Cloud-based Network Function Virtualization Using OPNFV Components

Villacís Francisco, Eng[1], Mejía David, MSc[1], Becerra Fernando, MSc[1], Bernal Iván, PhD[1]
[1]Faculty of Electrical and Electronics Engineering, Escuela Politécnica Nacional, Ecuador, francisco.villacis@epn.edu.ec, david.mejia@epn.edu.ec, fernando.becerrac@epn.edu.ec, ivan.bernal@epn.edu.ec

Abstract*– The growth in the demand for services requires an expansion of the network capacity, that is, of their ability to achieve the transfer of information and provide new functionalities; but the clearest impediment to such growth is the hardware limitation. Due to this problem, new trends in network architecture present virtualization as a solution. NFV (Network Function Virtualization) allows decoupling network services or functions from proprietary hardware and deploying them in virtual instances on a cloud environment. Thus, this work aims to implement simple network functions (Web server, DNS, vRouter, firewall) by using specific OPNFV components, which is an open software project focused on the development and evolution of NFV. The chosen components for the development of this work are Openstack and Tacker. Openstack is used as a virtual infrastructure manager, i.e., a cloud computing administrator; while Tacker is the component that allows the virtual network function management and orchestration. As a final product, the implementation of a complete cloud-based network has been achieved, demonstrating the benefits in the creation, execution and management of virtual network functions offered by NFV technology.

Keywords-- NFV, OPNFV, Openstack, Tacker, VNF.

I. Introduction

In this document, a theoretical review of the concepts that make up NFV technology is carried out, as well as its origin, benefits, and scope. The OPNFV project is introduced as a consolidated solution for the implementation and development of the Network Function Virtualization technology.

OPNFV will assist in the deployment of NFV Infrastructure (NFVI) and Virtualized Infrastructure Management (VIM). Inside the virtual machine installed on the server, a computing cloud will be built and virtualized network services will be deployed. Within the cloud infrastructure, network topologies will be implemented using the selected components of the OPNFV project. These components will be in charge of the orchestration and management of the services also called VNF (Virtual Network Functions), which, once deployed, will be able to clearly show their operation in each topology through the network functionalities offered by the computing cloud.

After reviewing the concepts, this study focuses on the explanation of the necessary software components for this project, both the Openstack cloud and the Tacker MANO service. Once the platform on which it works is installed, the illustration of the processes that are needed for the implementation of each of the VNFs that are deployed is carried out.

Digital Object Identifier (DOI):
http://dx.doi.org/10.18687/LACCEI2021.1.1.533
ISBN: 978-958-52071-8-9 ISSN: 2414-6390

NFV not only allows the creation of services, but also opens up a whole field of new tools for management, this is how, with the chosen components of the OPNFV project, it will seek to apply a monitoring policy for each network service, and thus create a different virtual network with new functions compared to a traditional network with services implemented on specific hardware. Such monitoring can be observed in the computing cloud environment.

Finally, the final service presented by each VNF created could be seen. The visualization of the operation of each VNF is directly linked to the correct delivery of its service and the observation of its ongoing monitoring policy. This is how the performance of each of the virtualized functions implemented is shown, through connection, access, name resolution tests, etc., as the case may be.

II. NFV

The Virtualization of Network Functions was defined for the first time by ETSI (European Telecommunications Standards Institute), in 2012, as the implementation of different network functions in software, which can be executed on a variety of standard servers, and which can be instantiated in various locations on the network as necessary, without the need for the installation of new physical equipment [1]. This definition has been updated over the years to become the specification of an architecture that allows the decoupling of services or network functions (routers, switches, firewalls, etc.) out of proprietary hardware, and its deployment in virtual instances over a cloud environment [2].

A traditional network architecture presents several problems, such as: limitations of flexibility, scalability restrictions, delays in solving market demands, difficulty of monitoring the network, high operational costs, difficulty of updating and migration, oversizing of the network, and interoperability problems between providers [3]. NFV seeks to directly solve the restrictions presented by traditional networks, and in addition, to bring new benefits and a transformation in architecture, orchestration and network management. Fig. 1 shows a comparison between a traditional network model and a model with NFV. The main benefits of this technology will be listed below [1]:

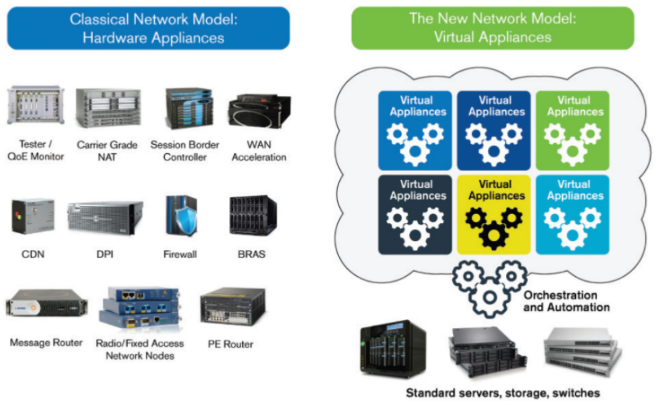


Fig. 1 Classic Network Model vs. Network model with virtualization [4]

- Equipment cost reduction. The purchase of specialized physical equipment is reduced, so the investment is less (Reduction of CAPEX or Expense Capital).

- Increase in the speed of response of the service against the demands of the market.

- Possibility of executing both final services and test projects, all on the same infrastructure, thus facilitating the validation of new solutions [3] and reducing development and operation costs (Reduction of OPEX or Operating Expense), as well as reducing time to market.

- Avoid network oversizing, since virtualization facilitates expansion and scaling when required. The growth of the network or the expansion of resources for a service are based on demand. Fig. 2 presents two comparative diagrams between Network Growth vs. Time for both a traditional network (left) and a network with virtualization (right). The black line represents demand for capacity and the broken blue line represents the capacity deployed by the service provider. For a network with virtualization, there are no times when the network lacks resources or, on the contrary, that it is oversized; but the network has a growth along with demand.

- Promote the development and innovation of different software ecosystems focused on networking.

- Optimize network configuration and its topology in real time (Virtual Network Function Management). The configuration actions that are carried out are scaling, operation changing, aggregation of new resources, state of virtualized network functions communication; all this without stopping the service.

- Support and interoperability between different hardware brands. Proprietary brand independence.

- Power consumption reduction by exploiting the sharing of several network functions within the same server and the use of existing hardware (servers) in data centers [3].

- Increased operational efficiency of the physical network. The network physical layer links are not underused.

- Ease in the migration of services. Due to the nature of the software, the virtualized network can be stored on a device such as a hard disk and installed on another server. Therefore, the service provided can be moved, copied or deleted efficiently without having to mobilize or buy new specific hardware equipment.

III. OPNFV

NFV looks for network equipment disconnection with specific brands. This disassociation is achieved through open software technology implementation, which goal is totally focused on NFV, or is partially helping this same purpose. The problem with this development is the compatibility between one project and another, as there is a large number of developers worldwide. To solve this problem, the OPNFV project was consolidated.

Open Platform for NFV (OPNFV) is an open software project focused on the development and evolution of NFV, which through integration, testing and implementation of the different open software projects, seeks to create an NFV reference platform for the transformation acceleration of business networks and service providers [5].

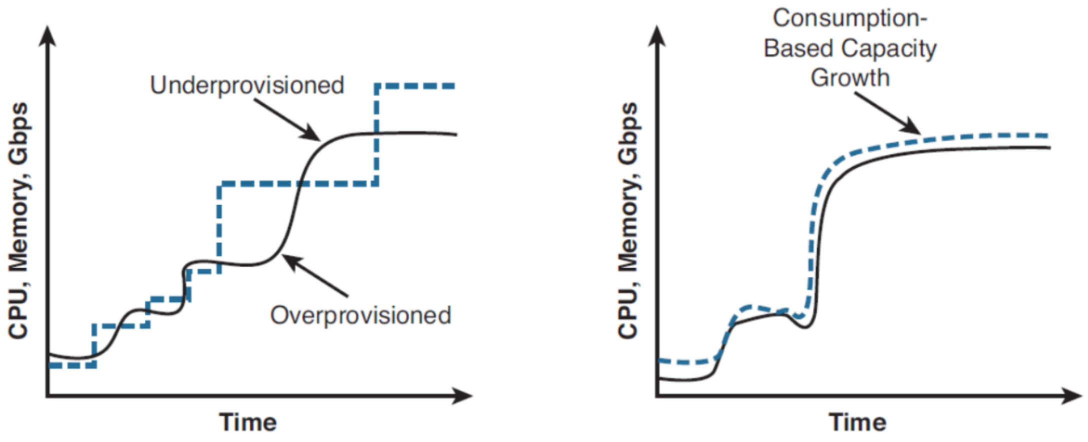


Fig. 1 Network growth vs. time in a traditional network and in a network with virtualization [3].

OPNFV takes projects for specific purposes such as: infrastructure virtualization, SDN controllers, cloud orchestration software, storage, data plane acceleration technologies, etc., and integrates them to create a common infrastructure for NFV.

The OPNFV architecture is based on the NFV architecture proposed by the ETSI, as shown on the left side of Fig. 3. The VNF layers, MANO and Infrastructure are observed, each with specific projects that interact for the implementation of a network with virtualization.

The right side of Fig. 3 shows the pillars of this project. The integration is based on the alignment of the different individual projects, that is, the search for compatibility between them and their grouping into common installers. The OPNFV project has a team of engineers who carry out continuous tests in their laboratories, such tests are about function, system, and performance. Both, the integration pillar, and the performance test are aligned in one pursuit, achieving new features for NFV technology. Finally, the entire project has a continuous integration and development approach, always cloud-based and with extensive documentation available to the user.

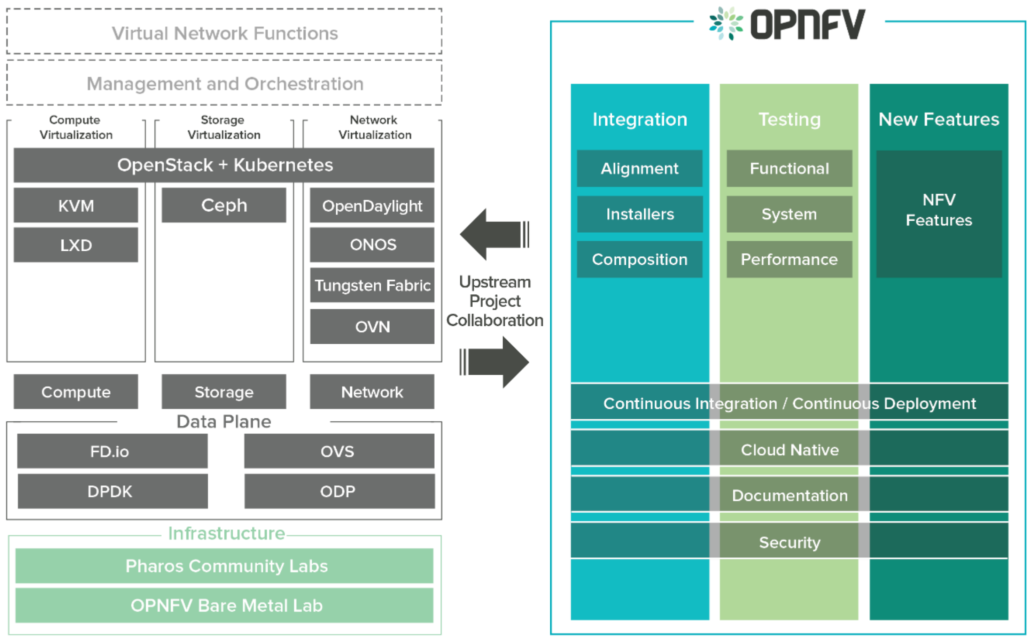


Fig. 3 Architecture diagram (left) and the pillars (right) of the OPNFV project [5]

In summary, the objectives of the OPNFV project are the following [6]:

- Develop an integrated and proven NFV software architecture;
- Include the participation of end users in the NFV development;
- Contribute and participate in open-source projects relevant to NFV;
- Establish an ecosystem of solutions for NFV and thus meet the needs of the end user; and,
- Promote OPNFV as the official platform for NFV.

OPNFV's philosophy is not to create projects specifically by itself, but to look for existing open software projects with features that benefit its objective, and incorporate them through a process of continuous integration, performance tests and compatibility implementation [7], as well, OPNFV seeks to show alternatives in each area of its infrastructure by not being restrictive in the choice of each component. Thus, OPNFV is a compilation of a long list of projects, which can be classified into four areas or layers: NFVI, MANO, SDN Controller and VIM.

The objective of this work is the creation of virtualized network functions on a cloud infrastructure. Thus, Openstack has been selected from among all OPNFV components, which functions as virtual infrastructure management software, that is, a cloud; likewise, for this work it has been selected Tacker, which will allow the administration and orchestration of virtualized network functions.

In the following sections, each of the OPNFV layers will be described, emphasizing the components that will be used in the development of this study.

A. *NFVI*

This layer consists of the network, storage and computing components, present in both hardware and virtualization software [8]. The three categories that make up the NFVI infrastructure are presented below:

- Compute virtualization: KVM, Docker
- Storage virtualization: Ceph
- Network virtualization / data plane acceleration: OVS, ODP, DPDK.

B. *VIM - Openstack*

Virtual Infrastructure Management (VIM) is the OPNFV layer that allows a cloud creation and management. The most used open software project for this purpose is Openstack.

Cloud computing is the set of virtual resources and software (operating systems, virtualization software, etc.), together with technical principles and approaches, which allow to create a computing infrastructure that provides services (Infrastructure as a Service), which functions as a platform (Platform as a Service) and in which applications (Software as a Service) can be hosted, all this coordinated by an administration and automation software [9]. Openstack seeks to integrate all this into an easy system to implement.

There are concepts of the different objects involved in the Openstack processes, which must be understood for the correct understanding and implementation:

- Instance: Virtual machine containing a virtual disk with an operating system. They are the individual virtual machines that are created and run on physical compute nodes within the cloud [10].
- Image: Template of the virtual machine file system for its deployment as an instance. Any number of instances can be displayed from the same image [10].
- Flavor: Describes the characteristics of memory, storage and computing resources for an instance. Example: 2 GB of RAM + 512 GB of storage + 2 virtual CPUs (vCPU).
- Volume: Storage unit that stores one or more instances. It is useful for service migration as it converts an instance to a file.

Openstack is structured by different components that make up the cloud computing system. Its architecture is shown in Fig. 4.

C. *MANO - Tacker*

According to ETSI, the Management and Orchestration layer also includes the functions of VIM and SDN Controller; however, for the purpose of this study, the definition of MANO will be restricted to include the functionalities of VNFM (Virtual Network Function Manager) and NFVO (Network Function Virtualization Orchestrator) [8].

- Virtual Network Function Manager (VNFM): Facilitates the initial configuration of a Virtual Network Function (VNF), manages the basic life cycle of a VNF (creation, update, deletion), monitors the deployed VNFs and manages the action in case of error and in need of escalation [12].

- Network Function Virtualization Orchestrator (NFVO): Verifies VIM resources and their allocation, has the ability to orchestrate multiple VNFs to create a wider network service in multiple VIMs [12].

It can be seen in Fig. 5 that Tacker is organized by several components. Upwards, it relates to Openstack's Horizon and provides the option to perform management actions through CLI, while downwards, it communicates with the cloud. Tacker, in addition to the components of VNFM and NFVO, provides a catalog, in which there is an important element for the development, this element is the VNFD (Virtual Network Function Descriptor).

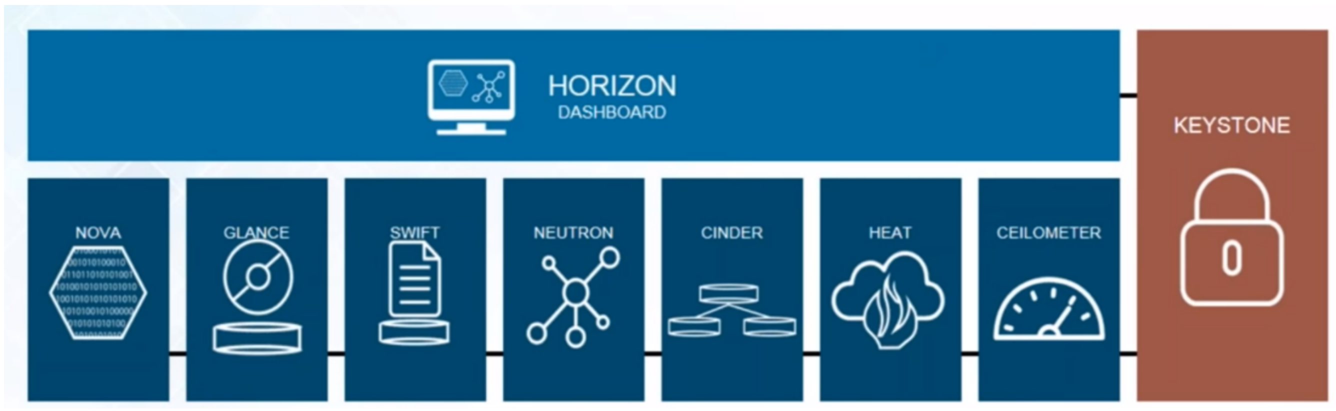


Fig. 4 Openstack Architecture [11]

Before the deployment of the Virtual Network Function, it must be entered into the cloud catalog with the characteristics with which it will be created. VNFD is a file that works with the standard TOSCA language (Topology and Orchestration Specification for Cloud Applications), which will allow the specification of three basic types of nodes [14]:

- Virtual Deployment Unit (VDU): Describes the Virtual Machine (VM) that will host the VNF and its features such as the operating system, memory capacity and storage. The VNFD file also specifies the monitoring policies that will be performed on this node, that is, on the service.

- Connection Point (CP): Describes the virtual network interface card of the VDU, that is, the network port of the virtual machine.

- Virtual link (VL): Describes the virtual link that the VDU will have with the Openstack Nova components, here it is specified to which cloud network the VDU will be connected. You can also create virtual links between VDUs.

Likewise, within this file scaling policies can be created, which allow you to increase or decrease the number of resources after the VNF is created as needed.

Openstack Heat orchestration service takes this VNFD file and translates it from the TOSCA format to a format that the cloud can understand [14]. The Tacker engine stores this descriptor in a VNF catalog where it can be started without having to make other settings.

Once the virtual function is deployed, it can be viewed as an instance running already in the Openstack cloud. This instance can be deleted, configured, or managed at the discretion of the network administrator.

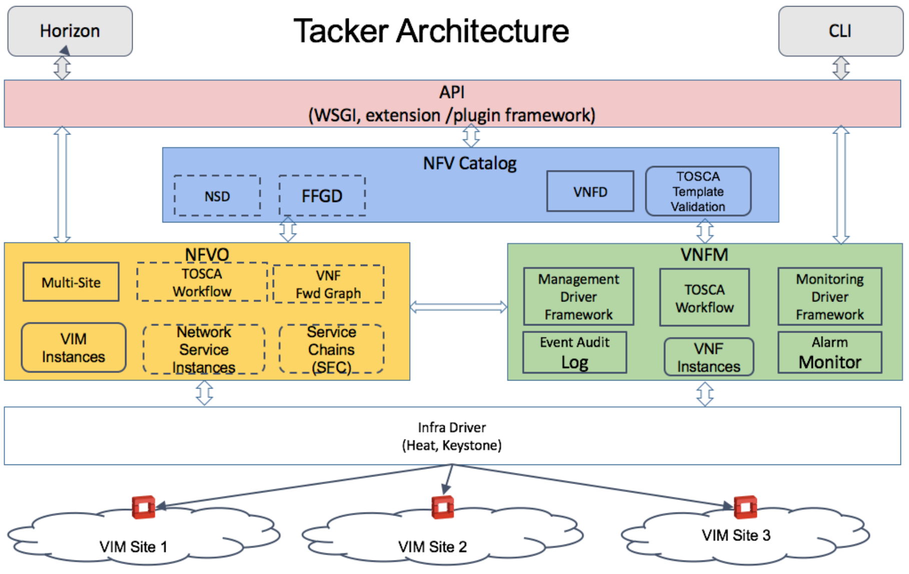


Fig. 5 Tacker Architecture [13]

IV. NFV Deployment

This project implementation was made on a DELL PowerEdge R630 physical server with Internet access. Within the physical server, a virtual machine is created, in which the cloud computing will be installed. Openstack and Tacker installation is done through the Devstack tool.

The design process is similar for each of the VNFs implemented. In this section, the creation of the different specific services is exposed. The key concepts to consider for the design of each virtualized network function are the following: service, need for network resources and method of visualization of the service in progress.

A. VNF – Web server

Within the VNFD configured for the service, it is observed that based on this script, a service will be created with a VDU1 instance, based on an image previously configured for the web server with Apache2. It will also implement a monitoring policy in which, pings will be made on the http port (port 80) of the instance and if those pings fail a respawn will be performed, which means that the instance will be deleted and a new one will be created automatically. This instance will be related to the connection point CP1 connected to the virtual link VL1, which in turn is connected to the net_mgmt network (192.168.120.0/24) of the cloud computing. Finally, the configuration of an escalation policy is made that will allow us to increase resources if necessary, after the VNF is instantiated.

B. VNF – DNS Server

In order to implement the DNS server, an OpenWRT image is used, which is a free software operating system that was specially developed for use as a virtual networking device, that is, it has the necessary packages to function as a DNS, firewall, vRouter, etc.

This VNFD will create a service with a VDU1 instance based on the OpenWRT image for the DNS server. A monitoring policy will also be implemented in which pings will be carried out periodically and, if these pings fail, a respawn will be performed. Finally, this instance will be related to connection points CP1 and CP2. CP1 connected to virtual link VL1, which in turn is connected to the net_mgmt management network (192.168.120.0/24) of the cloud computing. CP2 is related to the virtual link VL2 that is connected to the net0 network (10.10.0.0/24), which is a network created in the cloud.

A virtual machine test0 based on CirrOS is created and connected to the 10.10.0.0/24 network, in which the DNS server operation tests will be performed.

C. VNF – vRouter

This VNFD will create a service with a VDU1 instance based on the OpenWRT image for our vRouter, a monitoring policy will be implemented in which, pings will be performed periodically, and if these pings fail the failure will be reported to the tacker.log file. Finally, this instance will be related to connection point CP1, CP2 and CP3. CP1 is connected to virtual link VL1, which in turn is connected to the net_mgmt management network (192.168.120.0/24) of the computing cloud. CP2 is related to the virtual link VL2 that is connected to the net1 network (10.10.1.0/24), while CP3 is related to the virtual link VL3 that is connected to the net2 network (10.10.2.0/24). Both networks, net1 and net2 are networks created in the cloud using the Neutron engine.

The objective of this service is the connection between two virtual machines that are in different network segments, so that these machines are created, one in the net1 network (test1) and the other in net2 (test2). Both machines are created with the CirrOS image. On these machines, the routing table is configured so that the default gateway is the port of the vRouter that belongs to each network.

The vRouter and both test virtual machines have been added to the general topology of the project. The topology that brings together the 3 VNF created can be observed in Fig. 6.

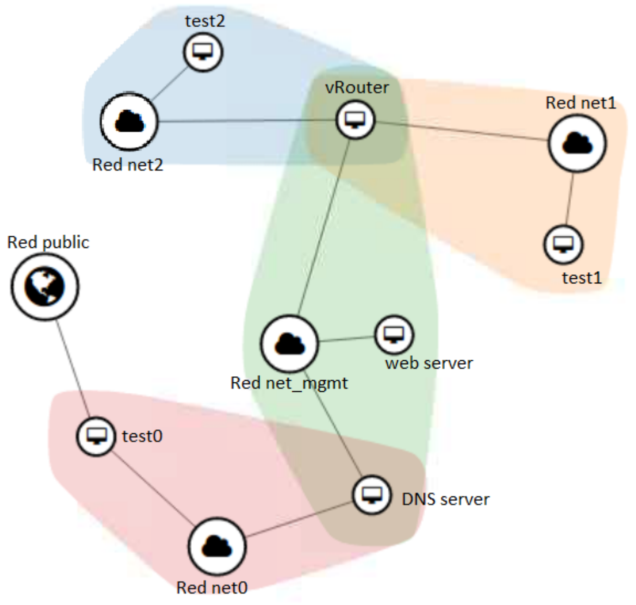


Fig. 6 Topology with 3 VNFs

D. *VNF – vRouter, DNS and Firewall*

Along with the virtualization of network functions, there is an additional benefit, which is the use of the same resource to provide several services at the same time. OpenWRT allows the configuration within the operating system for several network functions, this is how an instance that has several services at the same time, such as the routing, firewall and DNS service, could be created. This VNF is created in another project within Openstack, so it will be seen in a topology separating the previous ones, and can be seen in Fig. 7.

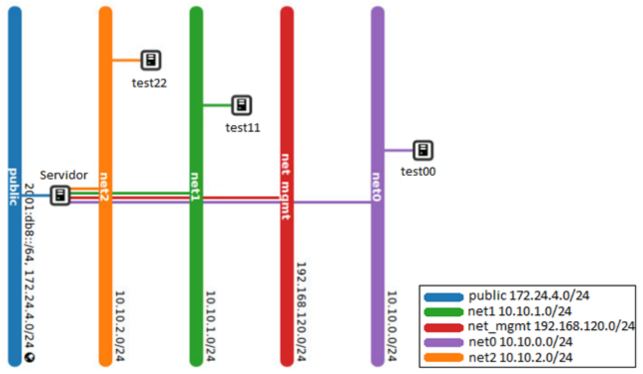


Fig. 7 Topology of the server: vRouter, DNS and Firewall.

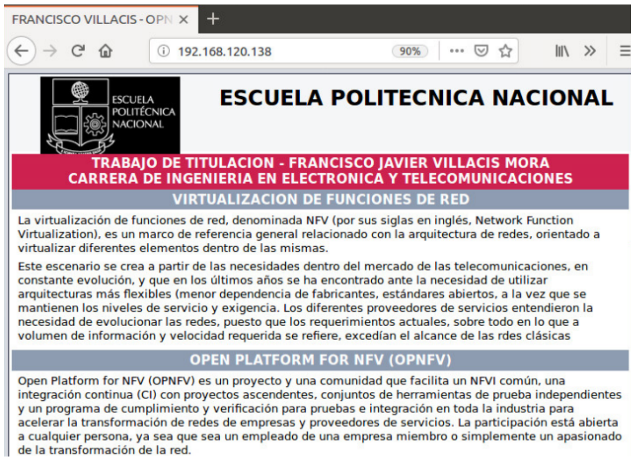
The VNFD for this VNF will create a service with a VDU1 instance based on the OpenWRT image and a flavor that includes 1024MB of RAM, 10GB of storage and 1 vCPU. A monitoring policy will be implemented in which pings will be carried out periodically and if these pings fail the failure will be reported to the tacker.log file. This instance will be related to connection points CP1, CP2, CP3, CP4 and CP5. CP1 is connected to virtual link VL1, which in turn is connected to the net_mgmt management network (192.168.120.0/24) of the computing cloud. CP2 is related to the virtual link VL2 that is connected to the net0 network (10.10.0.0/24); CP3 is related to the virtual link VL3 that is connected to the net1 network (10.10.1.0/24); CP4 connects to VL4 which is part of the net2 network (10.10.2.0/24); Finally, CP5 will be the server's connection to the internet, that is, it is connected to the public network (172.24.4.0/24).

V. Results

This section describes the final service presented by each VNF created, alongside with the benefits of the technology NFV for each topology.

A. *Virtual web server operation*

For the web server, a previous configuration was made, when creating the web-server-ubuntu image with the Apache2 program, so that a web page with basic project information is shown, as an example. The IP access to the web server is 192.168.120.138. Thus, when accessing it from the mother virtual machine, the service is deployed correctly, as shown in Fig. 8.


FRANCISCO VILLACIS - OPN

192.168.120.138

ESCUELA POLITECNICA NACIONAL

TRABAJO DE TITULACION - FRANCISCO JAVIER VILLACIS MORA
CARRERA DE INGENIERIA EN ELECTRONICA Y TELECOMUNICACIONES

VIRTUALIZACION DE FUNCIONES DE RED

La virtualización de funciones de red, denominada NFV (por sus siglas en inglés, Network Function Virtualization), es un marco de referencia general relacionado con la arquitectura de redes, orientado a virtualizar diferentes elementos dentro de las mismas.

Este escenario se crea a partir de las necesidades dentro del mercado de las telecomunicaciones, en constante evolución, y que en los últimos años se ha encontrado ante la necesidad de utilizar arquitecturas más flexibles (menor dependencia de fabricantes, estándares abiertos, a la vez que se mantienen los niveles de servicio y exigencia. Los diferentes proveedores de servicios entendieron la necesidad de evolucionar las redes, puesto que los requerimientos actuales, sobre todo en lo que a volumen de información y velocidad requerida se refiere, excedían el alcance de las rdes clásicas

OPEN PLATFORM FOR NFV (OPNFV)

Open Platform for NFV (OPNFV) es un proyecto y una comunidad que facilita un NFVI común, una integración continua (CI) con proyectos ascendentes, conjuntos de herramientas de prueba independientes y un programa de cumplimiento y verificación para pruebas e integración en toda la industria para acelerar la transformación de redes de empresas y proveedores de servicios. La participación está abierta a cualquier persona, ya sea que sea un empleado de una empresa miembro o simplemente un apasionado de la transformación de la red.

Fig. 8 Web service correctly deployed

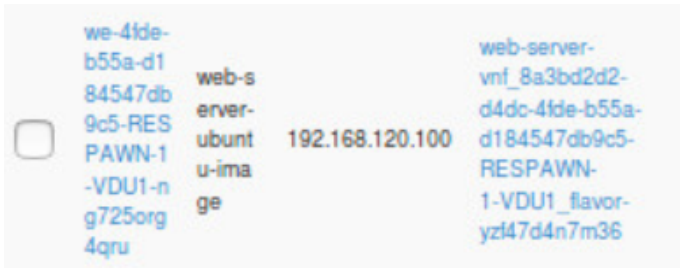
In the tacker.log file that was created to show the different events that are carried out in the VNFM, it can be seen that the monitoring planned for the service is executed. This monitoring is a ping on the http port of the instance. When pinging, you have a positive response message "driver_return True", which means that the instance responds correctly and is running.

The virtualized nature of the created instances allows their removal from the network when the administrator believes it is convenient. For example, if the web server is no longer necessary, it can be removed with the "Delete Instances" option.

To test the correct implementation of the monitoring policy for the web server, which planned a respawn in case of failure, the Apache program of the instance is stopped with the /etc/init.d.Apache2 stop command. Tacker was programmed to automatically perform the respawn, so the previously created instance is immediately removed and a new one is launched. Fig. 9 shows the new instance, which carries the RESPAWN-1 message in its name, which means that it is the first time Tacker had to perform this procedure on the service. The new IP address 192.168.120.100 with which it is built is also shown. If it is desired that the instance created after the respawn has the same IP address as the original instance, in the VNFD file it must be configured that the network port has a fixed IP.

B. Virtual DNS operation

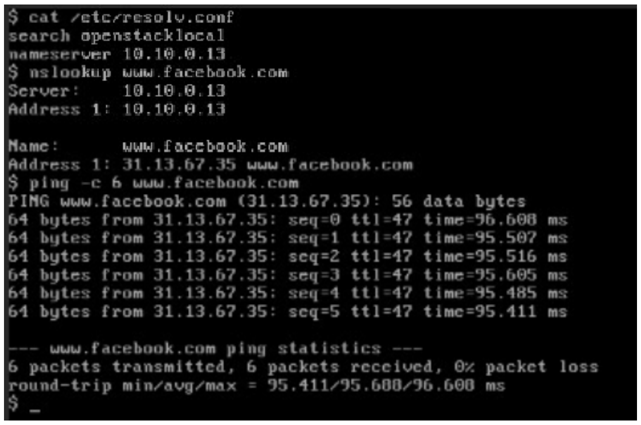
The instance created for this service uses the OpenWRT operating system, which allows the DNS configuration. Within the file /etc/config/dhcp the network domains to be translated are added. To demonstrate the operation in this project, a domain "www.facebook.com" has been added, which is a public address on the Internet. Once you have manually added the domain, you must execute the command /etc/init.d/dnsmasq reload on the instance so that the DNS server function is updated.



☐	we-4fde-b55a-d184547db9c5-RESPAWN-1-VDU1-ng725org4qru	web-server-ubuntu-image	192.168.120.100	web-server-vnf_8a3bd2d2-d4dc-4fde-b55a-d184547db9c5-RESPAWN-1-VDU1_flavor-yzf47d4n7m36

Fig. 9 Web server RESPAWN-1

Within the virtual machine test0, which allows to observe the correct operation of the translation of network domains to IP addresses, the command cat /etc/resolv.conf is executed, which shows that the IP address of the DNS server 10.10.0.13 is set as default DNS. The nslookup command shows a diagnosis of the proper functioning of the DNS server. The final test is performed with a ping to the domain name, it is clearly shown how it is translated to the IP address, which is on the Internet, and connection response is received. This process is shown in Fig. 10.



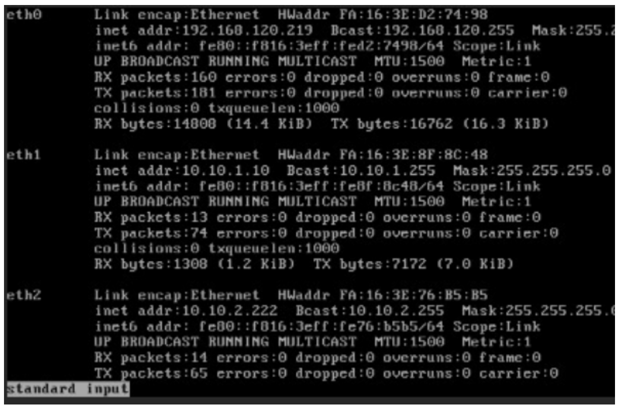
```
$ cat /etc/resolv.conf
search openstacklocal
nameserver 10.10.0.13
$ nslookup www.facebook.com
Server:    10.10.0.13
Address 1: 10.10.0.13

Name:      www.facebook.com
Address 1: 31.13.67.35 www.facebook.com
$ ping -c 6 www.facebook.com
PING www.facebook.com (31.13.67.35): 56 data bytes
64 bytes from 31.13.67.35: seq=0 ttl=47 time=96.608 ms
64 bytes from 31.13.67.35: seq=1 ttl=47 time=95.507 ms
64 bytes from 31.13.67.35: seq=2 ttl=47 time=95.516 ms
64 bytes from 31.13.67.35: seq=3 ttl=47 time=95.605 ms
64 bytes from 31.13.67.35: seq=4 ttl=47 time=95.485 ms
64 bytes from 31.13.67.35: seq=5 ttl=47 time=95.411 ms

--- www.facebook.com ping statistics ---
6 packets transmitted, 6 packets received, 0% packet loss
round-trip min/avg/max = 95.411/95.688/96.608 ms
$ _
```

Fig.10 DNS service correctly deployed

C. *vRouter operation*

Fig. 11 shows the interfaces correctly configured in the instance that will provide the routing service. The eth0 interface is connected to the management network, eth1 is connected to the net1 network with IP 10.10.1.10, while eth2 is connected to the net2 network with IP 10.10.2.222.

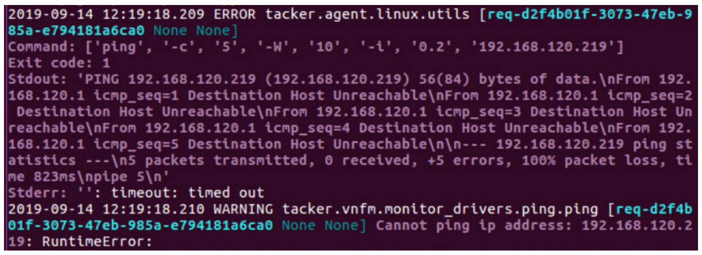
After configuring each virtual machine on the default gateway, it is possible to clearly demonstrate the connection between them. The test1 machine of the net1 network presents the IP 10.10.1.140 and the test2 machine of the net2 network presents the IP 10.10.1.246.



```
eth0      Link encap:Ethernet  HWaddr FA:16:3E:D2:74:98
          inet addr:192.168.120.219  Bcast:192.168.120.255  Mask:255.2
          inet6 addr: fe80::f816:3eff:fed2:7498/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:160 errors:0 dropped:0 overruns:0 frame:0
          TX packets:181 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:14808 (14.4 KiB)  TX bytes:16762 (16.3 KiB)

eth1      Link encap:Ethernet  HWaddr FA:16:3E:8F:8C:48
          inet addr:10.10.1.10  Bcast:10.10.1.255  Mask:255.255.255.0
          inet6 addr: fe80::f816:3eff:fe8f:8c48/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:13 errors:0 dropped:0 overruns:0 frame:0
          TX packets:74 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1308 (1.2 KiB)  TX bytes:7172 (7.0 KiB)

eth2      Link encap:Ethernet  HWaddr FA:16:3E:76:B5:B5
          inet addr:10.10.2.222  Bcast:10.10.2.255  Mask:255.255.255.
          inet6 addr: fe80::f816:3eff:fe76:b5b5/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:14 errors:0 dropped:0 overruns:0 frame:0
          TX packets:65 errors:0 dropped:0 overruns:0 carrier:0
standard input
```

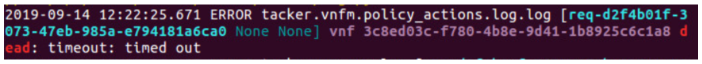
Fig.11 vRouter interfaces

The monitoring policy for this service is evidence of the benefit of NFV over virtualized services. In the tacker.log file we can see a periodic ping task which automatically checks the VNF connection. In order to demonstrate this configuration, the routing service is stopped on the vRouter. Immediately it can be seen in the log that the pings have an error message "cannot ping address 192.168.120.219" as shown in Figure 12. After a configured number of retries, finally, the "VNF dead" message appears in the tacker.log file. This can be seen in Figure 13.



```
2019-09-14 12:19:18.209 ERROR tacker.agent.linux.utils [req-d2f4b01f-3073-47eb-9
85a-e794181a6ca0 None None]
Command: ['ping', '-c', '5', '-W', '10', '-i', '0.2', '192.168.120.219']
Exit code: 1
Stdout: 'PING 192.168.120.219 (192.168.120.219) 56(84) bytes of data.\nFrom 192.
168.120.1 icmp_seq=1 Destination Host Unreachable\nFrom 192.168.120.1 icmp_seq=2
 Destination Host Unreachable\nFrom 192.168.120.1 icmp_seq=3 Destination Host Un
reachable\nFrom 192.168.120.1 icmp_seq=4 Destination Host Unreachable\nFrom 192.
168.120.1 icmp_seq=5 Destination Host Unreachable\n\n--- 192.168.120.219 ping st
atistics ---\n5 packets transmitted, 0 received, +5 errors, 100% packet loss, ti
me 823ms\npipe 5\n'
Stderr: '': timeout: timed out
2019-09-14 12:19:18.210 WARNING tacker.vnfm.monitor_drivers.ping.ping [req-d2f4b
01f-3073-47eb-985a-e794181a6ca0 None None] Cannot ping ip address: 192.168.120.2
19: RuntimeError:
```

Fig.12 VNF monitoring: vRouter



```
2019-09-14 12:22:25.671 ERROR tacker.vnfm.policy_actions.log.log [req-d2f4b01f-3
073-47eb-985a-e794181a6ca0 None None] vnf 3c8ed03c-f780-4b8e-9d41-1b8925c6c1a8 d
ead: timeout: timed out
```

Fig.13 VNF dead message: vRouter

D. vRouter, DNS and firewall operation

In the file /etc/config/firewall, the server presents the input, output, and forward options in ACCEPT status, that is, for all networks, inbound, outbound traffic and packet forwarding by the firewall is allowed. If you would like to restrict any of these options, the status should be changed to REJECT [15].

The rules of interaction between two specific zones are also configured, this is how you can specify the configuration for the forwarding of packets between each of the LAN networks with the WAN network.

The last service that is configured on the server is DNS. Domain configuration is done in the file /etc/config/dhcp. In order to demonstrate the correct functioning of the service, 3 domain names equivalent to the three test instances (test00, test11 and test22) and two domains found on the Internet are added.

With the configuration made on the server, it is possible to perform the tests in the instances of each network. The connection between LAN networks, the connection to the external network, the translation of the domain names and the operation of the firewall rules are tested.

VI. Conclusions

The principles and advantages of NFV were evidenced by the correct implementation and testing of virtualized network services, these being a web server, DNS, vRouter and a multifunction server (vRouter, DNS and firewall), along with different utilities, such as monitoring and scaling policies.

Through a theoretical study, the introduction of the fundamental concepts of NFV was carried out, as well as its origin, benefits, and scope. These concepts were used during the practical methodology of this project.

A general introduction of OPNFV was made and the components that allowed the implementation of Virtualized Network Functions on a computing cloud were chosen. These components are the virtualized infrastructure manager (VIM) Openstack and the administrator and orchestrator of VNF (MANO) Tacker.

The correct installation of the Openstack computing cloud was performed using the Devstack tool and the official documentation of said project. This installation is useful not only for projects focused on NFV but for any implementation and development that you want to perform on a cloud platform.

The commissioning of a virtualized multifunction server that includes routing, DNS and firewall services was carried out, demonstrating its correct implementation by connecting several LAN networks segments between them and the Internet, translating domain names and the implementation of firewall rules. This server proves the benefit NFV has of grouping different functions in the same resource.

The entire project was carried out on open software platform, in this way the possibility of creating a functional network with virtualized services was demonstrated, and at the same time independent of proprietary brand equipment and software.

A network was implemented with services that do not have oversize, that is, the virtual machines where the services are hosted are designed to provide only the functionality for which they are configured. At the same time, VNFs are shown as flexible services, which can be configured (expanded, deleted, etc.) easily if there are future expansion or improvement requirements.

By using two of its components, it was possible to expose OPNFV as a truly useful tool for the implementation of networks that intend to use NFV technology. The OPNFV ecosystem is really a solution when designing and implementing a network with NFV to meet the needs of the end user.

NFV allows the creation and elimination of as many virtual instances as necessary, as long as it does not exceed the use of physical resources to availability.

References

[1] ETSI, «Network Functions Virtualisation – Introductory White Paper» 22 10 2012. [Online]. Available: https://portal.etsi.org/NFV/NFV_White_Paper.pdf. [Accessed on: 8 24 2019].

[2] S. Arora, NFV Orchestration using OpenStack, University of Victoria: Department of Computer Science, 2017

[3] R. Chayapati, S. F. Hassan y P. Shah, Network Function Virtualization (NFV) with a touch of SDN, USA: Addison-Wesley, 2017.

[4] A. Tong y K. Wade, NFV and SDN Guide for Carriers and Service Providers, Hanover: Ciena Corporation, Blue Planet Essentials, 2017.

[5] M. Beierl, «OPNFV,» 13 11 2018. [Online]. Available: https://wiki.opnfv.org/. [Accessed on: 2 9 2019].

[6] OPNFV, «Our Mission,» 2018. [Online]. Available: https://www.opnfv.org/about/mission. [Accessed on: 2 9 2019].

[7] OPNFV, «Technical Overview,» 2018. [Online]. Available: https://www.opnfv.org/software/technical-overview. [Accessed on: 3 9 2019].

[8] A. Kapadia, «NFV Acceleration: An Introduction to OPNFV» edX, [Online]. Available: https://www.edx.org/course/an-introduction-to-opnfv. [Accesed on: 25 8 2019].

[9] Red Hat, Inc., «El concepto de cloud computing.,» 2019. [Online]. Available: https://www.redhat.com/es/topics/cloud. [Accessed on: 4 9 2019].

[10]Openstack Documentation, «Images and instances,» 23 8 2019. [Online]. Available: https://docs.openstack.org/ocata/admin-guide/compute-images-instances.html. [Accessed on: 9 10 2019].

[11]Edureka, «What Is OpenStack | OpenStack Tutorial For Beginners | OpenStack Training | Edureka,» 5 4 2017. [Online]. Available: https://www.youtube.com/watch?v=Kfj5XiNdJN0. [Accessed on: 4 9 2019].

[12]M. Ersue, ETSI NFV Management and Orchestration - An Overview, Vancouver, Canada: IETF #88, ETSI NFV MANO WG Co-chair.

[13]Openstack Documentation, «Tacker,» [Online]. Available: https://wiki.openstack.org/wiki/Tacker. [Accessed on: 5 9 2019].

[14]D. O. Briain, D. Denieffe, Y. Kavanagh y D. Okello, A report on the status of Network Functions Virtualisation, Carlow, Ireland: Department of Computing & Networking, Institute of Technology, 2017.

[15]OpenWRT Project, «Firewall configuration /etc/config/firewall» 14 9 2019. [Online]. Available: https://openwrt.org/docs/guide-user/firewall/firewall_configuration. [Accessed on: 15 9 2019].