Achieving interoperability and novel management of Virtual Network Functions in NFV-enabled Cloud Data-centre

MSc Research Project
Cloud Computing

Abheri Dutta
x16144198

School of Computing
National College of Ireland

Supervisor: Mr. Vikas Sahni

Student Name:	Abheri Dutta
Student ID:	x16144198
Programme:	Cloud Computing
Year:	2018
Module:	MSc Research Project
Lecturer:	Mr. Vikas Sahni
Submission Due Date:	13/08/2018
Project Title:	Achieving interoperability and novel management of Virtual Network Functions in NFV-enabled Cloud Data-centre
Word Count:	5716

Signature:	
Date:	17th September 2018

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Achieving interoperability and novel management of Virtual Network Functions in NFV-enabled Cloud Data-centre

Abheri Dutta
x16144198
MSc Research Project in Cloud Computing

17th September 2018

Abstract

In large Cloud, Data Centres infrastructure deployment of services by adopting Network Function Virtualization (NFV) and SDN technology will become a necessity. One of the major challenges towards the NFV enablement is its management and interoperability issue. It has major roadblocks in achieving disaggregation of infrastructure, complete software-based approach, network management and operational complexities etc. Currently, Virtual Network Functions (VNF) are available in multiple environments with multiple versions and packages. The VNF packages are using different codes, features, and sizes. This research paper proposes a heuristic VNF management framework to achieve deployment flexibility of VNFs and address interoperability in NFV enabled Cloud DC with heterogeneous resources. The analysis and the enhancement and test carried out in this project are proven to be a novel VNF management technique in NFV-enabled cloud data-center.

1 Introduction

Dutta (2018) In near future, the concept of Network function virtualization(NFV) is the most attractive approach with automated provisioning of services into cloud data-center instead of manual deployment of network services. To explain briefly, Network function virtualization(NFV) decouple the network functions from physical hardware and virtualizes the network services by software codes for each functionality. NFV technology has the potential to revolutionize large-scale service deployment, management and handle the operational task in cloud data-center network in a remarkably reduced time and lower cost and implement them on the commodity hardware. The virtualized network functions in NFV technology stands for VNFs. Orchestration of the virtual resources is an open challenge today because of dynamic coordination across multiple vendor environments, as one of the reason. Evolution of orchestration capability is an absolute need to simplify deployment and operation with heterogeneous resources which increase the complexity of managing these services.

Iglesias et al. (2017) describes on-boarding of various network functions to deploy network services on the cloud is always a painstaking task. Requires marathon configuration commands to establish the virtual environment. Dutta (2018) Monitoring resource availability in each of the virtual machines, measuring the threshold for dynamic scaling and configuring files to deploy network services are few tasks of developers consumes a huge amount of time to perform. Since resources such as CPU, memory etc are managed by individual VM's, the end to end service orchestration becomes a challenge.

Research Question: Can inter-operability and service orchestration be achieved by novel VNF management in NFV-enabled Cloud data-center with heterogeneous resources using heuristic?

According to Rotsos et al. (2017) One of the key challenges towards unified configuration and management of VNF interfaces is the existing VNF Managers currently lacks standardization. VNF appliances come in many different shapes and sizes and operate across all network layer. The different VNF interfaces causing a significant problem to enable automation in service orchestration.

With this motivation, Dutta (2018) a novel and unified VNF life-cycle management framework are designed to achieve the flexible deployment of VNFs in a heterogeneous Cloud-based environment. A functional testing is performed on the live cloud environment.OpenStack is used as the cloud application platform to set up the testing environment. Tacker orchestrator platform is used for developing the proposed novel VNF management framework to address the research question. Tacker library extends SDK for NFV orchestrator used in this development research. Python is used programming language for building VNF templates. The testing is carried out with a statistical analysis of real-time data in dynamically changing environments based on NFV MANO architecture.

Due to stringent time line the interoperability in multi-vendor topology test over the NFV environment will be done in next phase of the project in future.

This paper is categorized into three main sections followed by Introduction. The next section is Literature review includes a brief background of the research, analyze various approaches and researches performed on NFV Orchestration and VNF life-cycle management area. Next section, explains the research methods and tools used for implementation and testing with a high-level design model of the framework which includes brief implementation steps and the lab setup used during research In the final section concluded with detail analysis on evolution and results gathered during testing and future work.

2 Related Work

2.1 Background

Fernandez and Hamid (2015) mentions recently middleboxes are overcrowding the legacy network environment with the use of proprietary network equipments. Moreover, different middleboxes are needed for the different type of services to be deployed. Proprietary hardware is extremely expensive and mostly compatible with own software(OS), packages, protocols, features etc.Yi et al. (2018) states moreover designing proprietary protocol or Operating System incur a high cost and consume relatively more time. Multi-Service deployment in the large data-center environment becomes challenging. In such scenario use of Commercial-Off-The-Shelf(COTS) network equipments, which are providing more flexibility to use general (x86 based server) hardware for provisioning, multiple services in single hardware in Network Function Virtualization(NFV)technology environment looks more promising in the recent data center or cloud environment

Bellavista et al. (2015)mentions with this revolutionary approach flexibility is achieved in network deployment but NFV technology also brought challenges and opportunities along. Conceptually, network functions or services, such as routing, switching, NATing, traffic filtering etc are coded and deployed on top of a virtualized environment. These virtualized functions are called Virtual network functions (VNF), work exactly similar to the physical network.

A large community like ETSI, dedicated towards evolution and standardization of the product because of the potential NFV technology has to capture the cloud market due to its faster deployment capacity with lower cost. Yi et al. (2018) elaborates, in the area of Network function virtualization(NFV)mainly three major area focused for research work. One area is the integration of SDN with NFV and Success in this area would present an optimum network virtualization solution. Edge network like IoT or mobile network vendors highly interested to incorporate NFV technology due to its flexibility and light-weight architecture. The second category of the development on NFV towards VNF placement, service chaining issues etc. The other category of research is focused towards VNF management, deployment, vendor interoperability issues and Orchestration challenges. our research project is an attempt to address the management and interoperability issue of VNFs.

2.2 Literature Review

2.3 NFV Orchestration and Management,Design standards and TOSCA

Standardization bodies like OPNFV, IETF NFV collaborated with ETSI and proposed the extension to the popular MANO framework.Rotsos et al. (2017)mentions existing infrastructure management framework are reused by MANO, just like OpenStack so that external vendors can be supported. By multiple open source implementations, NFV MANO is matured now.Rotsos et al. (2017) explains that, MANO has design limitations. Current implementations use fail to accommodate certain VIM requirement like larger project support like Telecom vendors due to its use of the cloud management system(CMS) which is an OpenStack VIM. However, OPNFV and ETSI with joined effort introduced an open VIM through which addresses the gap.VNF image deployment request is not supported by Current MANO architecture.Fernandez and Hamid (2015)states the standardizations of NFV management framework is an utmost necessity, to address the interoperability issue.

2.4 TOSCA and VNF models

Artač et al. (2017) TOSCA stands for Topology and Orchestration Specification for Cloud Applications. Wettinger et al. (2014) explains, TOSCA is an emerging standardization model for service management and application integration in cloud infrastructure service deployment. A high level of service deployment and operational management is ensured by the TOSCA model in the vendor-independent environment. Rotsos et al. (2017) states TOSCA data-model generates standardized templates for application and services deployment and manage resources in open-source NFV related platform. Using TOSCA template many open-source projects building NFV orchestrators and VNF managers to deploy virtualized network functions in the Cloud environment.

2.5 NFV Orchestration and Management Challenges

Yi et al. (2018) describes although MANO defines a high-end API for service chaining, it does not support a well-defined standard or mechanism to take the decision based on the real-time internal state of Network functions(NF). The inadequate details of MANO API created confusion among Network Service vendors. Due to the in-efficiency of MANO API, various vendors are guided to create their own interpretations. According to Rotsos et al. (2017), unified configuration and management are the key challenges for service orchestrator automation. In ETSI MANO architecture, Element managers(EMS) is the component coupled with each VNF to provide information about the VNFs to the VNF Manager during its lifecycle. But the interface between VNF EMS layer and VNFM lacks standardizations due to the differences in shape sizes of VNF appliances which operate across all network layers.

2.6 VNF Lifecycle management and interoperability challenges

Yi et al. (2018) describes, the life cycle includes five states in ETSI VNF which are sequential, NULL, Instantiated Not Configured, Instantiated Configured Inactive, Instantiated Configured Active and Terminated.Dutta (2018)The NULL and Terminated are the start and end of the management cycle. The drawing of entire VNF life-cycle shown in below picture.

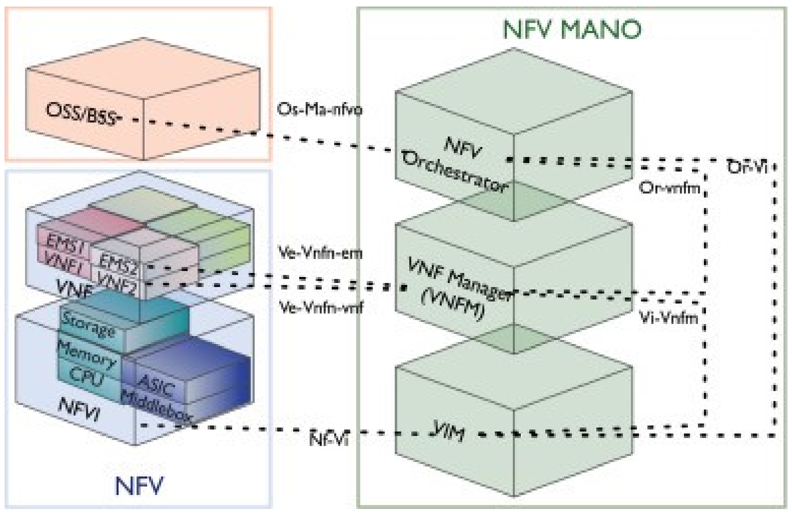


Figure 1: ETSI NFV management and orchestration architectureRotsos et al. (2017)

Yi et al. (2018)states ETSI VNF life cycle design is abstract in the real-time scenario. Dutta (2018) due to the fact, Enterprises implemented their own VNF life-cycle management. Cisco developed elastic service controller with sequences like VNF onboard, deploy, monitor, scale, heal, and update as lifecycle management. Similarly, Juniper added resource planning and VNF image management into the VNF life-cycle manager.Yi et al. (2018) mentions no unified interface or framework exist to coordinate with these orchestrators though many explored in this area.

Dutta (2018) ETSI is instrumental in the job of standardizing the MANO framework to build a vendor-neutral orchestrator, but as the paper describes, MANO's VNF life-cycle management lack real-world features to be adaptable in production deployment eg, on-boarding, auto-scaling, auto-healing, image management etc. External vendor-made orchestrators are in use whenever ETSI MANO fail to accommodate the customer requirement of providing services but all of them are faces compatibility and interoperability issues with other orchestrator or VNF managers.

2.7 Different APIs and analyze the frameworks

NFV technology brings down the cost of service provisioning by running Network functions on the inexpensive commodity hardware and by replacing high-end proprietary boxes.Ge et al. (2014) argues that the challenge is shared hardware degrade the performance of some network functions like NAT etc.Ge et al. (2014) proposes OpenANFV, a framework to achieve optimum performance using commodity hardware with VM machine. OpenANFV adopted Network Function Acceleration Platform (NEAP) technology and increase the performance of virtualized network functions instances successfully by running DPI, NAT services on the shared hardware(COTS) in cloud.Ge et al. (2014) elaborates i)Automated VNF management ii) Elasticity and iii) OpenStack are the capabilities supported by OpenANFV.

Khalid et al. (2016) argues that MANO APIs are not intelligent to observe and identify a state of each network traffic and incapable to take an independent decision based on the type of traffics. Though ETSI is consistently standardizing the MANO, the improvement in this area would further enhance placement of VNFs by measuring performance factors etc. A standard API should be detailed enough to manage VNF functions and adopt a generic approach to serve any specific implementation requirement.

Khalid et al. (2016) proposes a modern standardized Southbound API, to separate the operational task from core network function, in that manner API has full control over the network functions. The API becomes authorize to decide specific action about a given state of network traffic instead of just allowing it to the next VNF without instructions.Khalid et al. (2016) emphasis on 4 core operations in I. State management II. Service Chaining III. Bottleneck detection and IV. Configuration, in the proposed standard southbound API design.

But Khalid et al. (2016) brings about monumental action items to the vendors to realize the solution. First of all, the all VNF softwares to be modified accordingly in order to support the functionalists.Secondly, all the existing VNF Managers need modifications to support the operations proposed in the solution. However, if these roadblocks can be omitted this API offers a common interface to all the VNF managers. The task of a VNFM would be simpler and standardized by using a common software package to deploy VNFs.

2.8 Analysis of some VNF management framework and orchestrators

Multiple pieces of research are ongoing in the VNF management framework design area. Few VNF life-cycle management technique has introduced based on ETSI standard architecture as discussed below:

Shen et al. (2014) presents v-Conductor, an NFV management framework which helps to deploy end to end virtual network services based on MANO. It is the first ever VNF management model automates the provisioning.The model includes inventory management and compatible with backward resources to orchestrate while provisioning the network. Along with virtualized Network function, v-Conductor manages the physical network functions(PNF)too.Shen et al. (2014) states that the parts of v-Conductor are replaceable with products of third-party vendors too.

Szabo et al. (2017) presents FERO, does resource management by simultaneously creating the data-plane based on VNF traffics based on the resource hardware available in the environment, which is a data-plane orchestrator. It designed with a programmable interface.FERO capable of orchestration in a multi-domain environment.

Dutta (2018) In the above paper,v-Conductor is described as the end to end VNF management which does not support inter vendor interface. Interoperability, management with other vendor products would include in future work. The other research work introduces FERO as an intelligent resource management methodology tested in docker environment which is not par with the big production environment and also it excludes VNF life-cycle management.

NFV MANO is lacking a optimal management and orchestration of VNFs.Dynamic resource management, dynamic allocation, intelligence to automate the network operation in larger scale and interoperability issue in multi-vendor and multi-function scenario are the few open challenges. Thus the heuristic management and deployment approach would help large deployment of Cloud data center infrastructure.

Related Papers	Works Done
Wettinger et al. (2014)	Concept of integrating Dev-Ops artifacts with TOSCA std.
Shen et al. (2014)	vConductor an automation tool to provision VNFs and PNFs
Ge et al. (2014)	OpenANFV framework to optimise performance in VM
Bellavista et al. (2015)	A solution to MCRVMP-N issue for VM placement in cloud
Kothandaraman et al. (2015)	Central mgmt.of VNFs isolating control-plane on SDN-NFV
Fernandez and Hamid (2015)	A NFV pattern build on cloud to offer SaaS services
Mijumbi et al. (2016)	Cloud4NFV an e2e management platform of VNFs on ETSI
Khalid et al. (2016)	Designing standard SouthBound API by decoupling VNF mgmt.
Cziva et al. (2016)	GNF,a management framework in Containerized NFV
Bernal et al. (2016)	Inter-VNF Communication with Open vSwitch
Sun et al. (2016)	SLA aware high performance NFV architechture
Hung et al. (2017)	VNFM orchestrator based on TOSCA model
Hu et al. (2017)	Netcontainer,VNF provisioning framework
Fawcett and Race (2017)	Siren,management tool to deploy services
Iglesias et al. (2017)	ORCA,An automated service operation by configuring VNFs
Szabo et al. (2017)	FERO,dataplane orchestrator for resource management
Li et al. (2017)	High performance VNF using DPDK on multicore processor
Yi et al. (2018)	FROG,a NFV Orchestrator for SDN and cloud
Dutta (2018)	Management of VNF in Cloud DC using heuristic

Table 1: Summery of related work on VNF Orchestration and Management Dutta (2018)

Dutta (2018) Likewise, various research works are in progress to standardize NFV orchestration and mitigate the interoperability challenges. Though the NFV technology innovated a few years ago and evolving continuously, VNFs are still having manageability orchestration challenges. Therefore, there is a greater need for a standardized orchestration and VNF management framework. After analyzing the previous research works thoroughly,-in this research paper, designing a heuristic framework to manage Virtual Network Functions to achieve manageability and speedy deployment in NFV-enabled Cloud data-center with heterogeneous resources. The proposed framework is developed as a form of new VNF Management and provisioning method based on TOSCA model by applying heuristic. The research methods and testing approach of the proposed solution are described in details in the following sections.

3 Methodology

In the research method section building up the lab environment established by four components:

Sl.No.	Description
1	Hardware setup
2	Openstack setup
3	Tacker setup
4	NFV templates to setup mgmt

Table 2: Research Environment

3.1 Hardware setup

Depending upon the project requirement used one Intel(R) Xeon(R) CPU E5506 @2.13GHz. The X86 64 Linux server installed with Ubuntu 16.4.Specifications of 23 GiB system memory. Installed KVM to build multiple VM instances for OpenStack and Tacker to build VNF manager.

3.2 KVM on Ubuntu 16.4

Kernel Virtual machine(KVM) KVM (2008) is a Open source software.It is a full virtualization solution for Linux on X86 hardware.KVM includes kvm.ko which is loadable kernel module for providing core virtualization infrastructure with kvm-intel.ko and KVM-amd.ko modules for processors. By using KVM we are running multiple VM machines on top of unmodified Linux image.

3.3 OpenStack

OpenStackOpenStack (2018a) is an open-source cloud platform used for research and development purpose. In 2010 NASA and RackSpace jointly invented this cloud operating system . External organization like Cisco, IBM, VMWare, RedHat, IBM supports and uses the OpenStack foundation for development and purpose.OpenStack provides pull of available resources to developers to work on the applications in the virtualized environment. Through OpenStack, Dashboard end-user can easily get access and allocate required resources as per need. Many components like Nova for computing, Neutron for networking, Horizon, Keystone, glance for image etc used in the project.

Bellavista et al. (2015) As per project requirement keystone, nova, glance, cinder, neutron, and horizon are installed by Devstack. Floating IP has been used, guests have access to the external world. The VMs are exposed to the public network via Floating IP for external network communication.

Installing OpenStack OpenStack (2018b) manually is a tedious and very lengthy process. Devstack has been used for OpenStack installation with required components mentioned above for the project setup. Devstack takes considerably less amount of time to have OpenStack environment ready including required components needed by modifying few configuration files required to perform the execution.

3.4 PfSense Firewall

Pfsense Pfsense (2004) is licensed by Apache 2.0 open-source software. This software used as a router or firewall by many small to medium size organizations as a security mechanism or routing platform.The product offers a varied range of features to support various needs of organizations network infrastructure. In the project, pfsense is used as a NAT device to map the internal private IP range of OpenStack 10.10.10.1/24 to convert with WAN IP segment (public IP)for external network access.

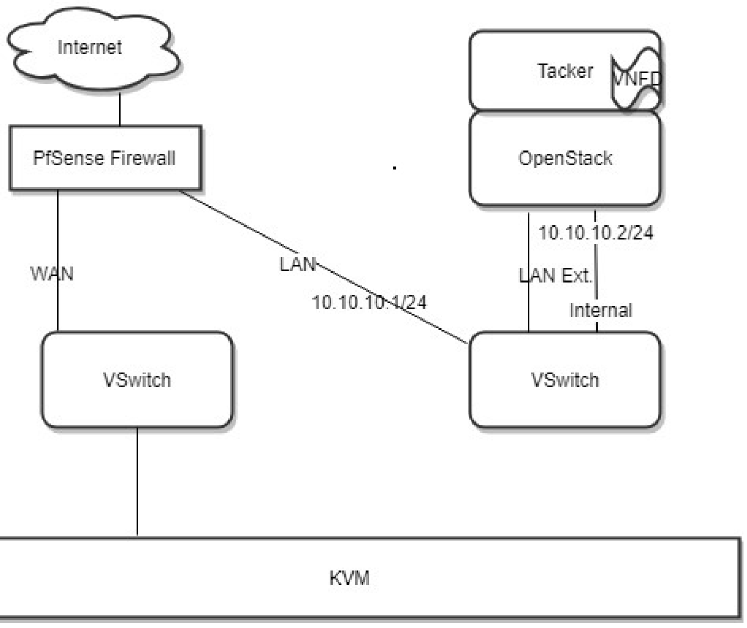


Figure 2: Test Lab setup for Research Project

3.5 Tacker

Katsalis et al. (2016) Tacker is NFV Orchestrator (NFVO) and official VNF manager for OpenStack to provision and operate Virtual Network Functions (VNFs) on the OpenStack NFV infrastructure. It is built by OpenStack based on ETSI MANO standard to orchestrate Network Services by using VNFs.

Various components needed to run NFV orchestrator service on OpenStack.Horizon is the component installed to use tacker commands to be used for creating the templates.VNFD templates use the TOSCA standard which is built after setting up Horizon. Heat and Keystone are the other two API's for NFV orchestration to function properly.

To run the environment of NFV Orchestrator on Tacker primarily four elements installed which are necessary.

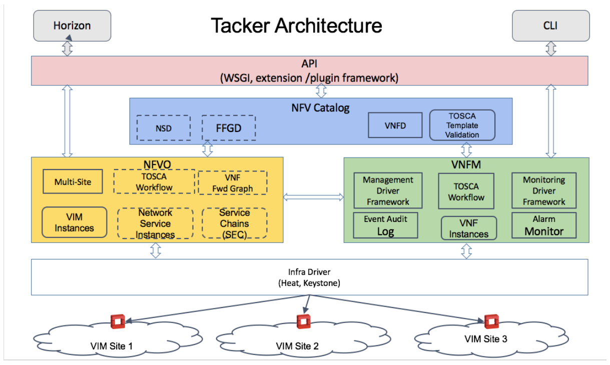


Figure 3: Tacker Architecture Tacker$_{O}$*penStack* (2015)

4 Implementation

In this research a virtual network infrastructure built on top of OpenStack. We used KVM on a physical Linux X86 box to create multiple Virtual machines to configure OpenStack. Tacker has used as the NFV orchestrator service here. Pfsense firewall has been used for network translation. Pfsense is used to Network Address Translation(NAT) for the inside private IP range to convert with Public Ip for communication with external vendor VNFs is the case for further research after building the VNF manager.

The VNF manager model created on top of Tacker Platform. For each of the network function with the same vendor, one VNFD template is written. For example, for routing, if in the network 4 routers are available from 2 different vendors 2 different templates are written for 2 different vendors. Similarly, for other network function different set of templates can be created in future. Each of the network function will be coded in Python. Tacker provides with required libraries and SDK to develop new VNF templates. Once the templates are ready, can be integrated with on the Tacker. In this project, VNF with routing functions is created and tested.

Yi et al. (2018) states Network management and orchestration layer (NFV MANO) is completely responsible for managing the virtualized environment in NFV, which includes resource orchestration, VNF provisioning, and management of VNF life-cycle. These jobs are assigned to Virtual Infrastructure Manager (VIM), NFV Orchestrator(NFVO) and Virtual Network Function manager(VNFM) based on ETSI. *VIM* manages the compute, storage and network resources running on VMs.But VIM is customizable to control only

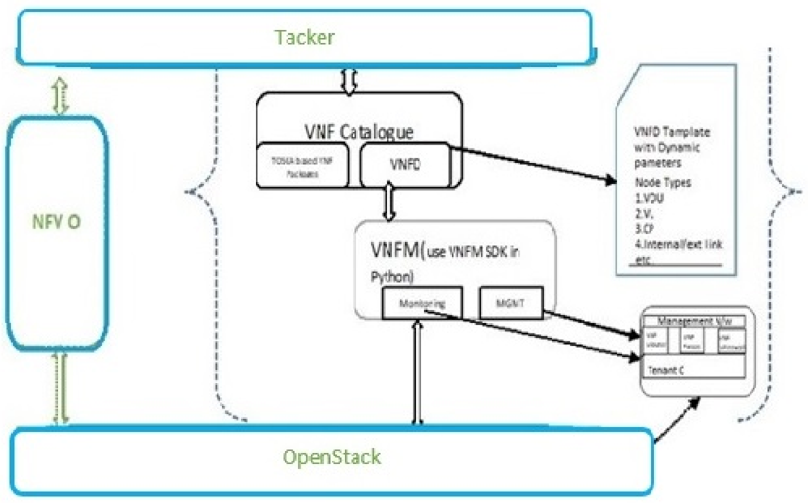


Figure 4: Design Model

one type of resources like a network or compute. *NFVO* responsible for orchestrating the VNF instances on NFV Infrastructure(NFVI). NFVO identify the optimal path for multiple VNFs to provision services after the provisioning of VNF instances. *VNFM* manages the life-cycle of multiple VNF instances includes modification, scaling, deployment etc. Single VNF manager(VNFM) manages Multiple VNFs However one VNF mapped with only single VNFM. *Network topology to connect VNFs*: Below diagram explains the connection and flow of traffic within VNFs. Virtual links (VL) are the logical connection between the endpoints which are physical or logical interfaces(NIC). Virtual links are the connections between the different VNFs.

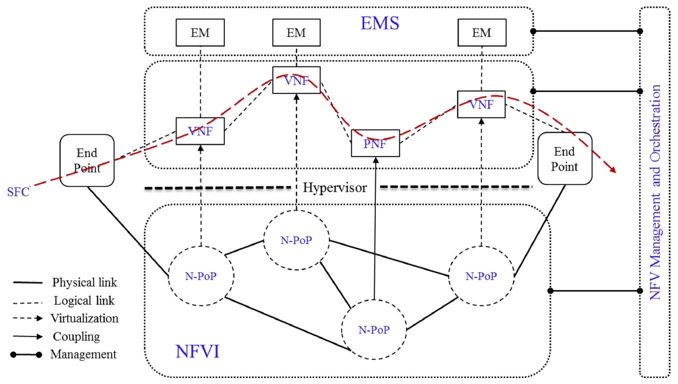
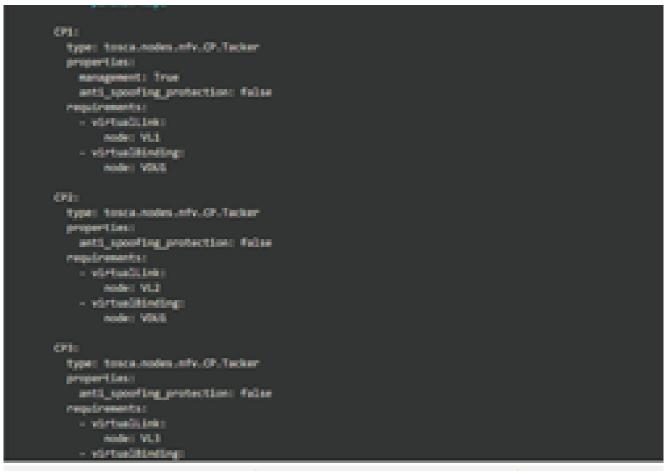


Figure 5: NFV MANO and VNF trafficsYi et al. (2018)

.

4.1 Novel framework to create VNF instances with unified templates

Ersue (2013) Currently, the challenge of existing VNFD templates is to use fixed values to deploy VNF instances, which means multiple VNF deployments is restricted to the single template. Each deployment requires writing a separate template to onboard a VNF. For example, with fixed Ip address second-time deployment of the instance without deleting the first one would trigger an error. It becomes tedious to write multiple templates to deploy a considerably large no of VNF deployment. Below figures shows the existing model of the VNF templates:



```
CP1:
  type: tosca.nodes.nfv.CP.Tacker
  properties:
    management: True
    anti_spoofing_protection: false
  requirements:
    - virtualLink:
        node: VL1
    - virtualBinding:
        node: VDU1

CP2:
  type: tosca.nodes.nfv.CP.Tacker
  properties:
    anti_spoofing_protection: false
  requirements:
    - virtualLink:
        node: VL2
    - virtualBinding:
        node: VDU1

CP3:
  type: tosca.nodes.nfv.CP.Tacker
  properties:
    anti_spoofing_protection: false
  requirements:
    - virtualLink:
        node: VL3
    - virtualBinding:
```

Figure 6: Existing VNF Template

In this research, a new framework allows creating VNF instances from a running network and the VNF templates capable of accepting dynamic values and further execute them to create similar VNFs running same services. As per the framework the IP-address, image name, network, management, vendor name are scriptable as dynamic fields using python. The advantage of this technique to enable end users deploy VNF instances in bulk numbers to provision network services faster in a bigger environment with cloud infrastructure. This unified approach assumed to bring better flexibility to the network operators and a seamless network service deployment with the lesser effort with much-reduced time-frame.

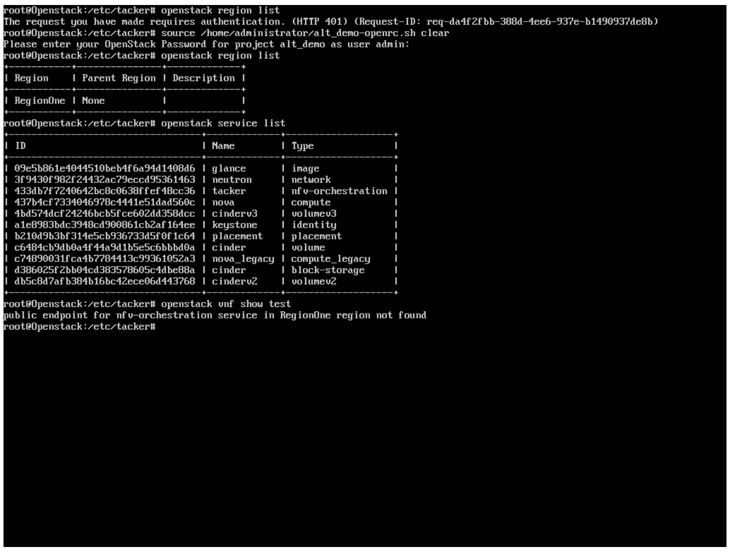
To test the generated VNF templates nfv orchestrator service instance deployed on OpenStack and in below section the testings are conducted to evaluate the results of the previous network based on the parameters provided.

5 Evaluation

Installed OpenStack through Devstack. Devstack supported by Ubuntu 16.04. The components such as Keystone, Nova, neutron, glance, and horizon are installed by devstack by modifying source openrc in the shell. Floating IPs will be available, guests have access to the external world.OpenStack command line tool manages devstack.

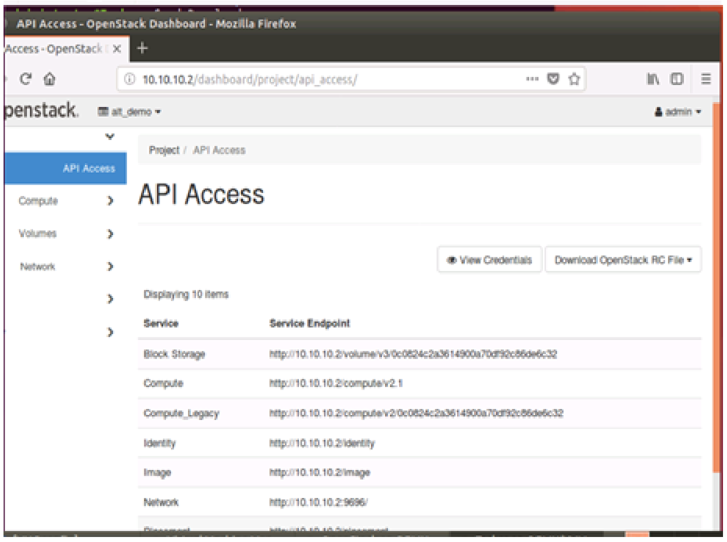
Once the OpenStack running successfully the Dashboard has been launched. The dashboard is used to manage VMs, networks, volumes, and images etc.

In order to run Tacker the local.conf file is created by manual. Tacker configuration installation has two options 1. all-in-one mode 2. standalone mode which only will install a standalone environment. All in one mode establish a complete devstack environment for Tacker. All-in-one mode is adopted in our project to setup Tacker environment.Once the local.conf file is created, execution of stack .sh is establishing the Tacker environment.



```
root@Openstack:/etc/tacker# openstack region list
The request you have made requires authentication. (HTTP 401) (Request-ID: req-da4f2fbb-388d-4ee6-937e-b1490937de8b)
root@Openstack:/etc/tacker# source /home/administrator/alt_demo-openrc.sh clear
Please enter your OpenStack Password for project alt_demo as user admin:
root@Openstack:/etc/tacker# openstack region list
+-----------+---------------+-------------+
| Region    | Parent Region | Description |
+-----------+---------------+-------------+
| RegionOne | None          |             |
+-----------+---------------+-------------+
root@Openstack:/etc/tacker# openstack service list
+----------------------------------+-------------+-------------------+
| ID                               | Name        | Type              |
+----------------------------------+-------------+-------------------+
| 09e5b861e4044510beb4f6a94d1408d6 | glance      | image             |
| 3f9430f982f24432ac79eccd95361463 | neutron     | network           |
| 433db7f7240642bc8c0638ffef48cc36 | tacker      | nfv-orchestration |
| 437b4cf7334046978c4441e51dad560c | nova        | compute           |
| 4bd574dcf24246bcb5fce602dd358dcc | cinderv3    | volumev3          |
| a1e8983bdc3948cd900861cb2af164ee | keystone    | identity          |
| b210d9b3bf314e5cb936733d5f0f1c64 | placement   | placement         |
| c6484cb9db0a4f44a9d1b5e5c6bbbd0a | cinder      | volume            |
| c74890031fca4b7784413c99361052a3 | nova_legacy | compute_legacy    |
| d386025f2bb04cd383578605c4dbe88a | cinder      | block-storage     |
| db5c8d7afb384b16bc42ece06d443768 | cinderv2    | volumev2          |
+----------------------------------+-------------+-------------------+
root@Openstack:/etc/tacker# openstack vnf show test
public endpoint for nfv-orchestration service in RegionOne region not found
root@Openstack:/etc/tacker#
```

Figure 7: Tacker command Lists


API Access - OpenStack Dashboard - Mozilla Firefox
10.10.10.2/dashboard/project/api_access/
openstack.
alt_demo
admin
API Access
Compute
Volumes
Network
Project / API Access
View Credentials
Download OpenStack RC File

API Access

Displaying 10 items

Service	Service Endpoint
Block Storage	http://10.10.10.2/volume/v3/0c0824c2a3614900a70df92c86de6c32
Compute	http://10.10.10.2/compute/v2.1
Compute_Legacy	http://10.10.10.2/compute/v2/0c0824c2a3614900a70df92c86de6c32
Identity	http://10.10.10.2/identity
Image	http://10.10.10.2/image
Network	http://10.10.10.2:9696/

Figure 8: OpenStack Dashboard

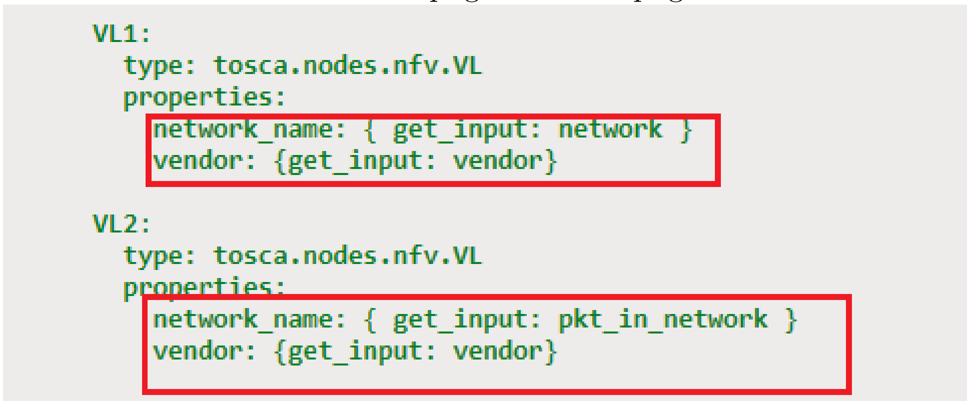
In the existing scenario, each time in VNFD templates fed with the various sections with appropriate values like image name, flavor, network, vendor, during each deploy of the VNF. This approach is factually assumed to hugely bring down the deployment time of multiple similar function VNFs for a larger deployment scenario.

5.1 Experiment / Case Study 1

In this project, during test identified variable values provided during deployment with getting input: variable name. For example, the image name: openwrt-15.05-x86-64-combined-ext4.img now replaced as: get input: image name. The image name is the variable that will keep the value for the variable image in a variables value file that is now provided at VNF deploy time.

The new templates look like the section as below :

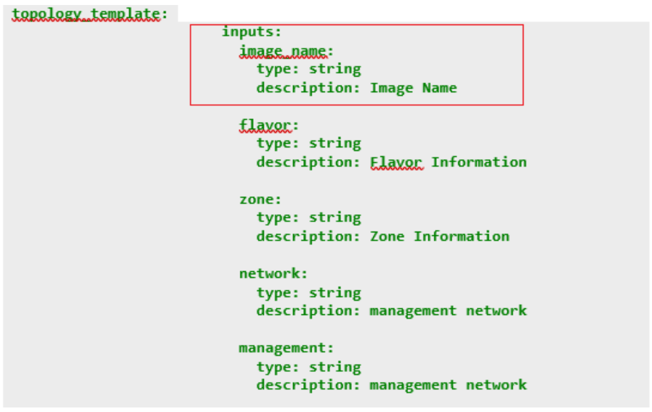
Currently, Variable fields are the image file, Connection Point and virtual links can variables2.png variables2.png



```
VL1:
  type: tosca.nodes.nfv.VL
  properties:
    network_name: { get_input: network }
    vendor: {get_input: vendor}

VL2:
  type: tosca.nodes.nfv.VL
  properties:
    network_name: { get_input: pkt_in_network }
    vendor: {get_input: vendor}
```

Figure 9: New VNF template with variable image field



```
topology_template:
          inputs:
            image_name:
              type: string
              description: Image Name

            flavor:
              type: string
              description: Flavor Information

            zone:
              type: string
              description: Zone Information

            network:
              type: string
              description: management network

            management:
              type: string
              description: management network
```

Figure 10: New VNF template with other variable fields

be modified to built multiple VNF files with the single template. Previously for each VNF function template needs to be created separately which is a time-consuming job for operators and vendors. By adopting the approach demonstrated in the project multiple similar function VNFs can be generated by using a single template which considerably reduces service deployment time and meets customer demand with less effort.

6 Conclusion and Future Work

Network Function Virtualization is one of the most desired ways to deploy services in the field of Cloud environment. While from past five years various approaches implemented to standardize NFV manageability and orchestration with help of improvised APIs and developing different frameworks and new models there is still a gap existed in managing and operating VNFs in a larger environment with heterogeneous resources. By implementing the framework that has been proposed in this research, it would be clearly possible to increase the speed of deployment and flexible to manage the VNFs in an NFV enabled Cloud infrastructure. In this project, a study and experimentation on VNF manageability through nfv-orchestrator service on OpenStack is demonstrated. Discussed the introduction to cloud computing and background of Network Function Virtualization and its manageability challenges in depth by analyzing related work and existing approaches advantages and challenges.

Through this research, Its successfully demonstrated that, the deployment of virtual network functions are quick and more flexible to manage by generating multiple VNFs for one service in the same time by inputting values to the variable fields in a single VNF template. This approach would be effective and more adoptable while provisioning larger network services in a Cloud environment. By this approach, end-user would have the capability to provide a more no of network services faster.

In the future work , the interoperability test with external vendor VNFs will be tested with multi-vendor topology scenario on the the NFV environment as discussed in the project as another challenge area . Also, the realistic data-sets from Survival Network design Library (SNDlib) can be used to traffic generation for testing the interoperability solution against different benchmark matrices with fewer scenarios in the next phase of the project.

7 Acknowledgment

I sincerely thank my mentor and supervisor Mr. Vikas Sahni who has been guiding me constantly throughout my research with invaluable ideas and inputs right from the start till submission of my project. I thank him for all the suggestions and his support extended to me in crucial moments to complete the research in time.

I express my gratitude towards my course director and the head of Cloud Competency Centre Dr.Horacio Gonzalez-Velez for guiding me constantly in the early stages of research and during the time of deciding and finalizing the topics. Finally, I would like to thank my family and friends for their continuous encouragement, assistance, and support and not losing faith in me during my research.

References

Artač, M., Borovšak, T., Di Nitto, E., Guerriero, M. and Tamburri, D. A. (2017). Devops: Introducing infrastructure-as-code, *Proceedings of the 39th International Conference on Software Engineering Companion*, ICSE-C '17, IEEE Press, Piscataway, NJ, USA, pp. 497–498. Core Ranking : B.
URL: *https://doi.org/10.1109/ICSE-C.2017.162*

Bellavista, P., Callegati, F., Cerroni, W., Contoli, C., Corradi, A., Foschini, L., Pernafini, A. and Santandrea, G. (2015). Virtual network function embedding in real cloud environments, *Computer Networks* **93**: 506 – 517. Cloud Networking and Communications II Core Ranking :A.

Bernal, M. V., Cerrato, I., Risso, F. and Verbeiren, D. (2016). A transparent highway for inter-virtual network function communication with open vswitch, *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, ACM, New York, NY, USA, pp. 603–604. Core Ranking :A*.

Cziva, R., Jouet, S. and Pezaros, D. P. (2016). Roaming edge vNFs using glasgow network functions, *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, ACM, New York, NY, USA, pp. 601–602. Core Ranking :A*.

Dutta, A. (2018). Achieving interoperability and novel management of virtual network functions in nfv-enabled cloud data-centre using heuristic.

Ersue, M. (2013). Etsi nfv management and orchestration-an overview.

Fawcett, L. and Race, N. (2017). Siren: A platform for deployment of VNFs in distributed infrastructures, *Proceedings of the Symposium on SDN Research*, SOSR '17, ACM, pp. 201–202. Core Ranking: A.

Fernandez, E. B. and Hamid, B. (2015). A pattern for network functions virtualization, *Proceedings of the 20th European Conference on Pattern Languages of Programs*, EuroPLoP '15, ACM, pp. 47:1–47:9. Core Ranking: B.

Ge, X., Liu, Y., Du, D. H., Zhang, L., Guan, H., Chen, J., Zhao, Y. and Hu, X. (2014). Openanfv: Accelerating Network Function Virtualization with a consolidated framework in openstack, *SIGCOMM Comput. Commun. Rev.* **44**(4): 353–354. Core Ranking A*.

Hu, Y., Song, M. and Li, T. (2017). Towards "full containerization" in containerized network function virtualization, *SIGOPS Oper. Syst. Rev.* **51**(2): 467–481. Core Ranking :A*.

Hung, Y. M., Chien, S. C. and Chunghwa, Y. Y. H. (2017). Orchestration of NFV virtual applications based on tosca data models, *2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 219–222. Core Ranking: C.

Iglesias, J. O., Aroca, J. A., Hilt, V. and Lugones, D. (2017). Orca an automata for configuring VNFs, *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference*, Middleware '17, ACM, New York, NY, USA, pp. 81–94. Core Ranking: B.

Katsalis, K., Nikaein, N. and Edmonds, A. (2016). Multi-domain orchestration for nfv: Challenges and research directions, *2016 15th International Conference on Ubiquitous Computing and Communications and 2016 International Symposium on Cyberspace and Security (IUCC-CSS)*, pp. 189–195.

Khalid, J., Coatsworth, M., Gember-Jacobson, A. and Akella, A. (2016). A standardized southbound api for VNFmanagement, *Proceedings of the 2016 Workshop on Hot Topics in Middleboxes and Network Function Virtualization*, HotMIddlebox '16, ACM, New York, NY, USA, pp. 38–43. Core Ranking: A*.

Kothandaraman, B., Du, M. and Sköldström, P. (2015). Centrally controlled distributed vnf state management, *Proceedings of the 2015 ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization*, HotMiddlebox '15, ACM, New York, NY, USA, pp. 37–42. Core Ranking: A*.

KVM (2008). Kvm.
URL: *https://www.linux-kvm.org/page/Main$_P$age*

Li, P., Wu, X., Ran, Y. and Luo, Y. (2017). Designing virtual network functions for 100 gbe network using multicore processors, *Proceedings of the Symposium on Architectures for Networking and Communications Systems*, ANCS '17, IEEE Press, Piscataway, NJ, USA, pp. 49–59.

Mijumbi, R., Serrat, J., Gorricho, J. L., Bouten, N., Turck, F. D. and Boutaba, R. (2016). Network function virtualization: State-of-the-art and research challenges, *IEEE Communications Surveys Tutorials* **18**(1): 236–262. cited 448.

OpenStack (2018a). Openstack architecture. https://docs.openstack.org/liberty/networking-guide.

OpenStack (2018b). Openstack installation steps.
URL: *https://docs.openstack.org/install-guide/openstack-services.html*

Pfsense (2004). Pfsense.
URL: *https://www.pfsense.org/download/*

Rotsos, C., King, D., Farshad, A., Bird, J., Fawcett, L., Georgalas, N., Gunkel, M., Shiomoto, K., Wang, A., Mauthe, A., Race, N. and Hutchison, D. (2017). Network service orchestration standardization: A technology survey, *Computer Standards and Interfaces* **54**: 203 – 215. SI: Standardization SDN and NFV. Core Ranking: B.

Shen, W., Yoshida, M., Kawabata, T., Minato, K. and Imajuku, W. (2014). vconductor: An NFV management solution for realizing end-to-end virtual network services, *The 16th Asia-Pacific Network Operations and Management Symposium*, pp. 1–6. Core ranking:C.

Sun, C., Bi, J., Zheng, Z. and Hu, H. (2016). Sla-nfv: An sla-aware high performance framework for network function virtualization, *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, ACM, New York, NY, USA, pp. 581–582. Core Ranking A*.

Szabo, M., Majdan, A., Pongracz, G., Toka, L. and Sonkoly, B. (2017). Making the data plane ready for NFV: An effective way of handling resources, *Proceedings of the SIGCOMM Posters and Demos*, SIGCOMM Posters and Demos '17, ACM, New York, NY, USA, pp. 97–99. Core Ranking :A*.

Tacker$_{O}penStack$(2015)$.Tackerarchitecture.$
URL:*https://wiki.openstack.org/wiki/Tacker*

Wettinger, J., Breitenbücher, U. and Leymann, F. (2014). Standards-based devops automation and integration using tosca, *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, UCC '14, IEEE Computer Society, London,UK, pp. 59–68. Core ranking: A.

Yi, B., Wang, X., Li, K., k. Das, S. and Huang, M. (2018). A comprehensive survey of Network Function Virtualization, *Computer Networks; Core Ranking :A* **133**: 212 – 262. Core Ranking :A.