# A report on the status of Network Functions Virtualisation

Diarmuid Ó Briain*†, David Denieffe*, Yvonne Kavanagh* and Dorothy Okello†
*GameCORE Research Centre,
Department of Computing & Networking,
Institute of Technology, Carlow, Ireland
Email: diarmuid.obriain@itcarlow.ie
†netLabs!UG Research Centre,
Department of Electrical and Computer Engineering,
College of Engineering, Design, Art and Technology,
Makerere University, Kampala, Uganda

***Abstract*—Cloud computing has been a disruptive force in computing over the last decade. The virtualisation computing paradigm developed in the 1990s, describes a single physical server that could host many Virtual Machines (VM) hosting guest Operating Systems (OS). Amazon Web Services (AWS) saw the opportunity to develop a compute resource on a pay-per-use basis and developed a product called Elastic Compute. Cloud services developed into classes, Infrastructure, Platform and Software as a Service. Enterprise and the consumer suddenly had choice not seen before, centralised across the Internet.**

**The traditional Service Providers (SP) were relegated to the role of bit stream delivery providers while Application Service Providers (ASP) developed centralised Over the Top (OtT) services on the SP networks. SPs found they could not compete with the scale of the globalised ASPs with their own application offerings. Things are changing however, SPs tired of paying for expensive networking hardware while ASPs benefit from cheap Elastic Compute and Storage launched an initiative to develop Network Functions Virtualisation (NFV). The idea is to develop Elastic Network and migrate away from expensive hardware based solutions, in a similar way to that being employed by the Data Centres with Software Defined Networking (SDN).**

**For Ugandan and other developing countries NFV presents a challenge and an opportunity, the challenge, to do nothing and be left behind as against the opportunity to embrace the change and use it to leap ahead and put in place the next generation of Internet.**

**There are three main projects, the Open NFV (OPNFV) project, hosted by the Linux Foundation that uses OpenStack as an upstream project for Virtual Infrastructure Manager (VIM). However in April 2016 with its Mitaka release, OpenStack introduced a new service called Tacker to provide the remaining Management and Network Orchestration (MANO) elements. Two other projects launched in 2016, one has its origins in the laboratories of Telefónica in Spain and now hosted by the European Telecommunications Standards Institute (ETSI) called Open Source MANO (OSM) and a second driven initially by the Chinese companies China Mobile and Huawei, hosted by the Linux Foundation originally called Open Orchestrator Project (OPEN-O) but recently merged with AT&T open source Enhanced Control, Orchestration, Management and Policy (ECOMP) on the 23 February 2017 to create the Open Network Automation Platform (ONAP) Project.**

**This paper describes the NFV concept, current MANO projects and considers OpenStack as a means to deliver it.**

## I. Network Functions Virtualisation

In October 2012 a group of SPs launched an initiative called NFV at the SDN & OpenFlow (OF) World Congress in Darmstadt, Germany (Chiosi, Clarke, Willis, Reid, Feger, Bugenhagen, Khan, Fargano, Cui & Denf, 2012). Having seen the impact of cloud computing, how new ASPs like Skype, Whatsapp and Facebook Messenger were selling OtT services in direct competition to their traditional voice and Short Message Service (SMS) services. In particular SPs experienced major falls in their once lucrative international roaming services (Heinrich, 2014). These same carriers have invested heavily in their network infrastructure, installing fibre networks while ASPs exploit their networks without associated costs due in part to Net Neutrality (A guide to the Open Internet, n.d.). NFV essentially virtualises

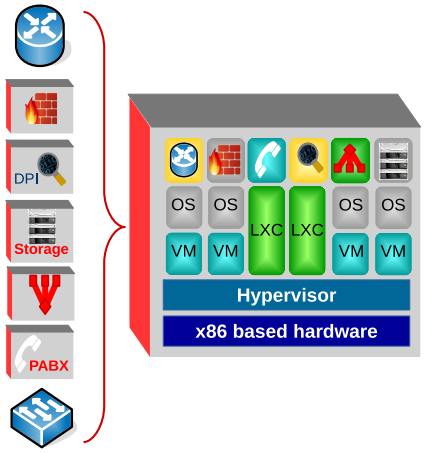


Figure 1. NFV Concept

services on a hypervisor running on Commercial Off The

Shelf (COTS) hardware called a virtual Customer Premise Equipment (vCPE) as demonstrated in Figure 1. Managing the hypervisor using IT cloud orchestration tools would allow new services to be established and torn down on the same device without the need for expensive site visits by technicians. High quality broadband links from the data centre to an vCPE at the customer premises means the SP is poised to maximise the benefit of investment by migrating services from physical devices to virtual functions.

The SPs formed an Industry Specification Group (ISG) under the ETSI. The initial phase of work for the group involved the development of applicable frameworks and standards (ETSI, 2012). Progress White Papers were issued in October 2013 (ETSI, 2013a) and October 2014 (ETSI, 2014). In January 2015 with the release of ETSI NFV ISG (ETSI, n.d.) documents the focus of the group switched from requirements to NFV adoption.

The NFV concept provides the possibility of two other new models of service delivery. Functions can be virtualised on Compute nodes in the SPs own data centre, called virtual Provider Edge (vPE) or the SP can use the customers own hypervisor to run functions in a model known as virtual Customer Edge (vCE). The vCE model could be particularly useful for a Global Service Provider (GSP) offering services on a partner Local Service Provider (LSP) network (Weldon, 2015).

Since the inception of NFV, both SDN and NFV have been seen to compliment each other without necessarily being inter-dependent. Having said that it is becoming increasingly evident that there is an inter-dependency and recent industry discussion of the term *Software Defined NFV* (SDNFV) (Pitt, 2016) underlines it. New NFV functional services which link multiple functions that may or may not be in the same location are being developed that require the flexibility and control of SDN to chain them.

NFV presents a number of significant challenges yet to be resolved. Issues like the performance trade-off between specialist hardware and virtualised functions on COTS hardware, the new network models provide new information security attack vectors. However these challenges are more than outweighed by the strength of the NFV case. The potential benefits are huge and major SPs normally associated with cautious, long-term planning and roll-out are jumping to NFV models even before the technological solutions are fully developed (Verizon, 2016). The carrot of increased speed to market as well as new services delivered on virtualised infrastructure is quite compelling and provide the motivation for development.

In September 2016 ETSI announced the second release of NFV (Dahmen-Lhuissier, 2016). This version introduced 11 new group specifications to detail functional requirements, interface descriptions and information models to define the management of virtualised resources as well as lifecycle management of network services and VNFs, fault/performance management of network services and capacity management of virtualised resources.

### A. *The NFV ecosystem*

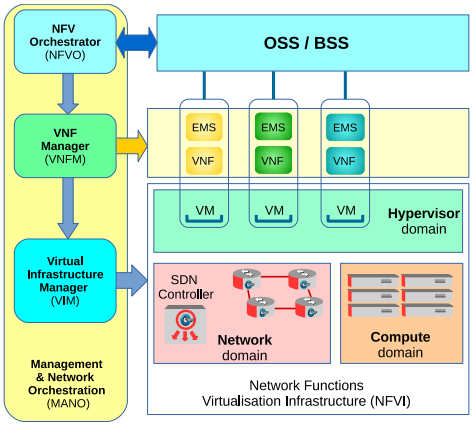


Figure 2. NFV Ecosystem.

Figure 2 outlines the overall NFV ecosystem (ETSI, 2013b). At its foundation is the *NFVI* which supports the Compute, Network and Hypervisor/Virtualisation domains. The Compute domain consists of the computer and storage hardware supporting the hypervisors. Virtualisation on these platforms is provided by existing hypervisor technologies like KVM, Xen, VMWare or a container technology like Docker. While networking can be provided utilising traditional switching and routing technologies it is becoming evident that this domain will become the reserve of SDN. *VIMs* are used to manage each domain within the NFVI.

The *NFV Orchestrator* (NFVO) orchestrates through the *Virtual Network Function Manager* (VNFM). The VNFM is responsible for the lifecycle, the Fault, Configuration, Accounting, Performance and Security Management (FCAP) of VMs and their *Virtual Network Functions* (VNF). The VNFM carries out these functions by instructing the relevant domain VIM. The associated ETSI standards are still in development and various vendor MANO solutions existing today are still unclear on the specific dividing lines between these functions. It is expected that the standards will harmonise in the medium term.

## II. OpenStack

*OpenStack* started in 2010, as a joint project of Rackspace Hosting and National Aeronautics and Space Administration (NASA). NASA contributed their Nebula platform, which later developed into Nova. Rackspace contributed their Cloud Files platform, which later became Swift. In April of 2011, the OpenStack *Bexar* release was introduced in Ubuntu. Later that same year, Debian GNU/Linux included OpenStack *Cactus*

in their distribution. In 2012, Redhat announced a preview of their OpenStack distribution. Since then, many others followed, including Oracle, HP, and Dell Vmware. Figure 3

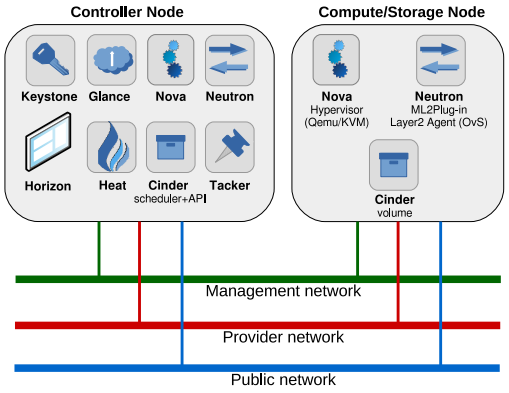


Figure 3. OpenStack.

shows a basic setup of OpenStack. At the ecosystem core is the Controller node, this hosts the major OpenStack services. Compute and Storage nodes can be connected to the Controller node such that capacity can be added to the system as required. On Compute nodes a Nova agent interfaces with the hypervisor and can be directed by the central Nova service while on Storage nodes the Cinder service can access volumes via the Cinder volume agent. Networking is achieved by the Neutron service on the Controller node managing layer 2 agents like Open vSwitch (OvS) or using the Modular Layer 2 (ML2) plugin to control SDN Controllers or legacy networking hardware. The OpenStack ecosystem has a number of networks. The *Public network* permits management access to the nodes as well as an access point to the Internet. The management network is a *Private network* used for intercommunication between the various OpenStack services while the *Provider network* is used as a public network for VMs and Containers (CT) instances.

### A. *Role of the OpenStack Foundation*

The OpenStack Foundation promotes the global development, distribution, and adoption of the cloud OS. It provides shared resources to grow the OpenStack cloud. It also enables technology vendors and developers to assist in the production of cloud software. (OpenStack Foundation, n.d.).

### B. *The structure of OpenStack*

OpenStack is a framework encompassing multiple services. There are a number of 'Core' services and a substantial number of 'Other' services that allows an organisation build for many configurations.

*1) **Nova 'Compute' Service**:* Nova is the central core project in OpenStack as it handles the Compute environment, the VM instance lifecycle. It implements VMs on a number of different hypervisors like Xen, KVM/Quick Emulator (QEMU), VMware vSphere. Nova installs an agent on the Compute node to manage the hypervisor. Nova spawns, schedules, and decommissions of VMs on demand.

*2) **Neutron 'Networking' Service**:* OpenStack Neutron is the OpenStack SDN enabler. It seperates the physical underlay network from logical VM instance overlay networks. In fact project users need not be aware of the underlay network topology. Such users can see an abstraction network at a higher level and SDN permits the project user to create logical networks that do not require the consideration of the underlying physical network. Neutron provides an ML2 Plug-in, a layer 2 agent to allow interfacing with the physical network architecture using a pluggable architecture that supports many networking vendors and technologies.

*3) **Cinder 'Block Storage' service**:* Cinder block storage provides persistent storage to instances. By default, the storage in the VM instances is ephemeral, non-persistent. Cinder allows for the attachment of persistent block devices to instances such that data can be saved. The Cinder scheduler and API on the Controller node interact with the Cinder volumes on Storage nodes to allow for functionality such as create volume, delete volume and attach.

*4) **Swift 'Object Storage' service**:* OpenStack Swift service provides an object-based storage model. This provides scalability where the Cinder 'Block Storage' service interacts with Swift 'Object Storage' service over a RESTful API which in turn can communicate with many, storage nodes. Swift uses a proxy service which, when it receives data from Cinder, creates chunks of data called binary objects.
Swift includes a replication algorithm which stores the binary objects on multiple storage nodes. Efficiency is also achieved because, the moment an application needs to retrieve the data, it will address the Swift proxy via the Cinder service which uses an advanced algorithm to determine exactly where the binary objects reside. It then sends calls to all the storage nodes that are involved, these are capable of working in parallel. The data will arrive at the Swift proxy, and onwards to the application via Cinder quickly and efficiently.

*5) **Keystone 'Identity' service**:* The Keystone Identity service is a core element in OpenStack and is used to authenticate and authorise. It also lists all current services and endpoints. To access the Nova service for example it must be defined as a service within Keystone. The endpoint provides a Uniform Resource Locator (URL) that provides access to the specific service. In Keystone, Users and Roles are created and they are assigned to Projects. A Project typically represents a customer of OpenStack.

*6) **Glance 'Image store' service**:* The Glance Image store service is used to store VM disk images. VMs, which are the actual instances are not installed each time, instead they are spawned off from an image. Glance provides the image store.

If an administrator wants to boot an instance, then the instance will be booted from the Glance image store.

*7) **Other Services**:*

- **Horizon** - Provides the Dashboard, which is a web-based user interface to manage the OpenStack Service
- **Heat** - Provides a service to orchestrate composite cloud applications, using a declarative template format through an OpenStack-native REST API
- **Ceilometer** - It is part of the Telemetry project and provides data collection services for billing and other purposes
- **Aodh** - Triggers actions based on defined rules against event data collected by Ceilometer
- **Monasca** - Multi-tenant Monitoring-as-a-Service (MON-aaS)
- **Trove** - Create and manage databases
- **Sahara** - provides a simple means to provision a data-intensive application cluster
- **Magnum** - provides for CT orchestration engines like Docker Swarm, Kubernetes and Apache Mesos. Magnum uses Heat to orchestrate an OS image which contains Docker and Kubernetes and runs that image in either VMs or bare metal in a cluster configuration
- **Ironic** - a bare metal provisioning program and was developed to deploy physical machines (and not VMs)
- **Tacker** - deploy and operate Network Services and Virtual Network Functions (VNFs) on an OpenStack as an NFV infrastructure platform.

## III. Open Platform NFV

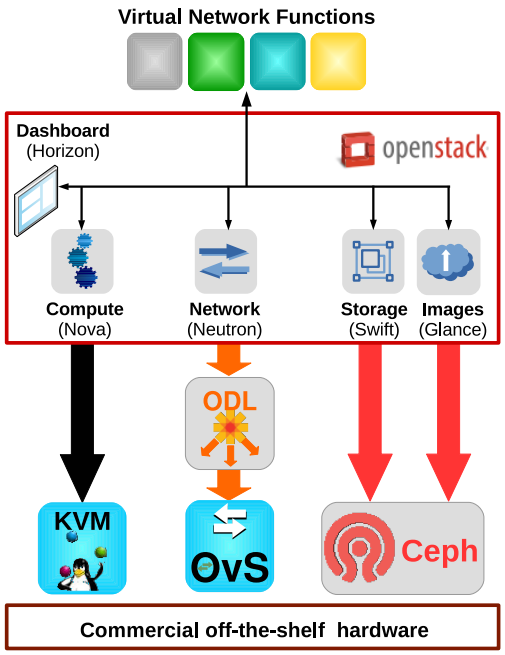


Figure 4. OPNFV example.

The Collaborative Project *OPNFV* was launched in October 2014 under the auspices of the Linux Foundation (Linux Foundation, 2014). Its intend is to make the ETSI NFV standards a reality through the provision of NFVI, VIM and open APIs to other NFV elements already in Free/Libre Open Source Software (FLOSS) projects to form an NFV architecture. The initial focus is the NFVI and VIM elements of the architecture and the project is achieving this through building interfaces between existing *upstream* FLOSS projects like OpenStack, Kernel Virtual Machine (KVM), Xen, LinuX Containers (LXC), OvS, Linux bridge, Data Plane Development Kit (DPDK), Open Dataplane (ODP) and the GNU/Linux kernel. Creating a functional reference platform in this manner will contribute to the goals of phase 2 of the ETSI NFV ISG.

A lab ready version of OPNFV was released in March 2016 called *Brahmaputra* (OPNFV, 2016) which was followed by the *Colorado* release in August 2016. Colorado while still a lab release doubled the number of scenarios supported. OPNFV has integrated upstream FLOSS projects to build the platform. In the Figure 4 example, OpenStack is seen as a major element, incorporated to provide MANO functionality. Nova provides Compute on a KVM hypervisor with Neutron interfacing over ML2 to the ODL RESTful API to control OvS switches. In this way OpenStack, KVM, ODL and OvS are considered upstream projects.

### A. *OpenStack Heat*

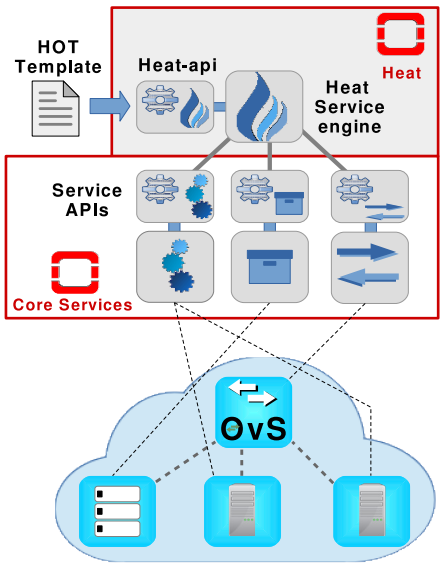


Figure 5. OpenStack Heat.

OpenStack has a specific *Orchestration* service (OpenStack Heat documentation, n.d.). While the Nova service can be used to create a VM instance or the Neutron service can

create networks there are many scenarios where there is a need to create an launch a group of VM instances, all of these VM instances need not be of the same type as the template can define individual VMs as required. These VM instances typically require a network and storage.
The *Heat* service provides an OpenStack orchestration engine that is designed to receive a *Heat Orchestration Template* (HOT) template (HOT Guide, n.d.) to launch multiple composite cloud applications. As demonstrated in Figure 5 the HOT template is applied to the Heat service via the *heat-api*, it is interpreted and the *heat-engine* creates jobs that are passed to the core services to create the cloud storage, network and VM instances as defined in the template. Heat has a second API called the *heat-api-cfn* which allows it to interpret AWS *CloudFormation* templates also.

HOT uses *YAML Ain't Markup Language* (YAML) which is an easily readable data serialisation language that is commonly used for configuration files (Ben-Kiki & Evans, 2009). In the first section of Figure 6 a simple HOT template in YAML can be seen which describes a single instance of Debian GNU/Linux. The *OS::Nova::Server* type is a resource for managing Nova instances, it is used to launch and manage the running VM instance. For more complex stacks with networks the *OS::Neutron::Net* type resource is used for creating and managing networks while the *OS::Cinder::Volume* type implements Cinder block storage devices.

The *openstack stack create* command interprets the

```
$ cat hot_example.yaml

heat_template_version: 2017-01-08

description: Example of single compute instance

resources:
  instance:
    type: OS::Nova::Server
    properties:
      flavor: m1.small
      image: Debian8.img
      key_name: instance_key
      networks:
        - network: private

$ openstack stack create --template hot_example.yaml Debian8_vm

$ openstack stack list
+--------------------------------------+------------+-----------------+----------------------+--------------+
| ID                                   | Stack Name | Stack Status    | Creation Time        | Updated Time |
+--------------------------------------+------------+-----------------+----------------------+--------------+
| d2009539-3ef9-4c04-aeb0-ee7569a124d5 | Debian8_vm | CREATE_COMPLETE | 2017-01-08T14:23:21 | None         |
+--------------------------------------+------------+-----------------+----------------------+--------------+
```

Figure 6. HOT Template.

HOT template and orchestrates the stack by calling on the *OS::Nova::Server* resource, in other words the Nova API. Finally when the stack instance(s) are launched the *openstack stack list* command will show a status of *CREATE_COMPLETE*. At this stage the VM instances are running and active.

### B. OpenStack Tacker

OpenStack has launched a project of its own called *Tacker* (Tacker Documentation, n.d.). The objective of Tacker is to produce a generic VNFM and NFVO which will make it easy to deploy and operate VNFs with OpenStack acting as the MANO. The Tacker API will be used to deploy VNFs either on remote customer networks as a vCE, on a SP provided vCPE or on the SP's own vPE infrastructure. This will form an important cog in the OPNFV architecture.

*1) **VNF Descriptor**:* VNFs are described in a VNF Descriptor (VNFD) file that is based on Organisation for the Advancement of Structured Information Standards (OASIS) Topology and Orchestration Specification for Cloud Applications (TOSCA) standard (TOSCA, n.d.). TOSCA is also a YAML formatted file. VNFs are described in the VNFD as three basic node types;

- *Virtual Deployment Unit (VDU)*:
  - describes the VM instance that hosts the VNF, for example the image to be used and the flavour, monitoring policies. etc.. *(tosca.nodes.nfv.VDU)*. It equates to Nova servers in OpenStack.
- *Connection point (CP)*:
  - describes the virtual NIC or Single Root I/O Virtualisation (SR-IOV) NIC that is virtually bound to a VDU. *(tosca.nodes.nfv.CP)*. It equates to Neutron ports in OpenStack.
- *Virtual Links (VL)*:
  - describes the logical virtual link entity that provides connectivity between VDUs. *(tosca.nodes.nfv.VL)*. It equates to Nova networks in OpenStack.

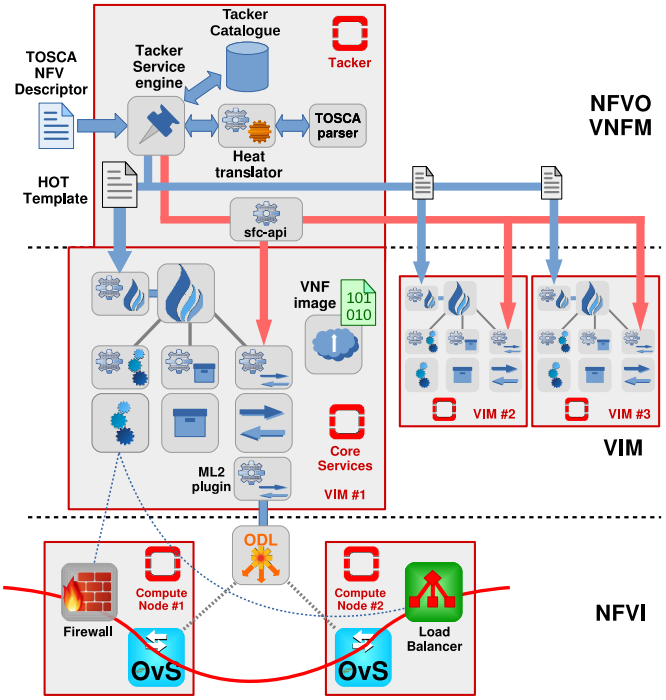


Figure 7. Tacker as NFVO/VNFM.

The diagram in Figure 7 outlines the complete NFV ecosystem. OpenStack core services act as VIM as defined in the ETSI NFV specifications managing the NFVI in the form of

OpenStack Compute Nodes, OpenDaylight SDN Controllers and legacy networking hardware as well as whitebox OF switches. The OpenStack Tacker service fulfils the role of NFVO and VNFM from the ETSI NFV specifications. It can orchestrate and manage multi-site VIMs to give the operator a single view for centralised management. In fact it can also act as NFVO/VNFM to a number of VIM type options:

- *OpenStack*
- *VMware ESXi*
- *AWS*

This means OpenStack Tacker can orchestrate with either AWS as a VIM using AWS NFVI or a VMWare ESXi as a VIM.

*2) **VNF setup**:* Before Tacker can have a virtual environment orchestrated it received a VNFD file. This file is sent to the *heat-translator* which uses the *TOSCA parser* to translate TOSCA to HOT format.
The Tacker engine stores the HOT template in a *Tacker catalogue*. When the VNF is launched the *tacker engine* passes the HOT template to the *heat-api* on the VIM site where it is to be orchestrated. The Heat service then orchestrates the VM instances, storage and network resources as defined in the HOT template. These resources can be managed via the Tacker service.

*3) **Function chains**:* On the networking side Tacker can manage function chains through the *Service Function Chaining* (SFC) API. This API manages SDN Controllers through the Neutron service *ML2 plugin*. SFC is a capability that allows for control of network packet flows to select paths that may not be the one chosen by the routing table.
It facilitates managing the network policy of an SDN Controller to direct traffic from one service to another, for example as demonstrated in Figure 7, traffic of a particular type being directed to a *firewall service* and then to a *load balancing service* even though these services are not necessarily in the routing path to the final destination. This flow control is managed through the flow control tables on the OvS switches.

*4) **VNF example**:* Before Tacker can orchestrate a VNF a few items need to be in place, these are demonstrated in Figure 8. The VNF image must exist in the Glance store. The addition of a Debian8 image is demonstrated with the *openstack image create* command. The next step is the creation of a TOSCA formatted YAML VNFD file, in the example *vnfd-debian8.yaml*. This YAML file contains the basic configuration of the VNF with *tosca.nodes.nfv.VDU.Tacker* defining the VDU capabilities, number of Central Processing Units (CPU), memory, disk size and other configuration for the VNF as necessary. The *tosca.nodes.nfv.VL* type defines the network and *tosca.nodes.nfv.CP.Tacker* links the defined VDU to the defined network. This VNFD is then added to the Tacker Catalogue with the *tacker vnfd-create* command. Finally the VNF is deployed with the *tacker vnf-create* command.

```
$ openstack image create vnfd-debian8 --disk-format qcow2 \
                                      --container-format bare \
                                      --file /path_to_image/Debian8.img \
                                      --visibility public


$ openstack image list
+--------------------------------------+--------------+--------+
| ID                                   | Name         | Status |
+--------------------------------------+--------------+--------+
| 7a0e5bae-e3bf-11e6-bf01-fe55135034f3 | vnfd-debian8 | active |
+--------------------------------------+--------------+--------+

$ cat vnfd-debian8.yaml
tosca_definitions_version: tosca_example_Debian8_NFV_1_0
description: Debian8 as a VNF

metadata:
  template_name: debian8-tosca-vnfd

topology_template:
  node_templates:
    VDU1:
      type: tosca.nodes.nfv.VDU.Tacker
      capabilities:
        nfv_compute:
          properties:
            num_cpus: 1
            mem_size: 1024 MB
            disk_size: 2 GB
      properties:
        image: debian8.img
        availability_zone: nova
        mgmt_driver: noop
        config: |
          param0: key1
    CP1:
      type: tosca.nodes.nfv.CP.Tacker
      properties:
        management: true
        order: 0
        anti_spoofing_protection: false
      requirements:
        - virtualLink:
            node: VL1
        - virtualBinding:
            node: VDU1
    VL1:
      type: tosca.nodes.nfv.VL
      properties:
        network_name: net1
        vendor: Tacker

$ tacker vnfd-create --vnfd-file vnfd-debian8.yaml --name vnfd-debian8
Created a new vnfd:
+-----------------+--------------------------------------------------------------------------------------+
| Field           | Value                                                                                |
+-----------------+--------------------------------------------------------------------------------------+
| description     | Debian8 as a VNF                                                                     |
| id              | 3f0706ac-2f15-4730-8f4a-f9b76ee18d34                                                 |
| infra_driver    | heat                                                                                 |
| mgmt_driver     | noop                                                                                 |
| name            | vnfd-debian8                                                                         |
| service_types   | {"service_type": "vnfd", "id": "0e43f875-690e-43f3-ae44-9cd67820e570"}               |
| project_id      | 3e08bde4e95c47178b0f491ed163179b                                                     |
+-----------------+--------------------------------------------------------------------------------------+


$ tacker vnf-create --vnfd-id 3f0706ac-2f15-4730-8f4a-f9b76ee18d34  --name vnfd-debian8
Created a new vnf:
+----------------+--------------------------------------+
| Field | Value                                         |
+----------------+--------------------------------------+
| description    | Debian8 as a VNF                     |
| id             | fd720c37-82da-4014-b7b5-96b8cd9b2030 |
| instance_id    | 69691051-ffe8-4402-8503-b480c96a1826 |
| mgmt_url       |                                      |
| name           | vnfd-debian8                         |
| placement_attr | {"vim_name": "VIM0"}                 |
| status         | PENDING_CREATE                       |
| project_id     | 3e08bde4e95c47178b0f491ed163179b     |
| vim_id         | f58df21d-0df4-442a-93a6-d8157437316c |
| vnfd_id        | 3f0706ac-2f15-4730-8f4a-f9b76ee18d34 |
+----------------+--------------------------------------+
```

Figure 8. Tacker example.

## IV. VNF Monitoring and Fault Management

The ETSI specifications have clear expectations for scalable, available, fault tolerant VNF and state that a scalable NFVI supporting thousands of VMs must have a high degree of automation in failure situations (ETSI, 2015). It specifies a fault management interface and performance management interface for virtualised Resources. The fault management interface reports fault alarms such as VM crashes, networking errors, VM to storage volume disonnections from the VIM to the VNFM while the performance management interface reports on resource consumption level. To meet the high demands of the specification the VNFM must react to this information to automatically deal with faults in such a way that the service availability classification level is maintained (ETSI, 2016).

- **Level 1** - Network operator control, Government, Emergency services. May require 1+1 redundancy with instantaneous switchover

- **Level 2** - Network operators as customers, Enterprise. May require 1:1 redundancy with fast switchover
- **Level 3** - Consumer Internet traffic. Best effort traffic. Satisfactory levels of availability.

### A. *Tacker VNF fault management*

In the initial release, Tacker has minimal support for monitoring VNFs. It monitors the VNF by using *ping* to the IP address of the VNF. If the function is browser based *curl* could be used. For example the following simple curl example shows that the VNF httpd daemon is operational:

```
$ curl -Is <VNF> | grep HTTP
HTTP/1.1 200 OK
```

Thus by means of ping or curl driver it can be determined if a VNF becomes unreachable. What Tacker cannot currently do however is monitor things like CPU and memory usage of VNFs. Currently if an unreachable VNF is detected Tacker can perform one or both of the following actions:

- **respawn** - Recreation of the VNF
- **log** - Note the failure in OpenStack logs

OpenStack has a seperate Telemetry project called *Ceilometer* whose function it is to provide data collection services and *Aodh* a service that tiggers actions based on defined rules against event data collected by Ceilometer. An example of their implementation with Tacker would see a policy like *tosca.policies.tacker.Alarming* establishing monitoring in Ceilometer. An alarm trigger of type*tosca.events.resource.utilization* would occur in Aodh if utilisation was detected to have risen above a threshold for example 85%. This could then cause the an additional VDU to be spawned (Tacker documentation, n.d. b).

Work is ongoing in the OpenStack project to design a generic monitoring framework. An alarm monitor driver in Tacker could monitor events collected by Ceilometer (or even Monasca MONaaS) where it collects data directly from the VNF like CPU and memory usage. To make this happen it is necessary to add support for inserting Ceilometer alarms to trigger scaling in Heat resource groups using a HOT template, this does not exist today (Tacker Specs documentation, n.d.)).

## V. Other solutions

While the OPNFV project is using upstream projects like OpenStack as the means to rapidly develop the required MANO elements, it is not the only solution. Telefónica's NFV Reference Laboratory (Telefónica NFV Reference Lab, n.d.) has been working on a MANO project since 2014 and folded its work into the OSM project when it launched in 2016 giving it a good head start (Hoban, Sepulveda, de Blas, Kashalkar, Shuttleworth, Harpe, & Velandy, 2016). The OSM project has attracted lots of other SP members like Telekom Austria, British Telecom (BT), Korea Telecom, the Norwegian SP Telenor and Canonical the company behind Ubuntu Linux. Another new project is the Linux Foundation ONAP (ONAP, n.d.) which also launched in 2016 as OPEN-O. It launched with significant funding from the Chinese companies China Mobile and Huawei while it has been joined by mainly vendors like Ericsson, Intel and Brocade. Redhat the opensource solution developer is a counterweight on OPNAP project to Canonical at the OSM project.

### A. *Open Source MANO*

OSM is a project hosted by ETSI to deliver an opensource MANO stack under the Apache Public License 2.0. The first release of OSM was in October 2016 is a basic implementation not suitable for commercial deployment. It is built on Ubuntu Server and can orchestrate and manage three VIM types:

- *Open VIM*: This is an OSM sub-project.
- *OpenStack*: OSM can use OpenStack as a VIM.
- *VMware vCloud Director*.

Like OpenStack, OSM can act as MANO to non OSM VIMs, in this case OpenStack and VMWare vCloud Director.

### B. *Open Network Automation Platform*

ONAP is another Linux Foundation project that aims to enable SPs to deliver end-to-end services across an architecture compliant with ETSI NFV specifications. ONAP has a strong SDN element as the project sees NFV and SDN as both part of the solution. In ONAP, SDN is seen as the interconnector between VNFs to provide end-to-end services. It has also adopted the TOSCA standard as a VNFD language. ONAP has three main functions:

- *Global Services Orchestrator (GS-O)*: provides end-to-end orchestration of any service on any network.
- *Network Function Virtualisation Orchestrator (NFV-O)*: Service life cycle manager, NFV resource manager and NFV monitor. Using the south bound SBI, NFVO interworks with several NFV SDN controllers, Multi-vendor VNFMs and different VIMs.
  - *NBI*: RESTful API between GS-O and NFV-O.
  - *SBI*: RESTful API to SDN Controllers (ODL and ONOS), OpenStack or VMWare as a VIM.
- *SDN Orchestrator (SDN-O)*: provides network service orchestration over SDN and legacy networks.

## VI. Conclusion

NFV is still in the early stages of development. ETSI have provided the early specifications for the architecture and a number of projects have evolved to fulfil them. The OPNFV project under the Linux Foundation saw the potential of using existing, what it terms *upstream* projects like OpenStack, ODL, ONOS, KVM and OvS. It sees its role as a facilitator for the development and evolution of the upstream projects NFV components across various open source ecosystems.

It runs the *Pharos Community Laboratory* as a distributed NFV test laboratory for the deployment and testing of OPNFV software releases. OPNFV has focused on building NFVI and VIM elements of MANO. Like the first three releases, the fourth and latest *Danube* release, OpenStack is

the upstream project for the VIM function.

The OpenStack project has worked in parallel on a service to include NFVO and VNFM in the OpenStack ecosystem. The first version of it called *Tacker* was released as part of OpenStack *Mitaka* release in March 2016. This provides a working NFV stack that can be used for NFV deployment testing.

Additionally there are two other main projects in the NFV space, the ETSI hosted OSM project and the Linux Foundation project ONAP. These aim to develop SP *carrier quality* MANO to accelerate moving NFV from the laboratory and trial deployments to mainstream.

NFV presents SPs with an opportunity to reduce their reliance on specialist hardware and migrate customers to COTS hardware supporting VNFs while also laying the foundation for the next stage of Cloud computing development. To support the tactile Internet of the future where applications require extremely low latency and no jitter while certain Internet of Things (IoT) applications have zero tolerance to any latency (Curry, 2016) it is necessary to move the applications closer to the end-user. Centralised applications that have been the hallmark of cloud computing will have to change and NFV presents a model for the cloudlet concept to develop. GSPs can have cloudlet capacity close to the end-user provided by the SP. This presents SPs the opportunity to move back up the value chain.

From a Ugandan and developing world perspective it is essential that this and other Software Defined technologies are embraced. They present a major opportunity to skip a technology generation and prepare for the Internet of the future.

## REFERENCES


Chiosi, M., Clarke, D., Willis, P., Reid, A., Feger, J., Bugenhagen, M., Khan, W., Fargano, M., Cui, C., Denf, H. (2012). Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action. In SDN and OpenFlow World Congress (pp. 2224).

A guide to the Open Internet. (n.d.). Retrieved 14 July 2016, from http://www.theopeninter.net/

Heinrich, E. (2014, June 23). Telecom companies count $386 billion in lost revenue to Skype, WhatsApp, others. Fortune. Retrieved from http://fortune.com/2014/06/23/telecom-companies-count-386-billion-in-lost-revenue-to-skype-whatsapp-others/

ETSI, (2012, October 3). ISG NFV Proposal. Terms of Reference of the Industry Specification Group on NFV (ISG NFV) Retrieved from https://portal.etsi.org/nfv/ETSI%20ISG%20NFV%20ToR_v5_4_October_2012.pdf

ETSI, (2013a). Network Functions Virtualisation Network Operator Perspectives on Industry Progress - 2013. Updated White Paper, (1). Retrieved from https://portal.etsi.org/nfv/nfv_white_paper2.pdf

ETSI, (2014). Network Functions VirtualisationNetwork Operator Perspectives on Industry Progress - 2014. Updated White Paper, (1). Retrieved from https://portal.etsi.org/Portals/0/TBpages/NFV/Docs/NFV_White_Paper3.pdf

ETSI. (n.d.). Network Functions Virtualisation. Retrieved 5 December 2015, from http://www.etsi.org/technologies-clusters/technologies/nfv

Weldon, M. K. (2015). The Future X Network: A Bell Labs Perspective. CRC PressINC. Retrieved from https://books.google.ie/books?id=DZEZjgEACAAJ

Pitt, D. (2016, June 14). What Is SDNFV & Why Should You Use It? Retrieved 10 July 2016, from http://www.lightreading.com/nfv/nfv-mano/what-is-sdnfv-and-why-should-you-use-it/a/d-id/724090

SDN-NFV Reference Architecture. (2016, February 1). Verison. Retrieved from http://innovation.verizon.com/content/dam/vic/PDF/Verizon_SDN-NFV_Reference_Architecture.pdf

Dahmen-Lhuissier, S. (2016, September 27). ETSI brings virtualization of telecommunication networks closer with announcement of NFV Release 2. Retrieved 23 January 2017, from http://www.etsi.org/index.php/news-events/news/1128-2016-09-news-etsi-brings-virtualization-of-telecommunication-networks-closer-with-announcement-of-nfv-release-2

ETSI, (2013b). Network Functions Virtualisation (NFV); Use Cases. V1, 1, 201310.

Linux Foundation. (2014, October 30). OPNFV - OPNFV - An open platform to accelerate NFV. Retrieved from https://www.opnfv.org//sites/opnfv/files/pages/files/opnfv_-whitepaper_103014.pdf

OPNFV. (2017, April 13). OPNFV issues its fourth release, Danube — Open Platform for NFV (OPNFV). Retrieved 21 May 2017, from https://www.opnfv.org/community/2017/04/13/opnfv-issues-its-fourth-release-danube

OpenStack Heat documentation. (n.d.). Retrieved 25 January 2017, from http://docs.openstack.org/developer/heat/

Heat Orchestration Template (HOT) Guide (n.d.). Retrieved 25 January 2017, from http://docs.openstack.org/developer/heat/template_guide/hot_guide.html

Oren Ben-Kiki, & Clark Evans. (2009, January 10). YAMLAintMarkupLanguage(YAML) Version1.2. Retrieved 23 January 2017, from http://www.yaml.org/spec/1.2/spec.html

Tacker Documentation. (n.d.). Retrieved 23 January 2017, from http://docs.openstack.org/developer/tacker/

TOSCA Simple Profile in YAML Version 1.0. (n.d.). OASIS. Retrieved from http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.0/os/TOSCA-Simple-Profile-YAML-v1.0-os.pdf

Telefónica NFV Reference Lab. (n.d.). Retrieved 26 January 2017, from http://www.tid.es/es/long-term-innovation/network-innovation/telefonica-nfv-reference-lab

Adrian Hoban, Alfonso Tierno Sepulveda, Gerardo Garia de Blas, Kiran Kashalkar, Mark Shuttleworth, Matt Harpe, & Rajesh Velandy. (2016, October). Open Source MANO. ETSI. Retrieved from https://osm.etsi.org/images/OSM-Whitepaper-TechContent-ReleaseONE-FINAL.pdf

ONAP. (n.d.). Retrieved 24 February 2017, from https://www.onap.org/announcement/2017/02/23/the-linux-foundation-announces-the-formation-of-a-new-project-to-help-accelerate-innovation-in-open-networking-automation

Alesander Curry. (2016, September 3). Edge Computing for network operators, opportunities, business models and partnerships.

ETSI GSNFV. (2015, July 1). Network Functions Virtualisation (NFV); Resiliency Requirements. ETSI. Retrieved from http://www.etsi.org/deliver/etsi_gs/NFV-REL/001_099/001/01.01.01_60/gs_NFV-REL001v010101p.pdf

ETSI GSNFV. (2016, April 20). Network Functions Virtualisation (NFV); Management and Orchestration; Vi-Vnfm reference point - Interface and Information Model Specification. ETSI. Retrieved from http://www.etsi.org/deliver/etsi_gs/NFV-IFA/001_099/006/02.01.01_60/gs_NFV-IFA006v020101p.pdf

Tacker documentation. (n.d.) Alarm monitoring framework . Retrieved 28 January 2017, from http://docs.openstack.org/developer/tacker/devref/alarm_monitoring_usage_guide.html

Tacker Specs documentation. (n.d.) Add alarm-based monitoring driver to Tacker . Retrieved 28 January 2017, from https://specs.openstack.org/openstack/tacker-specs/specs/newton/alarm-based-monitoring-driver.html

Ó Briain, D., Denieffe, D., Kavanagh, Y. & Okello, D. (2016). The move to a software defined future and the implications for Uganda. Presented at the National Conference on Communication (NCC), Mbarara, Uganda: Uganda Communications Commission.