An experimentation framework for mobile multi-tenant 5G networks integrated with CORE network emulator

Serrano Mamolar, Ana; Pervez, Zeeshan; Alcaraz Calero, Jose M.

Published in:
Proceedings of the 2018 IEEE/ACM 22st Interna:onal Symposium on Distributed Simula:on and Real Time Applica:ons (DS-RT)

DOI:
10.1109/DISTRA.2018.8600932

Published: 01/01/2018

Document Version
Peer reviewed version

An experimentation framework for mobile multi-tenant 5G networks integrated with CORE network emulator

Serrano Mamolar, Ana; Pervez, Zeeshan; Alcaraz Calero, Jose M.

Published in:
Proceedings of the 2018 IEEE/ACM 22st Interna:onal Symposium on Distributed Simula:on and Real Time Applica:ons (DS-RT)

DOI:
10.1109/DISTRA.2018.8600932

Published: 01/01/2018

Document Version
Peer reviewed version

An Experimentation Framework for Mobile Multi-Tenant 5G Networks integrated with CORE Network Emulator

Ana Serrano Mamolar
School of Engineering and Computing
University of the West of Scotland
ana.serrano@uws.ac.uk

Zeeshan Pervez
School of Engineering and Computing
University of the West of Scotland
zeeshan.pervez@uws.ac.uk

Jose M. Alcaraz Calero
School of Engineering and Computing
University of the West of Scotland
Jose.Alcaraz-Calero@uws.ac.uk

***Abstract*—Currently, there is a lack of tools for real validation of 5G scenarios. The increasing traffic demand of 5G networks is pushing network operators to find new cost-efficient solutions. The selected solution is a multi-tenancy approach that, together with user mobility will impose some architectural changes. This approach increases service dynamism making it necessary to have tools that provide these new capabilities to be able to validate each development. This work presents a novel experimentation framework for the emulation of 5G scenarios providing them with real-time user mobility and multi-tenancy. The functionality of this novel framework has been validated through different experiments.**

***Index Terms*—Network Emulation, 5G networks, Multi-tenant, Mobile network**

I. Introduction

The arrival of 5G networks is expected for 2020 when fully standardised 5G networks will deliver unprecedented levels of connectivity and hit the market. 5G will support multi-tenancy models, enabling operators to share infrastructures, reducing costs and energy consumption. This new paradigm forces 5G to ensure high flexibility regarding the topology and to be designed as sustainable and scalable technology. This architectural flexibility will also allow operators to enrich their portfolio with new services as platform offered as a service, network functions and infrastructure. Within this context, the network shall rapidly adapt to this new wide range of requirements across such a versatile architecture [1]. In order to quantify how new technical solutions would affect the quality of experience (QoE) of end-users, or how the 5G system would perform in a use case or any other aspects related to the 5G data plane, specific evaluation tests are needed. Some network operators are already applying changes in their infrastructures to be ready to provide 5G services to their users [2] [3]. Some of them already have a real testbed for running different experiments. However, there are more actors implied in the development of 5G networks and services other than infrastructure providers. These actors, such as software developers, integrators, vertical businesses, need to test their solutions in real 5G infrastructures, in order to check the feasibility of their solutions. The lack of tools to try new 5G-compliant services may hamper their entry into the 5G market. This work provides an experimentation framework for testing the data plane of 5G scenarios, providing two important capabilities of a 5G infrastructure: mobility and multi-tenancy.

978-1-5386-5048-6/18/$31.00 ©2018 IEEE

Since 5G infrastructures will be composed of tenants/operators that will share computational resources owned by the infrastructure provider, it is essential to guarantee that the traffic of each tenant is entirely isolated. Another requirement for 5G infrastructure is to provide users with complete mobility across antennas. Fig. 1 depicts a possible deployment of a 5G scenario where there are physical resources that are shared through a virtual layer; where the main architectural elements of the 5G architecture are also depicted. An interested reader may refer to Kim et. al. [4] for a comprehensive description of different architectural elements envisioned in the 5G architecture. In the figure, there are mobile users labelled as User Equipment (UE). A GPRS (General Packet Radio Service) Tunnelling Protocol (GTP) is used in order to assure user mobility and identify a UE connected to an antenna that belongs to one operator of the 5G network. Furthermore, in order to isolate each tenant/operator traffic, it is used VxLAN/GRE encapsulation. When looking for tools to test the capabilities of the network and its services it is essential to include the implementation of these two encapsulations protocols as a requirement, to make sure the services are 5G-compliant. This work represents, as to the best of our knowledge, the first emulator that provides these 5G capabilities.

The rest of this paper is organised as follows. Section II provides a brief literature review of previous works with tools for testing 5G scenarios. Section III provides details of the proposed solution design and implementation. In addition, Section IV shows some examples of the validation process. Finally, the conclusion and future work are provided in Section V.

II. Related works for 5G network research tools

Some open source tools allow networking researchers to validate their network designs. Most of them are not emulation but simulation tools. Nonetheless, different efforts have been

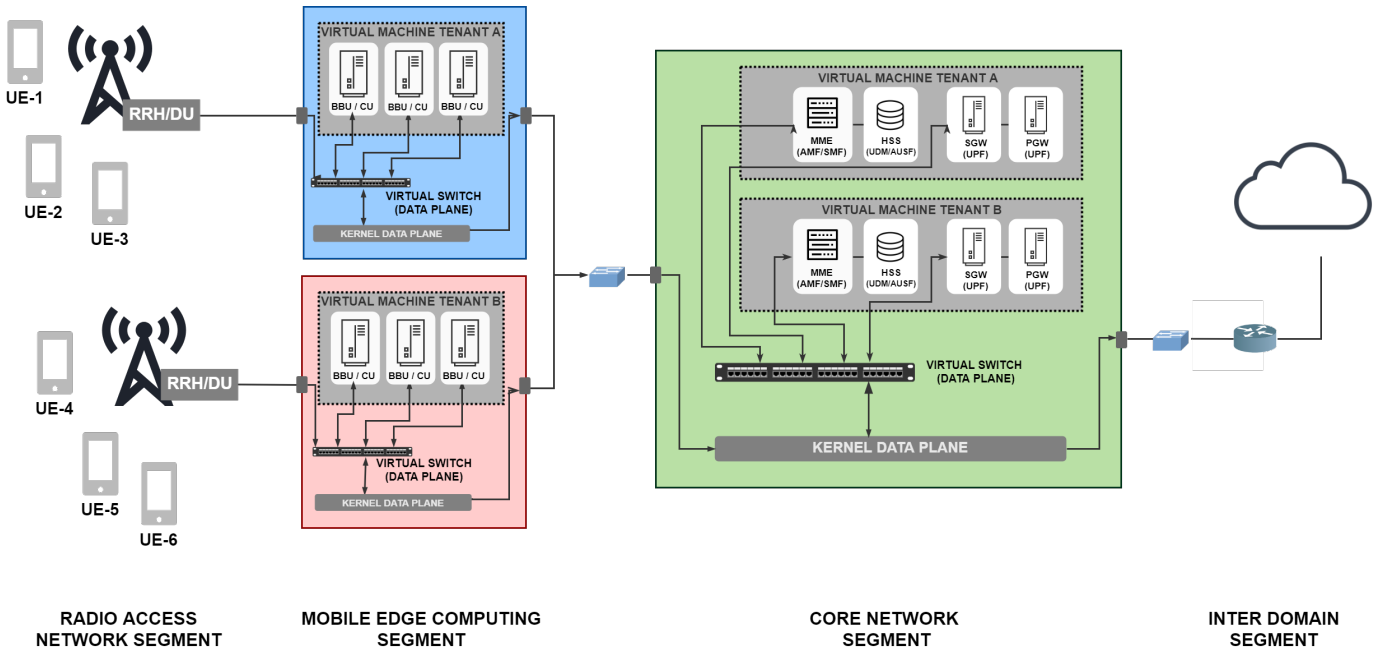

Fig. 1. Brief scheme of a 5G infrastructure.

made in the area of integrating simulation and emulation of 5G networks.

OpenAirInterface [5] is an open source initiative for LTE/5G network emulation. It provides an implementation of network nodes such as, eNodeB, UE, Evolved Packet Core (EPC) and Remote Radio Heads (RRH), that run on general purpose computing platforms. It allows to build LTE and 5G networks on personal computers and connect them to real User Equipment (UE) such a smartphone, or to software-based ones. ns-3 [6] is an open source network simulator written in C++ and python for developing new protocols and analysing complex architectures with support for LTE. Mezzavilla et. al. presented [7] a module for ns-3 to provide millimetre wave (mmW) models. mmW is considered as a key technology in 5G [8]. Omnet++ [9] is a component base C++ simulator for building network simulations. It can be used under Academic Public Licence. It has a framework approach, providing the basic machinery and tools to write simulations, without providing specific simulation components for computer networks. Different simulation models such as the Mobility Framework [10] or the INET framework [11] can be used. The INET framework includes a mmW module and a module to simulate LTE called SimuLTE [12] that can be used to simulate 5G networks. Matlab provides a simulation environment to test 5G algorithms for rapid prototyping and deployment. To simulate a 5G network with Matlab, many modules are needed, such as Simulink, Antenna toolbox or LTE system Toolbox. Algorithms can be tested with Software Defined Radio platform and FPGA hardware support. A discrete event simulator for R was presented in [13]. In [14], the authors presented the application of simmer for the design and analysis of three different 5G scenarios. The TRIANGLE EU project is developing a framework emulator [15] to help mobile app developers and device vendors to test their solutions utilising existing FIRE testbeds that combine proper realistic hardware and software components.

The Common Open Research Emulator (CORE) [16] is a major open-source tool that is widely used for both research and military purposes. For the best of our knowledge, there exists no framework based on CORE that provides the aforementioned 5G capabilities. The presented work offers a framework that combines CORE with an open Gateway GPRS Support Node (GGSN) and an open multilayer virtual switch.

Literature still lacks contributions in providing real emulation testbeds. Most of the tools found are simulators, which is not a good approach for measuring the performance of the network since they are not wholly aware of real hardware capabilities and their limitations [17]. On the other hand, most of the tools depend on built-in modules that manage the network at the link level. None of them provides a framework to manage multi-tenancy in a 5G network and explore multi-operator architectures. The main motivation of this research work is to provide a real emulation framework together with 5G network capabilities of mobility and multi-tenancy.

III. Framework Design

The framework proposed is based on CORE emulator [18], a tool for building virtual infrastructures. In contrast with simulators, CORE as an emulator builds a representation of a real x86-64 PC architecture using Linux containers and then creating virtual infrastructures by connected such containers within a network topology in real-time. This emulator allows developers to run real applications and protocols taking

advantage of virtualisation provided by the Linux network namespaces.

The CORE has a backend (core-daemon) that manages the emulated sessions and build emulated networks using kernel virtualisation. It uses bridging and packet manipulation for virtual networks. This backend is controlled via the graphical interface (core-gui). In order to allow running different services on different physical machines, CORE provides its socket-based API. To connect to a Linux network namespace of CORE and run commands in that namespace, CORE provides the vcmd program. This program is also used for setting up a node and running processes within it. In fact, this program is also used in this work to run scripts in particular nodes of the infrastructure to provide 5G capabilities.

A. Integration with CORE emulator

The main components of the CORE architecture are shown in Fig. 2, together with the extension presented in this work to provide 5G capabilities. The presented work include an installer of the framework. The main components installed by the installer, with their corresponding dependencies are: the CORE emulator [1], OpenVSwitch [2], and SGSN [3]. The CORE emulation provides the emulation foundations. OpenVSwitch is used to provide VXLAN/GRE encapsulation to allow multi-tenant isolation. SGSN is used to provide GTP and mobility support. Once the framework is installed, the user can start using it by deploying one of the templates scenarios. The interaction of the user through the framework is made by three entry points:

- The definition of a network scenario, following one of the templates provided in *.imn* format, the format that CORE emulator uses to define scenarios. These templates describe different network topologies which match several network business schemes. Thus, some templates consider different network domains and a different number of tenants per Infrastructure-as-a-Service (IaaS) compute, management planes for each tenant and/or for the IaaS itself. These scenarios defined in the templates only provide the network topology.
- The definition of the configuration file that describes important issues related to the deployment and the business logic of the network. For instance, in this configuration file, the user can indicate if it is needed to mirror the traffic in the 5G core segment for inspection purposes, either in the infrastructure provider, or the operators/tenants of such infrastructure, or even both of them, depending on the selected template. In that case, the mirror will be enabled. With this mirror enabled, a Network Intrusion Detection System (NIDS) could be deployed by the user in the service node included in the template, which is the one that will receive the mirrored information from the mirror node.
- The normal interaction with the CORE emulator through its core-gui or via the vcmd program to run any software in a given node, for example, the installation and configuration of the NIDS.

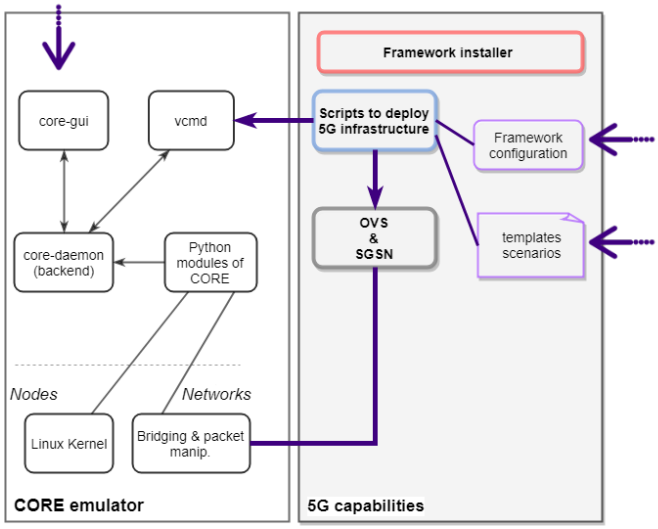

Fig. 2. Architecture of CORE with the extension for 5G capabilities.

B. Emulator capabilities

The framework together with the installation is also provided with several templates as a guidance of use. Every template represents a real 5G infrastructure, and any of them can be used by the user to test their implementations instead of implementing their own 5G scenario from scratch. An example of a template is shown in Fig. 3.

For any scenario, which follows the scheme of a template, this framework detects the significant roles of the 5G infrastructure to implement the necessary modifications to achieve a real 5G scenario. The most significant modifications are the building and configuration of encapsulation protocols in order to provide connectivity to the network. A brief diagram of those bridges is shown in Fig. 4 for a scenario where a compute is acting as IaaS, and it is allocating a virtual operator. In this figure, two users are represented as UE-1 and UE-2, an IaaS compute in the edge segment and another in the core segment of the 5G network, and a virtual operator allocated in Virtual Machines (VM) of these computes. In this case, the IaaS has different interfaces for the data path and one for control and management. In the specific case of the compute in the edge, interfaces 4 and 5 are the ones connected with the access segment, 0 is the one connected with the core segment through a central node, 2 and 3 are the ones that connect the compute with the virtual operator, and finally 1 is connected to the management node. In the core segment, the IaaS compute has an interface 0 which is the connection point with the edge segment, the interface 4 is the connection point with other domains, interfaces 2 and 3 connect the compute with its virtual tenant operator, and finally interfaces 1 and

[1]https://www.nrl.navy.mil/itd/ncs/products/core
[2]https://www.openvswitch.org/
[3]https://osmocom.org/projects/osmosgsn/wiki/OsmoSGSN

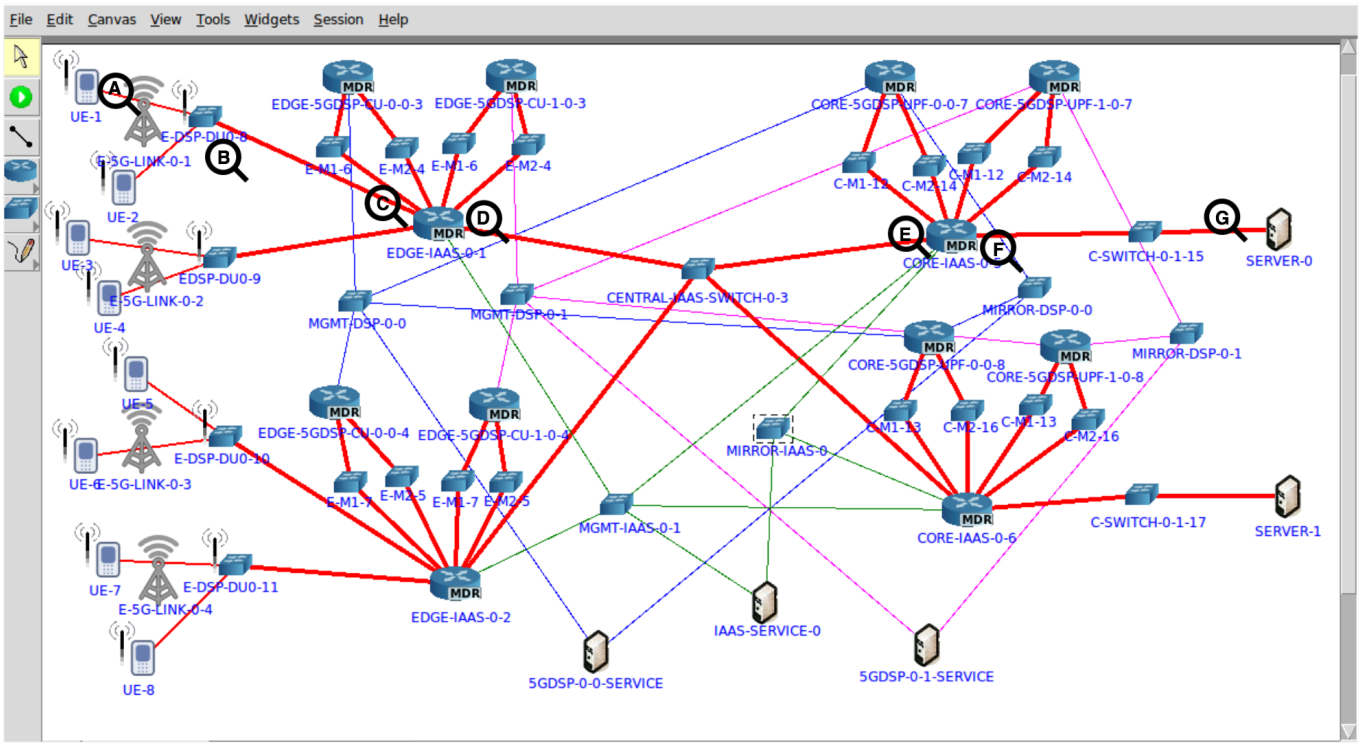

Fig. 3. Example of a template provided with the 5G framework.

5 are respectively connected to the core management node and the mirror node. Both user mobility and multi-tenancy should be configured to provide proper connectivity to the two users in Fig. 4. In order to provide the user's mobility, it is necessary to implement a GTP encapsulation protocol in the points indicated in the figure. Moreover, finally, to isolate tenant traffic it is necessary to encapsulate and de-encapsulate the traffic using VxLAN or GRE protocols in the bridges indicated in the figure. Thus, for every traffic sent from UE-1 to UE-2 through this infrastructure the traffic is encapsulated and de-encapsulated as shown in Fig. 4 from point A to G.

Fig. 3.

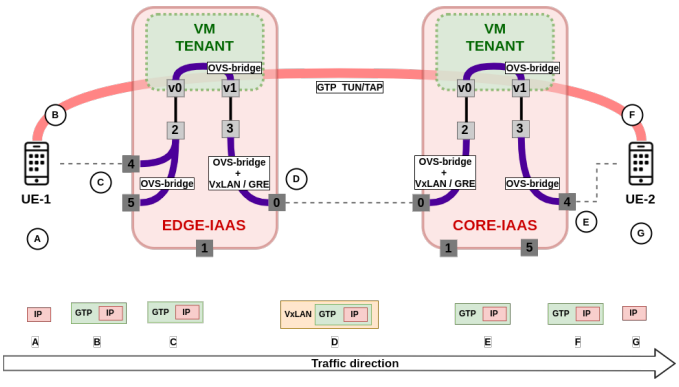

Fig. 4. Diagram of bridging and tunnelling for the case of an IaaS allocating one virtual operator, and traffic encapsulation provided.

The example shown in Fig. 4 could be reproduced with one of the templates provided with the framework. In such templates, there are included the nodes that play different roles: UEs, IaaS computes in the edge and core segment, virtual tenant operator VMs in the edge and core segment, a node for edge management, a node for core management, and a node for mirroring and inspecting the traffic. The framework identifies all those roles by parsing the names of the nodes that should follow a particular naming pattern depending on their role, listed below. These patterns are configurable through the configuration file. All of these nodes are represented in the template showed in Fig. 3.

The naming pattern and the roles are identified as follows:

- EDGE-IAAS-X-n : This node represents a compute in the edge segment that will act as an IaaS compute. Any node with the EDGE-IAAS prefix will be treated like that by this framework, applying the corresponding actions in the deployment of the scenario. Specifically, if this node is connected to one or more virtual operator node (EDGE-5GDSP-CU), the framework needs to build different bridges to provide the proper connectivity. One of the bridges will have to implement encapsulation to isolate the traffic of each tenant. X represents the administrative domain and n represents the id of the compute node.
- EDGE-5GDSP-CU-X-Y-n : This node represents a VM in the edge segment that will belong to a Virtual Operator allocated in an IaaS computer associated. X represents the id of the tenant, Y represents the id of the VM allocated in such compute and n is the id of the VM.

- CORE-IAAS-X-n : This node represents an IaaS compute in the core segment. If the IaaS is hosting a tenant, this node will be connected to a CORE-5GDSP-UPF node. X represents the administrative domain and n is the id of the compute node.
- CORE-5GDSP-UPF-X-Y-n : This node corresponds to the virtual machine of a tenant allocated in an IaaS compute in the core. X represents the id of the tenant, Y represents the id of the VM allocated in such compute and n is the id of the VM.
- IAAS-SERVICE-X-Y represents the node connected through the management network to the IaaS nodes. If the mirror is enabled, this node will receive all the traffic passing through the IaaS core segment nodes. In this node, any management service could be deployed, either through the vcmd program or the core-gui.
- 5GDSP-SERVICE-X-Y represents the node connected through the management network to the virtual operators. In this node, any management service could be deployed to control the nodes of the virtual operators. Depending on the business logic defined, this nodes will be managed either by the IaaS or the virtual operator itself.

C. Implementation

The integration of the framework with CORE is made via the vcmd program. This work has been implemented with bash scripting. Thus, different scripts are executed in the nodes of the scenario depending on its role in the 5G network, through this program. In order to facilitate the setup of the experimentation testbed to the user, an installer has been developed for Linux platforms, to install all the modules, their prerequisites packets as well as its configuration. Furthermore, a *start.sh* script is provided to run any experiment, once the framework has been installed. This script expects a configuration file like the one provided and an *.imn* format scenario following the templates provided. The configuration file allows the user to select the following parameters:

- Unique interface names. This is a patch in the original vnode python script of CORE in order to achieve a unique identifier for each interface. For this purpose, it combines the name of the node with the index of the interface. If this option is disabled, the interfaces of the scenario will be named as default (eth0 for instance) when the scenario is launched.
- Prefix names of the critical nodes.
- Encapsulation protocol for tenant isolation (VxLAN or GRE).
- Enable mirror if possible. If the scenario provided contains a node with the prefix of the mirror switch, and this option is enabled, the framework will execute the script to enable the mirror.

When the *start.sh* script is executed, the machinery of the framework gets the scenario and the configuration file and start running the scripts to build the needed bridges for this scenario and implement the encapsulation tunnels according to the topology given and the configuration defined. The implementation of the GTP tunnels has been made with Openggsn, running several scripts locally in the two nodes that compose the tunnel. The implementation of VxLAN and GRE encapsulation has been made with OpenvSwitch. Every ovs command from OpenvSwitch has to be run locally also, with unique process run and unique database per node to be managed. This is to avoid conflicts with duplicated names of interfaces and bridges. An extract of the script executed in each node to setup OpenvSwitch is shown in Listing. 1. When CORE instantiates a new node through the program vnoded, it creates a Linux network namespace, and listen on a control channel for commands. This control channel is a UNIX domain that follows the pattern /tmp/pycore.$sessionid/$nodename. Thus, for example /tmp/pycore.56342/EDGE-IAAS-0-0 refers to the node EDGE-IAAS-0-0 of session 56342.

```
mkdir -p openvswitch
ovsdb-tool create /tmp/pycore.$sessionid/$nodename.conf/
     openvswitch/conf.db "$ovspath"/vswitchd/vswitch.
     ovsschema
ovsdb-server /tmp/pycore.$sessionid/$nodename.conf/
     openvswitch/conf.db --remote=punix:/tmp/pycore.
     $sessionid/$nodename.conf/openvswitch/db.sock \
    --remote=db:Open_vSwitch,Open_vSwitch,manager_options \
    --private-key=db:Open_vSwitch,SSL,private_key \
    --certificate=db:Open_vSwitch,SSL,certificate \
    --bootstrap-ca-cert=db:Open_vSwitch,SSL,ca_cert \
    --pidfile=/tmp/pycore.$sessionid/$nodename.conf/
         openvswitch/ovsdb-server.pid \
    --log-file=/tmp/pycore.$sessionid/$nodename.conf/
         openvswitch/ovsdb-server.log \
    --detach \
    --unixctl=/tmp/pycore.$sessionid/$nodename.conf/
         openvswitch/ovsdb-server.ctl

ovs-vswitchd unix:/tmp/pycore.$sessionid/$nodename.conf/
     openvswitch/db.sock \
        -vconsole:emer \
    -vsyslog:err \
    -vfile:info \
    --mlockall \
    --no-chdir \
    --log-file=/tmp/pycore.$sessionid/$nodename.conf/
         openvswitch/ovs-vswitchd.log \
    --pidfile=/tmp/pycore.$sessionid/$nodename.conf/
         openvswitch/ovs-vswitchd.pid \
    --detach \
    --monitor \
    --unixctl=/tmp/pycore.$sessionid/$nodename.conf/
         openvswitch/ovs-vswitchd.ctl
```

Listing 1. Bash script extract for OpenvSwitch setup.

In the example shown in Fig. 4 the VxLAN encapsulation is achieved by following the steps detailed below. Listing. 2 shows an extract of the bash scripts where VxLAN encapsulation is implemented following those steps. In this extract $sessionid represents the identifier of the session managed by core-daemon, and $nodename is the name of the node defined in the .imn scenario.

Besides the bridge created between interfaces 4, 5 and 2 in Fig. 4, there is another bridge that will implement the

VxLAN encapsulation. First, an ovs bridge is created adding a port with the interface in the IaaS compute that is connected with the output interface of the tenant machine, 3 in Fig. 4. $brname refers to the name of the bridge. After creating the bridge, ip is added to the output interface of the IaaS, 0 in Fig. 4, that has no ip defined by default in the scenario. Then, new ports are added to the last bridge created in order to create the overlay network within the different IaaS nodes, which ips are represented by $iaas_nodes_ip in Listing. 2, setting an interface with the encapsulation type indicated in the configuration file, VxLAN or GRE, and represented by $tuneltype in Listing. 2. Note that it is necessary to pass the location of the database db.sock to every ovs command, which is the local database of each machine represented by a node of CORE.

```
sudo ovs-vsctl --db=unix:/tmp/pycore.$sessionid/$nodename.
     conf/openvswitch/db.sock add-br $brname
sudo ovs-vsctl --db=unix:/tmp/pycore.$sessionid/$nodename.
     conf/openvswitch/db.sock set bridge $brname stp_enable=
     true
ip link set $brname up
sudo ovs-vsctl --db=unix:/tmp/pycore.$sessionid/$nodename.
     conf/openvswitch/db.sock add-port $brname ${v_iface}
ip addr add $ipbridge dev ${p_iface}
i=0
for ip in "${iaas_nodes_ip[@]}"
do
  sudo ovs-vsctl \
  -- db=unix:/tmp/pycore.$sessionid/$nodename.conf/
       openvswitch/db.sock \
  add-port $brname ${tunneltype}_$i  \
  -- set interface ${tunneltype}_$i \
  type=$tunneltype     \
  options:remote_ip=$ip
  ((i++))
done
```

Listing 2. Bash script extract for VxLAN implementation.

IV. Validation of the Framework

In this section, the framework capabilities described in previous sections will be validated. This test has been run in a Virtual Machine created from scratch for the validation, with just the operating system installed, Ubuntu 16.04. Once the installer finished, a template has been tested. The template scenario that has been used is precisely shown in Fig. 3, which is a network with one domain, one IaaS and two tenants allocated in the IaaS. In this case, the management of each tenant is independent of the IaaS. Thus, in The selected encapsulation protocol for tenant isolation in this test has been VxLAN. The name of the interfaces of each node is not unique, and the mirroring is enabled. Fig. 3 shows that three different mirror-nodes mirror the traffic from the three managed entities, the IaaS, tenant-0 and tenant-1. The traffic is mirrored to three different service nodes IAAS-SERVICE-0, 5GDSP-0-0-SERVICE and 5G-DSP-0-1-SERVICE, that belong to the three entities respectively. This service nodes will manage their respective networks. Thus, each service node is connected to a management node, see prefix MGMT, that is connected to the corresponding nodes to be managed in their network. For the validation of the capabilities, some UDP traffic has been generated from the node labelled as UE-1, which belongs to tenant-0, to the node labelled as SERVER-0, and the traffic has been captured in different points of the infrastructure. These points are represented with a letter in a loupe in Fig. 3. When the *start.sh* script calls the bridging scripts, the framework first implements the needed bridges, as shown in Fig. 4. Then, it implements the GTP tunnels for user mobility. In this case, since there are two tenants per IaaS, four bridges are needed in each IaaS node. Furthermore, one bridge is implemented in each tenant node. These bridges are dynamically implemented with OpenVSwitch following Listing. 2 and they are built managing a unique database per namespace. Two of the bridges of each IaaS, one per tenant, will implement an interface type VxLAN for encapsulation and de-encapsulation. For the communication between the UE-1 and the SERVER-0 nodes, a tunnel has been dynamically built with Openggsn.

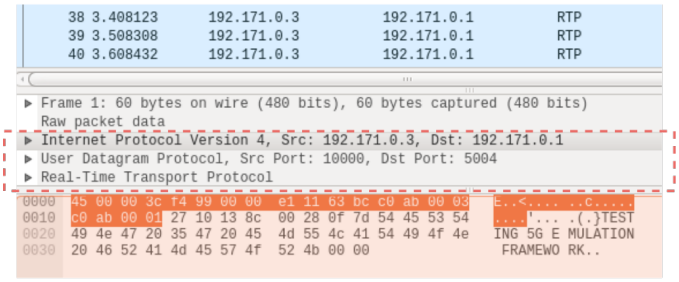

Fig. 5. Capture of the traffic in the tun0-00 interface of the client (point labeled as A in Fig.3). Before GTP tunneling.

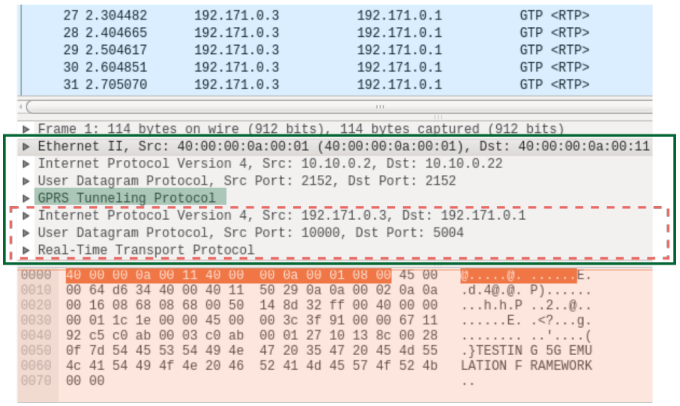

Fig. 6. Capture of the traffic in the eth4 interface of the IaaS compute in the EDGE (point labeled as C in Fig.3). Before VxLAN encapsulation.

Fig. 5 shows the traffic sent from UE-1 before applying any encapsulation. The traffic has been gathered precisely at point labelled as A in Fig.3. The traffic sent is an RTP over UDP flow. This user is connected to an antenna, that is connected to the Distributed Unit E-DSP-DU-0-8, that is in turn connected to the node EDGE-IAAS-0-1. In this scenario, each packet sent from UE-1 to SERVER-0 is first encapsulated over GTP.

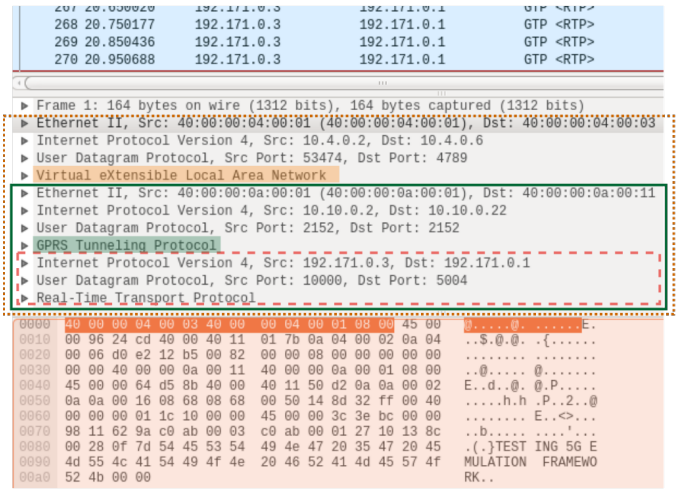

Fig. 7. Capture of the traffic in the eth0 interface of the IaaS compute in the EDGE (point labeled as D in Fig.3). After VxLAN encapsulation.

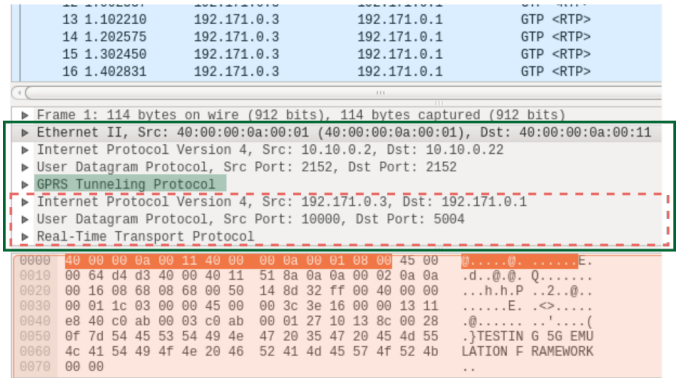

Fig. 8. Capture of the traffic in the eth4 interface of the IaaS compute in the CORE (point labeled as F in Fig.3). After VxLAN de-encapsulation.

Then the traffic enters into the EDGE-IAAS-0-1 at the edge through one of its interfaces. Fig. 6 shows the traffic captured in this input interface of the IaaS. The traffic has been gathered exactly at the point labelled as C in Fig. 3.

GPRS tunnelling protocol can be seen in Fig. 3. The input interface of the IaaS is connected through a bridge with the interface that is connected to the EDGE-5GDSP-CU-0-0-3, which represents the virtual machine of tenant-0 in this IaaS compute of the edge. The traffic goes through this tenant machine and returns to the IaaS node where it is finally encapsulated to provide tenant isolation. This encapsulation inserts the identification of the tenant. In this stage, the flow is double encapsulated as it is shown in Fig. 7. The traffic has been gathered exactly at the point labelled as D in Fig.3. After that, the traffic passes through the CENTRAL-IAAS-SWITCH-0-3 and pass through the CORE-IAAS-0-5 node where the traffic is VxLAN de-encapsulated. Fig.8 shows the traffic captured in the output interface of this node. Finally, the traffic is GTP de-encapsulated before being received by the destination user SERVER-0. The traffic has been gathered exactly at point labelled as F in Fig.3. This execution validates the dynamic encapsulation achieved in each of the different network segments of the architecture. The dynamic encapsulation is critical to allow users to run their services in a real 5G data plane where edge and core network segments are present, where dynamic GTP and VXLAN/GRE encapsulation are provided and where mobility is achieved by the control plane provided by the Openggsn software. To the best of our knowledge, this is the first attempt to provide an emulator that allows the user to define its own 5G infrastructure and thus it is automatically deployed with such capability.

This framework allows network researchers to test and benchmark their services while the industry moves towards 5G. Any solution for network managing could be tested within this framework in a real 5G data plane, including users, tenant and infrastructure management. The dynamism of this framework allows the user to explore different network monitoring schemes, and different network management approaches, as well as new network protection strategies.

V. Performance Evaluation

In this section, a performance evaluation of CORE emulator together with an assessment of the time overhead associated with the proposed extension are detailed. The evaluation has been made in terms of time consumed by the CORE to deploy an scenario and then, by the proposed extension to deploy the 5G capabilities. The evaluation tests have been executed for different scenarios varying the number of nodes.

The topology of each scenario used for the experiments follows the same scheme as showed in Fig. 3, but with just one tenant per IaaS. The scenarios used for the validation have been created ranging the number of IaaS entities from 1 to 100, resulting in scenarios ranged from 59 nodes to 3713

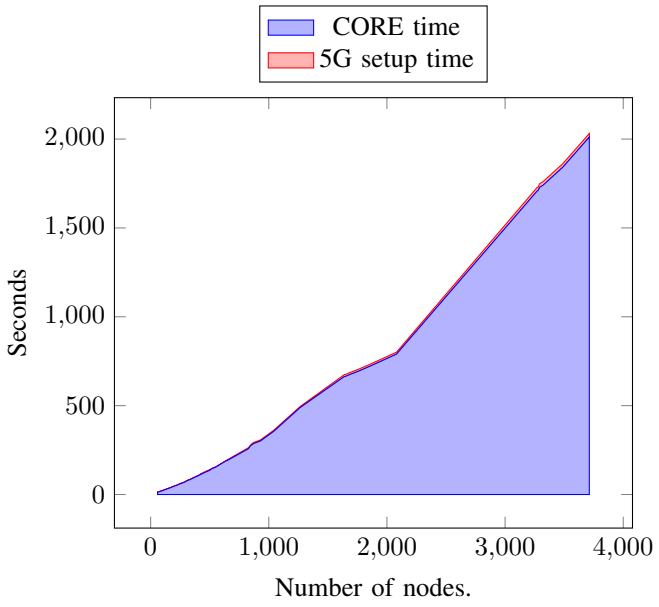

Fig. 9. Results of the time spent by CORE emulator to deploy scenarios with different numbers of nodes and the overhead added by the proposed extension to deploy the 5G connectivity.

nodes. The scenarios have been executed on an Intel Core i7 CPU 4.20 GHz, 32GB RAM hosting a Virtual machine with 8GB RAM and 4 cores. For the experiment, each scenario has been deployed with core-gui in batch mode, which is without launching the graphical interface, and then the 5G capabilities have been applied with the extension proposed. Both times, the time of CORE and the time of the extension that manages the 5G connectivity have been measured in the experiment and showed in Fig. 9. In Fig. 9 it can be seen how the time spent by CORE grows proportionally with the number of nodes. Also, it is shown how the overhead added by the proposed extension is a tiny part of the total time. For instance, for the biggest scenario, with 3713 nodes and 5760 links, CORE takes an average of 2011 seconds to deploy it, while the extension proposed takes 20 seconds to set-up the 5G connectivity, that means, tenant isolation and user mobility.

VI. Conclusion

In this paper, a novel framework for the emulation of 5G scenarios is provided. This framework is integrated with CORE emulator allowing the creation of different scenarios which topologies match several 5G business architectures. Furthermore, this framework provides the scenarios with user mobility and multi-tenancy as key capabilities in a 5G infrastructure. Some validation tests have shown the dynamic encapsulation happening in the 5G infrastructure along the different network segments in order to allow users to check their compliance with these requirements imposed by 5G infrastructures. The experimental tests deploying scenarios with different sizes prove a minimal overhead added to the CORE emulator by the extension proposed regarding deployment time. As a future work, this framework will explore the distributed mode of CORE emulator to provide higher hardware capabilities.

Acknowledgment

This work has been funded in part by the European Commission Horizon 2020 5G PPP Programme under grant agreement number H2020-ICT-2014-2/671672 SELFNET (Self-Organized Network Management in Virtualized and Software Defined Networks). This work has been additionally funded by the UWS 5G Video Lab project.

References

[1] "5G Vision." [Online]. Available: https://5g-ppp.eu/wp-content/uploads/2015/02/5G-Vision-Brochure-v1.pdf

[2] A. Prasad, Z. Li, S. Holtmanns, and M. A. Uusitalo, "5G micro-operator networks A key enabler for new verticals and markets," in *2017 25th Telecommunication Forum (TELFOR)*. IEEE, nov 2017, pp. 1–4. [Online]. Available: http://ieeexplore.ieee.org/document/8249272/

[3] E.-M. Oproiu, I. Gimiga, and I. Marghescu, "5G Fixed Wireless Access-Mobile Operator Perspective," in *2018 International Conference on Communications (COMM)*. IEEE, jun 2018, pp. 357–360. [Online]. Available: https://ieeexplore.ieee.org/document/8430184/

[4] J. Kim, D. Kim, and S. Choi, "3GPP SA2 architecture and functions for 5G mobile communication system," *ICT Express*, vol. 3, no. 1, pp. 1–8, 3 2017.

[5] OpenAirInterface Software Alliance, "OpenAirInterface - 5G software alliance for democratising wireless innovation," 2015. [Online]. Available: http://www.openairinterface.org/

[6] "ns-3." [Online]. Available: https://www.nsnam.org/

[7] M. Mezzavilla, M. Zhang, M. Polese, R. Ford, S. Dutta, S. Rangan, and M. Zorzi, "End-to-End Simulation of 5G mmWave Networks," *IEEE Communications Surveys & Tutorials*, pp. 1–1, 2018. [Online]. Available: http://ieeexplore.ieee.org/document/8344116/

[8] F. Boccardi, R. Heath, A. Lozano, T. L. Marzetta, and P. Popovski, "Five disruptive technology directions for 5G," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 74–80, feb 2014. [Online]. Available: http://ieeexplore.ieee.org/document/6736746/

[9] "OMNeT++ Discrete Event Simulator - Home." [Online]. Available: https://www.omnetpp.org/

[10] "Mobility Framework (MF) for OMNeT++." [Online]. Available: http://mobility-fw.sourceforge.net/

[11] "OMNeT++ Discrete Event Simulator - Home." [Online]. Available: https://www.omnetpp.org/

[12] "SimuLTE - LTE User Plane Simulator for OMNeT++ and INET." [Online]. Available: http://simulte.com/

[13] I. Ucar, B. Smeets, and A. Azcorra, "simmer: Discrete-Event Simulation for R." [Online]. Available: https://arxiv.org/pdf/1705.09746.pdf

[14] I. Ucar, A. Hernández, P. Serrano, and A. Azcorra, "Design and Analysis of 5G Scenarios with simmer: An R Package for Fast DES Prototyping." [Online]. Available: https://arxiv.org/pdf/1801.09664.pdf

[15] "An End-to-End Testing Ecosystem for 5G The TRIANGLE Testing House Test Bed." [Online]. Available: https://www.triangle-project.eu/wp-content/uploads/2016/06/triangle_eucnc16_cameraready.pdf

[16] J. Ahrenholz, C. Danilov, T. R. Henderson, and J. H. Kim, "CORE: A real-time network emulator," in *Proceedings - IEEE Military Communications Conference MILCOM*, 2008. [Online]. Available: https://www.researchgate.net/publication/261091924

[17] L. Angrisani, D. Capriglione, G. Cerro, L. Ferrigno, and G. Miele, "Experimental analysis of software network emulators in packet delay emulation," in *2017 IEEE International Workshop on Measurement and Networking (M&N)*. IEEE, sep 2017, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/document/8078382/

[18] "Common Open Research Emulator (CORE) — Networks and Communication Systems Branch." [Online]. Available: http://www.nrl.navy.mil/itd/ncs/products/core