Demonstration of Orchestration of 5G Core Network Functions with a Satellite Emulator

Ayoub Bergaoui*, Youssouf Drif*†, Tanissia Djemai*,
Olivier Gremillet*, Ahmad Al Sheikh* and Pascal Berthou†
*IRT Saint-Exupéry, France; Email: firstname.lastname@irt-saintexupery.com
†IRIT, Toulouse, France;
†LAAS, Toulouse, France; Email: pascal.berthou@laas.fr

***Abstract*—In this demonstration, we focus on the orchestration of Network Functions (NFs) following the Network Function Virtualization (NFV) paradigm. The deployed NFs are 1) 5G core network, 2) User Equipment (UE) and Radio Access Network (RAN) simulator, and 3) Satellite Emulator.**

This orchestration combines different types of NFs. We will show how we can orchestrate 5G Core networks as a Kubernetes Network Function and Satellite network elements that can be seen as Virtual or Physical NFs. The aspects linked to the NF auto-scaling will also be demonstrated with a focus on container services.

This work was performed in a research project dealing with the integration of satellite backhaul links in 5G networks. This particular context is exposed and also the intended scenarios. The demonstration platform hosting the services to be deployed is also described: architecture, components and infrastructure.

***Index Terms*—Satellite Networks, 5G, Network Slicing, SDN, NFV, Orchestration**

I. Introduction

A. The need for orchestration

Future network services are likely to be developed using NFs. Their deployment will be done by an orchestrator, which will be able to manage all operations independently of platform or environment. European Telecommunication Standards Institute (ETSI) has put lots of effort to coordinate the definition of Management and Orchestration (MANO) functionalities (like NFV abstraction [1]). In addition, a software solution called Open Source MANO (OSM) is actively developed [2].

There are many advantages related to the use of orchestration, like : i) increased control of the application life cycle, ii) automated service configuration, iii) scaling of services, iv) unified management of heterogeneous platforms.

This demonstration aims to demonstrate them in our dedicated context.

B. Context and overall project objectives

Our work is part of a research project dealing with Satellite Integration in fifth generation mobile (5G). Satellite networks are considered as Non Terrestrial Networks (NTNs) in a Third Generation Partnership Project (3GPP) context. NTNs were defined by the 3GPP to describe and integrate heterogeneous networks into 3GPP networks. In link with this research project, the considered use case for the satellite is Backhauling. It means that the satellite link is used to relay the data of a next generation Node B (gNB) located in the RAN towards the Core Network (CN).

C. Similar work

At the European level, the integration of the satellite into 5G has been analysed in several projects such as SaT5G [3] and SATis5 [4]. Our work takes place in the continuity of their initial findings, where the need of an orchestrated solution for Satcom NFs was evidenced. As NFs have been mainly used so far by telco operators, their usage for Satcom networks has to be popularized. Some work has been done by other teams using the ONAP orchestration platform.

II. Description of the orchestration platform

A. Possible and selected software frameworks

As explained previously, our platform will integrate the following components, with the corresponding selected implementation:

- Management and Orchestration tool: the OSM software has been selected, as it is much lighter than the ONAP solution.
- Virtual Infrastructure Managers: as a Virtualized Infrastructure Manager (VIM), we have used the Openstack solution (for the orchestration of virtual machines), and a Kubernetes Cluster (for the orchestration of containers).
- 5G Core Network / RAN: the implementation of Free5GCore has been selected [5] because this tools was identified as having the most functions compliant with 3GPP Release 15. As a consequence, the UERANSIM was used, as it has a common origin with Free5GC.
- Satellite Emulator: the opensand emulator has been used [6].

B. Platform Architecture

Our platform is described in Figure 1, with different layers corresponding to the infrastructure, the infrastructure as a service and the platform as service at the top.

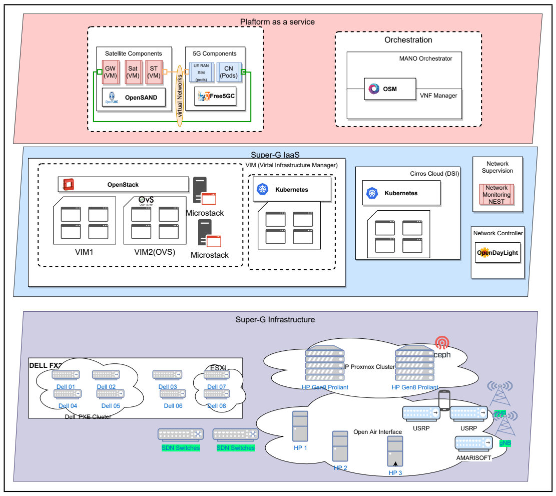


Figure 1. Project/Demonstration Infrastructure

At the bottom, our infrastructure is depicted. Our main components are 4 compute nodes based on HP-Gen8 servers with 16 CPUs, 96 GB of RAM, and a shared storage of 34 TB (using 3 storage servers). In addition, 8 Dell servers are also available as well as 3 Lab PCs.

Above, the proxmox hypervisor is used to provision Virtual Machines on the HP server, to enable the service layer for infrastructure to offer IAAS. A set of 4 Virtual Machines (VMs) can be used to build an OpenStack private cloud. As an alternative, a single VM can host Microstack, a simplified single package for OpenStack. A set of 5 VMs can be used to instantiate a Kubernetes cluster (nodes for worker nodes, master node). This introduces an additional level of virtualization, but eases the management of the software.

Finally, the top-level offers the services on the platform managed by the orchestration software for this demonstrator.

III. Demonstration

A. Orchestration of 5G services

1) Available solutions and adaptation: **Our demo will show how to convert the original docker CN into a Kubernetes function**.

The Free5GCore and UERANSIM are based on containers and their default installation is done using docker compose. Our orchestrator requires a Kubernetes environment for the deployment of containers, that will be deployed via either a helm charts package or juju bundles.

For that, we used the Kompose tool which transforms the docker compose into k8s manifests. We can use Kompose in two ways: by working directly with our Compose files, or by translating them into YAML Kubernetes files (manifests), which can then be deployed using `kubectl`. The second method was used.

Figure 2 shows the prerequisites for all services that are orchestrated in the rest of this demonstration and the possible actions that the user has to perform to orchestrate these services.

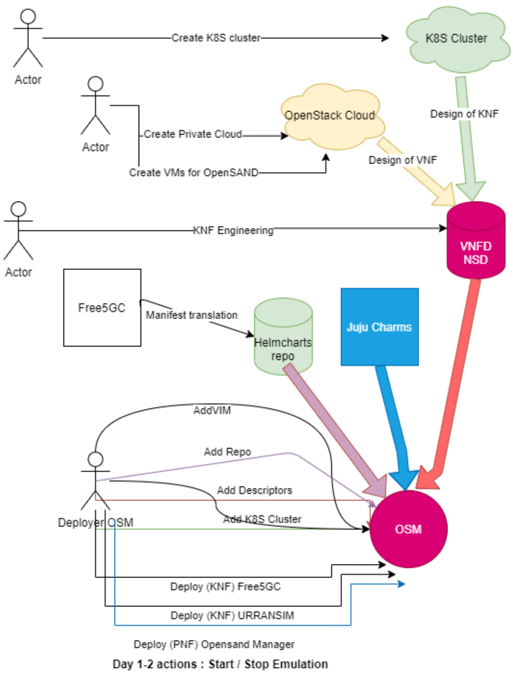


Figure 2. User interface diagram platform

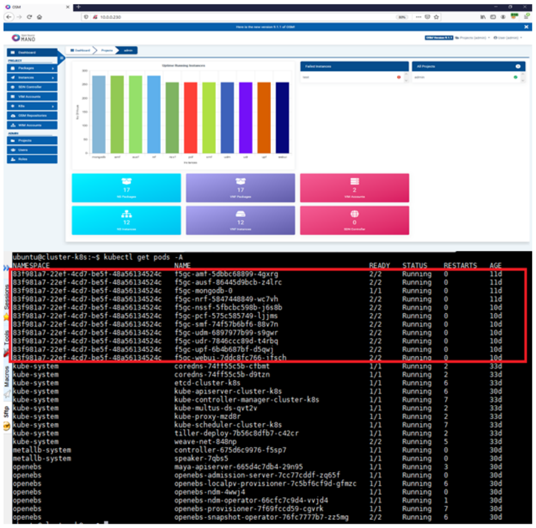
2) Orchestrated Solutions: **Our demo will show an orchestrated 5G Core network deployed using OSM on a Kubernetes cluster.**

OSM is used to orchestrate the 5G services: Free5GCore and UERANSIM. They are handled as KNF (Kubernetes-based Network Function). The following operations have to be performed:

- The first step, is that user should create a proper cluster Kubernetes and add it to map OSM and add to the helm repo developed previously for our containers
- After that, we will create and add our vnf and ns descriptor and launch our NS from OSM.
- OSM will retrieve the helm chart package, from a public github directory, which obviously contains the kubernetes manifests.
- Once the packages have been retrieved, OSM will deploy them in kubernetes through the Resources orchestration (RO) module.

As an output we will have pods deployed in our

Kubernetes cluster where each pod is represented by a 5G component (pod AMF, pod UPF, pod UE, pod RAN...). Figure 3 shows the deployed pods on kubernetes client and its corresponding information on the OSM User Interface.



```
ubuntu@cluster-k8s:~$ kubectl get pods -A
NAMESPACE                             NAME                                          READY   STATUS    RESTARTS   AGE
83f981a7-22ef-4cd7-be5f-48a56134524c  f5gc-amf-5dbbc68899-4gxrg                     2/2     Running   0          11d
83f981a7-22ef-4cd7-be5f-48a56134524c  f5gc-ausf-86445d9bcb-z4lrc                    2/2     Running   0          11d
83f981a7-22ef-4cd7-be5f-48a56134524c  f5gc-mongodb-0                                1/1     Running   0          11d
83f981a7-22ef-4cd7-be5f-48a56134524c  f5gc-nrf-5847448849-wc7vh                     2/2     Running   0          11d
83f981a7-22ef-4cd7-be5f-48a56134524c  f5gc-nssf-5fbcbc598b-j6s8b                    2/2     Running   0          10d
83f981a7-22ef-4cd7-be5f-48a56134524c  f5gc-pcf-575c585749-ljjms                     2/2     Running   0          10d
83f981a7-22ef-4cd7-be5f-48a56134524c  f5gc-smf-74f57b6bf6-88v7n                     2/2     Running   0          10d
83f981a7-22ef-4cd7-be5f-48a56134524c  f5gc-udm-6897977b99-s9gwr                     2/2     Running   0          10d
83f981a7-22ef-4cd7-be5f-48a56134524c  f5gc-udr-7846ccc89d-t4rbq                     2/2     Running   0          10d
83f981a7-22ef-4cd7-be5f-48a56134524c  f5gc-upf-6b4b687bf-d5qwj                      2/2     Running   0          10d
83f981a7-22ef-4cd7-be5f-48a56134524c  f5gc-webui-7ddc8fc766-jfsch                   2/2     Running   0          10d
kube-system                           coredns-74ff55c5b-cfbmt                       1/1     Running   2          33d
kube-system                           coredns-74ff55c5b-d9tzn                       1/1     Running   2          33d
kube-system                           etcd-cluster-k8s                              1/1     Running   6          33d
kube-system                           kube-apiserver-cluster-k8s                    1/1     Running   6          30d
kube-system                           kube-controller-manager-cluster-k8s           1/1     Running   7          33d
kube-system                           kube-multus-ds-qvt2v                          1/1     Running   2          33d
kube-system                           kube-proxy-mzd8r                              1/1     Running   2          33d
kube-system                           kube-scheduler-cluster-k8s                    1/1     Running   7          33d
kube-system                           tiller-deploy-7b56c8dfb7-c42cr                1/1     Running   2          33d
kube-system                           weave-net-848np                               2/2     Running   5          33d
metallb-system                        controller-675d6c9976-f5sp7                   1/1     Running   0          30d
metallb-system                        speaker-7qbs5                                 1/1     Running   0          30d
openebs                               maya-apiserver-665d4c7db4-29n95               1/1     Running   3          30d
openebs                               openebs-admission-server-7cc77cddf-zq65f      1/1     Running   0          30d
openebs                               openebs-localpv-provisioner-7c5bf6cf9d-gfmzc  1/1     Running   6          30d
openebs                               openebs-ndm-4wwj4                             1/1     Running   0          30d
openebs                               openebs-ndm-operator-66cfc7c9d4-vvjd4         1/1     Running   1          30d
openebs                               openebs-provisioner-7f69fccd59-cgvrk          1/1     Running   7          30d
openebs                               openebs-snapshot-operator-76fc7777b7-zz5mg    2/2     Running   6          30d
```

Figure 3. 5G orchestration results in k8s and osm

3) *Orchestration of a satellite link:* **The demo will show the orchestration of the Satellite emulator (OpenSand) using OSM on an OpenStack.** Our Satellite emulator is represented as a Physical Network Function (PNF) and composed of 3 VMs (sat, gw, st).

On Figure 2, we can see that our virtual machines have been set up beforehand, and the orchestration in this part will add what we call “Day-1” and “Day-2”. For that, we must follow these steps:

- The user should first create an OpenStack environment, install and create the virtual machines of our emulator OpenSand in advance.
- The user must develop juju charms to orchestrate this pnf, for example, in our case, we decided to create start/stop actions for our emulator OpenSand
- After that, we will create and add our pnfd and nsd and launch our ns from OSM.
- Juju will communicate to OSM to start emulation or stop emulation (a.k.a. “Day-2” action) when the user issue the request.

B. Autoscaling

Autoscaling is the capability for a service to adapt the required resources for its current need. **In our demo, we will show how we made our strategy of autoscaling for KNFs (pods/ containers in Kubernetes) and VNFs (Virtual machines in OpenStack).** There are two methods for doing autoscaling in kubernetes: the Horizontal Pod Autoscaler (HPA) and the Vertical Pod Autoscaler (VPA). The principle in general for HPA, as its name suggests, when a User Plane Function (UPF) pod for example becomes overloaded, ram and cpu wise, the number of runnings will be modified upwards on downwards depending of events. The VPA concept, unlike HPA, adds resources directly to the pod when consumption exceeds a threshold set in the script. The two options are depicted Figure 4 and will be demonstrated

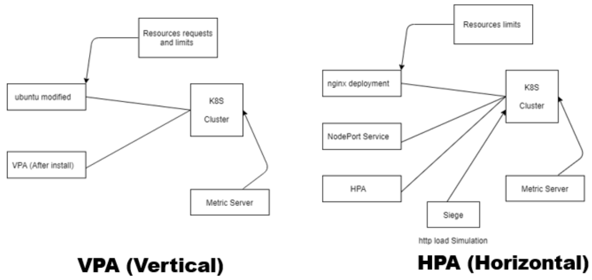


Figure 4. Autoscaling

C. Orchestration of end to end services

To launch all our network services, we create and deploy a network slice in OSM. It should be noted that there is a network service for each 5G component (i.e. 11 network services) in addition to the OpenSand orchestration network service. This makes a total of 12 network services. To avoid launching them each separately, we use the network slicing method offered by OSM. For that, we create what we call Network Slice Template (NST). **In our demo, we will show such a complete scenario**.

IV. Conclusion and Future Work

This demonstration shows the orchestration of a rich variety of network services that enables to provide a platform to test satellite NFs.

Future work will be conducted to integrate more components for more flexible scenarios. For example, some of the components described in [7] will be included.

References

[1] ETSI GS NFV 003, “Network Functions Virtualisation (NFV): Terminology for Main Concepts in NFV,” ETSI, Group Specification, 2014.

[2] “OSM.” [Online]. Available: https://osm.etsi.org/

[3] “H2020 SaT5G |.” [Online]. Available: https://www.sat5g-project.eu/

[4] “SATis5 | ESA’s ARTES Programmes.” [Online]. Available: https://artes.esa.int/projects/satis5-0

[5] “free5gc.” [Online]. Available: https://www.free5gc.org/

[6] Thales Alenia Space, CNES, Viveris Technologie, “Opensand,” https://opensand.org/content/home.php.

[7] Y. Drif *et al.*, “An Extensible Network Slicing Framework for Satellite Integration into 5G,” *International Journal of Satellite Communications and Networking*, 2020.