC, 2019, pp. 791–804.
[34] S. Mangiante, G. Klas, A. Navon, Z. GuanHua, J. Ran, and M. D. Silva, "VR is on the edge: How to deliver 360° videos in mobile networks," in *Proc. Workshop Virtual Reality Augmented Reality Netw.*, 2017, pp. 30–35.
[35] C. VNI. (2020). *Cisco Visual Networking Index: Forecast and Trends, 2017–2022*. Accessed: May 22, 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/ annual-internet-report/white-paper-c11-741490.pdf
[36] W. Shi, Y. Sun, and J. Pan, "Continuous prediction for quality of experience in wireless video streaming," *IEEE Access*, vol. 7, pp. 70343–70354, 2019.
[37] Q. Wang, J. Alcaraz-Calero, R. Ricart-Sanchez, M. B. Weiss, A. Gavras, N. Nikaein, and X. Vasilakos, "Enable advanced QoS-aware network slicing in 5G networks for slice-based media use cases," *IEEE Trans. Broadcast.*, vol. 65, no. 2, pp. 444–453, Jun. 2019.
[38] S. Marinova, V. Rakovic, D. Denkovski, T. Lin, V. Atanasovski, H. Bannazadeh, L. Gavrilovska, and A. Leon-Garcia, "End-to-End network slicing for flash crowds," *IEEE Commun. Mag.*, vol. 58, no. 4, pp. 31–37, Apr. 2020.
[39] *FFmpeg*. Accessed: May 22, 2020. [Online]. Available: https://www.ffmpeg.org/
[40] L. 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 Afr. Conf. (WAC)*, Aug. 2019, pp. 1–7.

JOÃO PAULO ESPER has been a Computer Science Researcher and a member with the Laboratory Computer Networks and Distributed Systems, Federal University of Goiás (UFG), Brazil, since 2017. He has participated in research projects funded by FAPEG and RNP, including Novel Enablers for Cloud Slicing (NECOS) from Joint Calls BR-EU. His research interests include wireless networks, software-defined networks, and virtualization and resource allocation.

ABDALLAH S. ABDALLAH (Member, IEEE) received the Ph.D. degree from the Bradley Department of Electrical and Computer Engineering, Virginia Tech, in 2016. He has experience with the Telecommunication Industry, such as Broadcom Company and Intel Company. He has been a Faculty Member with the Penn State Erie-The Behrend College since 2017, where he is currently an Assistant Professor of computer engineering. He believes in how critical undergraduate research for bridging between teaching and research, so he utilizes related pedagogical techniques in the classrooms. His research interests include wireless networks, image processing, video streaming over HTTP, applied machine learning for resource allocation in wireless networks, and computer vision for the Internet of Things (IoT) devices. He serves as a Regular Reviewer for *IEEE Network Magazine* and IEEE Access journal.

STUART CLAYMAN (Member, IEEE) received the Ph.D. degree in computer science from University College London, in 1994. He was a Research Lecturer with Kingston University London and University College London. He is currently a Principal Research Fellow with the EEE Department, University College London. He has coauthored over 60 conference papers and journal articles. He has extensive experience in commercial arena undertaking architecture and development for software engineering, distributed systems, and networking systems. His research interests include software engineering and programming paradigms, distributed systems, virtualised compute and network systems, network and systems management, sensor systems and smart city platforms, and artificial intelligence systems.

WALDIR MOREIRA received the bachelor's and master's degrees in computer science and the Ph.D. degree in telecommunications from the Universidade do Minho, Porto de Aveiro, in 2005, 2008, and 2014, respectively. He was with GERCOM, UFPA, Brazil, in 2006, INESC, Porto, Portugal, in 2009, COPELABS, Portugal, in 2010, and Regional Catalão, UFG, Brazil, in 2015, where he was also involved in different national and European projects as a Team Member or a Coordinator. He is currently a Senior Scientist with Fraunhofer Portugal Research Center for Assistive Information and Communication Solutions (AICOS) involved in information and communication technologies for development (ICT4D). His publications and current research interests include wireless adhoc, mesh, social-aware, cooperative, opportunistic, information-centric, and software-defined networking and routing.

ANTONIO OLIVEIRA-JR (Member, IEEE) received the B.S. degree in computing and the M.S. degree in electrical engineering from the Federal University of Uberlândia, Brazil, in 2000 and 2003, respectively, and the Ph.D. degree in computer science (Ph.D. Program in computer science) from Minho, Aveiro, and Porto Universities (MAP-i), Portugal, in 2014. Since 2004, he has been a Professor with the Institute of Informatics (INF), Federal University of Goiás (UFG). From 2007 to 2008, he was a Researcher with the Telecommunications Institute (IT-Porto). From 2008 to 2010, he was a Researcher with INESC-TEC, Porto. From 2011 to 2014, he was also a Researcher with SITILabs and COPELABS (Association for the Research and Development of Cognition and People-Centric Computing), Lisbon. He was an Invited Senior Scientist with Fraunhofer Portugal AICOS (he spent his sabbatical in 2019). He is currently an Adjunct Coordinator with GT-SOFTWAY4IoT funded by RNP and a member with the Laboratory Computer Networks and Distributed Systems. His main research interests include networking areas, such as wireless networks, the Internet of Things (IoT), softwarization, virtualization, intelligent networking (IA/ML), 5G/6G, and NGN.

SAND LUZ CORREA received the degree in computer science from the Universidade Federal de Goiás (UFG), in 1994, the M.Sc. degree in computer sciences from the University of Campinas (UNICAMP), in 1997, and the Ph.D. degree in informatics from the Pontifical Catholic University of Rio de Janeiro (PUC-Rio), in 2011. She has been a Professor and a Researcher with the Institute of Informatics, UFG, since 2010, where she is currently an Associate Professor. In 2015, she spent her sabbatical with the Inria Saclay Research Centre, France, in 2020, and Virginia Tech, USA. She has participated with some international research projects, including two from joint calls BR-EU. Her research interests include cloud computing, SDN, data plane programmability, sensor systems, and smart city platforms.

KLEBER VIEIRA CARDOSO (Member, IEEE) 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 COPPE, Universidade Federal do Rio de Janeiro, in 2002 and 2009, respectively. He has been a Professor and a Researcher with the Institute of Informatics, UFG, since 2009. In 2015, he spent his sabbatical with the Inria Saclay Research Centre, France, in 2020, and Virginia Tech, USA. He is currently an Associate Professor with the Institute of Informatics, UFG. He has participated with some international research projects, including two from joint calls BR-EU. He has coordinated several national-sponsored research and development projects. His research interests include wireless networks, software-defined networks, virtualization, resource allocation, and performance evaluation.