

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO



Creation of a Marketplace for NFV Functions

Luís Pedro Sousa Pinto

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

FEUP Supervisor: Ricardo Morla

External Supervisor: Pedro Tavares

July 25, 2017

Abstract

Virtualisation is gaining worldwide importance with its ability to support the development of network applications with the efficiency and flexibility of software applications. Motivation for investment in this type of technology is both financial and technical, the latter driven by better resource utilization and easy migration workload.

NFV is one such virtualisation concept. NFV allows the replacement and complementing of physical network devices with virtual network functions. Network functions such as NAT, DNS and IMS can thus be separated from the physical hardware for software through this concept.

This thesis discusses motivations and solutions for creating a Marketplace for NFV virtual functions (VNFs). The Marketplace solution will be able to accelerate the Procurement process in NFV. The proposed solution is defined to work as a Dynamic VNF Catalogue and with the concept of Yellow Pages for the customers and it is based on OpenStack. It will allow VNFs to be published, downloaded and instantiated on-demand.

Resumo

A virtualização está a ganhar importância mundial com a sua capacidade de suportar o desenvolvimento de aplicações de rede com eficiência e flexibilidade de aplicações de software. A motivação para o investimento neste tipo de tecnologia é tanto financeira como técnica, esta última impulsionada por uma melhor utilização dos recursos e uma fácil carga de migração.

O NFV é um desses conceitos de virtualização. O NFV permite a substituição e complementação de dispositivos físicos de rede com funções de rede virtuais. Funções de rede como NAT, DNS e IMS podem, portanto, ser separadas do hardware físico para software através desse conceito.

Esta tese discute motivações e soluções para criar um Marketplace de funções virtuais do NFV (VNFs). A solução do Marketplace será capaz de acelerar o processo de aprovisionamento no NFV. A solução proposta é definida como um dinâmico catálogo de VNFs, com o conceito de Páginas Amarelas para os clientes e é baseada em OpenStack. Isto permitirá que as VNFs sejam publicadas, descarregadas e instanciadas sob solicitação.

Agradecimentos

Em primeiro lugar, agradeço ao meu orientador Ricardo Morla por toda a ajuda e dedicação. A sua experiência, disponibilidade, orientação e crítica construtiva foram essenciais para o desenvolvimento desta dissertação e eu não poderia estar mais agradecido pelo seu compromisso.

Gostaria de agradecer ao meu co-orientador Pedro Tavares por me ter dado a oportunidade de conhecer o mundo empresarial, fornecendo-me toda a ajuda necessária durante esta tese de mestrado. A sua ambição e profissionalismo demonstrou ser uma fonte de inspiração para nunca desistir em alcançar a perfeição.

O agradecimento mais profundo à minha namorada, Mafalda Rodrigues, pelo incentivo, compreensão e encorajamento, durante todo este período. Incondicionalmente, esteve o tempo todo ao meu lado e, nos momentos mais difíceis que não foram raros neste período, sempre me fez acreditar que chegaria ao final desta difícil etapa.

Quero demonstrar a minha gratidão à minha amada família nomeadamente à minha mãe, Ana Sousa, ao meu pai, Fernando Pinto, e ao meu irmão, Rui Sousa, por me terem dado condições necessárias de poder realizar esta dissertação. O eterno apoio, ajuda financeira e encorajamento permitiram que fosse possível integrar-me um meio diferente e concluir este projeto de mestrado.

Quero agradecer ao meu primo, João Oliveira, e à namorada, Ana, por me terem recebido e apoiado durante a fase inicial deste longo projeto. Proporcionaram-me condições para me adaptar facilmente num novo meio e criar laços com novas pessoas.

Gostaria também de agradecer à minha tia, Olinda Sousa, ao meu tio, Jorge Oliveira, ao meu primo, Pedro Oliveira, à esposa, Inês, e a nova aquisição da família, a pequena Ritinha, por estarem sempre presentes durante este percurso e por me terem dado alegrias e forças para continuar a lutar pelo objetivo de finalizar a dissertação.

Quero agradecer aos meus colegas de curso pelos momentos de entusiasmo partilhados em conjunto.

Finalmente, gostaria de gratificar à Deloitte, S.A. e à equipa TEE onde trabalhei durante o período desta tese de mestrado. Permitiram-me participar num ambiente inspirador, inovador e encorajador que eles compartilham. Um agradecimento especial aos meus colegas da TEE, que ofereceram tempo e experiência como participantes na fase de avaliação deste projeto de tese de mestrado.

Luís Pedro Sousa Pinto

*“The great accomplishments of man have resulted from
the transmission of ideas and enthusiasm”*

Thomas John Watson

Contents

1	Introduction	1
1.1	Context	1
1.2	Objectives	2
1.3	Motivation	3
1.4	Structure	3
2	State of the Art	5
2.1	Technical Background	5
2.1.1	Overview of Virtualisation	5
2.1.2	NFV	7
2.1.3	OpenStack	12
2.1.4	Marketplace	18
2.2	Related Work	20
2.2.1	GSMA Marketplace	20
2.2.2	Aptoide	22
2.2.3	T-NOVA: Marketplace for NFV Functions	25
2.3	Conclusions	41
3	VNF Marketplace	45
3.1	Characterisation	45
3.1.1	Authentication and Authorisation	48
3.1.2	VNF Catalog Module	50
3.1.3	Request Module	52
3.1.4	Billing System	53
3.1.5	SLA Management	55
3.1.6	SLA Control System	56
3.1.7	Dashboard	57
3.1.8	Analytics Module	61
3.1.9	Interaction with OpenStack	62
3.2	Development	66
3.2.1	VNF Marketplace	66
3.2.2	OpenStack	71
3.3	Summary	74
4	Results	75
4.1	VNF Marketplace Functional Results	75
4.1.1	Authentication and Authorisation Module Functional Results	76
4.1.2	VNF Catalog Module Functional Results	79

4.1.3	Request Module Functional Results	85
4.1.4	Billing Module Functional Results	87
4.1.5	SLA Management Functional Results	90
4.1.6	SLA Control Module Functional Results	92
4.1.7	Dashboard & Analytics Module Functional Results	93
4.2	Functional Results of the interaction with OpenStack	95
4.2.1	VNF Publication	95
4.2.2	VNF Deployment	96
4.3	Tests Results	98
4.3.1	Time to Install VNF Marketplace	98
4.3.2	VNF Marketplace Performance	100
4.3.3	OpenStack Performance	106
4.3.4	Interviews	108
5	Conclusions and Future Work	115
5.1	Satisfaction of Objectives	115
5.2	Limitations	116
5.3	Future Work	116
A	Use Cases	117
A.1	Use Cases Details	118
B	Requirements Specification	125
B.1	Authentication and Authorisation Module	125
B.2	VNF Catalog Module	126
B.3	Request Module	128
B.4	Billing Module	129
B.5	SLA Management	131
B.6	SLA Control	132
B.7	Dashboard	133
B.8	Analytics Module	138
B.9	OpenStack	138
C	Dashboard Views, Requests and Responses	141
C.1	Customer	141
C.2	Seller	145
C.3	OpenStack	148
D	Files & Detailed Results	151
D.1	<i>Requirements.txt</i> File	151
D.2	<i>Settings.py</i> File	152
D.3	<i>Configuration.properties</i> File	155
D.4	<i>Local.conf</i> File	156
D.5	<i>VNFMarkerplaceInstallation.txt</i> File	156
D.6	<i>SLAManagementReqInstallation.txt</i> File	156
D.7	OpenWRT VNFD	157
D.8	Loading Static Files - CPU and RAM Results	158
D.9	Seller VNF Publication - CPU Results	159
D.10	Seller VNF Withdraw - CPU Results	161

D.11 Customer Actions - CPU Results	162
D.12 OpenStack with VNF Marketplace - CPU Results	163
D.13 OpenStack - CPU and RAM Results	166
D.14 VNF Marketplace installation times	167
D.15 SLA Management installation times	168
D.16 Time Results	168
References	171

List of Figures

2.1	<i>One-to-many</i> virtualisation; Multiple OS instances run concurrently on a single server	6
2.2	Many-to-one virtualisation; Multiple servers run as a distributed system, giving the illusion that there is just one server	6
2.3	Difference between the classical network appliance approach and NFV (figure 1 in [3])	7
2.4	NFV Architecture Framework (figure 4 in [7])	11
2.5	Tacker High Level Architecture (figure 1 in ¹⁶)	16
2.6	Sample Tacker VNFD Template	17
2.7	Different types of Virtual Marketplaces	19
2.8	Aptoide Anti-Malware Platform	24
2.9	T-NOVA Architecture Framework (figure 20 in [23])	26
2.10	Orchestrator Platform Architecture and Interfaces (figure 23 in [23])	27
2.11	NFS High Level Architecture (figure 2 in [25])	28
2.12	NFS Web Application Architecture (figure 3 in [25])	29
2.13	Marketplace internal blocks and external interfaces (figure 22 in [23])	31
2.14	SLA Lifecycle (Figure 2-11 in [26])	32
2.15	Billing Module Architecture (figure 12 in [26]) with typical accounting process (figure 1 in [27])	35
2.16	RABC Architecture (figure 2-6 in [26])	36
2.17	Brokerage Platform Architecture (figure 2-8 in [26])	37
2.18	Brokerage Platform Architecture (figure 2-7 in [26])	38
2.19	Stakeholders view from dashboard (figure 2-5 in [26])	38
3.1	VNF Marketplace Architecture combined with OpenStack and SLA Management module	48
3.2	Registration, Login and General Flow of user's tasks with Authentication and Authorisation Module	49
3.3	VNF Lifecycle	51
3.4	VNF Catalog High Level Architecture	51
3.5	Customer's Requests Status	52
3.6	Request Module Architecture	53
3.7	Billing System Module Architecture	54
3.8	SLA Management Module Architecture	56
3.9	SLA Control Module Architecture	57
3.10	Administrator Home Page	58
3.11	Customer Dashboard Menu	60
3.12	Seller Dashboard Menu	61

3.13 Request and Response Header to get Authentication Token from OpenStack . . .	62
3.14 VNF Functional View with Tacker	63
3.15 Requests and Responses to Upload a VNF Image into Glance	64
3.16 VNF instances status transitions	65
3.17 Virtualenv Creation and Activation Procedure	67
3.18 VNF Marketplace Topology	71
3.19 Conclusion of Devstack installation	72
3.20 OpenStack Topology	73
3.21 Security Groups from OpenStack	74
4.1 VNF Marketplace Main Page	76
4.2 Imports of some static files of VNF Marketplace	76
4.3 VNF Marketplace Registration - Phase 1	77
4.4 VNF Marketplace Registration - Phase 2	77
4.5 Registration Request and Response	78
4.6 Customer Dashboard Home Page	78
4.7 Edit Profile Dashboard Menu	79
4.8 Edit Profile Request and Response	79
4.9 VNF Publication in Basic Mode - Phase 1	80
4.10 VNF Publication in Basic Mode - Phase 2	80
4.11 VNF Publication in Basic Mode - Phase 3	81
4.12 VNF Publication in Basic Mode - Request and Response	81
4.13 VNF Publication in Premium Mode - Phase 2	82
4.14 VNF Publication in Premium Mode - Phase 3	82
4.15 VNF Publication in Premium Mode - Request and Response	83
4.16 VNF Marketplace Catalog	83
4.17 VNF Marketplace Catalog with search	84
4.18 VNF Marketplace with search - Request and Request	84
4.19 Customer's VNFs	85
4.20 Start VNF - Request and Response	85
4.21 Customer's VNF Download Response	85
4.22 New Request Dashboard View	86
4.23 New Request - Request and Response	86
4.24 Seller Requests Dashboard View	86
4.25 Customer Requests Dashboard View	87
4.26 Customer Requests - Request and Response	87
4.27 VNF Payment Options	88
4.28 Paypal Dashboard View	88
4.29 Customer Billing Dashboard View	89
4.30 Customer Billing - Request and Response	89
4.31 PDF with VNF Purchase (Customer)	90
4.32 Gmail account with the invoices	90
4.33 SLA Template	91
4.34 Creation of SLA Agreement	92
4.35 SLA Control SLA Information - request and reponse	92
4.36 Customer SLA Dashboard View	93
4.37 Customer SLA Details Dashboard View	93
4.38 Customer Statistics Dashboard View	94
4.39 Seller Statistics Dashboard View	94

4.40	Tacker VNF Catalog - Horizon view	95
4.41	Post VNFD to Tacker - Request and Response	96
4.42	Deploy VNF - Request and Response	96
4.43	OpenStack Networks with a deployed VNF and a Neutron Router	97
4.44	Console with the VNF deployed	98
4.45	Loading time results of static files	101
4.46	Cirros VNF Publication time results	101
4.47	Ubuntu VNF Publication time result	102
4.48	Cirros VNF Withdraw time results - Seller	103
4.49	Ubuntu VNF Withdraw time result - Seller	103
4.50	Cirros VNF Purchase time result	104
4.51	Ubuntu VNF Purchase time result	104
4.52	Cirros VNF Withdraw time result - Customer	105
4.53	Ubuntu VNF Withdraw time result	105
4.54	Ubuntu VNF Withdraw time result	107
4.55	Number of VNFs	107
4.56	Simplicity in purchasing VNFs as Customer	109
4.57	Simplicity in selling VNFs as Seller	109
4.58	Availability of information for the Customer and Seller	110
4.59	Diversification of payment methods (Customer)	110
4.60	Intuitive registration	111
4.61	Statistics evaluation for both actors	111
4.62	Graphical interface evaluation for both actors	112
4.63	Evaluation of communication between Customer and Seller	112
4.64	Overall performance	113
A.1	VNF Marketplace Use Cases	117
C.1	Creation of payment with credit card - Request and Response	141
C.2	Delete VNF (Customer) - Request and Response	141
C.3	Payment successful with credit card - Request and Response	142
C.4	Stop VNF - Request and Response	142
C.5	Purchase VNF in Basic Mode	143
C.6	Purchase VNF in Premium mode - SLA Flavor 1	143
C.7	Purchase VNF in Premium mode - SLA Flavor 2	144
C.8	Purchase VNF in Premium mode - SLA Flavor 3	144
C.9	Download VNF Dashboard View	145
C.10	Payout with credit card - Request and Response	145
C.11	Delete VNF (Seller) - Request and Response	145
C.12	Get Seller's VNFs - Request and Response	146
C.13	Reply to a Customer's Request - Request and Response	146
C.14	Get Seller's VNFs Dashboard View	147
C.15	Get Seller Invoice (PDF)	147
C.16	Seller Billing Dashboard View	148
C.17	Get VNF Instance Status - Request and Response	148
C.18	OpenWRT Deployed	148
C.19	OpenWRT Config Directory	149
C.20	OpenWRT with Firewall configured - Part 1	149
C.21	OpenWRT with Firewall configured - Part 2	150

C.22 OpenWRT Ping to Internet	150
-----------------------------------------	-----

List of Tables

2.1	SLA per Service (Table 2-8 in [26])	31
2.2	SLA Management Module information (Design - Table 2-9 in [26])	33
2.3	Accounting module information (Design - Table 2-10 in [26])	33
2.4	SP dashboard view (Table 2-5 in [26])	39
2.5	FP dashboard view (Table 2-6 in [26])	39
2.6	Customer dashboard view (Table 2-7 in [26])	40
2.7	Characterisation of the analysed marketplaces	43
3.1	Administrator Special Dashboard View	58
3.2	Customer VNF Marketplace Dashboard View	59
3.3	Seller VNF Marketplace Dashboard View	60
4.1	VNF Marketplace installation time measurements	99
4.2	SLA Management installation time measurements	100
A.1	Use Case 1	118
A.2	Use Case 2	119
A.3	Use Case 3	119
A.4	Use Case 4	120
A.5	Use Case 5	120
A.6	Use Case 6	121
A.7	Use Case 7	121
A.8	Use Case 8	122
A.9	Use Case 9	122
A.10	Use Case 10	123
A.11	Use Case 11	124
B.1	Authentication and Authorisation Module Software Requirements	126
B.2	VNF Catalog Module Software Requirements	128
B.3	Request Module Software Requirements	129
B.4	Billing Module Software Requirements	131
B.5	SLA Management Software Requirements	132
B.6	SLA Control Module Software Requirements	132
B.7	General Dashboard Software Requirements	133
B.8	Administrator Dashboard View Software Requirements	134
B.9	Customer Dashboard View Software Requirements	136
B.10	Seller Dashboard View Software Requirements	138
B.11	Analytics Module Software Requirements	138
B.12	OpenStack Software Requirements	139

D.1	VNF Marketplace detailed installation time measurements	167
D.2	SLA Management installation time measurements	168
D.3	Time Measurements of Requests-Responses	170

Listings

D.1	VNF Marketplace Server Requirements	151
D.2	VNF Marketplace Settings File	152
D.3	SLA Management Configuration File	155
D.4	Devstack Configuration File	156
D.5	VNF Marketplace Installation	156
D.6	SLA Management Requirements Installation	156
D.7	OpenWRT VNFD	157
D.8	Loading CPU and RAM Results	158
D.9	VNF Publication CPU Results	159
D.10	VNF Withdraw (Cirros) CPU and RAM Results with seller	161
D.11	VNF Purchase and VNF Withdraw CPU Results with Customer	162
D.12	Stakeholders Actions with CPU Results	163
D.13	Number of VNFs with CPU and RAM Results	166

Abbreviations

AA	Access Control
API	Application Programming Interface
APK	Android Package
APP	Application
APT	Advanced Packaging Tool
B2B	Business-to-Business
B2C	Business-to-Customer
BSS	Business Support System
C2C	Customer-to-Customer
CAPEX	Capital Expenditure
CDN	Content Delivery Network
CDR	Charge Data Record
CLI	Command Line Interface
COTS	Commercial Off-the-Shelf
CP	Connection Point
CPU	Central Processing Unit
CSS	Cascading Style Sheets
DB	Database
DC	Data Centre
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DPI	Deep Packet Inspection
EJB	Enterprise Java Bean
EMS	Elements Management System
ETSI	European Telecommunications Standards Institute
FP	Function Provider
FW	Firewall
GUI	Graphical User Interface
HOT	Heat Orchestration Template
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as-a-Service
IAB	In-App Billing
ICMP	Internet Control Message Protocol
ICT	Information and Communications Technology
ID	Identifier
IDS/IPS	Intrusion Detection System/Intrusion Prevention System
IMS	IP Multimedia Subsystem
IP	Internet Protocol

ISV	Independent Software Vendor
IT	Information Technology
IVM	Infrastructure Virtualisation and Management
JDBC	Java Database Connectivity
JDK	Java Development Kit
JPA	Java Persistence API
JSON	JavaScript Object Notation
JSR	Java Specification Request
JWT	JSON Web Token
KPI	Key Performance Indicator
JS	JavaScript
JSON	JavaScript Object Notation
JWT	JSON Web Token
KVM	Kernel-based Virtual Machine
NAT	Network Address Translation
NF	Network Function
NFS	Network Function Store
NFV	Network Functions Virtualisation
NFV-MANO	Network Functions Virtualisation Management and Orchestration
NFVI	Network Functions Virtualisation Infrastructure
NFVIaaS	Network Functions Virtualisation Infrastructure as-a-Service
NFVO	Network Functions Virtualisation Orchestrator
NIC	Network Interface Controller
NUMA	Non-Uniform Memory Access
OASIS	Organization for the Advancement of Structured Information Standards
OEM	Original Equipment Manufacturer
OPEX	Operational Expenditure
OS	Operating System
OSM	Open Source Mano
OSS	Operations Support System
OVA	Open Virtualization Appliance
OVF	Open Virtualization Format
PDF	Portable Document Format
PNF	Physical Network Function
QoS	Quality of Service
QR code	Quick Response code
RAM	Random Access Memory
RBAC	Role Based Access Control
REST	Representational State Transfer
RFX	Request for X (information, proposal, quotation, bid)
RFI	Request for Information
RFP	Request for Proposal
RFQ	Request for Quotation
RFB	Request for Bid
SDK	Software Development Kit
SDN	Software-Defined Networking
SHA	Secure Hash Algorithms
SIM	Subscriber Identifier Module

SLA	Service Level Agreement
SME	Small and Medium-sized Enterprise
SMTP	Simple Mail Transfer Protocol
SP	Service Provider
SQL	Structured Query Language
SR-IOV	Single-Root Input/Output Virtualisation
TLS	Transport Layer Security
TOSCA	Topology and Orchestration Specification for Cloud Applications
UDI	Unique Device Identification
UDR	Usage Data Record
URL	Uniform Resource Locator
UUID	Universally Unique Identifier
VAT	Value-added Tax
VCPU	Virtual Central Processing Unit
VDU	Visual Display Unit
VIM	Virtual Infrastructure Manager
VL	Virtual Link
VLAN	Virtual Local Area Network
VM	Virtual Machine
VNF	Virtual Network Function
VNFC	Virtual Network Function Component
VNFD	Virtual Network Function Descriptor
VNFM	Virtual Network Function Manager
VR	Virtual Reality
VoIP	Voice over Internet Protocol
WAN	Wide Area Network
XML	Extensible Markup Language
YAML	Yet Another Markup Language

Chapter 1

Introduction

1.1 Context

Analysing the world of telecommunications, it is possible to conclude that telecom networks experience huge advances in network technology. These advances are due to industry telecommunications inventions during the last century. For instance, the first working fiber-optical data transmission system in the 60s [1], the origin of Internet in the 80s and the development of VoIP in the 00s.

Over the last decade, the data rates required by services and applications have increased. Operators have to continuously purchase and operate new physical equipment due to their demand which requires operational staff formation to learn new skills and long deployments of network equipment. To counter this demand, the obvious solution was using the virtualisation approach to telecommunications networks through the implementation of SDN and NFV.

NFV, designed and proposed by ETSI [2], consists in the decoupling of network functions from dedicated hardware devices in order to allow network services, that are now being carried out by dedicated hardware devices, to be hosted on virtual machines. VNFs belongs to the NFV concept and are the virtual appliances required to replace hardware network functions devices. To summarise, NFV is an overarching concept, while a VNF is a component of NFV framework. For this dissertation, VNFs are the main product of the developed Marketplace.

The other emerging paradigm to improve network virtualisation is SDN. SDN allows network operators to program line-speed network devices in software rather than in hardware to respond quickly to changing business requirements [3]. NFV and SDN can complement each other but NFV can be implemented without SDN. To enhance performance and facilitate operation procedures, these two solutions can be combined because NFV can withstand SDN by providing the infrastructure upon which the SDN software relies on [3].

For this thesis, the implementation of the Marketplace consists of multiple business processes, where vendors can sell their VNFs and customers can buy them. This Marketplace has two main components: the Back-End and the Front-End. All components were built from scratch with the help from some available frameworks for developers. There are third party components present in the Marketplace to complement it. For instance, the Marketplace uses a third party server to create templates and agreements for SLAs and OpenStack to deploy VNFs.

The Marketplace is defined as VNF Marketplace and has the potential to support a large number of different VNF types and can be integrated in any environment. Also, since it is only a prototype, all its components can be upgraded to provide a better and more appealing Marketplace. NFV can revolutionise the telco world and, with this Marketplace, the VNF distribution can easily be made.

1.2 Objectives

The purpose of this thesis is to study and propose management and business processes on a Marketplace for VNFs and to validate these processes by means of a prototype. To achieve this main goal, the following activities were performed:

- Study existing Marketplaces to collect different management and business processes;
- Plan use cases, requirements and an architecture for the Marketplace;
- Develop the planned Marketplace (Web Application) to host VNFs;
- Adapt the Marketplace to communicate with OpenStack.
- Install, configure and operate VNFs between the Marketplace and OpenStack;

1.3 Motivation

This master's thesis project proposes an approach to fulfil the need for VNF distribution. Since NFV can contribute to a dramatic change in the telecommunications industry landscape, network operators, enterprises and end users are searching for a way to enter in the NFV world. This Marketplace could be the platform that provides just that because of its ability to distribute VNFs. NFV allows resource optimisation, an improvement in the Agile Procurement service, the reduction of CAPEX and OPEX, significant costs savings (reduced equipment costs), better operational efficiency and greater scalability and flexibility [4].

The Marketplace is expected to have benefits for the customer with the sale of VNFs. Using VNFs will cause a major impact on COTS platforms, more specifically on easily purchased products such as standard operating systems and industry servers [5] and the pretended Marketplace can easily provide them to interested customers.

The proposed Marketplace is not only expected to give access to the NFV world but also allow services and virtual network functions of a variety of developers and companies to be published and negotiated. The Marketplace will allow clients to search for and choose the services and virtual appliances that best match their needs. Lastly, with this Marketplace, VNFs can be easily distributed in different ways and has the special functionality of communicating with OpenStack which is an important tool for NFV and SDN.

1.4 Structure

This document is split in the following chapters:

- Chapter 2 provides the State of the Art that gives an introduction of virtualisation, analyses NFV and OpenStack, provides an approach of the pretended Marketplace for this thesis and explores three different Marketplaces;
- Chapter 3 presents the VNF Marketplace in greater detail including its characterisation and development;
- Chapter 4 introduces and analyses the results of the evaluations used for this master's thesis project;
- Chapter 5 gives conclusions and suggests future work.

Chapter 2

State of the Art

This chapter gives two separated main sections: Theoretical Background and Related Work. Initially, section 2.1.1 describes concepts of virtualisation in section 2.1.1. After that, section 2.1.2 explains NFV by portraying an overview of NFV, the NFV requirements and the NFV architecture framework. Section 2.1.3 explains OpenStack, a software that was used for this thesis in order to deploy and test VNFs. Next, section 2.1.4 introduces the theoretical concept of the Marketplace that is requested for this dissertation. Section 2.2.1 characterise GSMA Marketplace which is a Marketplace platform dedicated to the mobile industry. Section 2.2.2 describes an Android Marketplace, Aptoide. Lastly, section 2.2.3 deepens T-NOVA project, including the Marketplace developed for NFV Functions. These Marketplaces will help to understand the Marketplace's topology and functionality in order to develop the Marketplace for this dissertation.

2.1 Technical Background

2.1.1 Overview of Virtualisation

Virtualisation can be defined as the abstraction of physical resources into logical units and multiple physical resources can appear as a single logical unit [6]. When a single physical resource (host) appears as many logical units, there is an important computer software in the virtualisation called *hypervisor*. An *hypervisor* is responsible for the abstraction of the physical resource and for the creation of a separate OS environment, giving to each VM that is created the illusion of being the only operating system running on the host server [6]. This case is illustrated bellow (figure 2.1).

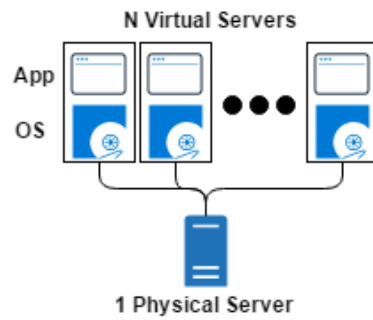


Figure 2.1: *One-to-many* virtualisation; Multiple OS instances run concurrently on a single server

In a telecommunications network context, a good example of a *one-to-many* virtualisation case is a VLAN, where a single physical network is partitioned into many logical ones.

Another event is when multiple physical resources appear as a single logical unit. One example is that of a *load balancer*. In this case, there are multiple physical web servers but all their details are hidden by the *load balancer*, which exposes a single virtual IP address to the web clients. When the web clients connect to the virtual IP address to obtain a service, they have the illusion that there is a single web server but, in fact, the service is using multiple servers [6] (figure 2.2).

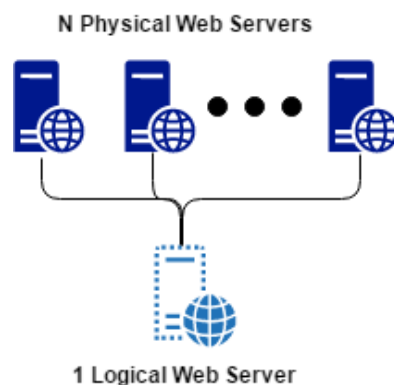


Figure 2.2: *Many-to-one* virtualisation; Multiple servers run as a distributed system, giving the illusion that there is just one server

There are two disruptive concepts that are being explored in the virtualisation field: NFV and SDN. These two concepts are closely related to the area of telecommunications networks since both aim to achieve network virtualisation.

2.1.2 NFV

NFV, as was already referenced in chapter 1, is defined by ETSI. The aim of NFV is to replace and to complement physical network devices with virtual network functions [3]. For instance, is possible to virtualise network functions from network elements that physically exists in traditional networks such as residential nodes or switching elements [3]. With virtualisation of physical network devices, the network equipment vendors will innovate faster and reduce their manufacturing costs [7].

Figure 2.3 illustrates a classic network appliance approach on the left side and a simple NFV architecture approach on the right side. The classical approach show several physical network devices that can be bought and form networks and the NFV approach display the connection between the virtual network appliances and network equipment types. The network equipment is deployed to form a cloud environment in order to VNFs work.

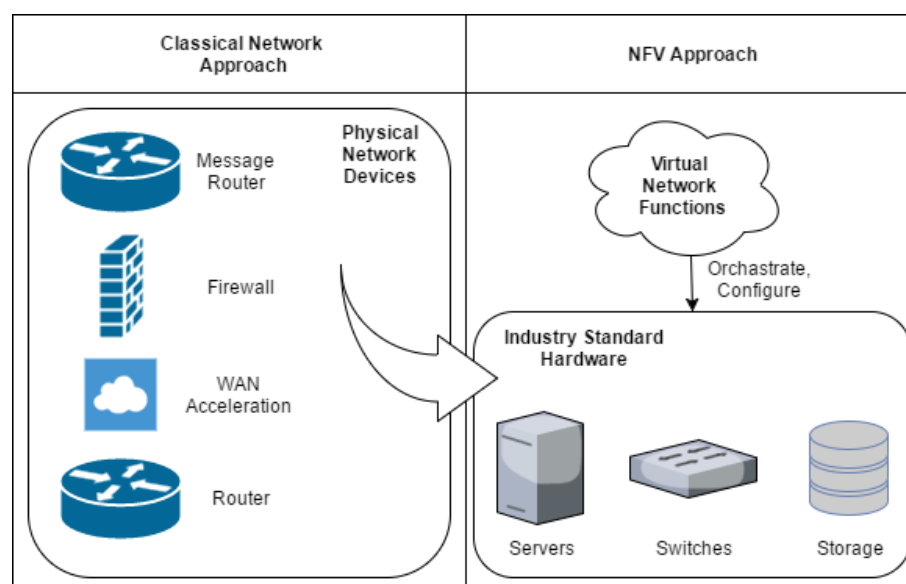


Figure 2.3: Difference between the classical network appliance approach and NFV (figure 1 in [3])

2.1.2.1 NFV Requirements

There are specific requirements put on NFV that facilitate the allocation of infrastructure resources to VNFs [7]. The following requirements, stated by ETSI GS NFV for NFV requirements [8], concentrate in the differences introduced by NFV processes to describe the business level and technical requirements for a NFV framework:

- Portability – Addressing the capabilities to load, execute and move software functions per different standard data centres environments;
- Performance – Classifying qualifications required to describe the infrastructure requirements for specific performance targets of software functions;
- Management and Orchestration – Placing requirements for mechanisms in order to manage and orchestrate the virtual functions lifecycle, the infrastructure resources and the operations performed on those;
- Elasticity – Marking the capabilities to provide an easier way to scale up/down and in/out hardware resources as traffic demands increase/decrease;
- Security – Planning aspects to be analysed as the virtualisation environment can be exposed to external attacks in ways that are not expected in current telecom architectures;
- Resilient and Network Stability – Indicating to the efficiencies demanded to limit disruption and return to normal operation in order to secure service availability and continuity, without functionalities becoming point of failures;
- Service Continuity – Stating the capabilities needed for the continuous delivery of service in conformance with the service specification and SLA requirements;
- Operations – Targeting the requirements needed for automation of operational functions;
- Energy Efficiency – Marking the technical capabilities that will help minimise the energy consumption of large scale virtualised networks;
- Migration and coexistence with existing platforms – Indicating to the requirements to support a transition path from today's networks where non-virtualised networks coexist with virtualised ones without disruption of the services or impacts to the users.

2.1.2.2 NFV Architecture Framework

The NFV architecture framework (figure 2.4), standardised by ETSI, is divided into three main building blocks:

- Virtualised Environment;
- NFV Infrastructure (NFVI);
- NFV Management and Orchestration (NFV-MANO);

2.1.2.2.1 Virtualised Environment

This block is where the VNFs and EMS are. VNFs are software implementations of network functions [7] such as IMS and CDN which can be placed at and migrated to where they are needed most. One VNF can be extended into multiple components like, for example, over multiple VM where each one is responsible for hosting a single component [7]. These components are referred as virtual network function components (VNFC) [9]. VNFs are typically connected into service chains (an ordered set of VNFs designed as SFC) to provide network service (NS) [10]. EMS is responsible for the VNFs functional management as well as the configuration of network functions, security management and flaw management.

2.1.2.2.2 NFVI

The NFV Infrastructure gives the hardware and software components to build the environment required to support VNFs [7]. Figure 2.4 illustrates hardware resources that are assumed to be COTS hardware including computing, storage and network hardware. NFVI creates a virtualised layer (contains hypervisors and VMs/containers) to abstract and to virtualise the hardware resources in order to decouple the VNF software from the underlying hardware [7].

Since one of the NFV objectives is to allocate VNFs in a cloud, NFVI is composed of NFV infrastructure points-of-presence (NFVI-PoPs) [11]. NFVI-PoPs are where the VNFs, including resources for computation, storage, and networking, are deployed¹. NFVI networks interconnect the computing and storage resources contained in an NFVI-PoP¹. This may include specific switching and routing devices to allow external connectivity¹.

¹ <https://www.sdxcentral.com/nfv/definitions/nfv-infrastructure-nfvi-definition/>

2.1.2.2.3 NFV-MANO

This last block covers the management and orchestration segments of physical and software components present in the NFV architecture framework. In other words, NFV-MANO supports the Virtualised Environment and the NFV Infrastructure [7] components. NFV-MANO has also a connection with the OSS/BSS in order to implement the requirements set by them and to allow NFV be part of an existing network-wide management landscape [7]. The NFV-MANO block is formed by three specific functional elements (figure 2.4): NFV Orchestrator, VNF Manager and Virtual Infrastructure Manager.

The Orchestrator is responsible for the orchestration of NFVI resources through various VIMs completing the Resource Orchestration functions and the lifecycle management of network services across multiple VNFMs, fulfilling Service Orchestration Services [12]. Also, NFVO communicates with the VNF and NS catalog [12].

The VNF Manager has the role in managing VNFs based on dynamic service demands in order to provide scalable and automated VNF lifecycle management [12]. Additionally, VNFM does the FCAPS (Fault, Configuration, Accounting, Performance and Security Management) for the virtual part of the VNF. The difference between VNFM and EMS is the VNFM does the VNF management of its virtual components while EMS does the management of functional components. For instance, in case where Mobile core is virtualised, the EMS will do the management of the functional part (for example issues related to mobile signalling), while the VNFM will do the management for the virtual part (bringing up the VNF itself)².

The last functional element is the Virtual Infrastructure Manager and it has the functionality of controlling and managing the NFVI hardware resources, generally within one Operator's Infrastructure domain [12]. The VIM can be deployed by multiple VIMs in order to administrate a certain type of NFVI resource or multiple ones (NFVI-PoPs) [12]. This block also exposes APIs for managers to provide better control and administration of NFVI resources.

² <http://www.telecomlighthouse.com/a-cheat-sheet-for-understanding-nfv-architecture>

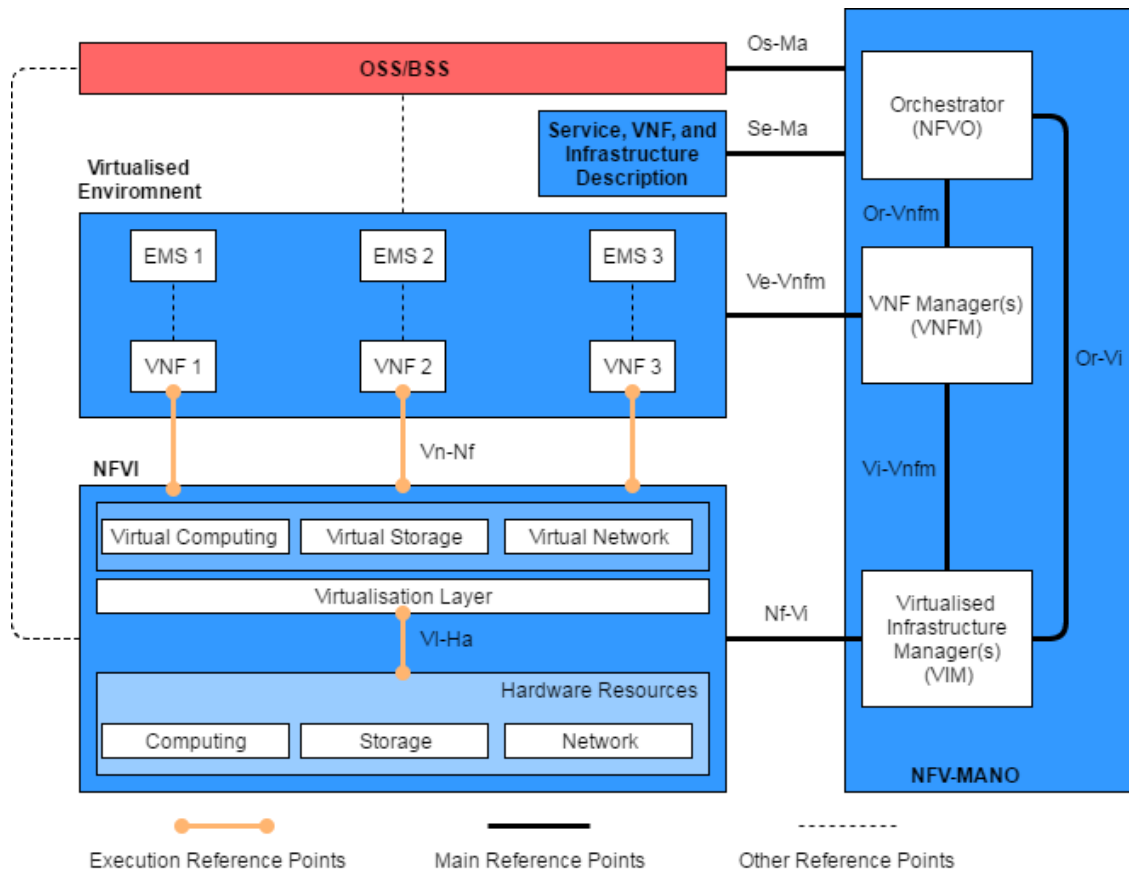


Figure 2.4: NFV Architecture Framework (figure 4 in [7])

The Service, VNF and Infrastructure Description block contains several templates describing the deployment and operational behaviour of Network Services and VNFs [13]. This block is where NS and VNF catalogs are present and each template is stored in the respective catalog [14]. The templates are referenced as descriptors (NSD and VNFD) and each one is associated to one VNF or NS [9]. Also, this block has VNFFG Descriptors present in NS catalog to allow NSs create a chain between the VNFs [12]. In other words, VNFFGD is a deployment template which describes a topology of the Network Service or a portion of the Network Service, by referencing VNFs, PNFs and Virtual Links that connect them [12].

In conclusion, these building blocks were designed to be adapted to an existing OSS/BSS environment [7] and to the reuse of existing solutions such as cloud management systems. As was explained in each building block, every functional element has its own responsibility to others elements which are represented as reference points in the NFV architecture framework.

2.1.3 OpenStack

Previously, companies had expensive data centres, deploying a server could take a long time and there was no option to react quickly. To solve this problem, it was created a solution called IaaS Cloud Computing³. IaaS Cloud delivers self-deployment of virtual machines in a cloud and each virtual machine can be used as a server⁴. Also, IaaS replaces the traditional data centre by reacting very quick and in a flexible way to business needs. For the customer perspective, IaaS has the advantage of allowing pay for used computing time only³.

One IaaS solution that was created is OpenStack. OpenStack started in 2010 as a joint project by Rackspace and NASA⁵. It is an open source solution and is directed by OpenStack Foundation. Not only OpenStack Foundation works in the development of OpenStack but also many IT vendors⁶. It is a win-win situation since IT vendors help OpenStack deliver flexible computing and, in retribution, their products become compatible with OpenStack.

OpenStack uses several services to provide a flexible and simple implementation of an IaaS Cloud solution. These services have different statuses that are defined by OpenStack Foundation [15]:

- Integrated - fully accepted services in OpenStack environment;
- Incubated - services candidates to be integrated;
- Third-party - third-party services which are not official;

OpenStack services offer three ways of communication: RESTFul API which can be easily used by developers that are creating Apps or others services, command-line interface (CLI) for admins and Horizon for a browser interface⁵. The next section introduces several official services which were used in this thesis.

³ <https://azure.microsoft.com/en-gb/overview/what-is-iaas/>

⁴ <https://www.ibm.com/blogs/cloud-computing/2014/02/what-is-infrastructure-as-a-service-iaas/>

⁵ <https://en.wikipedia.org/wiki/OpenStack>

⁶ <https://www.OpenStack.org/foundation/companies/>

2.1.3.1 OpenStack Services

2.1.3.1.1 Horizon

Horizon is a service that delivers the same functionality as the CLI but with a Web User Interface (Dashboard). The Dashboard can be used by cloud administrators, tenant administrators and tenant users⁷. Users can use the Dashboard to deploy virtual machines and, with the same interface, cloud administrators can configure virtual networks. Horizon allows many options but, for full control of OpenStack, the command line access is required [15].

The main advantage of Horizon is making OpenStack much easier to use. With this service, it is simple to manage and use most part of OpenStack functionalities. In case of some error or malfunction in the system, trying to use Horizon is not the best option since it does not give enough information about them [15].

Horizon is a great and optional tool to OpenStack users depending on what their preferences are but, in case of something wrong happens, users will need to know how to use CLI to correct it.

2.1.3.1.2 Keystone

Keystone is a service about identity (authentication, authorisation and accounting). It allows users and roles to access specific services within specific tenant environments⁸. With Keystone, users can log into Horizon or OpenStack console and start using OpenStack services. Every user has a specific role, for example, can be a member or an administrator.

Keystone works with authentication tokens and it is a central point to get information about OpenStack services⁹. For instance, if a user wants to know about the installed services, he just needs to run a keystone command in CLI to get that information. Also, Keystone stores all identity information in the database (MariaDB in case of using Linux distribution). In conclusion, Keystone is an important service of OpenStack. If Keystone does not exist or it does not work, every OpenStack services will not communicate with each other.

⁷ <https://docs.OpenStack.org/developer/horizon/intro.html>

⁸ <https://docs.OpenStack.org/developer/keystone/>

⁹ <https://wiki.OpenStack.org/wiki/Keystone>

2.1.3.1.3 Cinder

When a user spins off an OS from an image, it will have persistent writes that cannot be guaranteed¹⁰. In other words, when the user spins off the OS, the write will be local to the machine that is hosting the instance and if it is moved around, it will be lost. With Cinder, that will not happen since it delivers a block storage service.

Cinder provides block storage services for persistent storage which is ideal for cloud users [15]. Usually, Cinder is added to the virtual machine as a separate disk and uses an object store as its physical storage backend¹¹. This allows every time the virtual machine is moved to the cloud, the object storage does not lose any write or storage during the process.

2.1.3.1.4 Glance

Glance is an image service in OpenStack. It allows instances to be deployed from an image without being installed since installation is not flexible in cloud environments¹². Glance images work together with flavors which are templates to define the hardware properties of the instances that are going to be deployed [15].

Default images can be downloaded from the Internet for different OS. For instance, one of the most common images for testing used in OpenStack is Cirros (it has only 13 MB). Glance also allows users to create their own images according to their specific needs. In short, Glance is an important service in order to deploy virtual machines. Taking NFV in consideration, all VNFs need to be deployed in a cloud and Glance is a great solution.

2.1.3.1.5 Nova

Nova is a service that provides a cloud computing controller, supporting a wide variety of hypervisors and emulators¹³. For example, the default hypervisor which Nova supports is KVM since OpenStack is Linux oriented.

Nova is not the hypervisor itself but has an interface which communicates with them. Nova service is responsible for the direct placement of virtual machines [16]. If a virtual machine needs to be allocated to an hypervisor, Nova will perform that task. In conclusion, Nova provides scalable, on demand, access to compute resources which is an important service in a cloud environment.

¹⁰ https://www.vmware.com/support/ws3/doc/ws32_disks2.html

¹¹ <https://docs.OpenStack.org/ocata/config-reference/block-storage/block-storage-overview.html>

¹² <https://docs.OpenStack.org/developer/glance/>

¹³ <https://github.com/OpenStack/nova>

2.1.3.1.6 Neutron

Neutron service is defined as the provision of software defined networking in OpenStack. It allows creating logical networks (broadcast domains) and routers (L3 agent) [16]. Also, Neutron has the ability of separate traffic from different tenants in a multi-tenant cloud.

This service has other components as well¹⁴:

- Neutron Server - provides connection to Neutron API and its extensions. Also, it enforces the network model and IP addressing of each port;
- Plugin Agent - manage local virtual switch (vswitch) configuration on compute node;
- DHCP Agent - delivers DHCP service to all tenants networks and it is responsible for maintaining DHCP configuration.

With Neutron, it is possible to create networks in a cloud which virtual machines can be connected. For VNF testing, it will be an essential tool to test connectivity between different VNFs that are not connected to each other or are connected in different networks.

2.1.3.1.7 Heat

Heat is the definition of orchestration in OpenStack. It orchestrates collections of virtual machines that belong together (stacks) to make deployment easier¹⁵. This orchestration engine, in order to orchestrate and launch the stacks, needs to read a Heat Orchestration Template (HOT), in form of text file, that can be treated like a code.

HOT is a template that defines all the information of services and applications hosted on the cloud environment¹⁵. In other words, it specifies what makes up the stack. HOT contains 5 different components:

- Description - describes the purpose of the template;
- Resources - contains the objects that are created;
- Properties - Specifies of the template, such as the flavor, the key pair and the image;
- Parameters - Properties of specific resources;
- Output - What is passed back to the user (can be through the CLI or Horizon).

It is important to notice that HOT can also be defined to one virtual machine. The objective of Heat is, instead of taking all the procedures to deploy one or more virtual machines, the user only needs to submit a HOT file to the orchestration engine and the service will do all the deployment processes.

¹⁴ <https://docs.OpenStack.org/security-guide/networking/architecture.html>

¹⁵ <https://wiki.OpenStack.org/wiki/Heat>

2.1.3.1.8 Tacker

Tacker is an OpenStack service that replaces the VNFM and NFVO from NFV architecture. It has a generic VNFM and NFVO to deploy and operate VNFs and network services, respectively¹⁶. Also, Tacker uses OpenStack as NFVI and VIM. Figure 2.5 illustrates the Tacker architecture for a better understanding.

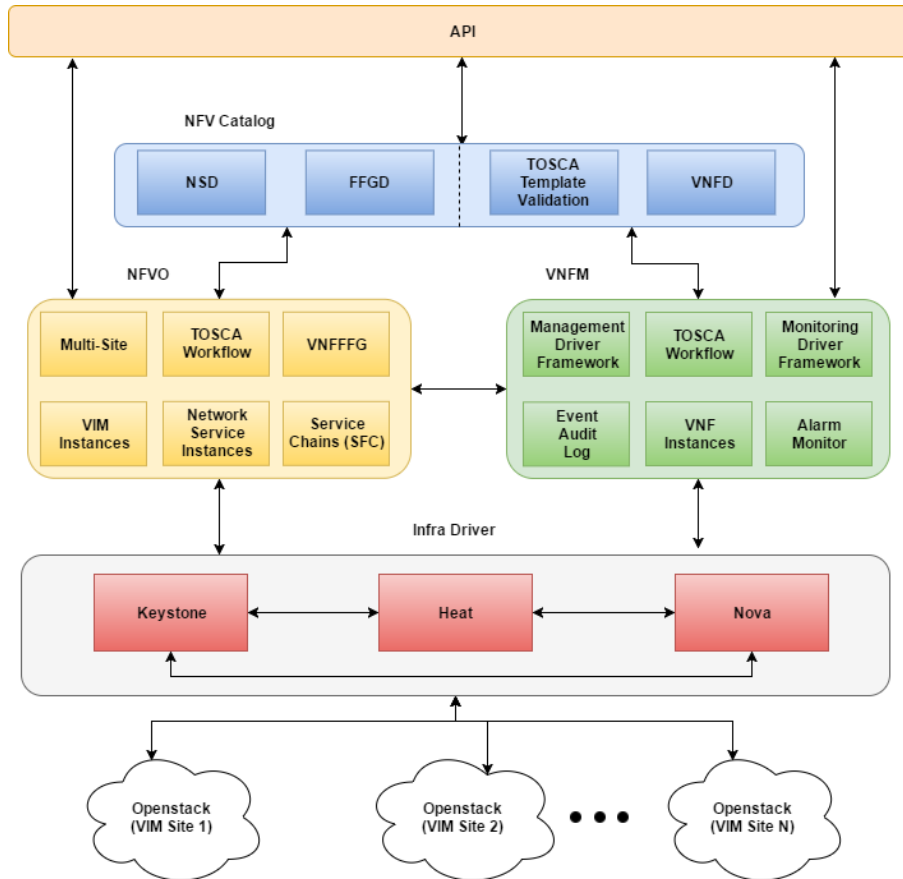


Figure 2.5: Tacker High Level Architecture (figure 1 in¹⁶)

According to figure 2.5, Tacker can deploy VNFs and services to multiple VIMs. To monitor the VNFs, Tacker uses the Monitoring Driver Framework. Since a VNFM needs to manage the VNFs, Tacker implemented a Management Driver. From Tacker Github¹⁷, it is possible to observe that there are two different types of management drivers and three types of monitoring drivers. For the management drivers, there is the noop driver, normally used for VNF testing with Cirros image and the OpenWRT driver for OpenWRT image. For the case of monitoring drivers, exists the Ceilometer driver which is responsible for connecting with Ceilometer service (Telemetry service from OpenStack), the HTTP Ping driver which will ping the VNF from its URL and port and, lastly, the Ping driver which will send ICMP packets using the ping command from Linux utils.

¹⁶ <https://wiki.OpenStack.org/wiki/Tacker>

¹⁷ <https://github.com/OpenStack/tacker>

Tacker uses Heat for deploying VNFs which means that all the VNF descriptors have a template form that Heat can read. Regarding this case, since Heat has a template translator¹⁸, Tacker VNFDs are in TOSCA standard. Tacker VNFD uses TOSCA because is an OASIS standard language to describe the VNF topology (considering the VNFs), their components, relationships, and the processes that manage them [17]. Any user that wants to deploy a VNF in OpenStack, needs to upload the image of the VNF and a TOSCA VNFD file (with YAML format) containing all the VNF information. Figure 2.6 shows a sample of a Tacker VNFD template.

```
tosca_definitions_version: tosca_simple_profile_for_nfv_1_0_0

description: Demo example

metadata:
  template_name: sample-tosca-vnfd

topology_template:
  node_templates:
    VDU1:
      type: tosca.nodes.nfv.VDU.Tacker
      capabilities:
        nfv_compute:
          properties:
            num_cpus: 1
            mem_size: 512 MB
            disk_size: 1 GB
      properties:
        image: cirros-0.3.5-x86_64-disk
        availability_zone: nova
        mgmt_driver: noop
        config: |
          param0: key1
          param1: key2

    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: net_mgmt
        vendor: Tacker
```

Figure 2.6: Sample Tacker VNFD Template (figure in¹⁹)

In the template, it is possible to observe VDUs which corresponds to a VM (VNFC), the VDU properties, Connection Points and Virtual Links. Connection Points are used to connect the internal virtual link or the outside virtual link (virtual NIC or a SR-IOV) and each one needs to bind a VDU²⁰. Virtual Links provide connectivity between VDUs and, in this case, the VDU is connected to a network²⁰. In conclusion, the Tacker service will be crucial for this dissertation since gives the functionality to manage and deploy VNFs.

¹⁸ <https://github.com/OpenStack/heat-translator>

¹⁹ https://docs.OpenStack.org/developer/tacker/install/getting_started.html

²⁰ https://docs.OpenStack.org/developer/tacker/devref/vnfd_template_description.html

2.1.4 Marketplace

The focus of this dissertation is the creation a Marketplace of VNFs. In order to develop the Marketplace, it is necessary to analyse the meaning of the Marketplace and the connection between the procurement process and NFV.

In the context of Marketplace for this dissertation, it can be defined as the virtual location where the customers can compare and buy VNFs that pleasure their needs²¹. The act of finding, acquiring, buying products or services from an external source, in this case from the Marketplace, is called procurement²².

The Marketplace of VNFs needs to allow network operators set up VNFs to offer them to their customers. With the Marketplace, is possible to provide VNFs to customers and, consequently, download and install them into their virtual environments. For that, it requires to address challenges such as [18]:

- Procurement from vast ecosystem including traditional suppliers and new entrants;
- Ability for separate procurement from different suppliers of NFV platforms and products;
- Simplify current procurement processes;
- Inconsistent catalogs of NFV products and platforms.

There is a wide range of different types of virtual Marketplaces that have been created. For this State of the Art, in section 2.2.1, section 2.2.2 and section 2.2.3 are analysed three types of Marketplaces to provide a better understanding. Also, these three Marketplaces are linked to telecom and/or software applications. The Marketplaces have B2B and B2C solutions that are the same pretended for this dissertation. Figure 2.7 illustrates some examples of virtual Marketplaces including the Marketplaces studied for this topic.

²¹ [https://en.wikipedia.org/wiki/Market_\(place\)](https://en.wikipedia.org/wiki/Market_(place))

²² <http://www.businessdictionary.com/definition/procurement.html>

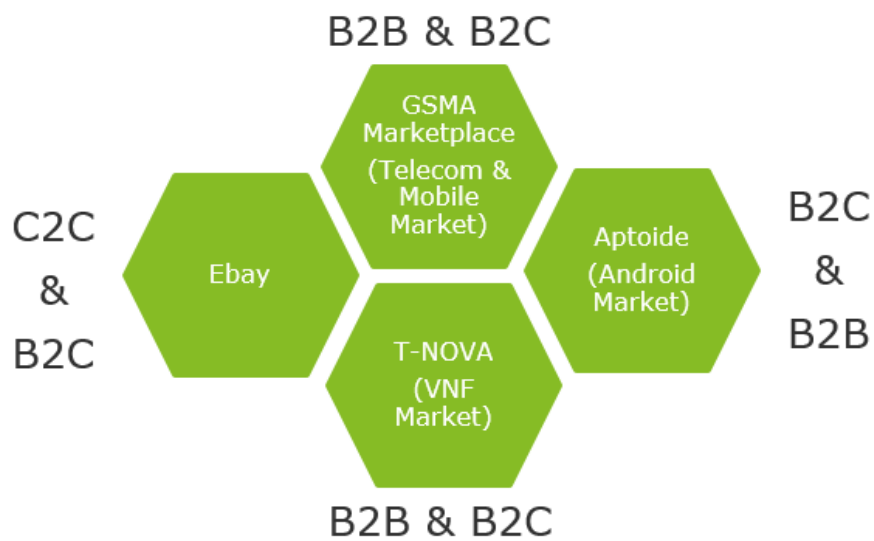


Figure 2.7: Different types of Virtual Marketplaces

Since there are already vendors selling their VNFs, this Marketplace has to adapt or innovate to the way they are sold. Normally, VNF vendors sell their VNFs with license fees. This provides security and validation. Also, there are vendors that associate a license per VNF, which means the license A only works in the VNF image A. For instance, if a user requests a trial version of Cisco Cloud Service Router 1000v (vRouter), Cisco forces the user to deploy the VNF, get the UDI serial number and, at Cisco Website, the user needs to register the product by inserting the UDI. After that, the user gets the license to insert into the VNF.

Finally, section 2.3 concludes the State of the Art. It is mentioned limitations and similarities of the three Marketplaces.

2.2 Related Work

2.2.1 GSMA Marketplace

The GSMA Marketplace is an online commerce platform dedicated to the mobile and telecom industry. It enables buyers and sellers being connected to buy and sell all types of products and services such as SIM cards and cell phones²². Using this online Marketplace will improve the efficiency of development and procurement process²³.

Before addressing GSMA Marketplace's advantages and functionality, it is important to explain business and management abbreviations (RFX, RFI, RFP, RFQ and RFB) that are used in the GSMA Marketplace. RFX is a catch-all term that captures all references to RFI, RFP, RFQ and RFB²³. In the above context, RFX is a document that seeks information (RFI), a proposal (RFP), a quotation (RFQ) and a bid (RFB)²⁴. RFI identifies lists of new potential suppliers or determines if current suppliers are willing or able to supply a new product or service²⁵. RFP is a request for proposal and it is sent to interested suppliers who have been pre-qualified as a potential source of a product or service for additional details on the product or service²⁵. RFQ aims to when the buyer knows what he wants but needs information on how sellers would meet their requirements and how much it will cost²⁶. Lastly, RFB is practically the same as RFQ but, instead of request requirements and costs of products and services for the final decision, it is used to formalise a bid to the sellers²⁷.

With the GSMA Marketplace, the buyer's advantages are the free access to the Marketplace; the reduce time, cost and travel to purchase; simplify and shorten the procurement process; competitive offers and bids; and, finally, a research tool to find potential suppliers or partners²³. In the case of the sellers, they benefit with a cost-effective way of winning their business by reducing time, cost and travel to sell, they can promote their products and services to buyers, they can simplify sales cycle and research new markets and competitors²³.

This Marketplace enables buyers to post a new RFX/RFP on GSMA Marketplace's display board, connect with sellers, find new products and services suppliers and watch a demo on how to improve their company's procurement²³. The sellers can instantly bid on a RFX from GSMA Marketplace's display board, connect with buyers, find new sales leads, watch a demo on how to win more costumers and open a seller account²³.

²² <https://www.gsmapmarketplace.com/faq/>

²³ <https://www.gsmapmarketplace.com/overview/>

²³ <http://www.sourcinginnovation.com/glossary/RFX.php>

²⁴ <http://searchcrm.techtarget.com/answer/What-does-RFX-mean>

²⁵ <http://sourcinginnovation.com/wordpress/2006/06/13/rfx-defined/>

²⁶ <https://www2.humboldt.edu/its/node/1633>

²⁷ <http://413x4nd3r.blogspot.pt/2012/10/rfp-vs-rfb-vs-rfq-vs-rfi.html>

The GSMA Marketplace has several types of companies for the "buyer" role and for the "seller" role. From the information given by GSMA²², the types of companies that fulfil the buyers are: mobile network operators, mobile virtual network operators, fixed and satellite network operators, network infrastructure players, device manufacturers and other companies that buy from telecom companies. In the case of the sellers, the companies are: network equipment providers, app developers, device manufacturers, silicon and hardware components, mobile network operators, consultants and outsourcing specialist, mobile virtual network operators, Platform/Service Providers, OSS providers, application and SPs, system integrators and other companies that sell extensively into the telecoms industry. This Marketplace does not restrict only to the types of companies presented above and it allows more companies to enter in the online commerce platform.

Buyers have a free account in the GSMA Marketplace but sellers have to choose between three accounts types with different costs²²:

- Basic Account: an entry level option with a Company Directory listing and a basic storefront;
- Gold Account: an enhanced level of full RFX participation, preferred listing on the Company Directory and improved storefront capabilities;
- Platinum Account: the premium experience enables you to contact buyers directly and also includes a multimedia storefront with more sales collateral and video, priority listing in the Company Directory and additional user licenses.

The buyers can also pay an additional fee to post live opportunities in the market²³ and to get additional information related to such things as patents and trademarks for the products they are considering purchasing²⁸.

The GSMA Marketplace has a feature called Research Centre which allows both buyers and sellers evaluate potential business partners, research competitors, identify the market opportunity and risks, explore product/service categories and investigate trends and developments²⁹. The Research Centre has also features that provide the activities and benefits for buyers and sellers such as an innovate research tool (Web Visualisation Tool), proprietary sources that allows analysis on a variety of topics (Press Releases, News, Published Reports, Social Media), collaborative research capabilities, a research project curation and proactive research notifications²⁹. The Web Visualisation Tool has a Heatmap which allows users to quickly limit their search pages that are relevant to their needs²⁹.

²⁸ <https://www.internetretailer.com/2015/05/08/new-gsma-marketplace-emerges-b2b-portal-mobile>

²⁹ <https://www.gsmamarketplace.com/researchcentreoverview/>

In conclusion of this section, the GSMA Marketplace has several features that can be implemented in the proposed Marketplace for this thesis. Although the GSMA does not give any type of information of the GSMA Marketplace architecture framework and how all these features and mechanisms are implemented in a more deeper way, it gives a background that the marketplace architecture framework is not all the focus for this thesis. It is important to focus on the buyer and the seller side in order to understand their needs and to provide them with an appealing, efficient and dynamic marketplace.

The different types of accounts for sellers are understandable to be implemented in the Marketplace because it is a way to support all the service that GSMA Marketplace provides but, for this dissertation, it is a topic that is unnecessary. One of thesis goal is to give buyers the best way to compare and buy VNFs and to sellers present all their products/services in a dynamic and efficient way.

2.2.2 Aptoide

Aptoide is an Android Application Store where users can download apps for their Android devices³⁰. It has a distributed approach which means that has multiple stores managed by partners instead of a central store (like Google Play)³⁰. In order to work with multiple repositories, Aptoide was inspired by the APT³¹ packaging manager to search for sources where the application is stored (e.g. with apt-get)³². Today, Aptoide has implemented several versions of it such as Aptoide TV for Android Smart TV, Aptoide Kids for children devices, Aptoide Lite (lighter version in Android device processing and storage) and a VR App Store³².

As a user, it is possible to create a free account to manage an Android app store³³. After that, the users can upload, backup and manage apps into the store, see the statistics of their apps downloads among others KPIs, set store to public or private, generate a QR code to share their stores to friends or to twitter account, manage store appearance, set favorites apps and stores, insert apps/stores/users into a blacklist and manage their own Editors Choice and Revenue Share³⁰. The management of their own apps has several features. The user can distribute them as was explained above, see the statistics of one specific app, edit them, and set a price for them (can be free)³⁰.

³⁰ <https://www.youtube.com/user/Aptoide>

³¹ <https://wiki.debian.org/Apt>

³² <https://en.wikipedia.org/wiki/Aptoide>

³³ <https://www.aptoide.com/partners>

The benefits of owning an Android market with Aptoide are³⁰:

- Easy to customise;
- Fast and scalable;
- Excellent and flexible conditions;
- Competitive Cost.

There are 3 types of main actors in Aptoide: Developers, Partners/Subscribers and clients.

The Aptoide's client role is very similar to the client role in Google Play. The client is a simple user who can download apps present in the stores, review apps, use the social timeline feature to see friend's downloaded apps, comment and/or like friend's activity, search for apps even in others stores like, for instance, search for Facebook App and download it at the official Facebook Website and, finally, he can use the downloader manager to see the downloaded app's history³⁰.

The Developer is responsible for creating the apps and distribute them into the Aptoide App Market. In order to do that, the Developer can upload apps through Aptoide Website, Aptoide Uploader App and Aptoide Backup App³⁴. After the upload, Aptoide will certify the app only if it is free from malware or virus³⁰.

Finally, the Partners focus on creating their own android market with specific apps and distribute them in the Aptoide community. Everyone can be a partner and, normally, who wants to be a partner are OEM, telcos and integrators³³. In terms of subscription, the Partners have three choices/models³³:

- Free – create free account and store, personalise and manage the store, select own or Aptoide brand for the store and upload apps into the store;
- Freemium – Same functionality as the Free model but the Aptoide content in the store is free and the Partner content is premium. The clients can browse free content but have to subscribe to see the premium content. The subscription lifecycle is managed by the Partner. It has a revenue share with Aptoide (store publicity);
- Premium – Also have the same requirements and functionality from the Free model but the users can download the apps only if they are subscribed to Premium Partner. The subscription lifecycle is managed by the partner and, per active subscriber, the Premium partner has to pay to Aptoide only once.

³⁴ <https://aptoide.zendesk.com/hc/en-us/categories/115000018466-Getting-Started>

Accordingly to Aptoide³³, Partners have 50% of Ads revenue and the paid apps revenue are distributed to Partners (10%), Developers (75%) and Aptoide (10%). It has also a personalised customisation for the revenue but annual fees are applied³³. The system which is responsible for this distribution is called Aptoide Open IAB³³. The Aptoide Open IAB allows to sell digital content from inside applications and offers unique content to users reaching higher engagement and long term value³⁵. It is also possible to use the service to sell a wide range of content, including downloadable content such as media files or photos, virtual content (e.g. game levels or potions), premium services and features, and more³⁵. With Aptoide IAB, the Developers have higher payout rates up to 85%, the localised payment options help the users reach newcomers in emerging markets, has a plugin for Unity and an easy implementation through API³⁶.

The security system in Aptoide³⁷ is illustrated in figure 2.8. This system exists in order to prevent apps with malware or virus since everyone can install APK through unknown sources. A report from Malwarebytes [19] explains in between June and November 2016, over 1 billion malware on nearly 100 million Windows and Android devices were detected. The report focused only on this numbers to analyse what malware had more impact on users (Ransomware). This numbers are notable and demonstrate the danger of installing apps without Google or Aptoide trust stamp. Aptoide has also a non-automated system (internal team) that verifies all the apps at the Aptoide Android Market to make sure everything is safe and legal³⁰.

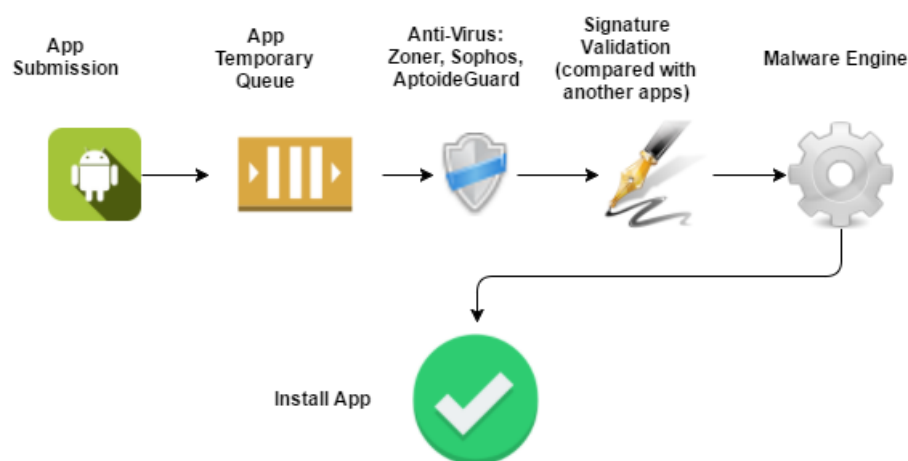


Figure 2.8: Aptoide Anti-Malware Platform

³⁵ <https://developer.android.com/google/play/billing/index.html>

³⁶ <https://www.aptoide.com/page/iab>

³⁷ <https://aptoide.zendesk.com/hc/en-us/communities/posts/115000205943-Need-help>

The piracy and plagiarism are important facts to take in consideration. In several reports and complaints³⁸, there are certain users in the Aptoide community that upload apps from another's Developers. For instance, exists user's uploaded paid apps of Google Play Store in Aptoide for free³⁹. The Aptoide internal teams do not only focus on preventing apps with malware but also in piracy³⁰.

In conclusion, Aptoide is a good alternative to Google Play Store. It gives another perspective of Marketplace and how it interacts with the Android world. For this dissertation, Aptoide has very similarities since the users will download VNFs like Android apps. Both are software applications but with different goals and environments. Since the VNF Marketplace will cover all VNFs from developers and companies, it is important to focus on testing and security to guarantee secure and functional VNFs to customers.

2.2.3 T-NOVA: Marketplace for NFV Functions

T-NOVA, NFaaS over Virtualised Infrastructure, is a European FP7 Large-scale Integrated Project focused on NFV [20]. The primary goal is the design and implementation of an orchestrator framework for the automated provision, configuration, monitoring and optimisation of NFaaS over virtualised network and IT infrastructures [21].

T-NOVA allows network operators to deploy VNFs for their own needs and to offer them to interested customers [20]. It also gives to customers a way of eliminating the need of acquiring, installing and maintaining specialised hardware at their premises by providing Virtual Network Appliances such as gateways, firewalls, proxies, IDS/IPS, traffic analysers and transcoders [20]. Finally, T-NOVA has a Marketplace for NFV [20] which will be analysed to help in the future development of the Marketplace for this thesis.

Figure 2.9 illustrates the T-NOVA architecture with three layers [22]:

- Marketplace;
- Orchestrator Platform;
- ICT Platform (IVM).

³⁸ https://www.reddit.com/r/androiddev/comments/2u3aav/apparently_my_app_is_available_for_free_at_aptoide

³⁹ <https://www.b4x.com/android/forum/threads/aptoide-piracy.40186>

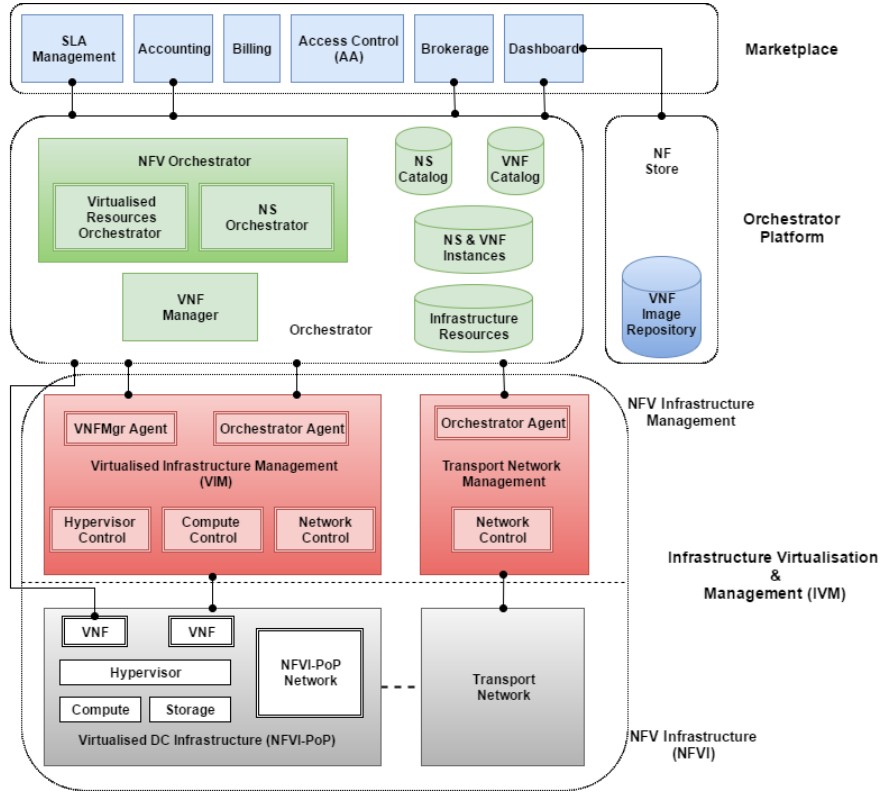


Figure 2.9: T-NOVA Architecture Framework (figure 20 in [23])

2.2.3.1 ICT Platform

The Infrastructure Virtualisation and Management layer was adapted from ETSI's NFVI and NFV-MANO to achieve the necessary performance for high traffic volume or delay intolerant network functions [20]. The ICT Platform is composed of two blocks: NFVI and NFV Infrastructure Management. NFVI has a Virtualised DC Infrastructure where are located the VNFs, hypervisors, NFVI resources and a WAN. NFV Infrastructure Management has a VIM (section 2.1.2.2.3) responsible for controlling and managing Virtualised DC Infrastructure and a WAN Infrastructure Connection Management to support and manage the WAN. To guarantee the best deployment of NFV and SDN at the cloud environment, T-NOVA added more intelligence into orchestration process. This means T-NOVA exposed the underlying physical and virtual landscape of the computing/storage and networking environment into which a VNF must be deployed and a SDN must manage [20]. VNF placements were focused by T-NOVA because the allocation decisions in virtualised environments should have a certain attention to avoid effects such as "noisy neighbours" that can adversely affect co-located VNF's performance [20, 24]. Finally, T-NOVA highlighted the context location for VM instantiation in order to avoid excessive latency and network overload due to inefficient traffic redirection to remote VMs [20].

2.2.3.2 Orchestration Platform

The Orchestration Platform is responsible for addressing the automated management and operations of networking and IT resources to accommodate VNFs, lifecycle management and deployment of VNFs and Network Services [20, 23]. To enable this capability, the orchestrator needs to spread cloud assets throughout the network to not restrict them into data centres [20]. Figure 2.10 shows the Orchestrator Platform modules and interfaces. The Orchestrator Platform itself was adapted from ETSI's designed NFV architecture with the addition of some blocks. The connection between the Orchestrator and the Marketplace is the functional entity which establishes the business and operational management of T-NOVA [23]. The NFVO is responsible for acting as an application that interacts directly to the Marketplace and it orchestrates all the incoming requests towards the others elements of the architecture [23]. The repositories present in the NFVO support operations procedures describing the available VNFs, NSs and the resources in the virtualised infrastructure [23]. The VNFM, already introduced in section 2.1.2.2.3, is responsible for managing VNFs with specific lifecycle management procedures.

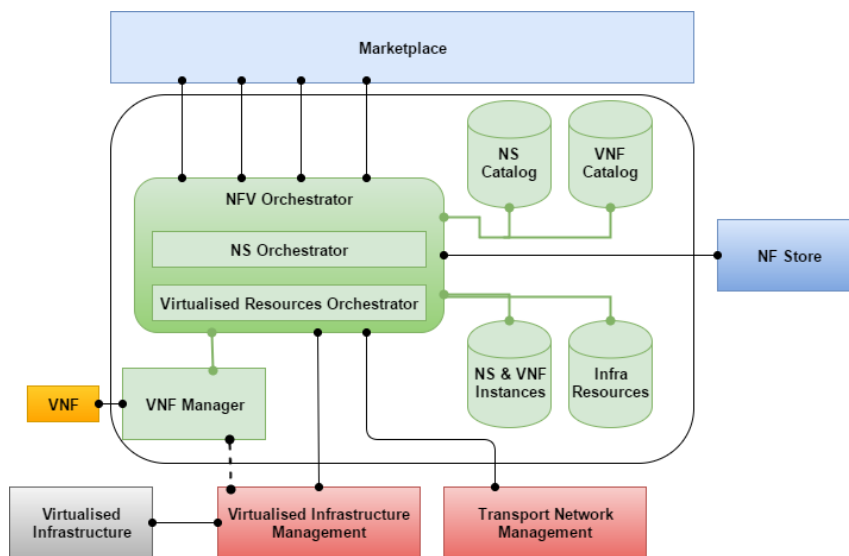


Figure 2.10: Orchestrator Platform Architecture and Interfaces (figure 23 in [23])

2.2.3.3 Network Function Store

It is important to introduce the Network Function Store to understand a part of the T-NOVA Marketplace functionality. The NF Store contains network functions images and metadata descriptions by several third-party developers and it allows costumers to select the VNFs, download and install them in their existing connectivity services and configure them according to their needs [20, 25]. It has a connection to the Marketplace, through the Brokerage Platform, to permit services requests and initiation from the clients [25]. In figure 2.11 is possible to observe a high level of the NS Store.

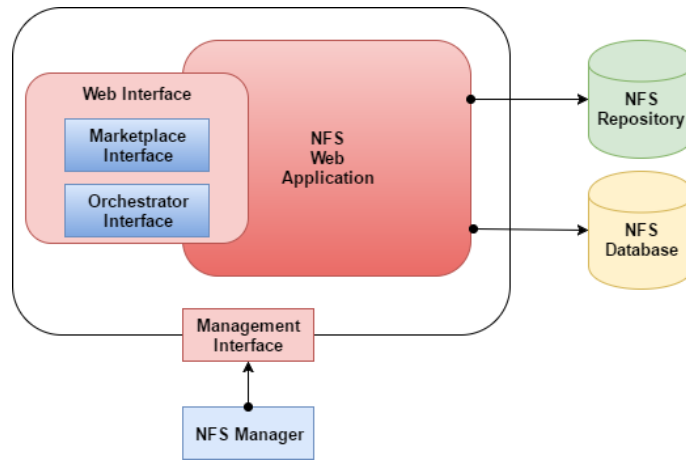


Figure 2.11: NFS High Level Architecture (figure 2 in [25])

The high level illustrates various blocks [25]:

- NFS Repository – contains the VNF images;
- NFS Database – responsible for containing the VNF metadata and data for the Web Application;
- NFS Web Application – is the logic application in the NF Store which interacts over the exposed interfaces, implements NF Store functionality and administrate the repositories. It implements the interfaces to the repository, database and web interface;
- NFS Manager – application responsible for managing the NF Store.

The NF Store provides REST interfaces to the orchestrator and the Marketplace [25]. It is also implemented a shell interface (Management Interface) provided by the NFS Manager and it is used to manage the NF Store service as a standard Linux service [25]. The NF Store function is a web application running on an application server called Apache TomEE [25].

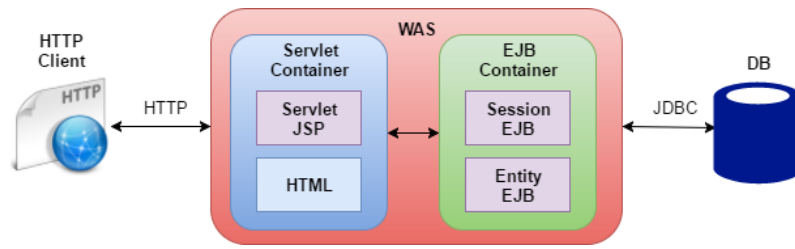


Figure 2.12: NFS Web Application Architecture (figure 3 in [25])

Figure 2.12 represents the Web Application architecture implemented in NF Store. The Web application is based on standard JSR-318 Enterprise Java Bean (EJB) and it is used two different features: Stateless EJB and Singleton EJB [25]. The Stateless EJB is a feature used to perceive the NF Store direct interactions giving to the users the possibility to use the interfaces also when VNF image operations (download, upload, modify) are active [25]. This feature prevents to have the interface locked and unusable for many time because, for example, the time for downloading a VNF image depends on image's length and that can affect the Web Application interface [25]. Finally, Singleton EJB is used for the NF Store Manager Service, realising the synchronisation centre of the Web Application and to execute operations in an exclusive way [25].

The Database is developed by using H2, an open source Java SQL database with a small footprint, and it is saved as a standard file into the Linux file system [25]. The database model is made using JPA and the connection to the DB uses JDBC interface [25]. The main entities of the database model are VNFDDescriptor and VNFFFile [25]. The others entities in the database are used for Brokerage interface to improve search performance [25]. VNFDDescriptor is used to save information in String form about VNF metadata [25]. Lastly, The VNFFFile model is a saved VNF image file at the file system to describe a one-to-many relation between VNF images and VNFs [25].

In finalisation, NF Store is a T-NOVA component relevant for this dissertation. It gives the knowledge of how a Marketplace exposes VNFs for customers and how it interacts with the Brokerage Platform in order to advertise Functions Providers and/or Service Providers VNF/services. With this, it is now possible to understand how the catalogs and databases can be implemented and connected to a possible web application through the interfaces that can be equal to those introduced in the high level architecture (figure 2.11).

2.2.3.4 Marketplace

T-NOVA implemented a Marketplace of VNFs that enables the introduction of VNFs into the market. With the marketplace, it will attract new entrants to the networking markets which is particularly important for SMEs and academic institutions who can influence the T-NOVA architecture by developing innovative NFs as software modules that can be included in the NF Store [20]. The T-NOVA Marketplace allows network services and functions by a variety of developers to be published and traded. Also, it authorises customers to browse the Marketplace and select the services and virtual appliances that best match their needs, as well negotiating the associated SLAs and billing models [20,26]. The main T-NOVA Marketplace functions are publication of resources and NF advertisement, VNF discovery, resource trading and service matching and, finally, customer-side monitoring and configuration of the offered services and functions (Dashboard) [20,26].

According to T-NOVA architecture (figure 2.9), there are six modules present in the Marketplace [26]:

- SLA Management (section 2.2.3.4.1) – component that stores all the SLA agreements among the involved parties, checks if the SLAs have been fulfilled or not, and informs the accounting system for the pertinent billable items (penalties or rewarding);
- Accounting (section 2.2.3.4.2) – module which is in charge of registering all business relationships (subscriptions and SLA evaluations) and making the related information available for the billing system (for each user);
- Billing system (section 2.2.3.4.3) – component that produces the bill for a customer on behalf of the Service Provider (based on the information stored in the accounting module). A bill will be produced for the SP on behalf of its own suppliers (FPs);
- Access Control (section 2.2.3.4.4) – component that regulates who is allowed to access the T-NOVA system and what is it allowed to do (different Stakeholders);
- Brokerage Platform (section 2.2.3.4.5) – platform that receives requests for VNFs and it will present the most suitable offerings (trading);
- Dashboard (section 2.2.3.4.6) – graphical Front-End displaying monitoring/utilisation/status information about the provisioned T-NOVA services. It will have different views for each of the stakeholders accessing the T-NOVA marketplace.

Figure 2.13 provides a better understanding of the Marketplace internal functionality with external interfaces to the Orchestration Platform.

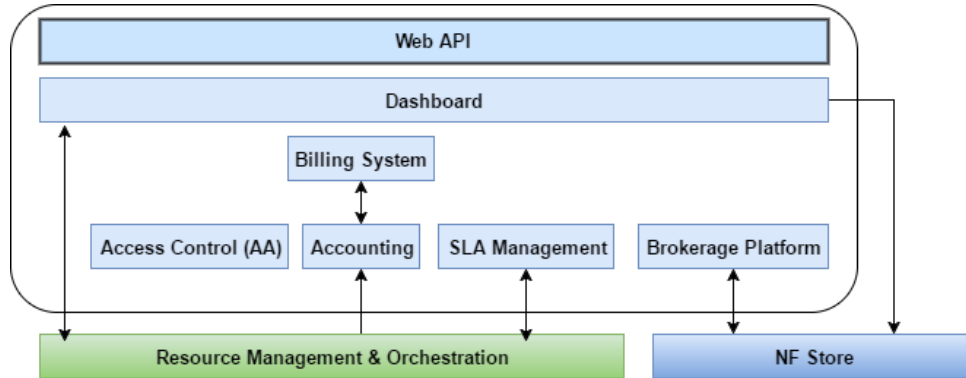


Figure 2.13: Marketplace internal blocks and external interfaces (figure 22 in [23])

2.2.3.4.1 SLA Management

SLA represents the formalisation of the QoS in a contract between the customer and the SP, for example, which describes the service that is delivered, its properties and the obligations of each party involved [26]. This contract establishes in case the guarantee is fulfilled or violated, rewards or penalties, monetary or not, can be applied, respectively [26]. According to T-NOVA [26], there are a SLA agreed between FP and a SP and between the SP and its customers, per each service. It also refers that the same service could have different levels associated [26]. Table 2.1 represents different SLA levels with different prices per service.

SLA per Service
service1, SLA11, price11
service1, SLA12, price12
service2, SLA21, price21
service2, SLA22, price22

Table 2.1: SLA per Service (Table 2-8 in [26])

T-NOVA SLA Management Module provides information for later accounting, depending on the terms and conditions gathered in the SLA and on whether this SLA has been met by all parties or not [26]. This module is based on two main steps [26]:

- Paper-signed contract between the customer and the SP, and between the SP and FPs, including the description of the QoS and the penalties to be applied;
- eContract: It is automatically negotiated between parties for each customer, depending on the demand. Always based on a paper-signed framework contract.

Also, the SLA Management Module is able to provide the following functionalities: publication, discovery, and negotiation of SLAs requirements in order to manage the SLA lifecycle that can be split into several steps (figure 2.14) [26].

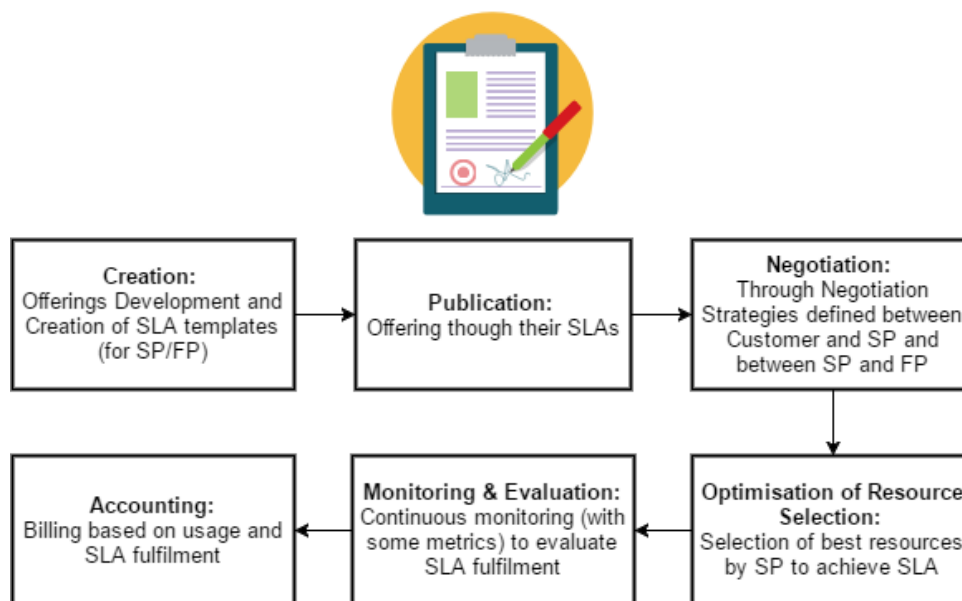


Figure 2.14: SLA Lifecycle (Figure 2-11 in [26]))

Finally, table 2.2 describes a high level SLA Management Module with information that shall be stored in the SLA Management.

Item	SLA template specification	SLA contract	SLA fulfilment	Billable items
	Input from the SP and FP from the dashboard	Input from the dashboard as the output of the SLA negotiation	Input from the monitoring system in the orchestrator	Output of the SLA Management Module (to be sent to the accounting module)
VNF ID				
Service ID				

Table 2.2: SLA Management Module information (Design - Table 2-9 in [26])

2.2.3.4.2 Accounting Module

This module, as was introduced above, is in charge of registering all the business relationships and events that will be used in the Billing module. The Accounting module will be the intermediate component between the billing system and the rest of the system [26]. Table 2.3 gives a high-level information that shall be used by the accounting module.

Type	It could be a service or a standalone VNF
Instance ID	Instance ID of the service (or function) in the system once it is been instantiated. It is used for interactions with the Orchestration
Client	Purchaser of the service (Customer) or function (SP)
Provider	Seller of the service (SP) or function (FP)
SLA	SLA agreement ID of the transaction (in order to get possible penalties to be applied)
Status	Current status or the service/function: Running or stopped
Billing	Information on how to bill the Client
Dates	Date when the service/function was instantiated

Table 2.3: Accounting module information (Design - Table 2-10 in [26])

2.2.3.4.3 Billing System

T-NOVA Billing system is in charge of producing the bills for users and providers and revenue sharing reports for the SP on behalf of its suppliers at the end of a billing cycle. This module is adapted from a version of the Open Source Rating-Charging-Billing framework, Cyclops [26].

Cyclops is a dynamic rating-charging and billing solution for cloud services (designed for IaaS, PaaS and SaaS)¹⁵. It is composed by several micro-services that are used in T-NOVA Marketplace: UDR Service, Rating and Charging Service and Billing process [26]. The UDR harmonises the resource consumption data coming from the collectors, merges multiple data stream into a single output stream (UDR reports) and has Data APIs (usage data visualisation in the dashboard and the data prediction based on historical trends) [27]. The Rating and Charging system is made of rule-engine more data persistence and query interface [27]. It also converts UDR into charge records, the rating and charging is truly generic (customers configure the rules, has a RESTful interface and is highly programmable) and the CDR is stored back for visualisation (with associated timestamps) [27]. Finally, the Billing service is made like the Rating and Charging service but has different purposes. It handles customer retention policies (discounts, rebates, promotions, among others), SLA violations and penalties, geographical rules (VATs, sales-tax, currencies) and federated billing [27].

There is another micro-service used in T-NOVA billing system called Gatekeeper which allows process/user authentication and validation service [26,27]. The external applications send relevant data into Cyclops asynchronously over highly available message bus [26]. Lastly, figure 2.15 illustrates the adapted Cyclops high level architecture used in the T-NOVA billing system together with a typical accounting process.

¹⁵ <http://icclab.github.io/cyclops/>

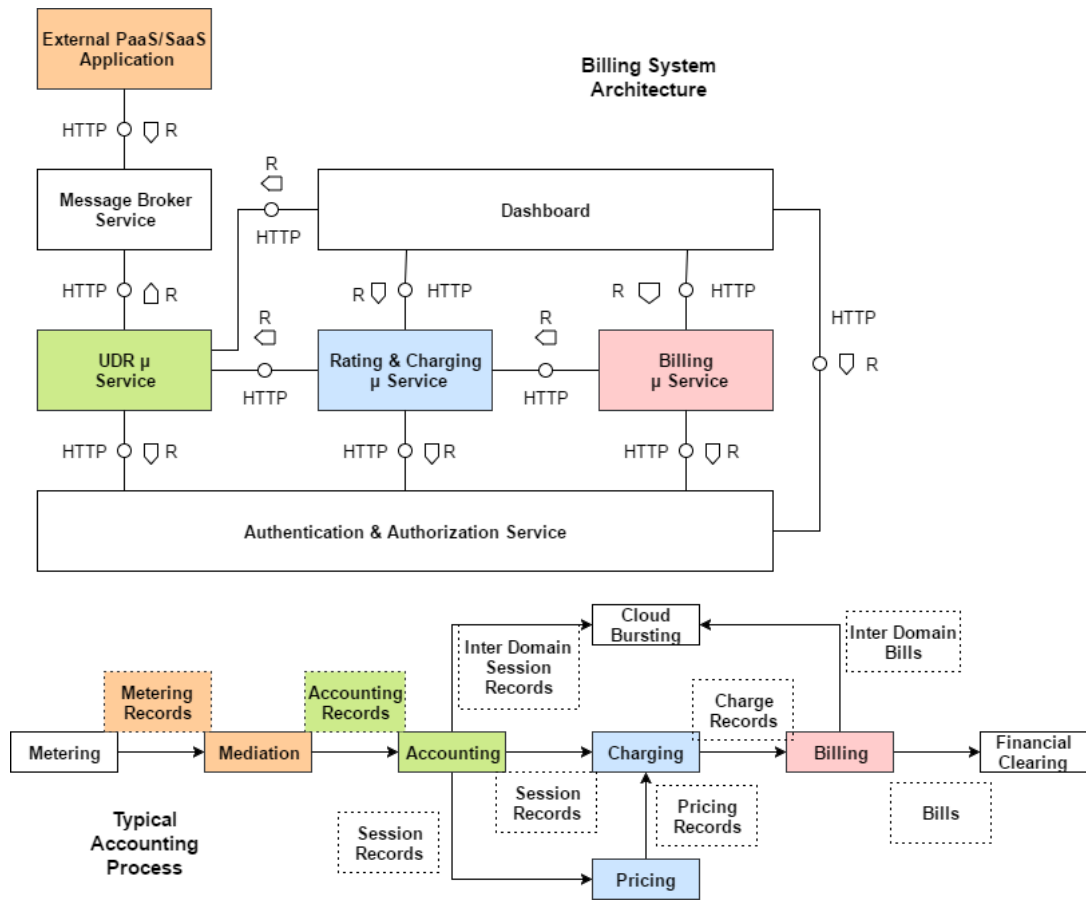


Figure 2.15: Billing Module Architecture (figure 12 in [26]) with typical accounting process (figure 1 in [27])

2.2.3.4.4 Access Control

T-NOVA established different stakeholders for the Marketplace and each of these stakeholders has a specific role with associated permissions. In order to administer security in a multi-user environment and associate the roles to each one of them, T-NOVA developed the Role Based Access Control [26]. The requirements gathered for this module are [26]:

- The different stakeholders should be authenticated before any operation on the T-NOVA system;
- The different stakeholders should be authorised to perform tasks that are associated with their roles and permissions;
- Roles are created accordingly to their functions in T-NOVA, and stakeholders are assigned roles based on their responsibilities and qualifications;
- Roles can be reassigned or granted new permissions if needed;
- Roles and permissions should be updatable and revocable.

The AA system (RABC) offers two main functionalities [26]:

- Authentication – Process by which the system will verify that a user of T-NOVA is exactly who he is claiming to be;
- Authorisation – Process by which a user is allowed to perform the tasks he desires to.

Figure 2.16 illustrates a high level architecture of RABC. The Authentication Manager enables a mechanism with different functionalities to the stakeholders that a user can register and login with username and password [26]. To register, the user needs to fill several information fields such as username, password, email, among others [26]. After the login, the Authentication Manager returns a JWT authentication token reflecting that the user is logged in [26].

The Policy Enforcement Service enables another mechanism that allows assigning users different roles resulting in different rights [26]. For instance, after a new user registers a user profile, the RBAC mechanism will associate the new user a specific role [26]. The roles associated with this project are [26]:

- T-NOVA Operator – In charge of the T-NOVA System;
- SP – Purchases several VNFs to compose a service to be sold to its final customers;
- FP – Uploads and upgrades a given VNF on the T-NOVA System;
- Customer – Entity interested in purchasing a T-NOVA Service.

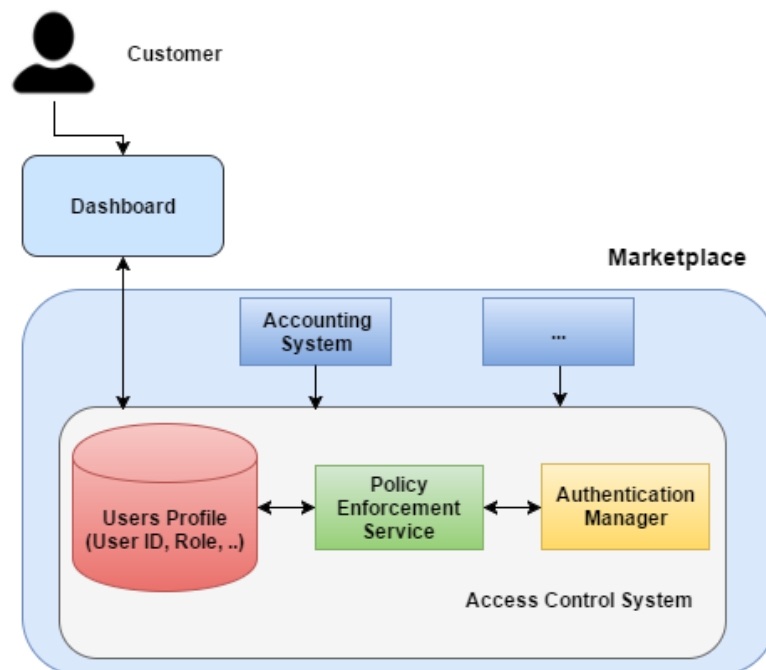


Figure 2.16: RABC Architecture (figure 2-6 in [26])

2.2.3.4.5 Brokerage Platform

This module, as was introduced above, allows FP and SP trading VNFs (especially through auctions in case one VNF is offered by more than one FP [26]) and the VNF discovery for seeking requested VNF. Figure 2.17 shows the Brokerage module architecture associated with users, dashboard and NFS. The SP can place their requests and requirements for the corresponding VNFs, receive offerings and make the appropriate selections, taking into account the price and the offered SLAs [26]. Trading policies such as the long-term lease, scheduled lease, short-term lease (these leasing types refer to the duration of VNFs exploration) or spot markets can be based either on fixed-price or auction-based strategies [26].

In order to maximise the auction payoff, T-NOVA focused in the first-price sealed-bid auction mechanism [26]. This mechanism avoids the exceed signalling overhead with the bidders simultaneously submitting sealed bids so that no bidder knows the bid of any other participant (blind auction) [26]. Consequently, bidders cannot change their bids after the announcement of the others bids [26]. The implementation of second-price sealed-bid is also possible to be implemented and T-NOVA developed as an optional feature [26]. This implementation grants a service more truthful because the first-price sealed bid does not guarantee to be truthful (truthfulness prevents market manipulation since the bidding is performed considering the true value of the item) [26].

The T-NOVA Brokerage Platform can change the pricing rule in a flexible manner as an extra policy in the brokerage module operation mechanism [26]. Additionally, the call price can be used to provide rational item valuation and the brokerage module will determine the proper call price for each VNF based on marketing factors [26]. Finally, the bidders can use their own valuation tools along with both the brokerage module, so that the bidders are able to learn the optimum call price [26].

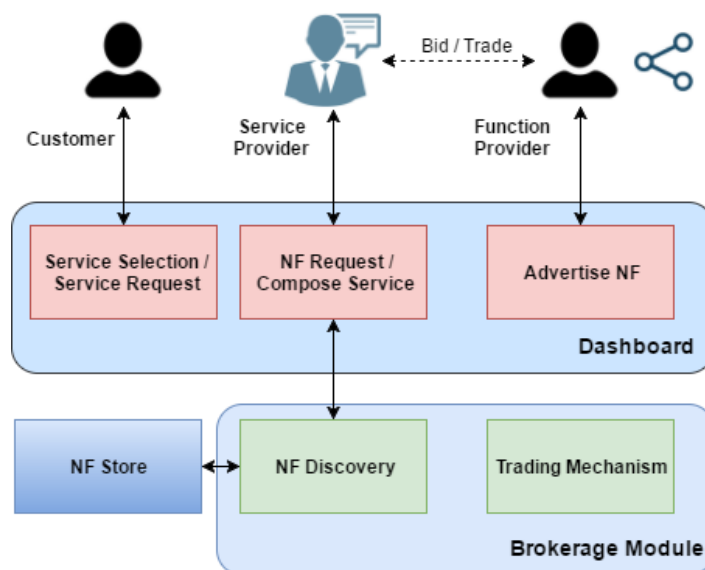


Figure 2.17: Brokerage Platform Architecture (figure 2-8 in [26])

Figure 2.18 reflects the trading process between the SP and the FP for a better understanding.

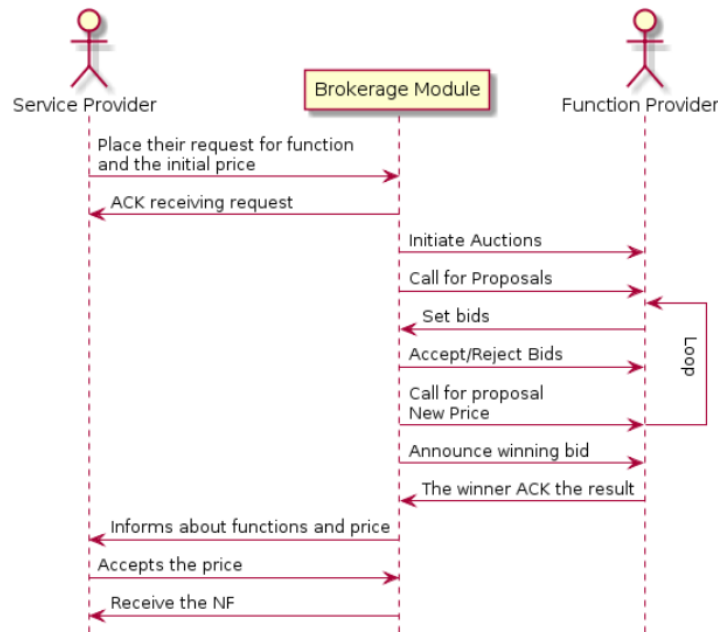


Figure 2.18: Brokerage Platform Architecture (figure 2-7 in [26])

2.2.3.4.6 Dashboard

T-NOVA implemented a dashboard to provide simplicity for the different T-NOVA users taking into account the different roles of the T-NOVA Marketplace. This dashboard host 3 views for the 3 basic stakeholders: SP, FP and customer [26]. Figure 2.19 illustrates different views and features implemented in the dashboard.

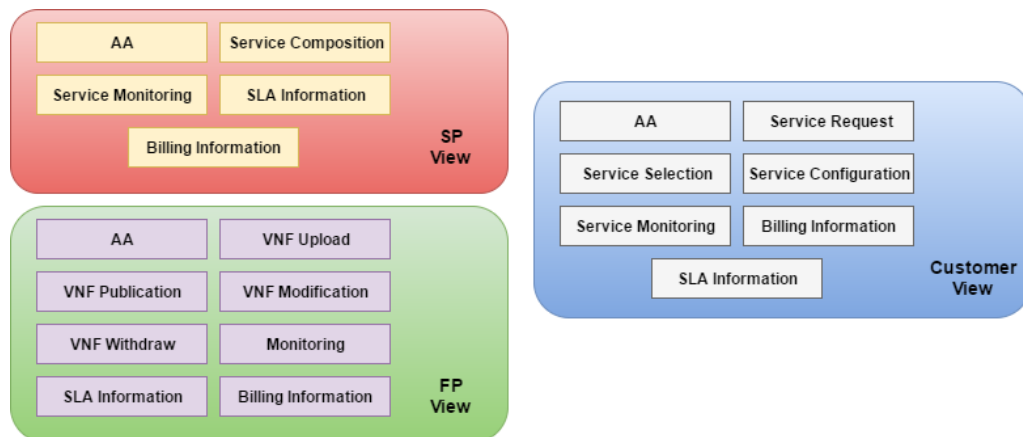


Figure 2.19: Stakeholders view from dashboard (figure 2-5 in [26])

For the SP view, table 2.4 describes each functionality in the dashboard.

Functionality	Explanation
AA	Authorisation and Authentication of the SP's role into the T-NOVA dashboard
Service Composition	Graphical wizard that will help the SP to compose a new NS starting from the brokerage among the FPs owning the available VNFs
Service Monitoring	Graphical representation of all monitoring data for a selected or "consumed" service
Billing Information	Graphical representation of the billing outcomes of selected of "consumed" service. There will be 2 types of billing information for the SP: Charges for the SP's customers (BSS functionality) and invoices on behalf of own suppliers (FP)
SLA Information	Details of the selected or "consumed" service based on how they respect the agreed SLA. The SP will have access to 2 different kinds of SLA contract and SLA monitoring information: SLA between SP and its customers (BSS) and SLA agreed with FPs

Table 2.4: SP dashboard view (Table 2-5 in [26])

The FP view of the dashboard is also illustrated in the next table (table 2.5).

Functionality	Explanation
AA	Authorisation and Authentication of the FP's role into the T-NOVA dashboard
VNF Upload	Graphical wizard that will help the FP to upload his VNF with the necessary parameters
VNF Publication	Graphical representation for the FP to provide the last check in order to publish the uploaded VNF
VNF Modification	Small graphical wizard that provides the ability to the FP to modify the uploaded VNF
VNF Withdraw	Graphical representation that gives the FP the ability to remove an already published or uploaded VNF
VNFs Monitoring	Graphical representation of all monitoring data for a selected or "consumed" NF
Billing Information	Graphical representation of the billing outcomes for a selected or "consumed" NF
SLA Information	Information of the selected or "consumed" NFs based on the agreed SLA and its fulfilment

Table 2.5: FP dashboard view (Table 2-6 in [26])

Finally, table 2.6 represents the customer view in the dashboard with the functionalities and short explanations for each one of them.

Functionality	Explanation
AA	Authorisation and Authentication of the customer's role into the T-NOVA dashboard
Service Request	Graphical representation of the Services/Functions returned by the T-NOVA business service catalogue
Service Selection	Graphical representation assisted by a check box providing the ability to the customer to select a service for consumption
Service Configuration	Small graphical wizard that providing to the customer predefined parameters for defining the selected service
Service Monitoring	Graphical representation of the data gathered from the monitoring modules
Billing Information	Graphical representation of the billing outcomes for a selected or "consumed" NF
SLA Information	Details of the selected or "consumed" service based on how they respect the agreed SLA

Table 2.6: Customer dashboard view (Table 2-7 in [26])

2.3 Conclusions

NFV is a concept that replaces network functions devices to virtual appliances. These appliances can be added to a standard data center hardware. Since NFV can be complemented with SDN, there is the possibility of enhancing performance, simplify compatibility with existing deployments and facilitate operation and maintenance procedures. With OpenStack, that possibility becomes viable. Using Tacker and Neutron services, every end user, enterprise and network operator can create virtual networks where can place the desired VNFs.

In terms of VNF performance, the VNF placement in a virtual environment is a factor to take in consideration for the Marketplace development and for VNF testing. The VNF placement respecting NUMA and dedicated VCPU mapping are crucial to obtaining reliable and high performance in the virtualised network [24]. Also, there is a NFV's scaling difficulty due to virtualisation overheads and it is restrained by the number of cores on the physical server [24].

Taking into consideration the Marketplaces discussed in the State of the Art, it is now possible to compare each one and note the benefits and functionalities that can be used in this dissertation. In the case of GSMA Marketplace, there was not much information about the Marketplace but the business processes and features developed are interesting. It gives another perspective of how the client interacts with the seller and a definition of what is a Marketplace. Aptoide has also an appealing and simple Marketplace. Users can simply search for Apps, download for free or not and create a business to sell Android Apps. T-NOVA was a complex and organised project that describes how VNFs and NFV services can be deployed in a virtualised environment and how NFV functionalities can be into the market.

In terms of comparison, T-NOVA has a better approach for this dissertation since it is in the same thematic and has more documents describing its architecture and functionalities. The Brokerage Platform for each Marketplace has different procedures. T-NOVA focus on auctions and trading between SPs and FPs. There is a dynamic and simple interaction between the customers and the SPs for obtaining network services. The GSMA Marketplace gives the clients the opportunity to perform a free registration, request services/products (RFX/RFP) in the marketplace and negotiate directly with the seller. In the case of Aptoide, the customers easily download or purchase the desired apps and see detailed information about them. The following table (table 2.7) characterises the Marketplaces in a more detailed form.

Modules	GSMA Marketplace	Aptoide	T-NOVA Marketplace
Authorisation and Authentication	Complete and distinctive. Seller and Customer with different permissions and authorised services	Complete and efficient. Customer, Developer and Partner with different permissions and authorised services	Simple and manageable. FP, SP and Customer with different permissions and authorised services
Catalog and Communications Module	Direct and simple communication between Stakeholders	No direct communication between Stakeholders. The dynamic communications present are when the Customer buy Apps (from Developer or Partner) and Partners use Developer's Apps	No direct communication between Stakeholders. The dynamic communications present are when SP/Customer buy VNFs/NSs and it is created SLA agreements
Catalog Module	Customer get product if Seller reply to RFX. Not dynamic but complete	Dynamic and fast acquisition of Apps	Dynamic and accessible acquisition of VNFs and NSs
Catalog Module	Product publication complete and concise. Sellers advertise their products with help of GSMA Marketplace	App publication simple and summed up. Creation of stores with lot of information	VNF/NS publication very accurate but without any information/help
Catalog Module	Products are not validated. Similar system as Ebay. The Customers can see statistics and information of the Sellers and then choose the best option that match their needs	Apps are certified and validated by an automated and non-automated system	No validation/certification of VNFs and no enough information about FPs/SPs. No statistics about VNFs
Community Module	GSMA has Customer community (Research Centre) where is possible to see the product's information, statistics of Seller's Products, social features and events	Aptoide has a community for Clients where they can review and comment products. Social features are also included	No Customer community

Request Module	Customer can request available and not available products (private or public)	Aptotide does not have Request Module for Customers/Partners. Customers can only download/buy available Apps and Partners can only choose available Apps for their stores	No Request Module available
Catalog and Search Module	Research Centre with published reports, Market statistics, Seller's information, among others	Search engine with filters. The engine is appealing and fast	No search engine or tool
Billing Module	Simple and has only the necessary payment type (direct payment). No need for subscriptions. It was not possible to find different options of payment (Paypal, credit card, etc). Transactions and billing information are showed to Stakeholders	Precise and with IAB. It has two different payment types: direct payment and subscriptions. It was not possible to find different options of payment (Paypal, credit card, etc), how subscriptions are made and if transactions and billing information are showed to Stakeholders	Complete, efficient and with two types: Pay-As-You-Go and Revenue Sharing. Bills have discounts if SLAs are not being fulfilled (violations). No payment options and Charging does not really charge Customer/SP or even payout to FP/SP
SLA Module	SLA are created and fulfilled between Stakeholders and GSMA Marketplace	SLA are created and fulfilled between Developers, Partners and Aptotide	SLA are created and fulfilled between FPs-SPs and between SPs-Customers
Billing Module (Market Share)	Not possible to check how it is the Market Share System	Just and configurable Market Share	No Market Share. Prices are not discussed and the payouts/payments are not executed

Table 2.7: Characterisation of the analysed marketplaces

All the mentioned Marketplaces have their flaws. For instance, in the GSMA Marketplace registration step, when the user clicks on submit button, sometimes there are some errors which are not informed. In case of Aptoide, it has the problems of the piracy and malware. Doing a search in Aptoide Market, it was identified a free game which at Google Play is paid. It was also identified as a reliable application. The company that developed the app is mentioned but it was not possible to see information about it and the application distributed in Aptoide Market is from Partner/Developer that probably has no connection with the company.

T-NOVA has also its limitations. For example, T-NOVA Marketplace does not help developers how they should edit and add VNFs. The same way that does not specify how a service can contain any orchestration particular functions such related to placement. There are others limitations present in T-NOVA. If new Network Services developments are to be accommodated, the SP needs to delete the NS, remove all instances and re-create it from the beginning. There is also a lack of VNF validation and certification before on-boarding a VNF into the system. Lastly, the customer can not request VNFs or even NSs, can not communicate with the seller and can only choose NSs from a catalog of pre-composed NSs.

In overall, Marketplaces need to focus and construct a strong and effective business model to achieve potential clients and vendors. There are some Marketplaces that sometimes do not plan the best business model even if the business idea is appealing and with great future. For instance, having a Marketplace that has great products but does not allow customers to search for what they want, it will be difficult selling the available products. Some Marketplaces as well have service limits that prevent customers/sellers using what they really desire. Not providing configurable service flavors for customers and give pre-defined products only from Marketplace business partners, can become a more "private" Marketplace and not so customer-oriented. Lastly, what normally happens with most of the Marketplaces is not giving the best support in case of some malfunction or doubt. For instance, if a client (enterprise) wants Amazon Web Store to be his Cloud Provider, he will need to invest in his network team to successfully manage the cloud because, to get technical support, it is needed to pay a fee. Marketplaces can have many advantages but it is important to focus in want the customer/seller desire and how the Marketplace can fulfil that.

In conclusion, there will be a mix of features present in the studied Marketplaces in this dissertation. Others components and functionalities will be added in order to provide an innovative and dynamic Marketplace to all the users that want to use it.

Chapter 3

VNF Marketplace

This chapter provides an explanation of the main topics for this thesis. Section 3.1 introduces the VNF Marketplace including goals, benefits, architecture and use cases. After that, section 3.2 explains the technical development of VNF Marketplace. Lastly, section 3.3 defines conclusions about VNF Marketplace and its components.

3.1 Characterisation

VNF Marketplace is a Marketplace that introduces the VNF commerce. It allows developers and enterprises to upload VNFs into the Marketplace and end-users/companies to purchase them. With VNF Marketplace, users have an appealing and concise web platform to trade and request VNFs. The web platform was developed to approve any type of VNF and, in case of upgrading the Marketplace, is possible to transfer the platform to other server environments and increase the performance. VNF Marketplace communicates with OpenStack but has the ability to communicate with other IaaS solution such as CloudStack. Lastly, one feature of the VNF Marketplace is the application (deployment) of VNFs in different cloud infrastructures. This is possible because Tacker can select a specific VIM (Openstack) to place the desired VNFs.

There are two actors present in the system: the Customer and the Seller. The customer can browse the VNF catalog, make requests if the desired VNF does not exist, see the associated SLAs and billing models, and, finally, buy the VNF that best matches his needs. The seller can publish VNFs into the Marketplace, identify the desired billing payments and see statistics such as the monthly revenue and the best selling VNF. VNF Marketplace also allows sellers to deploy VNFs in a cloud infrastructure (OpenStack). This will permit customers to buy VNFs without the need to configure and install the VNFs. Appendix A illustrates every use case present at the VNF Marketplace.

Comparing VNF Marketplace with the others studied in the State of the Art (section 2.2), it is possible to verify some similarities and innovations. Table 2.7 describes the characteristics of those Marketplaces that are comparable to the VNF Marketplace attributes. VNF Marketplace also has an equivalent Authentication and Authorisation module like the other Marketplaces. It provides different roles and permissions for stakeholders and a simple authentication system (with tokens). VNF Marketplace also delivers a dynamic interaction among Stakeholders but, in case of doubts, customers can see some details of the seller's profile and try to contact him. This functionality provides a better communication between Stakeholders.

VNF Marketplace has a VNF catalog with a mix of catalog features from the studied Marketplaces. The VNF publication is simple and accurate and the VNF acquisition allows customers to purchase and browse (also with a search engine) VNFs in a dynamic way, but if the desired VNF is not available, the customer can request it (Request Module). For this thesis, the VNF validation and certification was not implemented. The first objective of VNF purchase/VNF publication from this master's thesis project is to deliver VNFs in multiple ways to customers so that VNF validation/certification can be implemented. One way to develop the VNF validation is, instead of submitting VNFs directly to the VNF catalog, to manually test first and then certify them. After certification and validation, the VNFs are submitted to the catalog. This way of implementation prevents the VNF Marketplace from being automated and, like Aptoides has, the VNF catalog could have an automated system to avoid that. Also, the VNF publication from VNF Marketplace does not have the best approach to help sellers advertise their VNFs. It is referenced in all the fields of information that the seller must complete all of them but the differences and the reason for the fields are explained only in some cases.

The Billing Module is also implemented in the VNF Marketplace. Compared to the studied Marketplaces, it has a combination of billing functionalities from them. Not only it allows direct payment as it generates bills depending on the time of VNF consumption, it also offers different payment types and different ways to perform it. For instance, the VNF Marketplace permits customers to buy VNFs through Paypal or credit card. Also, the Market Share is implemented and customers must pay a fee to the VNF Marketplace accordingly to the selected SLAs and the VNF configuration/setup. Finally, the VNF Marketplace manages the same SLA Management as T-NOVA (an external module of VNF Marketplace). The only difference is that the SLA agreements are between the customers and the VNF Marketplace since the VNF configuration and deployment are not performed by the sellers.

The VNF Marketplace's main functions are:

- Publication of VNFs;
- VNF Discovery with search tool and catalog;
- Payment and payout system;
- Creation of requests for VNFs;
- Customer-side monitoring of the offered VNFs;
- Seller's revenue monitoring;
- Deployment of VNFs in Openstack.

Figure 3.1 illustrates the VNF Marketplace architecture complemented with OpenStack and the external SLA Management module. The requirements for each module can be found in Appendix B. There are seven internal modules present in the VNF Marketplace:

- Authentication and Authorisation – Module that allows users to authenticate and differentiates what each actor can do;
- Request System – Module that receives requests from customers, provides them to sellers and notifies customers in case of reply;
- Analytics – System that offers actors different types of graphical representations related to billing and VNF utilisation;
- Billing – Module that produces bills for the customer on behalf of the seller;
- VNF Catalog – Component that manages the VNFs, offers the possibility to upload VNFs to OpenStack and it allows customers to download them;
- SLA Control – System that registers SLA evaluations and communicates with SLA Management in case of pertinent billable items (penalties or rewarding).
- GUI (Dashboard) – Graphical front-end displaying certain services from VNF Marketplace. For each actor, there are different services displayed;

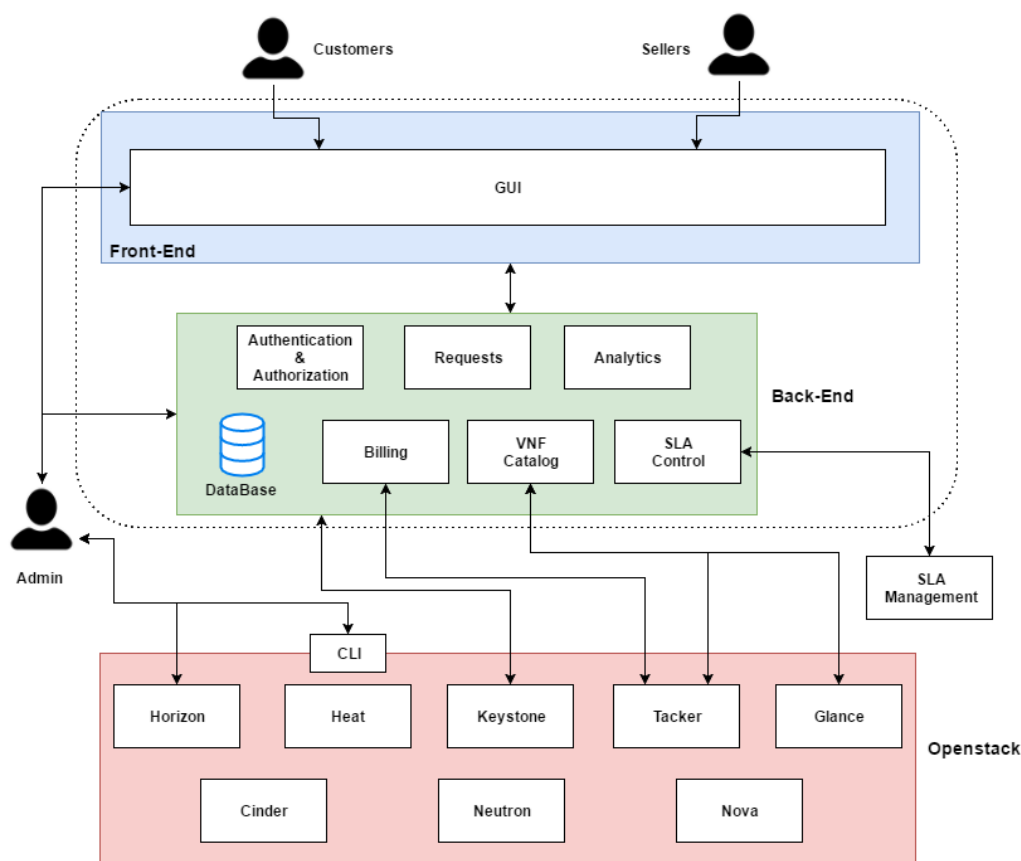


Figure 3.1: VNF Marketplace Architecture combined with OpenStack and SLA Management module

3.1.1 Authentication and Authorisation

The Authentication and Authorisation module was created to provide security in the VNF Marketplace. It establishes different permissions for each role and prevents unauthorised users from entering the Marketplace. It also provides roles to registered users. There are three types of roles in the VNF Marketplace:

- Customer – User for the purpose of purchasing and searching VNFs;
- Seller – User for uploading VNFs into VNF Catalog and sell them;
- Administrator – User who administrates all systems including OpenStack;

Before any user starts using a service on VNF Marketplace, registration is necessary. The user begins by selecting what role he wants to be (customer or seller) and then filling in informational fields such as username, password, email and company. After submission, the user is registered and allowed to enter VNF Marketplace.

The permissions and roles can be changed. Also, roles and permissions can be reassigned or granted to specific users if needed. In the VNF Marketplace, every time a user tries to perform a task, the Authentication and Authorisation module will verify if he is allowed to and if he is authenticated. When the user logs in, with the help of the Front-End, the user's role and permissions are saved to prevent the Authentication and Authorisation module communicate with database every time the user performs an action. Figure 3.2 illustrates the general procedure of this module:

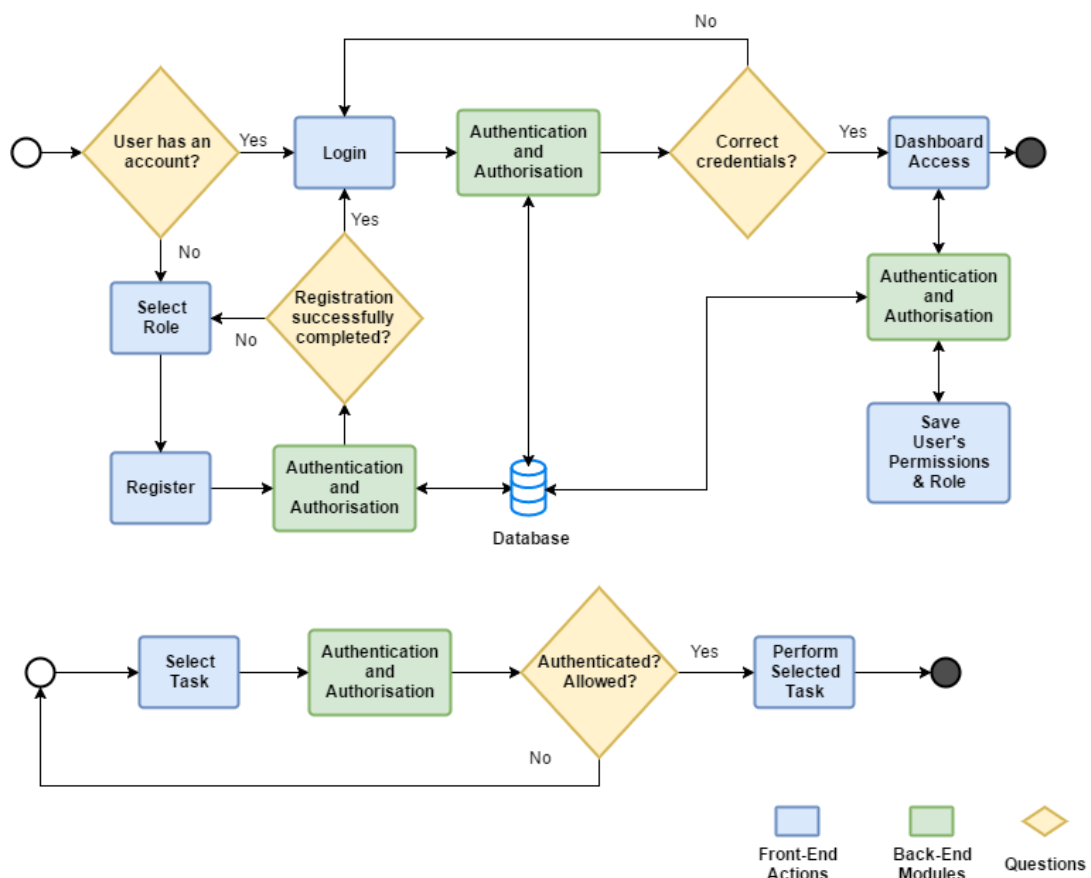


Figure 3.2: Registration, Login and General Flow of user's tasks with Authentication and Authorisation Module

3.1.2 VNF Catalog Module

This module allows sellers to publish VNFs in the VNF Marketplace, VNF discovery for seeking the requested VNF, the upload functionality of a specific VNF into OpenStack and also permits a customer buy the desired VNF. To store information about VNFs, the VNF Catalog module communicates with the Database.

The VNF publication has three phases: insertion of VNF information, selection of a purchase mode (uploading of VNF image and certain files as well) and the filling in of billing information. In case of VNF information, the seller needs to fill in information fields such as VNF name, description, version and type. It is important to use the version field in case of a seller upgrading his VNFs and the VNF type field to help customers search for specific types of VNFs in the catalog (e.g. vRouter, vFW, vDPI, etc).

VNF Marketplace defines two purchase modes: Basic and Premium. The Basic mode offers the customer the ability to download the desired VNF. The seller needs to upload the VNF image and a file with licenses. The licenses were introduced into the VNF Marketplace since there is a vast number of VNF vendors selling their VNFs with licenses. VNF Marketplace is only a prototype and focuses on selling VNFs, so the licenses are present in the text file instead of being generated per purchase or per production. This module can be adapted so it is possible to send licenses per VNF instead of pre-defined licenses. For instance, Brocade sends a late email with a license after the customer bought a VNF.

The Premium mode allows customers to buy VNFs without downloading them. This means the customer does not need to install and configure the VNF. For this being possible, the seller has to upload the VNF image and a VNFD which Tacker can read. Since VNF Marketplace communicates with OpenStack and Tacker, the VNFDs only work for Tacker. Figure 2.6 illustrate an example of a Tacker TOSCA template. The VNFD and VNF image are uploaded into the VNF Marketplace Database and into OpenStack as well (VNF image to Glance and VNFD to Tacker VNF Catalog). After the customer buy a specific VNF, Tacker deploy the VNF with VNF Manager. It is important to mention that VNF Marketplace can be easily adapted to other cloud environments unlike OpenStack.

Figure 3.3 illustrates the VNF lifecycle of the VNF Marketplace.

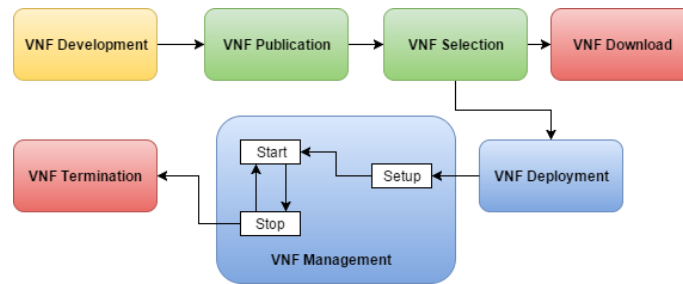


Figure 3.3: VNF Lifecycle

In the last phase, the seller has to specify a payment type (license payment in Basic mode and Pay-As-You-Go in case of Premium mode), the price, the period type and the currency. For License Payment, the period can be per month or year and for Pay-As-You-Go can be per hour, day or week. Pay-As-You-Go was introduced into the VNF Marketplace because it makes sense a customer pay for used computing time only.

VNF discovery permits customers find the desired VNF. VNF Catalog communicates with Database to see available VNFs and send the VNFs information to the Dashboard. VNF Catalog also allows customers to search for specific VNFs instead of browsing all of them. Figure 3.4 illustrates a high level architecture of the VNF Catalog with interfaces.

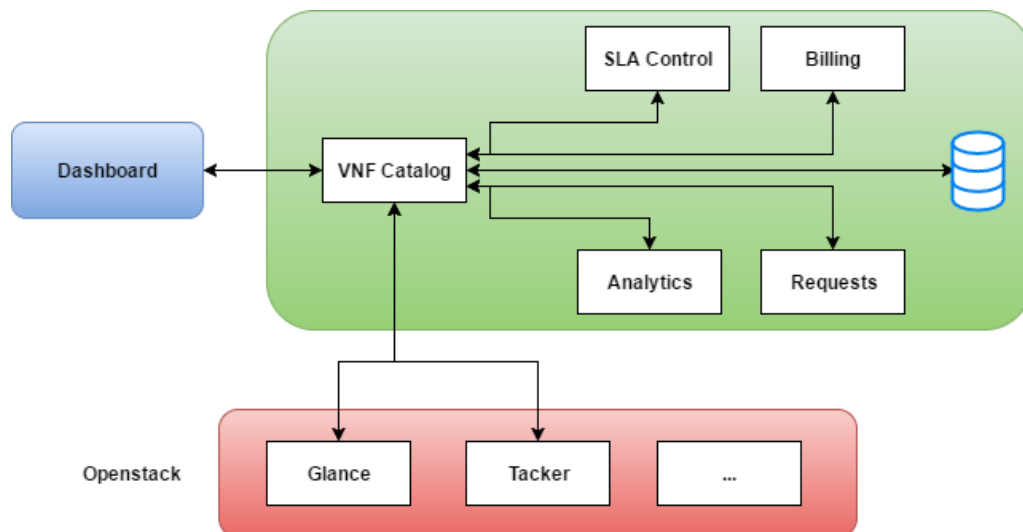


Figure 3.4: VNF Catalog High Level Architecture

3.1.3 Request Module

This module was introduced in the VNF Marketplace to help customers request VNFs that are unavailable in the VNF Catalog. Also, customers can request a specific VNF with different features from the available VNFs. For instance, if VNF Catalog has one vRouter which can run on different virtualisation platforms but not on the customer's desired platform, it can be requested.

It is very simple to the customer request a VNF. The customer only needs to specify a VNF name, description, type and the desired purchase mode. It is possible to exist a VNF at the catalog that is only available for the Basic Mode, so the customer can request it for the Premium mode. After the submission, the request is with an "Open" status and will become available to sellers reply. In the Requests section for sellers are the customer's requests and, if the seller select to reply it, it proceeds to VNF publication phase. The Request Module has the functionality that, after the seller clicks on "Reply" button, the VNF name, description, type and purchase mode are automatically filled on the VNF publication.

Posteriorly, the request status is defined as "Closed" and the customer can proceed with the purchase of the VNF. The following figure (3.5) illustrates the possible status from customers requests.

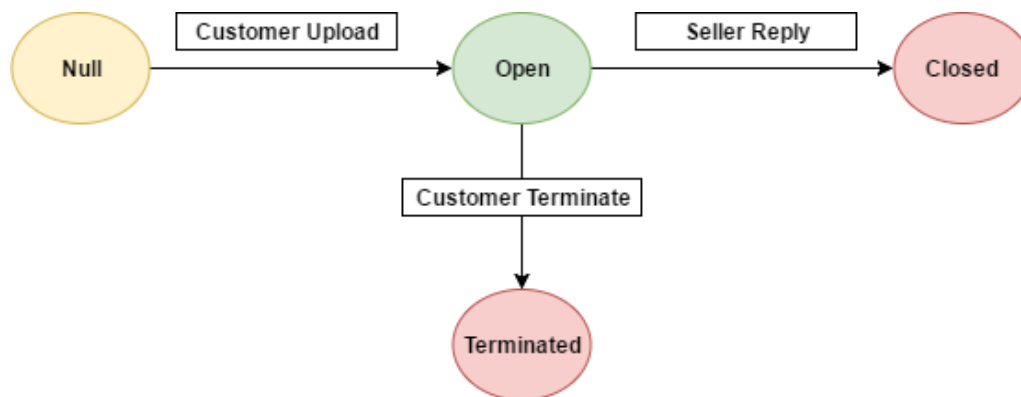


Figure 3.5: Customer's Requests Status

The customer's request is private but after the seller publish the requested VNF, the VNF will be available to all the customers since another customer can desire the same VNF. Figure 3.6 represents the Request Module architecture for other understanding.

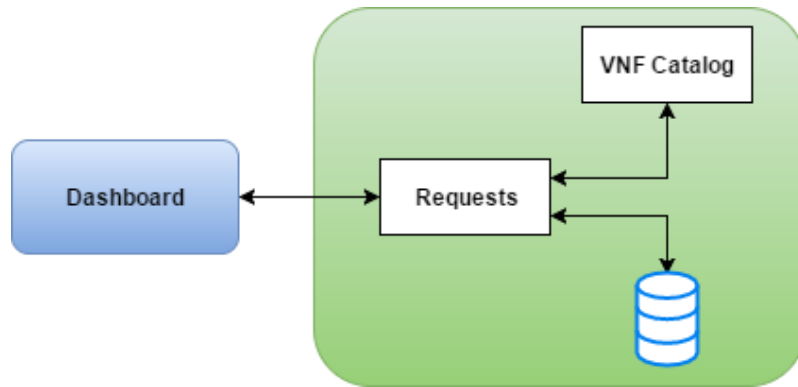


Figure 3.6: Request Module Architecture

3.1.4 Billing System

The Billing System Module allows payments and payouts on the VNF Marketplace. In other words, Billing Module is in charge of producing the bills for customers and revenue for sellers. The payments are performed with credit card and Paypal. Payouts are assured and every transaction creates a PDF file with the invoice. This module has its own Charging, Billing and Rating blocks, with different functionalities from Cyclops (section 2.2.3.4.3). In this module, the VNF price is not negotiable and, for every payment, the seller receives the exact amount he attributed for his VNFs.

Charging system has two different approaches. If the seller selects the Basic Mode, the customer is charged by credit card or Paypal payment. For that happen, it is necessary to get the price that seller introduced for the VNF. As was mentioned above, after payment concluded, it is created a PDF with the invoices for the customer and the seller check. This block uses Paypal Python SDK which assures the payments and payouts. Since the Basic mode uses license payment, it is not created bills for customers. In case the customers want to renew the license, it is necessary to buy again the VNF.

For the Premium mode, the first transaction is done like the Basic mode. The only needed requirement for this mode is the existence of a credit card for debit payment. This is needed because the payment is Pay-As-You-Go and the Charging block will communicate with Rating block to charge the customer for every bill he has. In this mode, the Charging system also uses Paypal Python SDK for every charge bill. Finally, there are different bills accordingly to the selected SLA parameters. For instance, VNF Marketplace has three different group of SLAs (flavors, section 3.1.5) and each one has different prices.

Rating system is a block of the Billing System module that monitors and rates the resource consumption data coming from OpenStack. Since it is needed for Pay-As-You-Go payment, this module is responsible for communicating asynchronously with Tacker and OpenStack. For each VNF active, it produces the bill according to the prices submitted by the seller. This block can be easily adapted to, instead of communicate with OpenStack, communicating with other cloud infrastructure.

Lastly, the Billing block is responsible for saving information from the Charging and Rating block in the Database. In case the Premium mode is active for some VNF, Billing block communicates with SLA Control to see if the SLAs are being fulfilled. In case they are not, it communicates with the Rating block and changes the rate with the respective discount (from SLA Agreement). This block also sends billing information to the Dashboard where the users can see every transaction (payment/revenue) made. Figure 3.7 illustrates a high level architecture of Billing Module with internal and external interfaces.

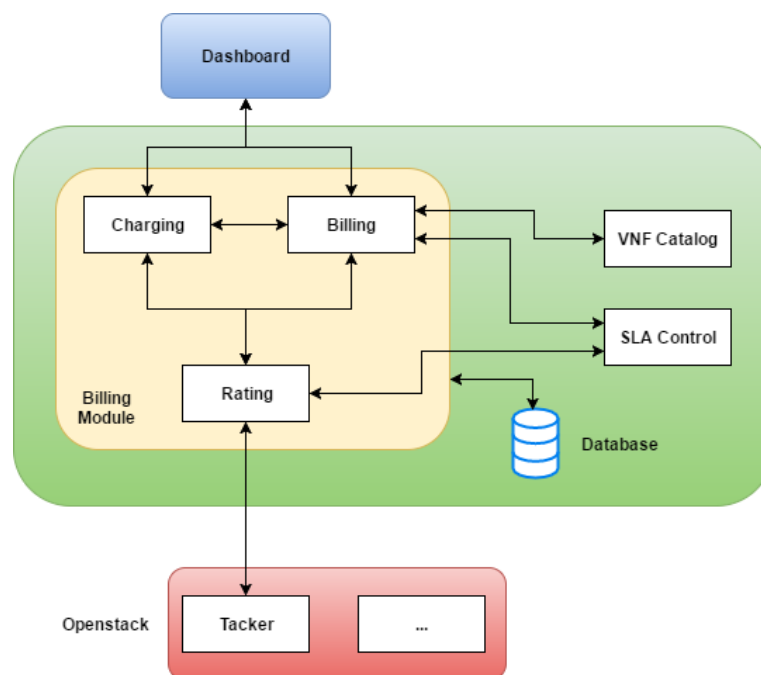


Figure 3.7: Billing System Module Architecture

3.1.5 SLA Management

SLA Management is an external module from VNF Marketplace. It is an open source SLA framework and it is the same SLA Management that T-NOVA used. The reason using this framework is because it is simple to operate, the SLA lifecycle manager system allows to asynchronously communicate with VNF Marketplace and OpenStack and it can be easily adapted in case the cloud infrastructure vary.

Taking section 2.2.3.4.1 as reference, this module has the same procedure and SLA lifecycle. Since is the VNF Marketplace that ensures the VNF deployment and utilisation at cloud environment, the agreements are done between customers and VNF Marketplace. So, after the VNF publication, VNF Marketplace creates SLAs differentiated by flavors. For instance, the first flavor ensures the customer for each VDU that, in case of CPU utilisation is greater than or equal to 75% during 2 breaches counts with a 360 seconds interval, it will have a discount of 2% on the bill. It is important to mention this flavors are pre-composed and, in the future, it can have more and personalised SLAs. Also, in case of sellers want to use the VNFs for own utilisation, it is possible to create some agreements and bills as well.

This framework allows SLAs to be negotiated but, for this first version of VNF Marketplace, the SLAs are pre-defined. The communication with VNF Marketplace is done by requests and responses. The first request is the creation of a SLA template. In case the customer buys a certain VNF, VNF Marketplace will send as request the SLA template with pre-defined SLA parameters and the template will be created at SLA Framework if everything went well. Next, VNF Marketplace will ask SLA Framework for an agreement using the previous SLA template. In case of success, the agreement will be created and SLA Framework will start automatically with enforcement jobs. An enforcement job is an entity which starts the enforcement of the agreement guarantee terms⁴⁰.

The monitoring system has to be configured at the SLA Framework to check if the terms of the agreements are being fulfilled. Due to lack of time and failed installation of OpenStack Ceilometer or Gnocchi services (Telemetry OpenStack services), the metrics were not introduced at the SLA Framework. Instead, VNF Marketplace used dummy metrics and SLA Framework assumed that every agreement is being fulfilled. In the future, this can be easily managed if the installation of those services is successfully done.

⁴⁰https://github.com/Atos-FiwareOps/sla-fra_mework#group-enforcement-jobs

Figure 3.8 represents a high level architecture between VNF Marketplace, SLA Management module and OpenStack (including interfaces). It is important to mention that exists a SLA lifecycle and the metrics are consistently obtained for a set time in the SLA Framework configuration.

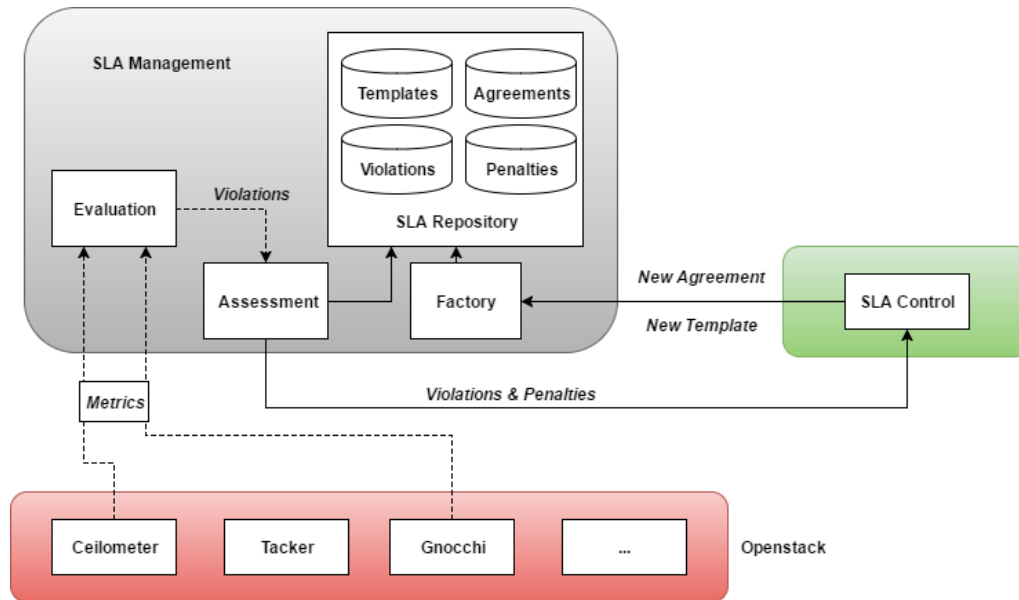


Figure 3.8: SLA Management Module Architecture

3.1.6 SLA Control System

This module is introduced at VNF Marketplace to provide a fast and better control of SLAs. Since SLA Management has its own Database, it was necessary to use VNF Marketplace Database to save additional information such as violations, discounts and related to specific bought VNFs. The module that communicates with the VNF Marketplace Database to receive and transmit that information is SLA Control. Also, it gives information about them to Billing System to perform the appropriate discounts in case of any penalty.

The SLA Control module communicates with Dashboard to provide information about any SLA. This allows customers check if the SLA agreements are being fulfilled. The communication between Dashboard and SLA Management could be done but, due to security and control reasons, SLA Control Module was created and permits sending more information that SLA Management can not provide.

Figure 3.9 illustrates the SLA Control connections inside VNF Marketplace and with SLA Management and Dashboard.

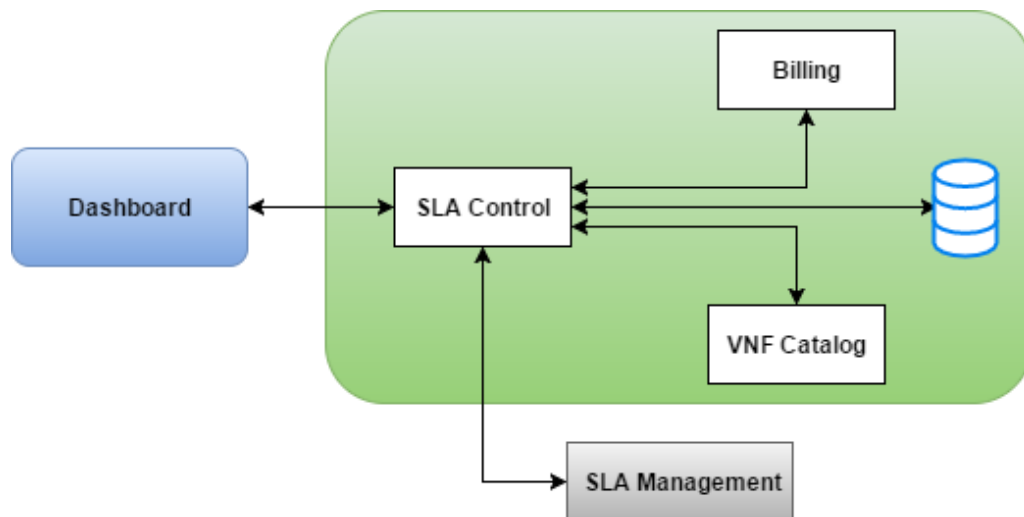


Figure 3.9: SLA Control Module Architecture

3.1.7 Dashboard

Like the Marketplaces referenced on the State of the Art, VNF Marketplace has also a Dashboard for the users. It provides simple different views depending on the role. Dashboard communication with the VNF Marketplace Back-End is done with HTTP requests. The Administrator role has a specific dashboard where is possible to use every service at VNF Marketplace. The Administrator view is represented at the following table (table 3.1):

Functionality	Explanation
Authorisation and Authentication	Login and authorisation of the Administrator role into the Administrator dashboard
Authorisation and Authentication	Creation/modification/withdraw of permissions and roles
User Creation	Graphical representation that will help the Administrator create a specific user
User Modification	Graphical representation to help the Administrator change any a specific parameter from an user
User Withdraw	Graphical representation that gives the Administrator the ability to remove an already created user

VNF Publication	Graphical representation that will help the Administrator upload a VNF with the necessary parameters
VNF Withdraw	Graphical representation that gives the Administrator the ability to remove an already published or purchased VNF
VNF Modification	Graphical representation to help the Administrator change any parameter from an already published or purchased VNF
Request Publication	Graphical representation to help upload requests with the necessary parameters
Request Withdraw	Graphical representation that gives the Administrator the ability to remove an uploaded Request
Request Modification	Graphical representation to help the Administrator change any parameter from an already published Request
SLA Modification	Graphical representation to help the Administrator change any SLA detail from an already VNF purchased or used
Billing Modification	Graphical representation that will help the Administrator modify any billing results for every VNF purchased or used

Table 3.1: Administrator Special Dashboard View

Figure 3.10 illustrates a view of the special Dashboard for Administrator:

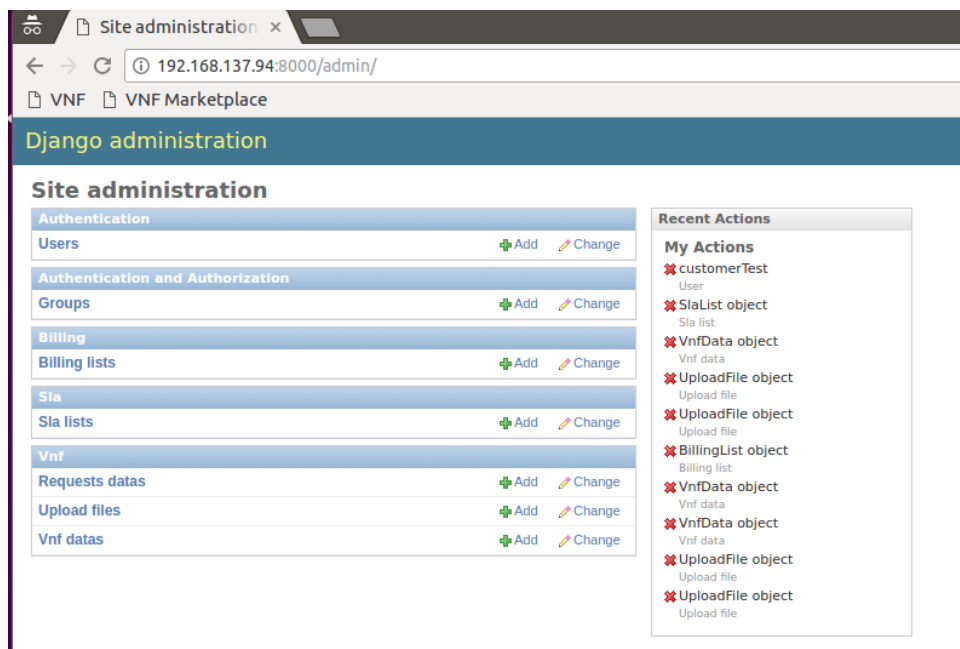


Figure 3.10: Administrator Home Page

For the customer view, table 3.2 describes every functionality that the customer is allowed to:

Functionality	Explanation
Authorisation and Authentication	Login and authorisation of the Customer role into the VNF Marketplace dashboard
VNF Selection	Graphical representation with search engine that will help Customer browse the VNF Catalog and buy the desired VNF
VNF Configuration	Graphical representation that provides the Customer a way to start/stop/download an purchased VNF, depending on the Purchase Mode
VNF Withdraw	Graphical representation that gives the Customer the ability to remove an purchased VNF
Request Publication	Graphical representation to help upload requests with the necessary parameters
Request Withdraw	Graphical representation that gives the Customer the ability to remove an uploaded Request
SLA Information	Graphical representation with details of purchased VNFs (in Premium Mode) to illustrate if the SLA Agreements are being fulfilled
Billing Information	Graphical representation of the billing results for every Customer's VNF purchased or used
Analytics	Graphical representation with multiple graphs about Customer's VNFs utilisation

Table 3.2: Customer VNF Marketplace Dashboard View

Figure 3.11 illustrates the Customer's view of the service Menu:

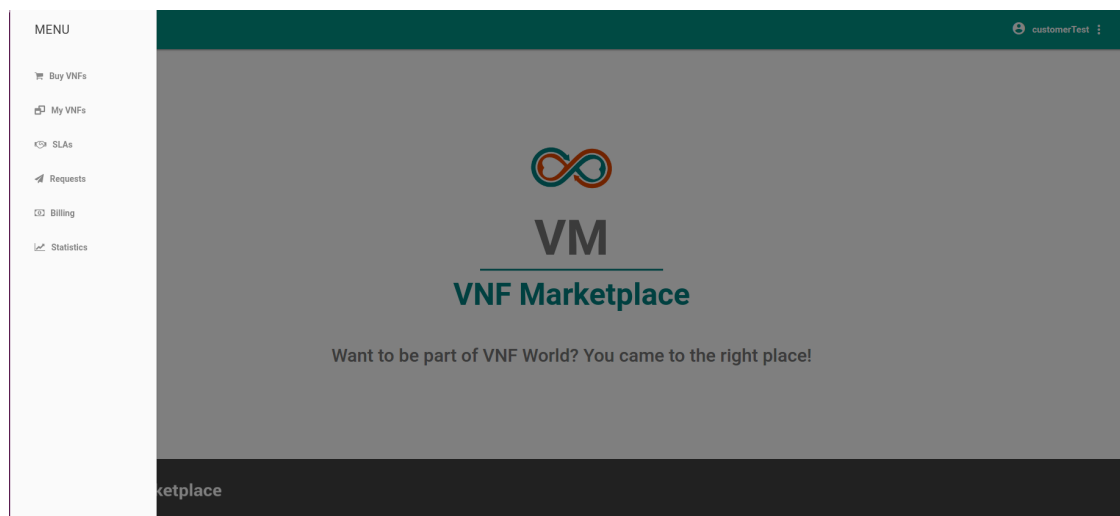


Figure 3.11: Customer Dashboard Menu

Finally, the seller view is represented at table 3.3:

Functionality	Explanation
Authorisation and Authentication	Login and authorisation of the Seller role into the VNF Marketplace dashboard
VNF Publication	Graphical representation that will help the Seller upload his VNF with the necessary parameters
VNF Withdraw	Graphical representation that gives the Seller the ability to remove an already published VNF
Requests Information	Graphical representation with search engine to help reply the customers requests
Billing Information	Graphical representation of the billing results for every Seller's VNF purchased or used
Analytics	Graphical representation with multiple graphs about billing outcomes and Seller's VNFs utilisation

Table 3.3: Seller VNF Marketplace Dashboard View

Figure 3.12 illustrates the Seller's view of the Service Menu:

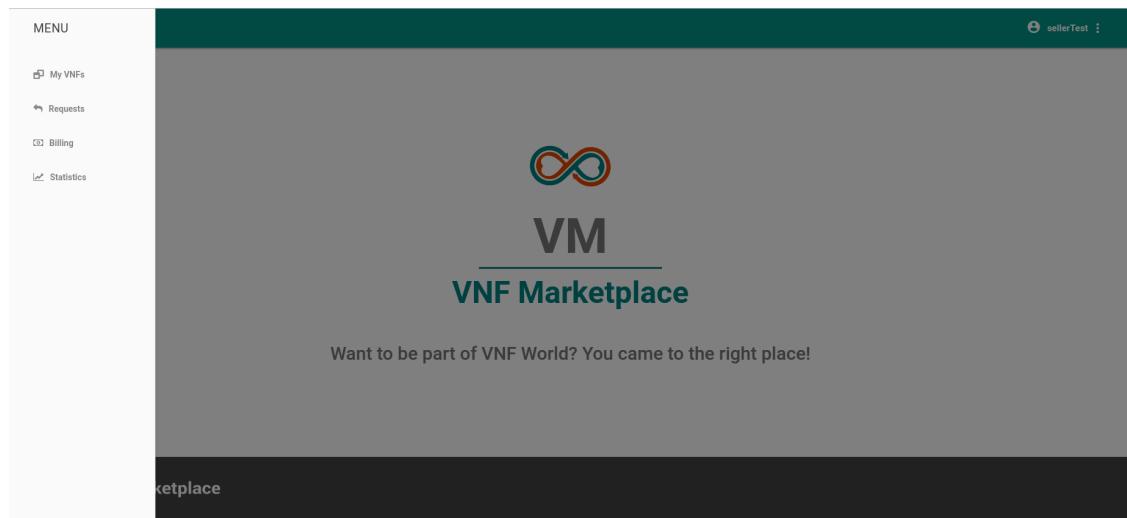


Figure 3.12: Seller Dashboard Menu

Any user has the ability to change details from the respective account at "Settings" section in the dashboard. This means that the user can change profile information such as first name, last name, country, company name, etc. In the future, "Settings" will have options to change and add billing information. This was not introduced since Paypal Python SDK is configured in *Sandbox* mode and real credit cards are not used.

3.1.8 Analytics Module

Analytics Module provides statistics to users at the Dashboard. This module also communicates with all internal modules to gather essential information. For instance, to give month revenue graph of a specific seller, it communicates with Billing Module and Database for that purpose. The Front-End is very important for this module because is where the graphical representations are created.

For this first version of VNF Marketplace, the Analytics Module is not too much essential since OpenStack metrics and monitorisation do not work. Front-End can handle every information from the other modules and it is not really necessary at this point using Analytics Module. In the future, in case of OpenStack Monitorisation works successfully and provides correct values from instances and VNFs (OpenStack API sometimes gives incorrect information) this module will be very important since it saves any information from OpenStack or other cloud environments. Front-End could also communicate with OpenStack but, for security reasons, it is better to communicate between two servers where common users do not have access.

3.1.9 Interaction with OpenStack

OpenStack is a critical software to VNF Marketplace deploy VNFs in case of the customer is interested in buying VNFs without installing or configuring them. After the OpenStack installation with Tacker and Heat, it is provided with the CLI and the Horizon for Administrator access. This will allow VNF configuration and maintenance. Taking VNF Marketplace in consideration, the communication between them will be through API. Every information that is needed or a task that must be performed, VNF Marketplace will send HTTP requests (with JSON as application data) and get responses from OpenStack. The requests need always to have an authorisation token from OpenStack. To get the authorisation token, VNF Marketplace must send a request to Keystone with *Admin* credentials and the response will have the token (present in the Headers) in case it did not occur an error (e.g. incorrect credentials). Figure 3.13 represents a request and response for authenticating with *Admin* credentials.

```

Request
{
  "auth": {
    "identity": {
      "methods": ["password"],
      "password": {
        "user": {
          "name": "admin",
          "domain": { "id": "default" },
          "password": "selectedpassword"
        }
      }
    }
  }
}

Response Header
Connection: close
Content-Length: 312
Content-Type: application/json
Date: Wed, 21 Jun 2017 17:49:48 GMT
Server: Apache/2.4.18 (Ubuntu)
Vary: X-Auth-Token
X-Openstack-Request-Id: req-a1b1b689-1bc7-4d01-8c3c-1eadb6b8c00c
X-Subject-Token: gAAAAABZSrG8spqyOuDyefBZU5hRe2533ZlvRRpXbRx5JV9ckklj5mCDBIHgQ_mCkqRMTOPhDoQ9SI1psQ
kemZFU_WXJSJs8rwt3aX5cpjb6YqgkMlhBQawH76iu5y9RUfZggbjv10hYwc_omDHvAPnYk8aw7I5SgA

```

Figure 3.13: Request and Response Header to get Authentication Token from OpenStack

When the seller uploads the VNFD into VNF Catalog from VNF Marketplace, it is sent a request to Tacker in order to create a VNFD. The same VNFD is saved into VNF Catalog from Tacker and, if there was no error, Tacker will send a success response informing the VNFD was created (with an ID associated). In Appendix C is present requests and responses from VNF Marketplace to Tacker, including the upload of a VNFD. Figure 2.6 represents a TOSCA Tacker VNFD sample that can be submitted into the VNF Catalog from Tacker.

The next figure (figure 3.14) shows the VNF composition with Tacker connections. This will provide a better understanding why TOSCA Tacker VNFD selects several parameters on the template.

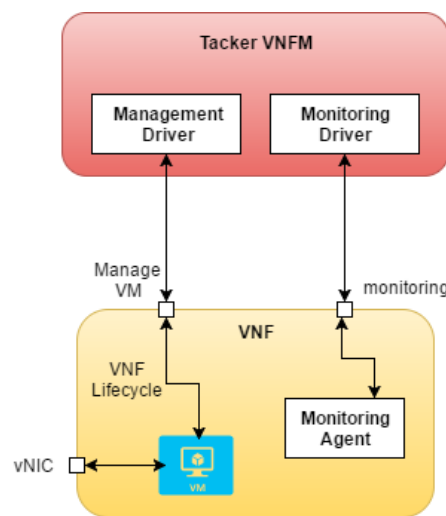


Figure 3.14: VNF Functional View with Tacker

At the VNF publication, the VNF image is also needed for the VNF deployment. After the VNFD submission, in case of success, VNF Marketplace sends a request to create and save the image into Glance. Since the VNF image can be with different types of disk formats, the request must not be in JSON application data. The possible disks formats which Glance can read are⁴¹:

- RAW – unstructured disk image format;
- VHD – common disk format used by virtual machine monitors from VMware, Xen, Microsoft, VirtualBox and others;
- VHDX – enhanced version of the VHD format which supports larger disk sizes among other features;
- VMDK – common disk format supported by many common virtual machine monitors (VMware);
- VDI – disk format supported by VirtualBox virtual machine monitor and the QEMU emulator;

⁴¹<https://docs.openstack.org/developer/glance/formats.html>

- ISO – archive format for the data contents of an optical disc (e.g. DVD-ROM);
- Ploop – disk format supported and used by Virtuozzo to run OS Containers;
- QCOW2 – disk format supported by the QEMU emulator that can expand dynamically and supports Copy on Write;
- AKI – Amazon kernel image;
- ARI – Amazon ramdisk image;
- AMI – Amazon machine image;

In case the VNF image needs a specific container format (OVF, OVA, Docker, etc) it is also important to send it to Glance. If it is not mentioned, Glance will assume it as bare (with no container or metadata envelope for the image). Figure 3.15 illustrates the sequence of requests and responses to upload a VNF image into Glance⁴².

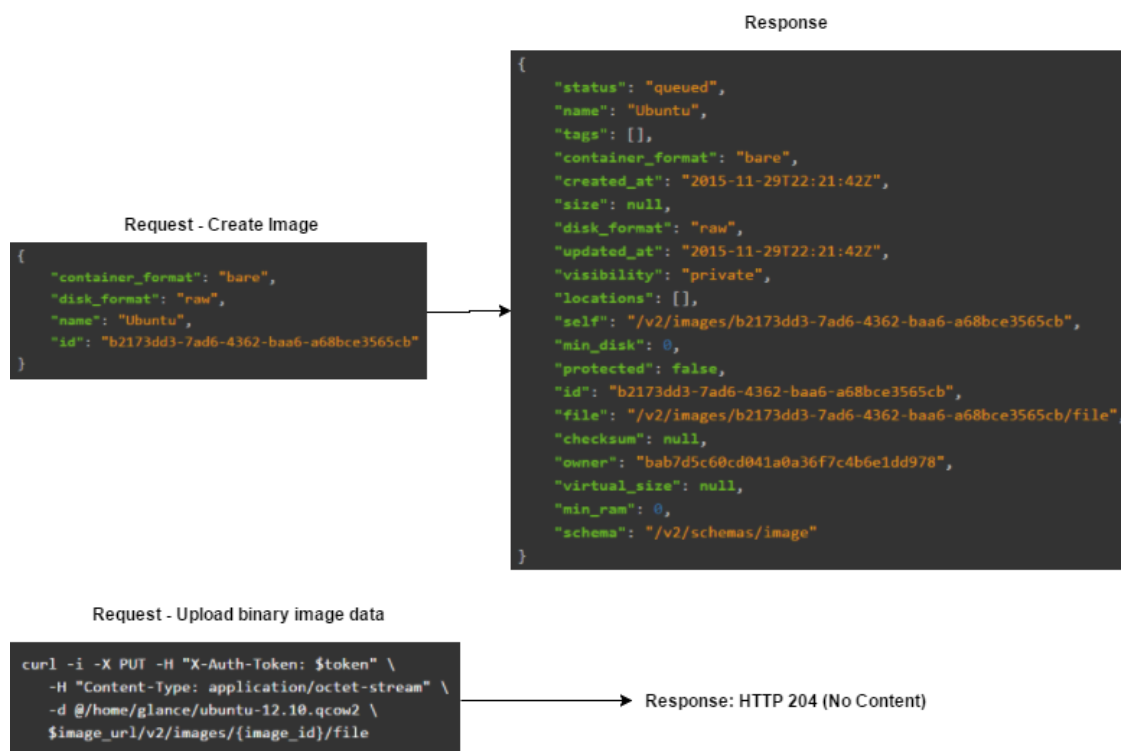


Figure 3.15: Requests and Responses to Upload a VNF Image into Glance

⁴²<https://developer.openstack.org/api-ref/image/v2/index.html>

If this two phases (VNFD and VNF image submission) are successful, VNF publication is completed and OpenStack is ready to deploy VNFs in case any customer buy the uploaded VNF. The next step is the VNF deployment. If the customer buys a VNF in Premium mode, there will be an essential communication between VNF Marketplace and Tacker. Since the VNFD is already at VNF Catalog from Tacker and the VNF image at Glance, it is only needed to send a request with the ID from the respective VNFD to deploy the VNF. Tacker will read the request, communicate with Heat in order to deploy the stacks (VNF can contain more than one VM/VNFC/VDU) and, in case of success, it will respond to VNF Marketplace. In the response are present the ID from the VNF if, by chance, it is needed to withdraw or scaling it and the IDs from the instances/VMs to manage them (stop/start). In Appendix C is also a request from the VNF Marketplace to deploy a VNF at OpenStack.

VNF Monitorisation can also be performed at Tacker. Due to lack of time and bad installation of Ceilometer (Telemetry service in OpenStack), monitorisation and alarms could not be implemented. These two functionalities are important to assure SLAs and, in case of some malfunction, the Administrator can see the logs and make the necessary changes. Following these two tutorials from Tacker Docs (⁴² and ⁴³), VNFs can have monitoring policies.

Since was not implemented, figure 3.16 illustrate the possible states and actions to VNF instances. It is also important to mention that, if VNFD has configuration parameters, the configuration phase is automatic. This is useful to deploy a VNF without configuring it manually after instantiating.

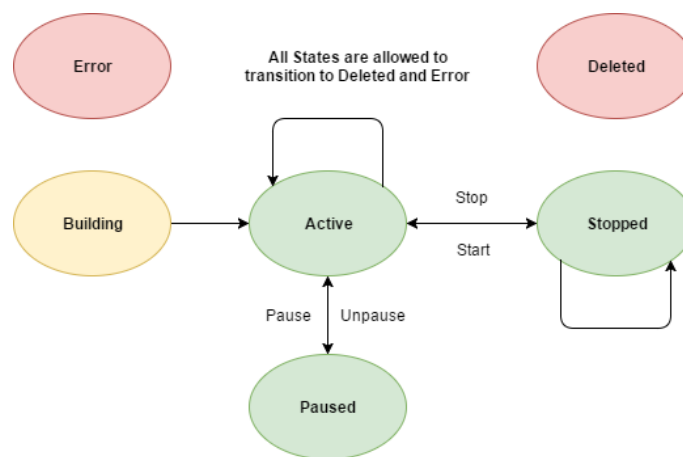


Figure 3.16: VNF instances status transitions

⁴²<https://docs.openstack.org/developer/tacker/devref/monitor-api.html>

⁴³https://docs.openstack.org/developer/tacker/devref/alarm_monitoring_usage_guide.html

OpenStack is very important to VNF Marketplace. Not only allows to create and administrate a cloud environment but also to deploy and manage VNFs. As was mentioned above, the VNF Marketplace can also communicate with other IaaS solutions. OpenStack has a vast number of services and functionalities that can be used in case of upgrading and add other activities into VNF Marketplace.

3.2 Development

After characterising VNF Marketplace, this section explains how every module was developed and how OpenStack and SLA Management were installed.

3.2.1 VNF Marketplace

3.2.1.1 Back-End

Since every module was created by scratch, it was needed an environment to develop the web application. The Laptop used had 12 GB RAM as memory, a Processor Intel Core i5-5300U CPU @ 2.30 GHz (4 CPUs), 500 GB disk space and Windows 8 distribution (64-bit).

The Back-End was created on a VM at VirtualBox with 64-bit Linux distribution (Ubuntu 16.04 LTS Desktop) with the following system requirements:

- 7 GB RAM;
- 2 CPUs (4 vCPUs);
- 100 GB disk space;
- Network – Bridge Adapter mode.

It was used in Bridge mode because OpenStack was connected to the Laptop private network. The base memory and processor could be lower than the configured but, for better performance, it was used the above configuration. After Ubuntu installation, it was needed to update the Ubuntu system and install Python (was installed Python 3.5). Since Back-End needed a test environment, it was installed Virtualenv. Virtualenv is a tool to create isolated Python environments and, with it, the Back-End is not dependent on the Ubuntu Python environment. After Virtualenv installation, it was necessary to create and activate the test environment. For that, figure 3.17 illustrates an example of this procedure.

```
marketserver@marketserver:~/Marketplace$ virtualenv env
Using base prefix '/usr'
New python executable in /home/marketserver/Marketplace/env/bin/python3.5
Also creating executable in /home/marketserver/Marketplace/env/bin/python
Installing setuptools, pip, wheel...done.
marketserver@marketserver:~/Marketplace$ source env/bin/activate
(env) marketserver@marketserver:~/Marketplace$
```

Figure 3.17: Virtualenv Creation and Activation Procedure

With the Python environment activated, the Back-End started to be developed. Python was essential since the framework used to create VNF Marketplace Back-End was Django. Django is a high level Python Web framework that encourages rapid development and clean, pragmatic design⁴⁴. To create a Django project it was used one command: ***django-admin startproject VNF-Marketplace***. This command not only creates the Back-End directory but also files that are going to be necessary to run VNF Marketplace server. For instance, the *settings.py* file has all the necessary configuration parameters to start the server. With Django, a SQLite3 database can be created. This database was used at VNF Marketplace and it was only needed to configure *settings.py* to the server start communicating with the database. In order to VNF Marketplace receive files such as VNF images, licenses files and VNFDs, it was created a directory to store them. The SQLite3 database only saves the path where these files are stored. To provide security, all files are stored with a UUID as the file name.

The easiest way to allow VNF Marketplace connect with other cloud infrastructures and Front-Ends is providing a fast and clean API. Nowadays, web applications are using REST API to fulfil those properties. Then, it was installed Django REST Framework. This framework allows communicating with the VNF Marketplace Back-End by HTTP messages. For this thesis, the only format present at the HTTP messages (requests and responses) is JSON. XML could also be used on HTTP messages but JSON is easiest to manage. At Appendix D is illustrated a text file (*requirements.txt*) with each tool used at the VNF Marketplace Back-End and, to install all of them, it is only necessary to run the command at the Back-End directory: ***pip install -r requirements.txt***.

For each internal module represented at the VNF Marketplace architecture, it was created a Django application. The creation of one application was performed by the following command (as for example): ***django-admin startapp VNFCatalog***. After creating and configure all application-s/modules, the server can now be instantiated with the command: ***python manage.py runserver (IP)***. Since Django already provide admin access, an Administrator Dashboard is created if at least one application allows the Administrator user access it and the database as well.

⁴⁴<https://www.djangoproject.com/>

The Billing module, as was mentioned in section 3.1.4, used Paypal Python SDK. This SDK provides Python APIs to create, process and manage payment. To test the billing processes, it was created a sandbox account at Paypal and a REST Application associated with the same account. Every request transmitted by VNF Marketplace is sent to the REST App and payouts/payments are processed inside it. The payouts are performed asynchronously for a better and fast performance. To create PDFs inside Billing module it was used the PDFKit tool. This JavaScript PDF generation library allows VNF Marketplace to create invoices PDFs from an HTML template. Finally, the Billing module used Django email properties (**django.core.mail**) to send emails to customers and sellers. The SMTP Back-End applied at VNF Marketplace was Gmail and it was created a Gmail account to send emails in name of VNF Marketplace.

The Authentication and Authorisation module used Django Rest Framework JWT to provide JSON Web Tokens for authentication. These tokens help users authenticate to every service in a fast and simple way (do not need to verify username and password, only token). Since is a token, the browser cookies can save it and, every request from VNF Marketplace Front-End to the Back-End, it is sent with the token as Header. Also, if the cookies already save the token, the user can close VNF Marketplace website tab, open it again and it will be not necessarily performing the log in procedure. The tokens have an expiration date but, when the user authenticates after expiration, a new token is created. The user's password is saved with SHA-256 hash encryption.

Finally, for the Back-End, the static files were configured in order to start constructing the Front-End. These statics files are present in a directory (inside VNF Marketplace project directory) and they have the purpose of storing all files that are required for the Front-End. For instance, in this directory, there are HTML, JS and CSS files from the Front-End. In *settings.py* was also added the template directory where is stored the index's HTML file of VNF Marketplace Front-End. The *settings.py* file of VNF Marketplace Back-End is present at Appendix D.

3.2.1.2 Front-End

After Back-End configuration and finalisation, the Front-End was developed. VNF Marketplace Front-End was developed with AngularJS. AngularJS is a Google Open Source Javascript Framework which helps to construct single-page applications. This framework adapts and extends traditional HTML to present dynamic content through two-way data-binding. Single-pages applications are similar to a desktop application. In other words, instead of each HTML file loading associated Javascript files, there is only a single page load (the index's HTML file loads all the Javascript, CSS and HTML files). To configure AngularJS, the index HTML file from VNF Marketplace Front-End must have a **ng-app** directive. This directive is the root element of an AngularJS application and it allows to inject every AngularJS directive and Javascript library into the Front-End application.

To navigate different HTML files at the VNF Marketplace Front-End, it was used AngularJS Routing. For instance, if a user clicked to see **Settings**, the AngularJS Routing (with HTML files directory already configured) tells **ng-app** to show the respective **Settings**' HTML file. Again, it is important to mention that all the HTML, Javascript and CSS files were loaded at the beginning of Front-End usage (the index's HTML file).

There were two AngularJS directives used to handle REST API Restful Resources: Restangular and ng-file-upload. Restangular is an AngularJS service that simplifies common GET, POST, DELETE, and UPDATE requests with a minimum of client code⁴⁵. With Restangular, the communication between the VNF Marketplace Front-End and the Back-End becomes possible. There is a good alternative to make requests to the Back-End designed by **\$http** which is an AngularJS core service. The HTTP requests (examples at Appendix C) used with Restangular are:

- GET – get specific information from the selected Back-End module;
- POST – submit/post specific information to the selected Back-End module;
- DELETE – delete specific information from the selected Back-End module;
- PUT – update specific information from the selected Back-End module.

All the requests and responses are handled by Restangular instead of one: Submission of VNF image, VNFD and license file simultaneously. To perform that task, ng-file-upload was used. Ng-file-upload is a lightweight AngularJS directive specifically to upload files⁴⁶. This directive was installed since uploading binary files (in this case the VNF image) with Restangular is not trivial and involves making difficult configurations. Also, with one request, it is needed to send the VNF image and the VNFD or the VNF image and the license text file (depending on the Purchase mode) to the Back-End which are two different types of files.

Finally, taking section 3.1.7 in consideration, every view is represented as a HTML file with the respective AngularJS controller. AngularJS controller is a regular Javascript object which controls the data of AngularJS applications. For instance, the VNF List's HTML file for the seller has an AngularJS controller responsible for getting information about the seller's VNFs (performs a request to the Back-End and send the response to the HTML).

⁴⁵<https://github.com/mgonto/restangular>

⁴⁶<https://github.com/danialfarid/ng-file-upload>

3.2.1.3 SLA Management

Following the SLA Management installation guide [40](#), it is simple to install the server. The requirements to install the SLA Core server are:

- Oracle JDK ≥ 1.6 ;
- MySQL ≥ 5.0 ;
- Maven ≥ 3.0 .

After installing the requirements (with the Virtualenv test environment or not), it is needed to download the project from Github [40](#). This will permit to start with the SLA Core installation. Next, it is necessary creating a MySQL database and a user as well. If the database and the user were successfully created, a specific file (*configuration.properties.sample*) needs to be saved as (*configuration.properties*) into the SLA Core parent directory.

The *Configuration.properties* file contains several parameters that can be configured. For instance, the SLA Enforcement jobs are configured inside this file. If the OpenStack VNF metrics are needed, *Configuration.properties* must get the file responsible for handling the metrics. Appendix D illustrates the *Configuration.properties* file used for VNF Marketplace SLA Management.

After the configuration file is completed, it is necessary to compile. To compile the server, it is used the command: ***mvn install***. If a change is made to the configuration file or to the server, it is required to compile again. Finally, to run the server and to start communicate with it, the mandatory command is ***./bin/runserver.sh***.

Figure 3.18 illustrates the topology used in this thesis with the VNF Marketplace server, the SLA Management module and OpenStack. The private network is created with a 8-Port Gigabit switch from D-Link. To Laptop 2 get Internet access, the Laptop 1 needed to share it.

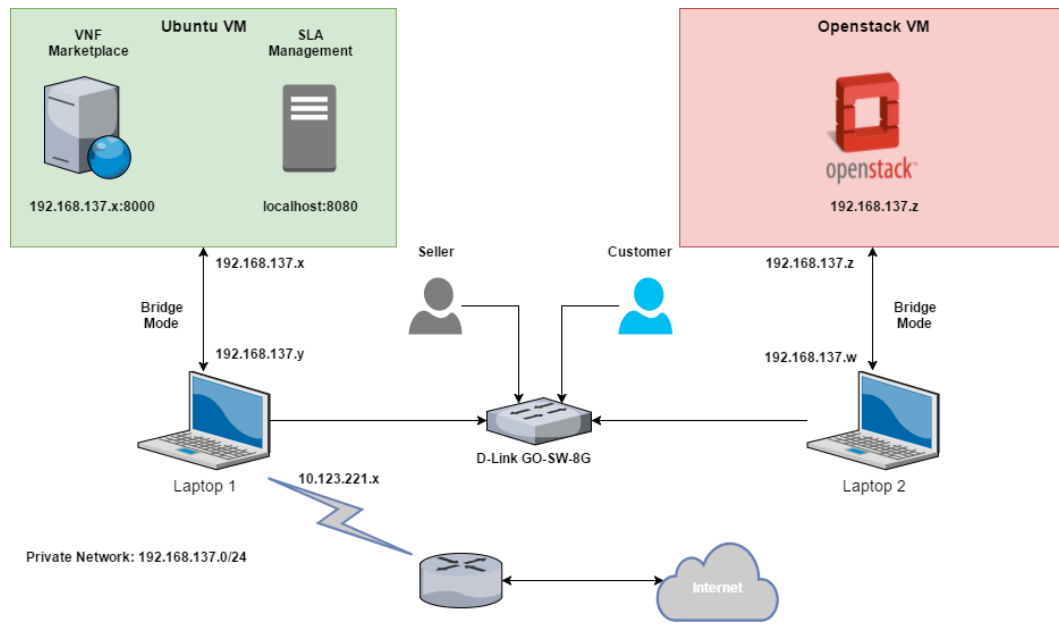


Figure 3.18: VNF Marketplace Topology

3.2.2 OpenStack

The figure above illustrates the OpenStack environment. It was used a Laptop with 12 GB RAM, a Processor Intel Core i5-2520M CPU @ 2.50 GHz (4 CPUs), 500 GB disk space and Windows 8.1 distribution (64-bit). The OpenStack VM was created with VirtualBox. It was used a 64-bit Linux distribution (Ubuntu 14.04 Server) with the following system requirements:

- 8 GB RAM;
- 2 CPUs (4 vCPUs);
- 100 GB disk space;
- Network – Bridge Adapter mode.

For this thesis, OpenStack was installed with Devstack. Devstack is a series of extensible scripts used to quickly bring up a complete OpenStack development environment⁴⁷. With Devstack it is not mandatory having great hardware requirements (Devstack needs at least 4 GB of RAM and 1 CPU to run). The first step was upgrading and updating the Ubuntu server. Next, it was created a user with passwordless sudo privileges. In other words, every task performed with sudo (a program that allows users to run programs with the security privileges) it is not necessary to enter the user's password.

⁴⁷<https://docs.openstack.org/developer/devstack/>

After the user's configuration, the Devstack repository was downloaded. The Devstack repository contains a script that installs OpenStack and the templates for configuration files. One configuration file was needed to be adapted for the pretended installation. The file is designed by *local.conf* and is present in the Appendix D. The *Local.conf* file is important since is where the user defines the desired OpenStack services and passwords. For this thesis, the *Local.conf* file contains the Tacker and Heat repository and other configurations such as the desired password for the Administrator user and the host IP of OpenStack. Lastly, it was only needed to run the installation script with the following command *./stack.sh*.

Normally, the Devstack installation process takes a bit of time. Figure 3.19 illustrates the conclusion of Devstack installation. It is possible to observe the total runtime of the installation, the host IP address, the default users and passwords and the Horizon's IP address. With the browser, the Administrator user can access the Horizon service and start using OpenStack.

```

=====
DevStack Component Timing
=====
Total runtime      2073

run_process        33
test_with_retry    5
apt-get-update     254
pip_install        325
wait_for_service   31
apt-get            70
=====

This is your host IP address: 192.168.137.221
This is your host IPv6 address: ::1
Horizon is now available at http://192.168.137.221/dashboard
Keystone is serving at http://192.168.137.221/identity/
The default users are: admin and demo
The password: secret
2017-06-05 09:11:37.061 | WARNING:
Services are running under systemd unit files.
For more information see:
https://docs.openstack.org/developer/devstack/systemd.html
2017-06-05 09:11:37.063 | Using lib/neutron-legacy is deprecated, and it will be removed in the future
2017-06-05 09:11:37.071 | stack.sh completed in 2073 seconds.

```

Figure 3.19: Conclusion of Devstack installation

After the conclusion of the Devstack installation, Devstack installed three networks for Tacker: *net0*, *net1* and *net_mgmt*. Also, by default, Devstack installs a public network to grant Internet access to the other networks and instances. Those three networks are, usually, used to manage and receive VNFs from Tacker. Since the VNFs instances need a tenant/project, during the Tacker installation it is installed a new tenant with *nfv* as name and *nfv_user* as the tenant's admin user. The *Nfv_user* does not have special privileges like the Administrator user of OpenStack. Then, the Administrator user of OpenStack was configured to be also an admin of *nfv* tenant.

Figure 3.20 illustrates the OpenStack topology after the conclusion of Devstack installation and configuration.

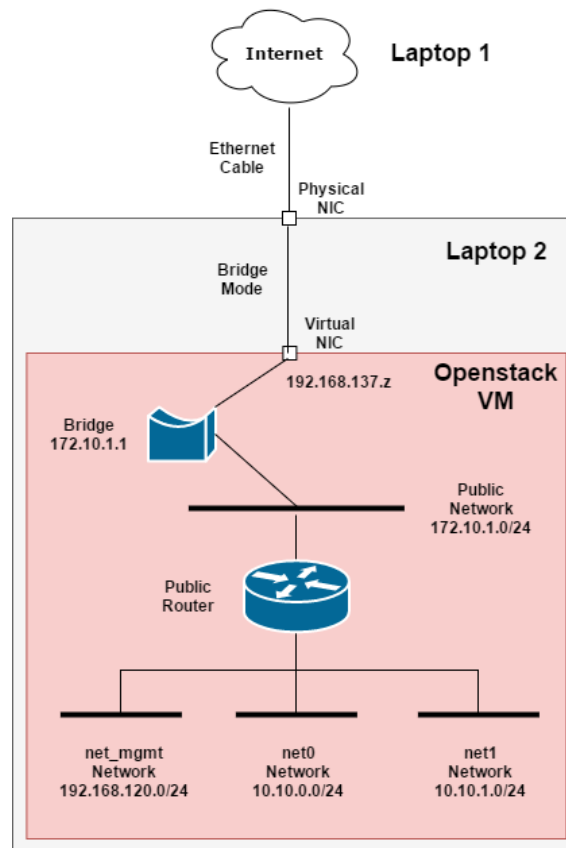
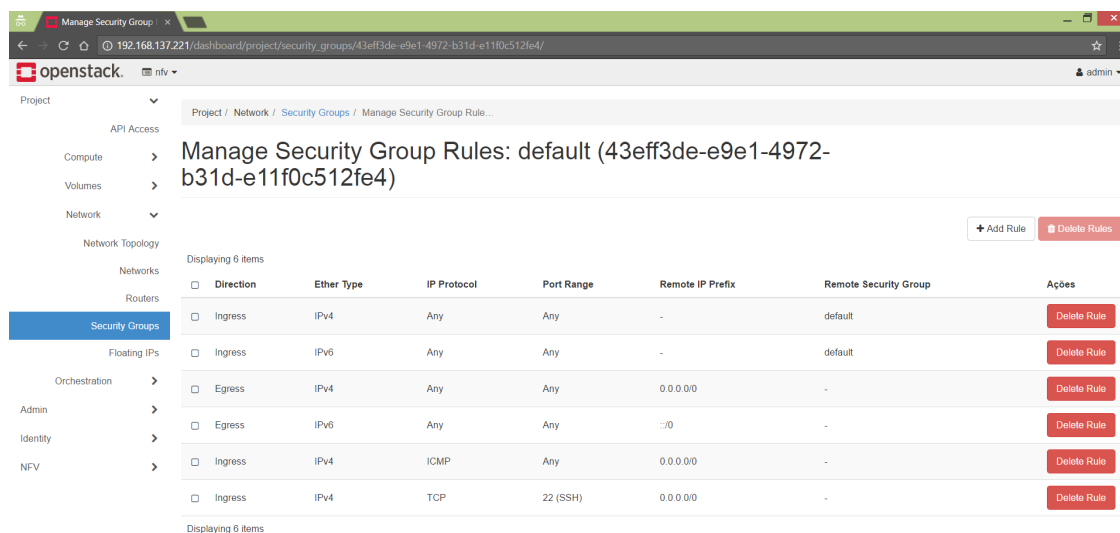


Figure 3.20: OpenStack Topology

To give access to the instances from the OpenStack VM, it was needed to configure the floating IPs to each VNF instance. A floating IP is a public IP and is used to communicate with networks outside the cloud. This configuration can be performed on the public Router which associate a free floating IP to the desired instance.

After associate a floating IP to an instance, it was needed to configure the OpenStack security groups. A security group is a named collection of network access rules that are used to limit the types of traffic that have access to instances. The following figure (figure 3.21) illustrates the security groups configured for this thesis.



Project / Network / Security Groups / Manage Security Group Rule...

Manage Security Group Rules: default (43eff3de-e9e1-4972-b31d-e11f0c512fe4)

+ Add Rule Delete Rules

Displaying 6 items

Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Ações
Ingress	IPv4	Any	Any	-	default	Delete Rule
Ingress	IPv6	Any	Any	-	default	Delete Rule
Egress	IPv4	Any	Any	0.0.0.0/0	-	Delete Rule
Egress	IPv6	Any	Any	:::/0	-	Delete Rule
Ingress	IPv4	ICMP	Any	0.0.0.0/0	-	Delete Rule
Ingress	IPv4	TCP	22 (SSH)	0.0.0.0/0	-	Delete Rule

Displaying 6 items

Figure 3.21: Security Groups from OpenStack

3.3 Summary

VNF Marketplace allows VNF vendors from all over the world sell VNFs to interested customers. Also, Tacker can provision and configure seller's VNFs. VNF vendors need to provide documents (VNFDs) defining single VNFs, in order for Tacker create and configure VNFs. Initially, VNFDs are stored in the VNF Catalog from Tacker and then VNFs are deployed if a customer buys the stored VNF. This service is an alternative to the customer and it has the advantage of not having to configure and install at customer's premises.

With VNF Marketplace, the business process is assured and the clients can buy VNFs without any problem. The users have an attractive and fast Dashboard with the essential information and actions.

Chapter 4

Results

This chapter discusses how VNF Marketplace and Openstack were analysed. Section 4.1 presents functional results of VNF Marketplace. Section 4.2 exposes functional results of Openstack. The last section (section 4.3) is divided into several tests performed among the VNF Marketplace, Openstack and the communication between them. The results are analysed and presented in each subsection.

4.1 VNF Marketplace Functional Results

After understanding the VNF Marketplace modules and how they were developed, it is necessary to present functional results. This is important since validates every module and functionality that was developed in this master's thesis project. Ideally all the functional results should be presented in this section. However, print screens of every VNF Marketplace were taken, and because of that, only the most important results for each module, are presented. Appendix C illustrates the others functional results of VNF Marketplace. Also, in this section, it is illustrated dashboard views and requests/responses accordingly to the respective functionality.

The tests were performed in the same topology illustrated in figure 3.18 and with the same hardware components mentioned above (section 3.2.1). All the print screens were taken at Ubuntu virtual machine. The Ubuntu VM has dynamic IP because the development was performed in different topologies and locations. Then, the private network assigns free IPs to each machine/VM accordingly to protocol DHCP. In order to perform this test, it was previously assumed that all the VNF Marketplace installations and configurations were already concluded. Openstack is also installed and configured. In case of some error, the system notifies the user of it but, for this section, it is assumed that everything is performed correctly. All the procedures of each module are detailed in section 3.1.

4.1.1 Authentication and Authorisation Module Functional Results

In this results, it is illustrated the registration, log in and profile. In other words, it is showed the registration and log in phase and the profile being saved at the Front-End. Before any action at VNF Marketplace, the user introduces the IP of the VNF Marketplace server in the Web Browser. Figure 4.1 illustrates the main page of VNF Marketplace before authentication.

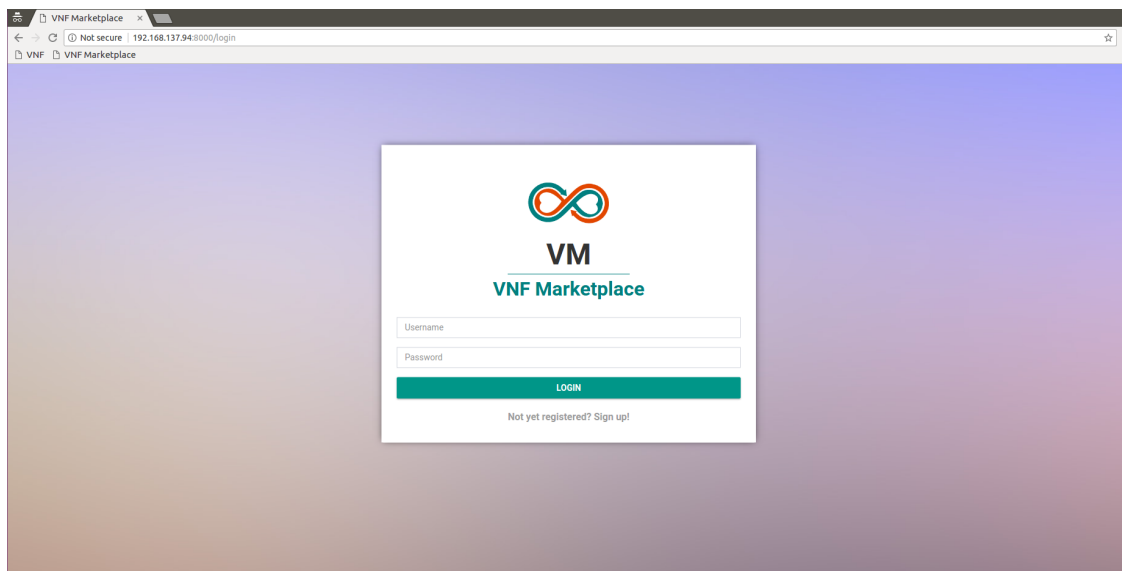


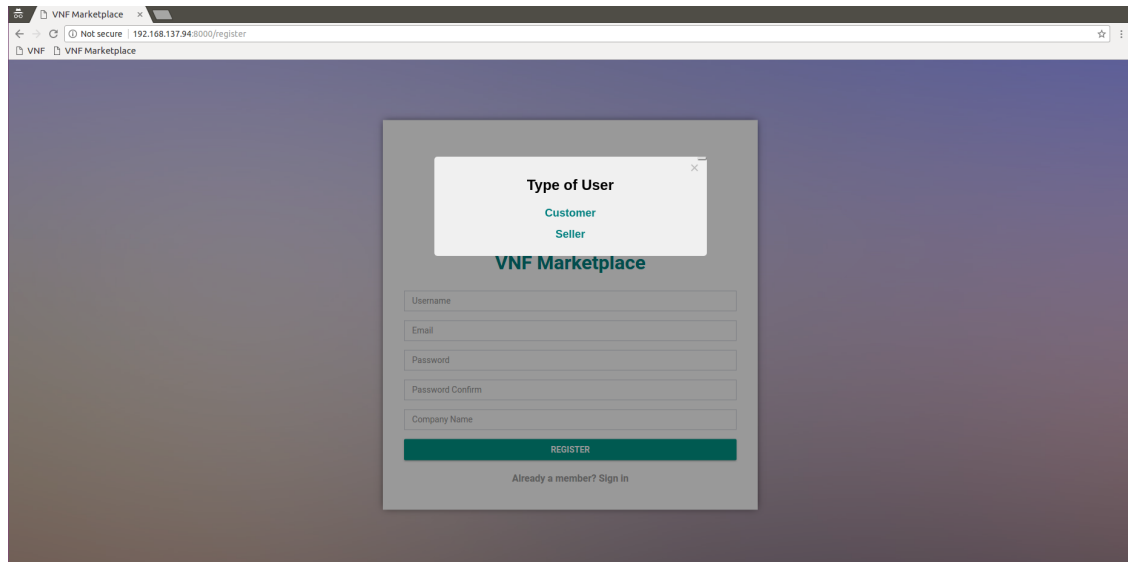
Figure 4.1: VNF Marketplace Main Page

When the user introduces the IP in the browser, the VNF Marketplace server collects all statics files inserted during the VNF Marketplace configuration/development. Without this files, the Front-End is not functional and the user does not have access to the Dashboard. Figure 4.2 presents some imports of those statics files. Not all are illustrated since the Front-End has a vast number of these files (HTML, CSS, JS and images).

```
[31/May/2017 19:26:25] "GET /static/bower_components/roboto-fontface/css/roboto/roboto-fontface.css HTTP/1.1" 200 12789
[31/May/2017 19:26:25] "GET /static/bower_components/font-awesome/css/font-awesome.css HTTP/1.1" 200 37414
[31/May/2017 19:26:25] "GET /static/bower_components/angular-rangeslider/angular.rangeSlider.css HTTP/1.1" 200 12544
[31/May/2017 19:26:25] "GET /static/css/styles.css HTTP/1.1" 200 5688
[31/May/2017 19:26:25] "GET /static/css/loader.css HTTP/1.1" 200 977
[31/May/2017 19:26:25] "GET /static/css/base.css HTTP/1.1" 200 19662
[31/May/2017 19:26:25] "GET /static/css/login.css HTTP/1.1" 200 1028
[31/May/2017 19:26:25] "GET /static/css/box.css HTTP/1.1" 200 10327
[31/May/2017 19:26:25] "GET /static/bower_components/bootstrap/dist/js/bootstrap.js HTTP/1.1" 200 69707
[31/May/2017 19:26:25] "GET /static/bower_components/codemirror/mode/yaml/yaml.js HTTP/1.1" 200 3693
[31/May/2017 19:26:25] "GET /static/bower_components/jquery/dist/jquery.js HTTP/1.1" 200 268039
[31/May/2017 19:26:25] "GET /static/bower_components/angular-ui-router/release/angular-ui-router.js HTTP/1.1" 200 175484
[31/May/2017 19:26:25] "GET /static/bower_components/angular/angular.js HTTP/1.1" 200 1249863
[31/May/2017 19:26:25] "GET /static/bower_components/codemirror/lib/codemirror.js HTTP/1.1" 200 358669
```

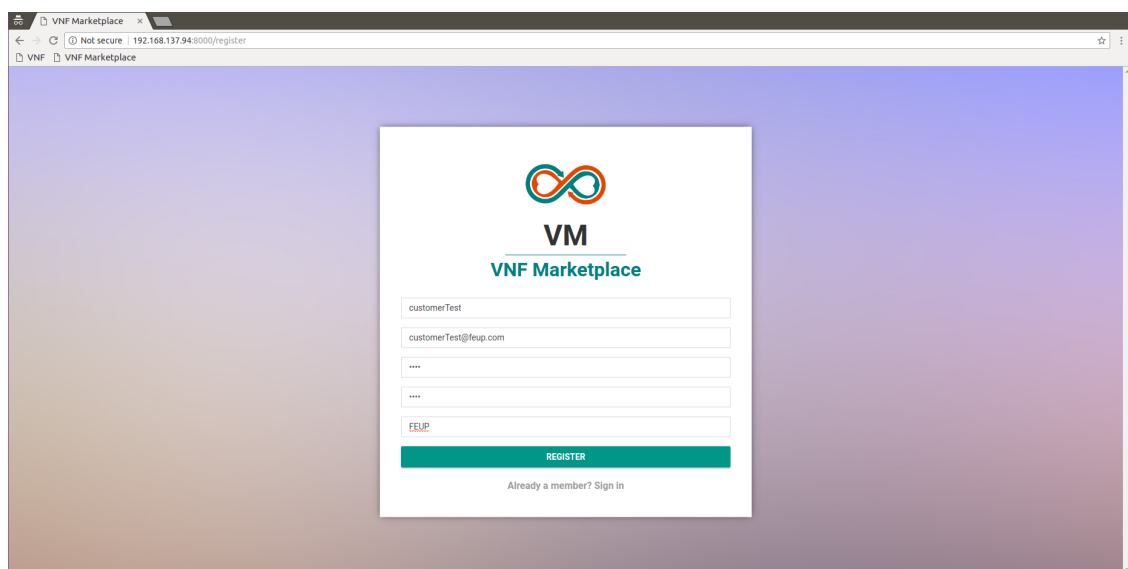
Figure 4.2: Imports of some static files of VNF Marketplace

After this, the user must sign up to be registered at VNF Marketplace. To simplify the registration phase, it is only illustrated one register. The registration is performed as customer. Figure 4.3 and figure 4.4 illustrate the selection of the role and the registration webpage, respectively.



The screenshot shows a web browser window with the address bar displaying "192.168.137.94:8000/register". The page title is "VNF Marketplace". A modal dialog titled "Type of User" is open, showing two options: "Customer" (selected) and "Seller". Below the modal, the registration form is visible, featuring input fields for Username, Email, Password, Password Confirm, and Company Name, followed by a "REGISTER" button and a link for "Already a member? Sign In".

Figure 4.3: VNF Marketplace Registration - Phase 1



The screenshot shows the same VNF Marketplace registration page, but the modal dialog is closed. The registration form is now filled with the following data: Username: "customerTest", Email: "customerTest@feup.com", Password: "****", Password Confirm: "****", and Company Name: "FEUP". The "REGISTER" button is highlighted in green, and the link "Already a member? Sign In" is visible below it.

Figure 4.4: VNF Marketplace Registration - Phase 2

Next, if no error occur, the registration is successfully done and the webpage is redirect to the log in phase. Figure 4.5 shows the request and the response of the registration to the Back-End. The reason why the the webpage is redirected to the login, after the user's registration, is related to a verification step of the email, that can me implemented as a security feature in the future.

```
{'group': 3, 'company_name': 'FEUP', 'username': 'customerTest', 'email': 'customerTest@feup.com', 'password_confirm': 'cust', 'password': 'cust', 'groups': [3]}
{'country': 'PT', 'company_name': 'FEUP', 'username': 'customerTest', 'first_name': '', 'last_name': '', 'city': '', 'password': 'pbkdf2_sha256$20000$LLbo3YvOnFZT$3n3QbXiYFx17dJoIAtrxcf+BESve2IcTrA3R8Msq9Dag=', 'groups': [3], 'address': '', 'id': 32, 'email': 'customerTest@feup.com'}
[20/Jun/2017 18:40:45] "POST /user-management/register/ HTTP/1.1" 201 262
```

Figure 4.5: Registration Request and Response

The login request is simple since only sends the username and password. The request is only a HTTP 201 (the request has been fulfilled, resulting in the creation of a new token). The token is used every time the user performs an action. Figure 4.6 illustrates the main page after the log in. The group number is the reference for each role. In this case, the number three is the Customer role/group.

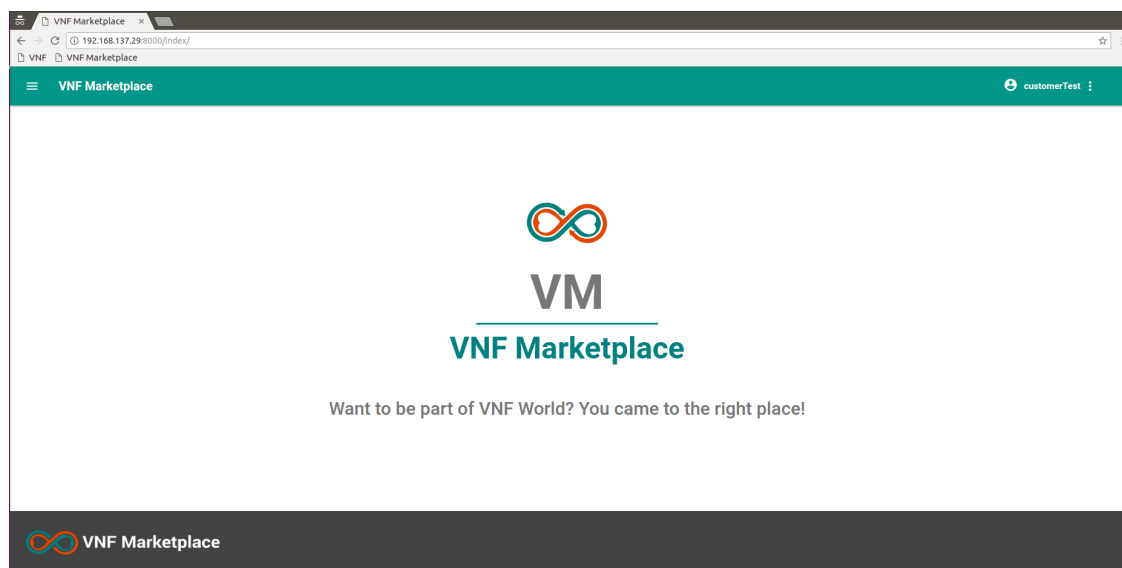


Figure 4.6: Customer Dashboard Home Page

Finally, figure 4.7 and figure 4.8 illustrate the profile update. This is important if the user want to change his profile and, for the future, this changes can be implemented to billing including the billing shipping address and credit card. The other requests, responses and Dashboard views are illustrated in Appendix C. Both seller and customer can edit their profiles.

Edit User Profile

Username: customerTest

Company: FEUP

Email: customerTest@feup.com

First name: Test

Last name:

Address:

City:

Country: Portugal

SUBMIT

Figure 4.7: Edit Profile Dashboard Menu

```
{'city': '', 'last_name': '', 'country': 'PT', 'id': 32, 'address': '', 'groups':
: [3], 'email': 'customerTest@feup.com', 'first_name': 'Test', 'company_name': '
FEUP', 'username': 'customerTest'}
{'city': '', 'last_name': '', 'country': 'PT', 'id': 32, 'address': '', 'groups':
: [3], 'email': 'customerTest@feup.com', 'first_name': 'Test', 'company_name': '
FEUP', 'username': 'customerTest'}
[20/Jun/2017 20:33:41] "PUT /user-management/profile/ HTTP/1.1" 200 175
```

Figure 4.8: Edit Profile Request and Response

4.1.2 VNF Catalog Module Functional Results

In this section is illustrated how a seller upload a VNF, how the customer browse in VNF Catalog and also a view of the purchased/owned VNFs. For the sake of simplicity, only some of the module functionalities are shown here.

To upload a VNF, the seller must fill the VNF information fields, select purchase mode, upload the VNF image and VNFD/license file (depending on the purchase mode) and fill the billing information. The next figures show the VNF publication at the Dashboard with the Basic Purchase Mode.

New VNF

←

Step 1
VNF Information

Step 2
VNF Composition

Step 3
Billing

1. VNF Information

VNF Name *

Virtual Router

VNF Description *

Virtual Router

VNF Version *

1.0.0

VNF Type *

vRouter

NEXT

Figure 4.9: VNF Publication in Basic Mode - Phase 1

VNF Marketplace - Google Chrome

VNF Marketplace

2. VNF Composition

Only Select One Mode:

☒ Basic Mode

Image Upload *

Select Image

Image: cirros-0.3.4-x86_64-disk.img

Licenses Upload *

Select License File

License File (.txt file): license.txt

Premium Mode

VNF Marketplace

Basic: Mode for VNF Download. Configuration and installation are necessary.

Premium: Mode for VNF Receiving. Configuration and installation are not necessary.

Figure 4.10: VNF Publication in Basic Mode - Phase 2

New VNF

←

Step 1 VNF Information

Step 2 VNF Composition

Step 3 Billing

3. Billing Information

Billing Model *

License Payment

Price per Period *

10

Period *

Month

Currency *

Euro

CREATE +

Figure 4.11: VNF Publication in Basic Mode - Phase 3

```
<QueryDict: {'license_file': [<TemporaryUploadedFile: license.txt (text/plain)>],
'file': [<TemporaryUploadedFile: cirros-0.3.4-x86_64-disk.img (application/x-r
aw-disk-image)>], 'typevnf': ['vRouter'], 'billing': [{'model': 'License Payment
', 'period': 'Month', 'price': {'value': 10, 'unit': '€'}}], 'modeID': ['Basic'], 'nam
e': ['Virtual Router'], 'description': ['Virtual Router'], 'id_image': ['bcd99c3
a-d6d6-4d55-848b-df6a11849a1c'], 'id_openstack': ['ID'], 'version': ['1.0.0'], '
vnfd_temp': ['']}>
{'owner': 'sellerTest', 'billing': '{"model": "License Payment", "period": "Month",
"price": {"value": 10, "unit": "€"}}', 'modeID': 'Basic', 'name': 'Virtual Router',
'owner_number': 33, 'id_image': 'bcd99c3a-d6d6-4d55-848b-df6a11849a1c', 'id_open
stack': 'ID', 'owner_company': 'FEUP', 'vnfd_temp': '/media/f29841bf-2b52-41bc-8
aaa-b565eed67956.yaml', 'license_file': '/media/4834c5dd-5271-4936-9858-1a412fc1
9060.txt', 'file': '/media/51f77702-73e0-4712-badc-57919a7c4b24.img', 'typevnf':
'vRouter', 'owner_email': 'sellerTest@feup.com', 'description': 'Virtual Router
', 'version': '1.0.0', 'pk': 7}
[21/Jun/2017 21:38:11] "POST /vnf-management/vnfs/ HTTP/1.1" 201 578
```

Figure 4.12: VNF Publication in Basic Mode - Request and Response

Figure 4.12 shows the request of the VNF Publication in the Dashboard and the response from the Back-End. It is possible to observe that the name files are with an UUID and the server saves the path where they are stored. Since the Database does not allow any empty file, VNF Marketplace has already templates configured and stored in the directory.

The next figures (figure 4.13 and 4.14) illustrate the VNF Publication for the Premium mode. Phase 1 is not illustrated because is just information fields and it was already shown above. Figure 4.15 has the VNF information submitted by the seller.

Basic Mode

☒ Premium Mode

Image Upload *



Select Image 

Image: cirros-0.3.3-x86_64-disk.img

VNFD Upload *

Select VNFD 

VNFD (.YAML file): sample-vnfd2.yaml

NEXT +

Figure 4.13: VNF Publication in Premium Mode - Phase 2

New VNF

←

Step 1
VNF Information

Step 2
VNF Composition

Step 3
Billing

3. Billing Information

Billing Model *

License Payment

Price per Period *

10

Period *

Month

Currency *

Euro

CREATE +

Figure 4.14: VNF Publication in Premium Mode - Phase 3

```

<QueryDict: {'license_file': [''], 'file': [<TemporaryUploadedFile: cirros-0.3.3-
x86_64-disk.img (application/x-raw-disk-image)>], 'typevnf': ['vFW'], 'billing'
: ['{"model": "Pay-As-You-Go", "period": "Hour", "price": {"value": 0.6, "unit": "$"}}'],
'modeID': ['Premium'], 'name': ['Virtual Firewall'], 'description': ['Virtual
Firewall'], 'id_image': ['0df9099b-1956-4143-8f90-c5234bcdec76'], 'id_openstack'
: ['ID'], 'vnfd_temp': [<TemporaryUploadedFile: sample-vnfd2.yaml (application/x-
yaml)>], 'version': ['1.0.2']]>

{'owner': 'sellerTest', 'billing': '{"model": "Pay-As-You-Go", "period": "Hour", "pr
ice": {"value": 0.6, "unit": "$"}}', 'modeID': 'Premium', 'name': 'Virtual Firewall',
'owner_number': 33, 'id_image': '0df9099b-1956-4143-8f90-c5234bcdec76', 'id_op
enstack': '7728df22-673c-472c-9a11-47540b1bc822', 'owner_company': 'FEUP', 'vnfd
_temp': '/media/a8802a52-c63c-4163-9d14-d79b9473ff9d.yaml', 'license_file': '/me
dia/8aadf26d-5ff9-4f2d-8dc4-d38a17ecaf15.txt', 'file': '/media/a201031c-fdc8-4f0
3-a8a5-33720caa8738.img', 'typevnf': 'vFW', 'owner_email': 'sellerTest@feup.com',
'description': 'Virtual Firewall', 'version': '1.0.2', 'pk': 8}

```

Figure 4.15: VNF Publication in Premium Mode - Request and Response

Comparing this response with Basic mode response, the field `id_openstack` saves the ID of the VNFD stored in the Tacker VNF Catalog. This is important because, with this ID, VNF Marketplace can request to deploy the pretended VNF. This is not displayed here, but later in section 4.2, it is illustrated the VNFD upload to OpenStack.

At this step, changing to the customer user, it is now possible to browse available VNFs. The customer can also search for specific types of VNFs instead of browsing all available VNFs. Figure 4.16 illustrates the Dashboard view to browse the VNF catalog and the other two figures (figure 4.17 and 4.18) illustrate the searching mechanism with the view and request/response respectively.

#	VNF Name	VNF Description	VNF Type	Modes	
8	Virtual Firewall	Virtual Firewall	vFW	Premium Base Price: 0.6 \$ + Setup: 30 \$ Payment Type: Pay-As-You-Go per Hour	SELECT VNF
7	Virtual Router	Virtual Router	vRouter	Basic Base Price: 10 € Payment Type: License Payment (available for one Month)	SELECT VNF

2 VNFs

Figure 4.16: VNF Marketplace Catalog

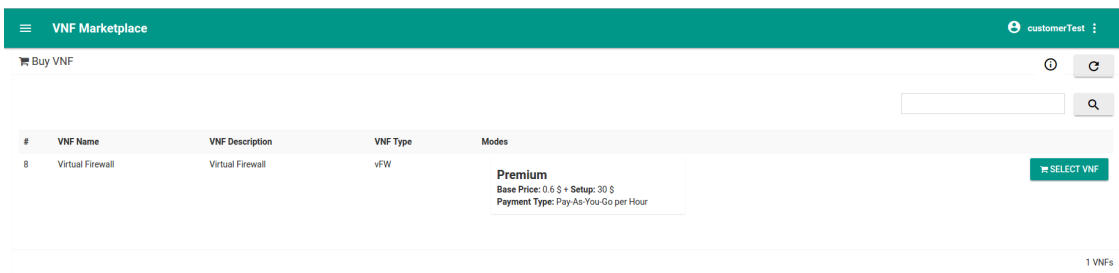


Figure 4.17: VNF Marketplace Catalog with search

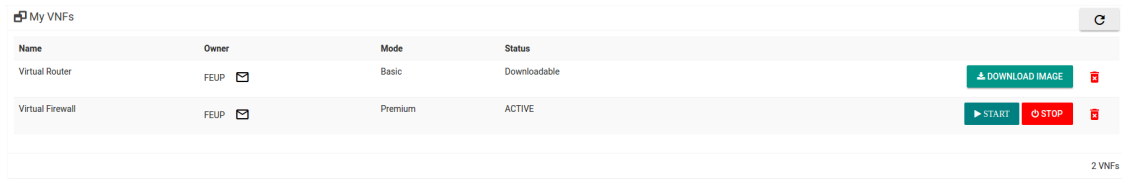
```
[OrderedDict([('pk', 8), ('name', 'Virtual Firewall'), ('file', '/media/a201031c-fdc8-4f03-a8a5-33720caa8738.img'), ('description', 'Virtual Firewall'), ('version', '1.0.2'), ('typevnf', 'vFW'), ('id_openstack', '7728df22-673c-472c-9a11-47540b1bc822'), ('vnfd_temp', '/media/a8802a52-c63c-4163-9d14-d79b9473ff9d.yaml'), ('license_file', '/media/8aadf26d-5ff9-4f2d-8dc4-d38a17ecaf15.txt'), ('id_image', '0df9099b-1956-4143-8f90-c5234bcdec76'), ('owner', 'sellerTest'), ('owner_number', 33), ('owner_company', 'FEUP'), ('owner_email', 'sellerTest@feup.com'), ('modeID', 'Premium'), ('billing', '{"model": "Pay-As-You-Go", "period": "Hour", "price": {"value": 0.6, "unit": "$"}}')])]
```

[21/Jun/2017 21:53:01] "GET /vnf-management/vnfs_type/?quer=search&typevnf=vFW HTTP/1.1" 200 612

Figure 4.18: VNF Marketplace with search - Request and Request

When the customer insert the desired VNF type in the search field, the REST request send through the URL the VNF type as a query parameter. The Back-End receives the query, filter the data according to the VNF type that customer insert and send the response to the Front-End.

Assuming that the customer has already purchased one or more VNFs, the customer can execute available functionalities to the VNFs. For instance, figure 4.19 illustrates the owned VNFs and what is possible to perform. To simplify, the next figures (figure 4.20 and 4.21) show requests and responses of performing the "start" action to a stopped VNF and the download of a VNF in Basic Purchase Mode.



Name	Owner	Mode	Status	
Virtual Router	FEUP	Basic	Downloadable	DOWNLOAD IMAGE
Virtual Firewall	FEUP	Premium	ACTIVE	▶ START ◻ STOP

2 VNFs

Figure 4.19: Customer's VNFs

```
{'hours_used': 0, 'paymentflag': False, 'owner': 33, 'FlavorKey': 'Flavor1', 'VNF
Type': 'vnf', 'paymentseconds': 0, 'VnfdID_openstack': '0a357bb5-3c0b-4606-bb46-e
8654330cdd8', 'vnfdImagePath': '/media/706834de-f7b1-4932-8e68-b528f1a6f2a7.img',
'vnfdTemplatePath': '/media/33dec217-1c21-44db-943b-39d75e430465.yaml', 'status'
: 'ACTIVE', 'name': 'Cirros VNF', 'LicenseFilePath': '/media/b0a68bfb-8b00-4325-8
351-1b9ea33f8598.txt', 'VnfdID': 33, 'owner_company': 'FEUP', 'owner_email': 'sel
lerTest@feup.com', 'templateID': '409', 'instanceID': '72623146-9ef4-4329-ba81-a9
f34672200f', 'serviceID': 'Premium', 'VnfdID_openstack': '7a9108c0-6e2b-4169-9e1a-
0c18d59bf15a', 'client': 'customerTest', 'agreementID': 'agreen33', 'VimID': 'VIM
0', 'pk': 25}
[22/Jun/2017 19:58:12] "PUT /vnf-management/cust_vnfs/25/ HTTP/1.1" 202 697
```

Figure 4.20: Start VNF - Request and Response

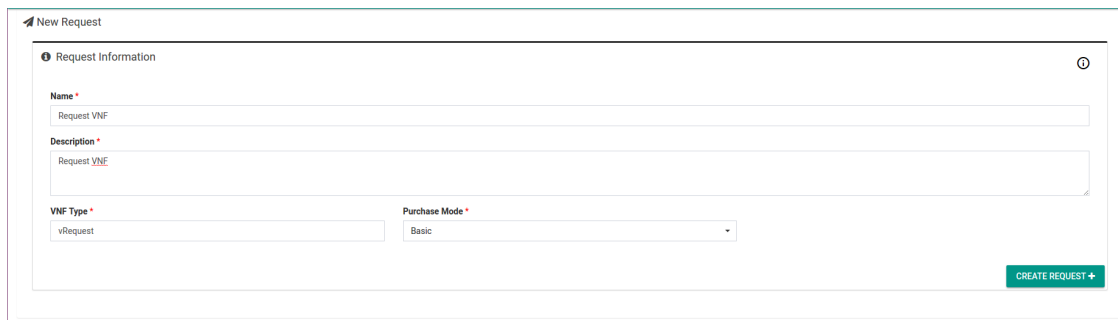
```
[22/Jun/2017 13:52:51] "GET /media/51f77702-73e0-4712-badc-57919a7c4b24.img HTTP
/1.1" 200 13267968
```

Figure 4.21: Customer's VNF Download Response

4.1.3 Request Module Functional Results

The Request Module allows the customers to request both non-available VNFs and existing VNFs with other Purchase Mode. In this subsection, it is illustrated how a customer upload requests, how the seller reply to them and, lastly, how a customer performs actions to upload requests (delete/select). Again, Appendix C has all the other requests, responses and Dashboard views of the Request Module functional results.

The following figures (figure 4.22 and 4.23) illustrate the view, request and response of a customer uploading a request to the system. The customer only need to fill the information fields, so that request is uploaded.



New Request

Request Information

Name *
Request VNF

Description *
Request VNF

VNF Type *
vRequest

Purchase Mode *
Basic

CREATE REQUEST +

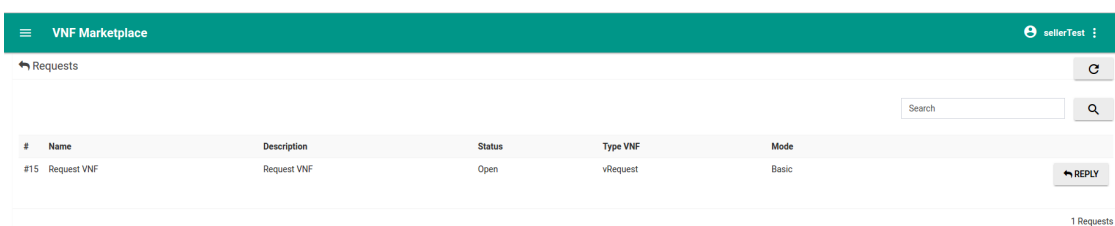
Figure 4.22: New Request Dashboard View

```
{'name': 'Request VNF', 'typevnf': 'vRequest', 'description': 'Request VNF', 'mode': 'Basic'}
{'vnfID': None, 'status': 'Open', 'owner': 'customerTest', 'pk': 15, 'description': 'Request VNF', 'client': None, 'name': 'Request VNF', 'mode': 'Basic', 'typevnf': 'vRequest', 'client_company': None}
[22/Jun/2017 17:05:07] "POST /vnf-management/requests/ HTTP/1.1" 201 182
```

Figure 4.23: New Request - Request and Response

Figure 4.23 illustrates the response from the Back-End. There is no VNF ID because the request is still with the "Open" status. After a seller reply to the submitted request, the Status will change to "Closed" status and the VNF ID will be associated with the ID of the seller's VNF. This is really important because it gives the ability to customers to buy the VNF from customer's requests.

After uploading, the seller can observe the uploaded request with an "Open" status. Figure 4.24 shows the available request at the Dashboard. The reply button redirect the webpage to the VNF publication.



#	Name	Description	Status	Type VNF	Mode
#15	Request VNF	Request VNF	Open	vRequest	Basic

1 Requests

Figure 4.24: Seller Requests Dashboard View

Finally, assuming that two requests were uploaded and one of them was replied by one seller, the customer can choose between delete both or select the replied request. The next figures (figure 4.25 and 4.26) illustrate the customer's requests view and the request/response to collect from them.

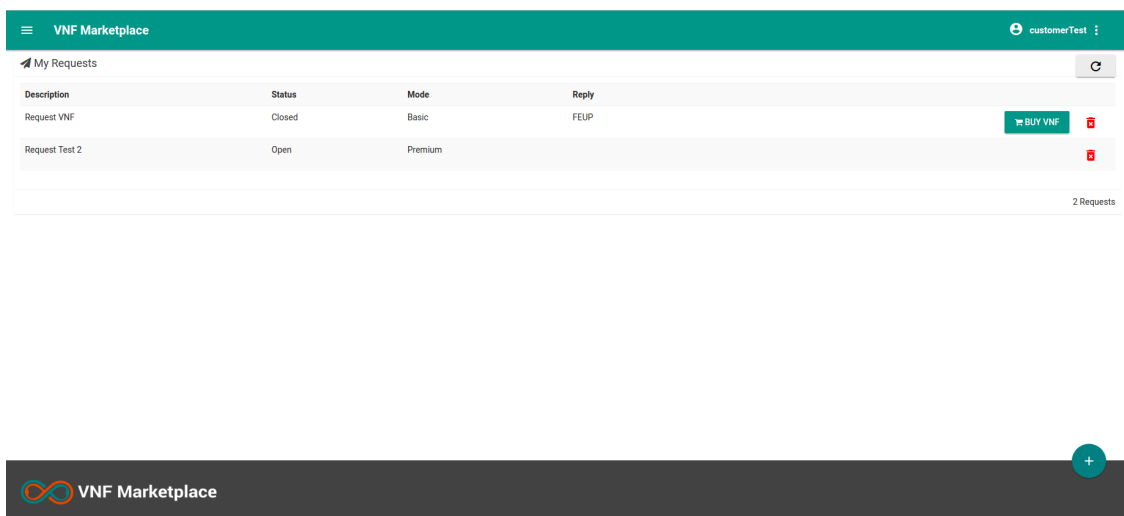


Figure 4.25: Customer Requests Dashboard View

```
[OrderedDict([('pk', 15), ('name', 'Request VNF'), ('description', 'Request VNF'), ('typevnf', 'vRequest'), ('owner', 'customerTest'), ('client', 'sellerTest'), ('client_company', 'FEUP'), ('status', 'Closed'), ('mode', 'Basic'), ('vnfID', 31)]), OrderedDict([('pk', 16), ('name', 'Request Test 2'), ('description', 'Request Test 2'), ('typevnf', 'vRequest'), ('owner', 'customerTest'), ('client', None), ('client_company', None), ('status', 'Open'), ('mode', 'Premium'), ('vnfID', None)])])
[22/Jun/2017 17:22:27] "GET /vnf-management/requests/ HTTP/1.1" 200 385
```

Figure 4.26: Customer Requests - Request and Response

4.1.4 Billing Module Functional Results

In this section the customers can find the available payment options, the process through what the sellers and the customers have access to detailed information about billing and also the way an email is sent with the invoice. Since VNF metrics are not being generated, it is assumed that there are no discounts. Appendix C contains the other requests, responses and Dashboard of the Billing Module functional results including the request and the response to check if the VNF is active or inactive.

Assuming that the customer wants to buy a VNF, after selecting the desired VNF, it is presented the multiple available options to purchase it. Those options include Paypal and credit card: If the VNF is in the Premium mode, it is only needed perform this action once. The Premium mode has the Pay-As-You-Go billing model so the customer only have to perform this action once. Figure 4.27 shows the payment option mentioned above and figure 4.28 illustrates the Paypal login to proceed with payment.

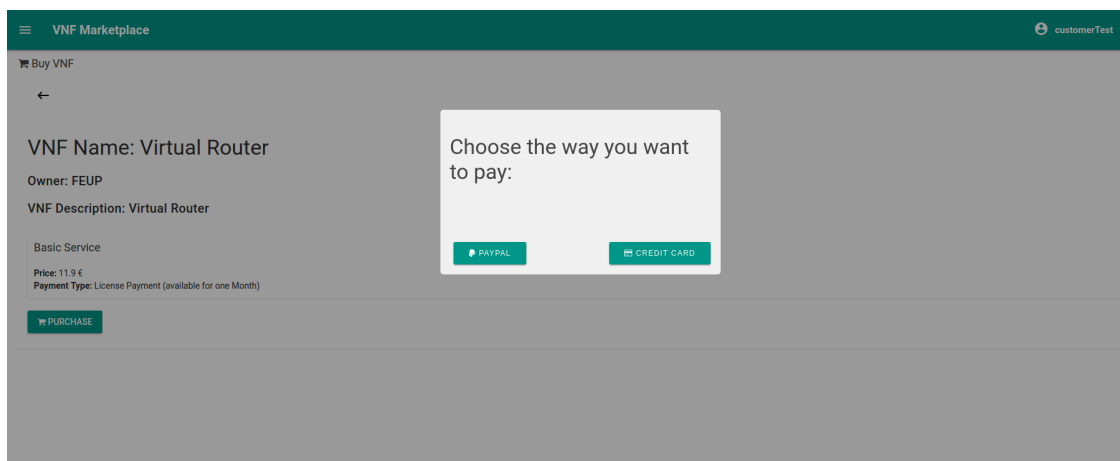


Figure 4.27: VNF Payment Options

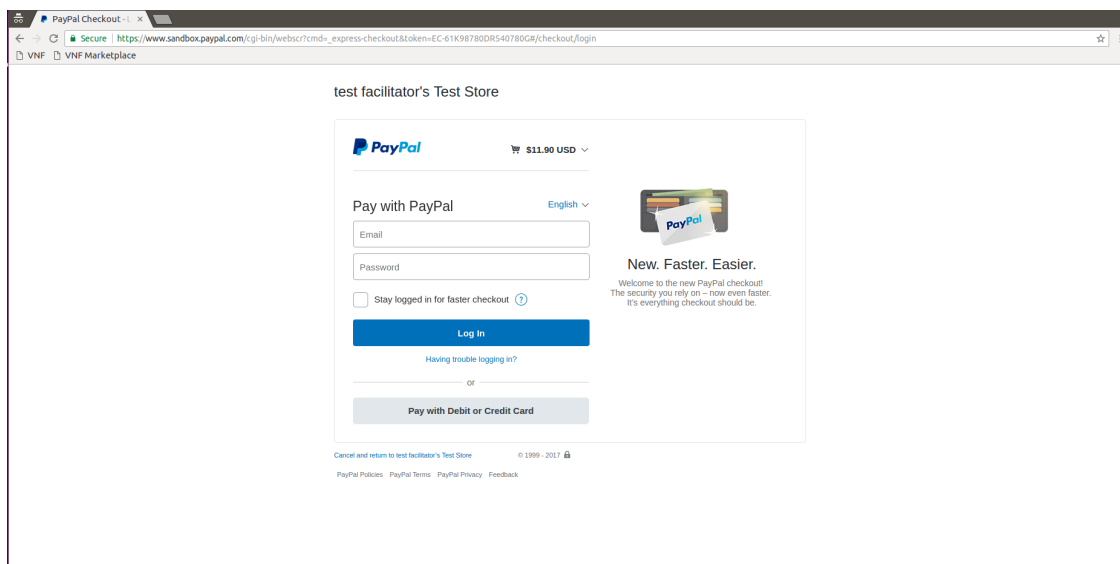
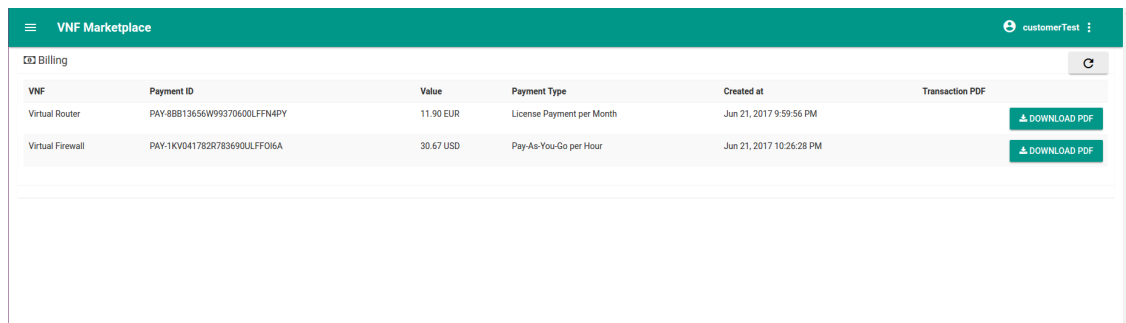


Figure 4.28: Paypal Dashboard View

After the customer buy one or more VNFs, the payments and payouts are performed in order to complete one of the objectives defined for the Billing Module and to accomplish the Market Share. Since the Dashboard and the Billing Module allow customers/sellers to see detailed information about billing, the next figures (figure 4.29, 4.30 and 4.31) show the view, the request, the response and the invoice to demonstrate it. The only aspect that changes in between the two Stakeholders are the PDF and the payload/payment. Then, for testing purposes, it is only showed for the customer case.



VNF	Payment ID	Value	Payment Type	Created at	Transaction PDF
Virtual Router	PAY-8BB13656W99370600LFFN4PY	11.90 EUR	License Payment per Month	Jun 21, 2017 9:59:56 PM	Download PDF
Virtual Firewall	PAY-1KV041782R783690ULFFOIA	30.67 USD	Pay-As-You-Go per Hour	Jun 21, 2017 10:26:28 PM	Download PDF

Figure 4.29: Customer Billing Dashboard View

```
[OrderedDict([('pk', 4), ('paymentID', 'PAY-8BB13656W99370600LFFN4PY'), ('payloadID', 'XZNP7P2KTSRVC'), ('value', 10.0), ('value_cust', '11.90'), ('currency', 'EUR'), ('owner', 'sellerTest'), ('client', 'customerTest'), ('vnfID', 'Virtual Router'), ('paymentType', 'License Payment per Month'), ('created', '2017-06-21T20:59:56.484335Z'), ('transactionClientPDF', '/media/64094c64-1e12-4c99-b781-425249fa8db1.pdf'), ('transactionSellerPDF', '/media/b5e67434-1e0b-4867-85de-139e0af4a6a9.pdf'))], OrderedDict([('pk', 8), ('paymentID', 'PAY-1KV041782R783690ULFFOIA'), ('payloadID', 'U6YFEADCWZ8SQ'), ('value', 0.6), ('value_cust', '30.67'), ('currency', 'USD'), ('owner', 'sellerTest'), ('client', 'customerTest'), ('vnfID', 'Virtual Firewall'), ('paymentType', 'Pay-As-You-Go per Hour'), ('created', '2017-06-21T21:26:28.982817Z'), ('transactionClientPDF', '/media/99c82718-639a-44f2-8dd5-6c15a23d319f.pdf'), ('transactionSellerPDF', '/media/458849af-4cea-4789-9a14-403e8c12eafd.pdf'))])
[21/Jun/2017 22:53:03] "GET /billing-management/billing_cust/ HTTP/1.1" 200 857
```

Figure 4.30: Customer Billing - Request and Response

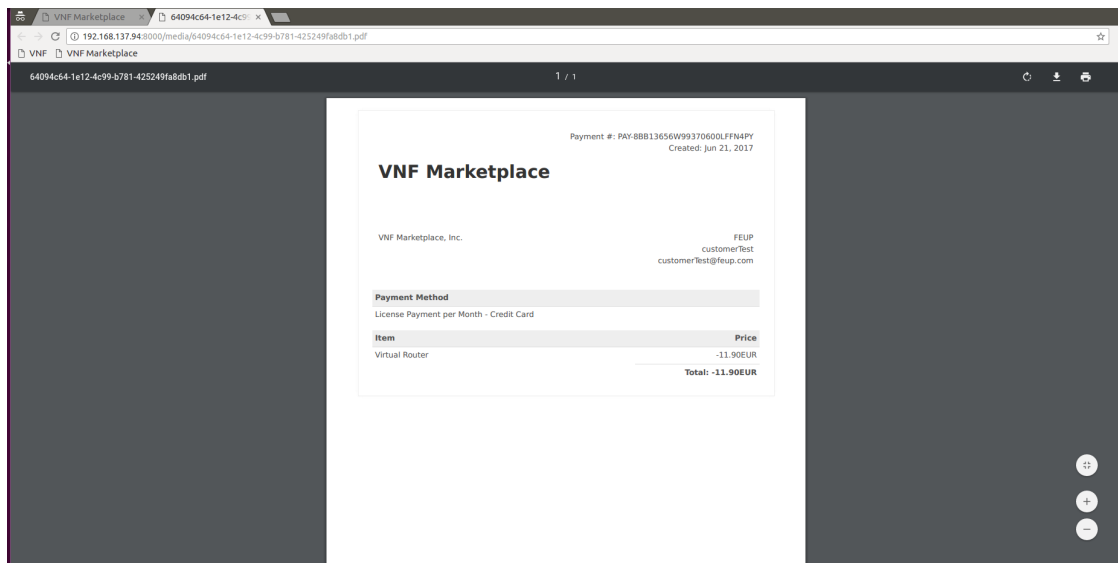


Figure 4.31: PDF with VNF Purchase (Customer)

Lastly, it was implemented a feature to allow customers/sellers to get the invoices without the Dashboard access. To fulfil that purpose, an email mechanism was developed and, every time a customer buy a VNF, the invoices are sent to the email of both Stakeholders (seller/customer). Besides that, invoices are in the PDF format. Figure 4.32 illustrates the Gmail account with both emails. For testing purposes, it was used the same email.

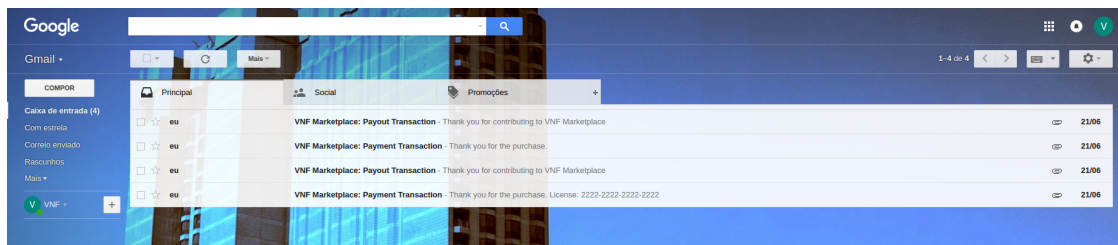


Figure 4.32: Gmail account with the invoices

4.1.5 SLA Management Functional Results

The SLA Management Module allows the VNF Marketplace to create SLA templates and SLA agreements. In this subsection, it is illustrated requests and responses between them to demonstrate the development of the pretended SLAs. According to section 3.1.5, the metrics are not being generated at OpenStack so it is assumed that does not exist SLA violations.

Before deploy a VNF, it is necessary to create SLA templates and SLA agreements. The VNF Marketplace use pre-defined SLA parameters which must be inserted in the SLA template and the SLA agreement. Assuming that the customer bought a VNF in the Premium Purchase Mode with the SLA parameters from *flavor 1* (first group of SLA parameters), the next figures (figure 4.33 and 4.34) show requests and responses in order to create the pretended SLA template and SLA agreement.

```

VNF_SLA
{
  "context": {
    "agreementInitiator": null,
    "agreementResponder": "18",
    "service": "vFW",
    "serviceProvider": "AgreementResponder",
    "templateId": "vnf8"
  },
  "name": "Virtual Firewall_Flavor1",
  "templateId": "vnf8",
  "terms": {
    "allTerms": {
      "guaranteeTerms": [
        {
          "businessValueList": {
            "customBusinessValue": [
              {
                "count": 1,
                "penalties": [
                  {
                    "expression": 2,
                    "type": "Discount",
                    "unit": "%",
                    "validity": "P2H"
                  }
                ]
              }
            ]
          },
          "name": "cpu_util",
          "qualifyingCondition": null,
          "serviceLevelObjective": {
            "kpitarget": {
              "customServiceLevel": "{\\\"policies\\\": [{\\\"count\\\": 2, \\\"interval\\\": 360}], \\\"constraint\\\": \\\"cpu_util GT 75\\\"}",
              "kpiName": "cpu_util"
            }
          },
          "serviceScope": null
        }
      ],
      "serviceDescriptionTerm": {
        "name": "requirements",
        "serviceName": "Flavor1"
      },
      "serviceProperties": [
        {
          "name": "MonitoredMetrics",
          "serviceName": "default",
          "variableSet": {
            "variables": [
              {
                "location": "/monitor/cpu_util",
                "metric": "xs:double",

```

Figure 4.33: SLA Template

All the SLA parameters from *flavor_1* are presented here. In section 3.1.5 is explained what SLA parameters are included in this flavor. This template is defined at SLA Management Github⁴⁰ and must follow this topology. The context area has the provider ID (VNF Marketplace), the template ID and others. Analysing the SLA template further, the guarantee terms have the SLA parameters pre-defined by VNF Marketplace. In a more detailed way, the Business Value List contains the appropriate discount and the Service Level Objective has the condition to perform the discount. Since this prototype is on the first version, the customer does not have too many options. The Service Properties has the type of metric the SLA Management must evaluate but OpenStack is not generating them.

```
POST http://localhost:8080/sla-service/agreements/ 201 Location: http://localhost:8080/sla-service/agreements/agreem8
PUT http://localhost:8080/sla-service/enforcements/agreem8/start 202
```

Figure 4.34: Creation of SLA Agreement

To post a SLA Agreement, it is necessary to send the customer ID and the template ID. The creation of SLA agreement is only performed if the customer selects the desire SLA flavor and purchases the respective VNF. If the SLA agreement was successfully created, SLA Management starts with the enforcement jobs to fulfill the SLA agreement.

4.1.6 SLA Control Module Functional Results

In order to improve the communication between VNF Marketplace and the SLA Management system, the SLA Control module was developed. The next results show the collected information of the SLAs agreements (violations, agreement identifier, provider, etc). This information is available to each customer that bought a VNF. Again, since Openstack does not generate any VNF metric, the violations are not saved in the SLA Control Module even if the module is prepare for that.

The first interaction of this module is when a customer buy a VNF. The SLA agreement is created and SLA Control must serve as an intermediary between VNF Marketplace and the SLA Managment Module. Also, it is necessary to collect essential information to provide to the customers. Figure 4.35 illustrates the request and the response to save that information. Lastly, the other figures (figure 4.36 and 4.37) show the SLA details in the customer's dashboard. It is assumed that the customer already performed a VNF purchase in Premium mode.

```
{'flavorId': 'Flavor1', 'VnfId': 8, 'ProviderId': 'FEUP', 'AgreementId': 'agreem8', 'productType': 'vnf', 'VnfName': 'Virtual Firewall'}
{'flavorId': 'Flavor1', 'pk': 6, 'AgreementId': 'agreem8', 'VnfId': 8, 'dateTerminated': None, 'ProviderId': 'FEUP', 'SLAPenalties': 0, 'dateCreated': '2017-06-21T21:26:32.194103Z', 'ClientId': 'customerTest', 'productType': 'vnf', 'VnfName': 'Virtual Firewall'}
```

Figure 4.35: SLA Control SLA Information - request and reponse

The reason why the Provider ID is saved with the seller and not with the VNF Marketplace is for further development at the VNF Marketplace. At the future, it can be implemented the possibility for sellers to configure and install the VNFs in their premises. With this improvement, they have direct interaction with their own VNFs and they only need to configure and assure the pre-defined SLAs.

Agreement ID	VNF Product	Provider	Date Created
agreem8	Virtual Firewall	FEUP	Jun 21, 2017 10:26:32 PM

Figure 4.36: Customer SLA Dashboard View

←

VNF Name: Cirros VNF

Agreement: agreem33

Flavor: Flavor1

For each VDU:

Monitoring Parameter: CPU Utilization

In case CPU Utilization is Greater than or equal 75% during 2 breaches count with a 360 seconds interval: 10% discount in the payment

Figure 4.37: Customer SLA Details Dashboard View

4.1.7 Dashboard & Analytics Module Functional Results

For testing purposes, these two components are together because the Dashboard performs all the analytics objectives. As was shown in section 3.1.8, the Analytics Module was created and implemented in the VNF Marketplace but does not have any functional connection with the other modules. The Analytics Module will be important in the future to get all the information from the other modules. This will provide statistics of billing, searching, VNF monitoring, among others. It will help to produce KPIs for market studying and other purposes. The Dashboard could also be analysed in this section but, since all the above functional tests already provide dashboard views, it is only illustrated the graphical representation of some statistics. The requests and the responses are in Appendix C.

For the customer's and seller's Dashboard view, the analytics are presented in the "Analytics" service at the Dashboard menu. Considering that the customer has already bought VNFs, figure 4.38 shows the customer's dashboard view with some graphs. It is provided VNF statistics without VNF metrics and monitoring due to bad installation of OpenStack Ceilometer/Gnocchi (OpenStack Telemetry Service).



Figure 4.38: Customer Statistics Dashboard View

Finally, assuming that the seller has sold some of his VNFs, figure 4.39 illustrates statistics from the Dashboard about billing and other parameters.

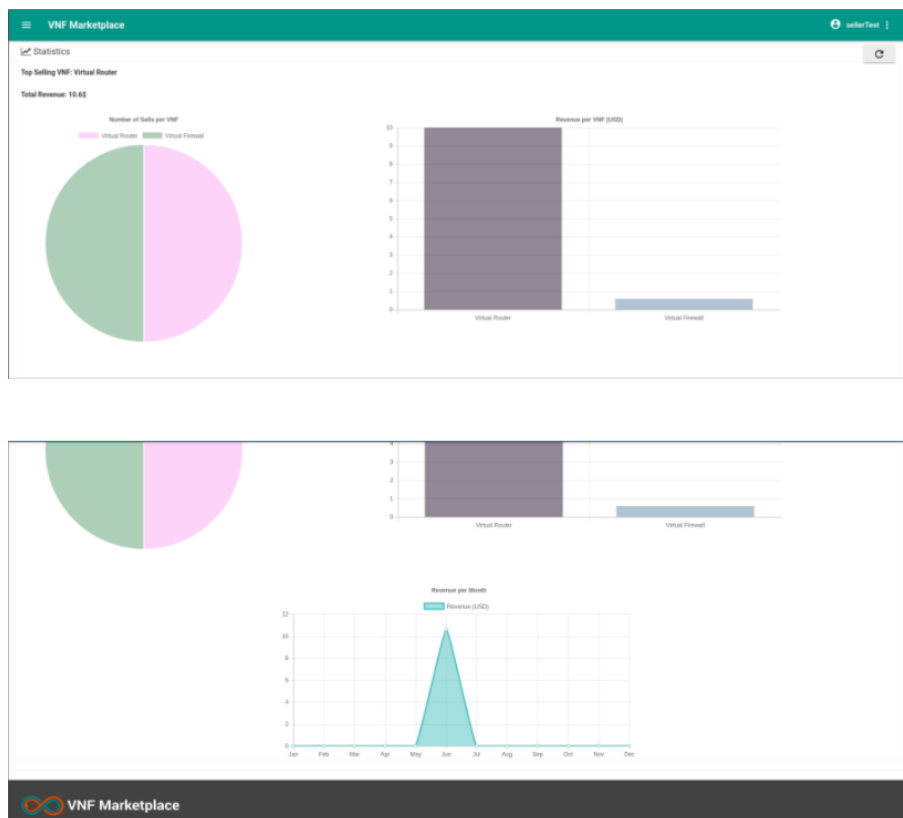


Figure 4.39: Seller Statistics Dashboard View

4.2 Functional Results of the interaction with OpenStack

After understanding OpenStack, the interactions with VNF Marketplace and how it was installed, it is necessary to present functional results. In this section are only presented the results obtained from the communication between VNF Marketplace and OpenStack. This is important since validates one objective of this master's thesis project. The results are illustrated with Horizon views and requests/responses accordingly to the respective functionalities.

The tests were also performed in the same topology illustrated in figure 3.18 and with the same hardware components mentioned above (section 3.2.2). All the print screens were taken at the Ubuntu virtual machine. The OpenStack VM has also a dynamic IP. For this tests, it is assumed that all the VNF Marketplace installation and configuration has already been performed. OpenStack is also installed and configured. After configuration, OpenStack and the OpenStack VM have the same topology mentioned in section 3.2.2. In case of some error, OpenStack notifies both Administrator and VNF Marketplace but, for this section, it is assumed that everything is performed correctly. Lastly, all the interactions with the VNF Marketplace are detailed in section 3.1.9.

4.2.1 VNF Publication

Before deploying VNFs, Tacker needs to store the VNFD and Glance the VNF image. During VNF publication phase, the seller needs to upload the VNFD and the VNF image into VNF Marketplace. This will allow the intended interaction with OpenStack to be successful. Taking figure 4.16 in consideration, it is possible to observe a successful VNFD and VNF image upload. To provide another perspective, figure 4.40 illustrates the Horizon view of VNF Catalog. Figure 4.41 shows the VNFD upload from the VNF Marketplace.

Name	Description	Service Types	Catalog Id
Virtual Firewall	Virtual Firewall	-	7728d022-673c-472c-9a11-47540b1bc822

Figure 4.40: Tacker VNF Catalog - Horizon view

With this response, VNF Marketplace knows that Tacker VNFM is deploying the VNF and then the customer will start avail the purchased VNF. The response also shows another information such as the Heat template (Tacker communicates with Heat to transform VNFD into a Heat template), the VIM ID, the VNF name (it function like an ID) and the instance ID which will be very useful to configure the instance every time the customer wants to stop/start the VNF.

To check if the VNF is really deployed, it is necessary to analyse first the VNFD template before access the VNF. The uploaded VNFD mentioned above (section 4.2.1) is also illustrated in figure 2.6 (only the name of VNF image is different). It is possible to observe that the VDU will have 1 CPU, 512 MB of RAM and 1 GB disk space. The VNF name is different but is a Cirros image. The VNF will also be connected to the *net_mgmt* network. After studying the VNFD, figure 4.43 shows the topology of the virtual networks inside OpenStack (cloud environment) after the VNF deployment. The VNF is associated to a dynamic IP accordingly to the *net_mgmt* network. Finally, to use the deployed VNF, OpenStack provide (through Horizon) a console to configure the VNF. Figure 4.44 illustrates the console demonstrating the deployed VNF.

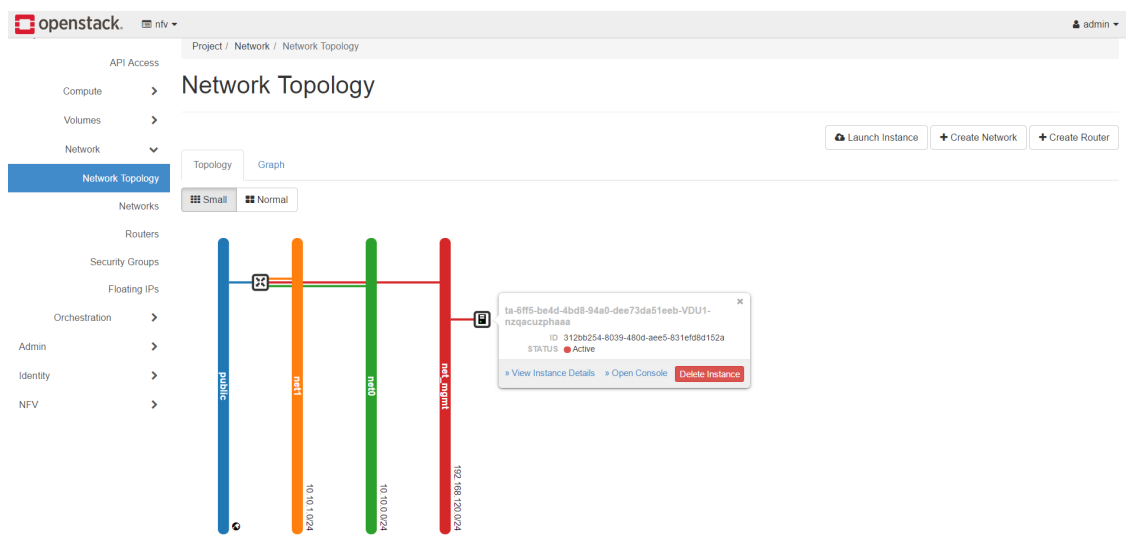


Figure 4.43: OpenStack Networks with a deployed VNF and a Neutron Router

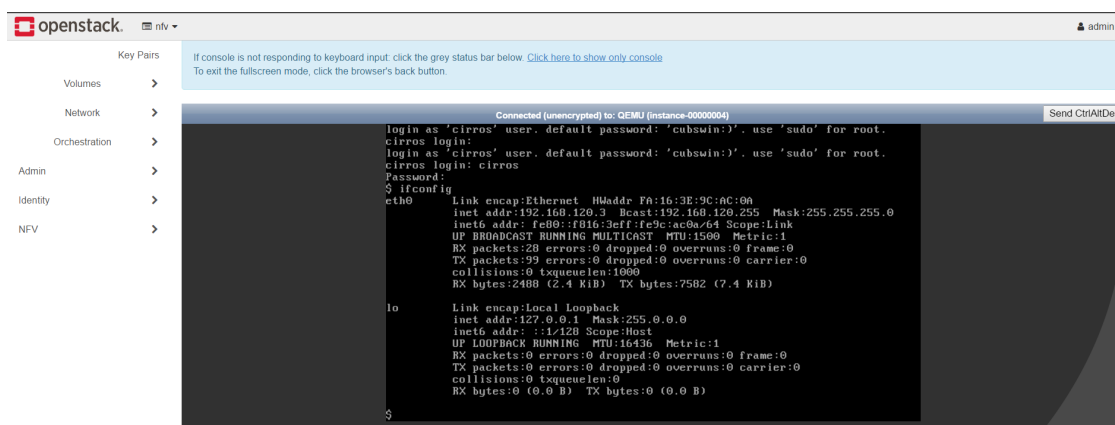


Figure 4.44: Console with the VNF deployed

4.3 Tests Results

This section presents different types of tests for both VNF Marketplace and OpenStack. Analysing only functional results is not enough to measure the potential of VNF Marketplace. Then, it was performed tests of performance and interviews for different results and analyses. The SLA Management was not individually evaluated in this section (only on installation) because it were performed global tests to VNF Marketplace. Section 4.3.1 analyses the time required to install VNF Marketplace. Section 4.3.2 presents performance tests to VNF Marketplace. Next, section 4.3.3 also analyses performance tests but for OpenStack. Finally, section 4.3.4 present results of the performed interviews.

4.3.1 Time to Install VNF Marketplace

A global metric to consider when evaluating web servers is how quickly operators can install that server on a host machine. This is very important since the hardware specifications of physical/virtual machines are always upgrading and, sometimes, it is needed to migrate all the systems in previous/out of date machines to new ones. The most important aspect of this, is that if it takes too long to install, then the operators will lose interest in the use of this system.

The tests to get installation time were performed in the same machine mentioned in section 3.2.1 with VMs similar to Ubuntu VM. It was not measured the installation time of Ubuntu because it is assumed that the operator has already an operational machine to place the VNF Marketplace server. Also, Windows OS was not used for this tests because of the hardware requirements to deploy it. Definitely, all this installation procedure is dependent of operator's experience and exist the possibility of an operator works better with Windows OS than with Linux. The download of the package of VNF Marketplace server was also not evaluated because it is dependent of operator's Internet velocity. For reference, the package (.zip) size of VNF Marketplace server has 56,8 MB.

Ideally, the tests for this section could be performed by participants to get different results accordingly to their experience with Linux and Python but that was not executed. VNF Marketplace does not have scripts to make a Python environment but, in the package, there is one text file with all the commands to execute at terminal. Appendix D mention the file with all the commands to install VNF Marketplace. The VNF Marketplace database has no data but roles and permissions are already configured. Table 4.1 illustrates, for each VM, the average installation phase time. For more details, Appendix D contains the time of each command. To get the time, whenever a command was executed, it was added the *time* Linux command as well.

Installation Phase	VM1	VM2	VM3	VM4	VM5	Average
Install Python	13.540s	10.982s	15.046s	11.831s	12.324s	12.745s
Create test environment	8,668s	7.587s	6.005s	7.457s	7.732s	7.490s
Install VNF Marketplace Requirements	63.110s	53.233s	55.880s	62.339s	57.901s	58.491s
Run Server	19.227	20.780s	13.214	15.683s	13.783	16.537s
Total	104.545s	92.582s	90.145s	97.310s	91.740s	95.263s

Table 4.1: VNF Marketplace installation time measurements

Analysing the results, the VNF Marketplace server needs, in average, 1 minute and 35 seconds to be installed (95,263 seconds). This case is only applied to servers that do not have Python installed. If a server has already Python and does not want to use a test environment, this average time suffers an important change. Also, there is the possibility that the operator's server may have installed some VNF Marketplace requirements. The time to run the VNF Marketplace server is with 16.537 seconds because of the creation of an Administrator. Finally, some of this installation phases are dependent on the Internet and therefore the measurements can suffer an inconsistent change.

Since VNF Marketplace needs SLA Management to create SLAs for customers, it is necessary to install it as well. Appendix D contains a file with the commands to install SLA Management Requirements. To install SLA Management, it was followed the tutorial from the Github⁴⁰. Table 4.2 shows, for each VM, the average installation phase time. It is assumed the SLA Management server is already at the VMs.

Installation Phase	VM1	VM2	VM3	VM4	VM5	Average
Install Requirements	157.772s	151.894s	191.946s	156.692s	179.939s	167.649s
Create MySQL Database	26.891s	30.012s	25.253s	25.724s	23.765s	26.329s
Compiling	180.932s	201.810s	166.634s	220.808s	171.162s	188.269s
Run Server	23.281s	26.089s	20.333s	39.463s	22.112s	26.256s
Total	388.876s	409.805s	404.166s	442.687s	396.978s	408.503s

Table 4.2: SLA Management installation time measurements

Comparing with VNF Marketplace server, SLA Management needs a bit more time to install. The reason behind it is because the SLA Management server is a Java server and it needs Java libraries. The Java JDK takes more time to download and to install. Analysing the results, using Python is faster and easier to manage. If the total average measurements of both servers are summed up, the time required to perform a full installation of the VNF Marketplace is 8 minutes and 24 seconds (503,766 seconds). This means that 81% of the total installation time is from the setup of SLA Management.

In conclusion, if an operator wants to install VNF Marketplace from scratch and without scripts, the installation time could go up to approximately 15 minutes. This include download, some configurations and installation. It is assumed that the operator knows how to use Linux commands and is familiar with Python. It is also possible for the operator to download the SLA Management already configured but it is necessary to install Java JDK, create a MySQL Database and run the server. This can reduce some time but not significantly.

4.3.2 VNF Marketplace Performance

Another critical metric used to analyse web servers is the performance. After the installation, some tests were performed in order to analyse the VNF Marketplace performance. To execute these tests, VNF Marketplace was in the same machine mentioned in section 3.2.1. Ubuntu VM had the same specifications and it was used the topology illustrated in figure 3.18. OpenStack was installed in a VM with 8 GB RAM, 2 CPUs and 100 GB disk space (section 3.2.2). Both machines had internet connection and OpenStack was already configured to receive VNFs.

To collect time measurements between requests and responses, it was developed a mediator (Middleware) inside the VNF Marketplace Back-End. During some tests were collected CPU utilisation traces (Appendix D). A decision to stop testing VNF Marketplace was taken when the result did not seem to change much between the rounds. Results related to customer's requests, registration, log in, statistics, list of available/owned VNFs, SLA and billing details are included in Appendix D and not deepened in this section because they are fast to be executed and do not cause any critical changes in performance.

Firstable, it was tested the load time of all the statics files. This is important because if the server does not react quickly at the beginning, users will lose interest in the web platform. Figure 4.45 illustrates the loading time to get all static files. The median amount of time required is 0.626 seconds. The free RAM does not suffer any significant changes but CPU utilisation (in system time) exceeds 10%.

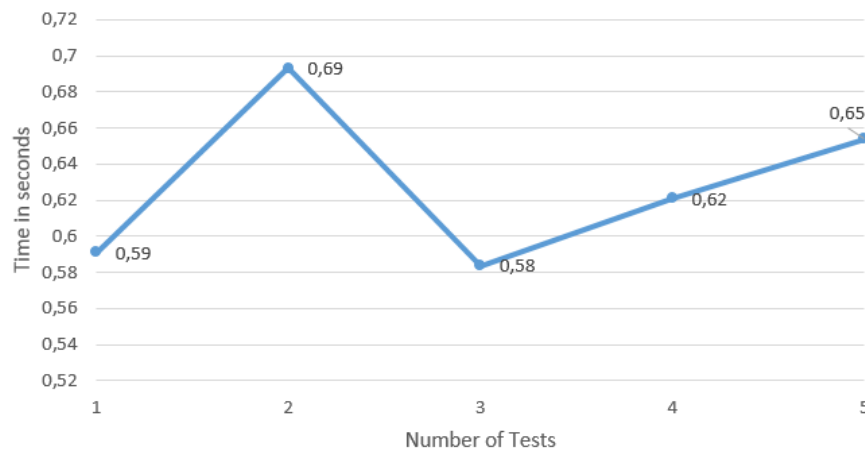


Figure 4.45: Loading time results of static files

After loading the Dashboard, two tests were performed: VNF publication in the Basic mode and the other in the Premium mode. To provide different times for each Purchase mode, it was used two VNF images: Ubuntu image (Ubuntu 16.04 Server) and Cirros image. The Cirros image only has 12,6 MB of size and Ubuntu has 829 MB. The following figures (4.46 and 4.47) illustrate the VNF publication results with Cirros and Ubuntu image for each purchase mode. For more details, Appendix D illustrates the time results and the CPU traces of both evaluations.

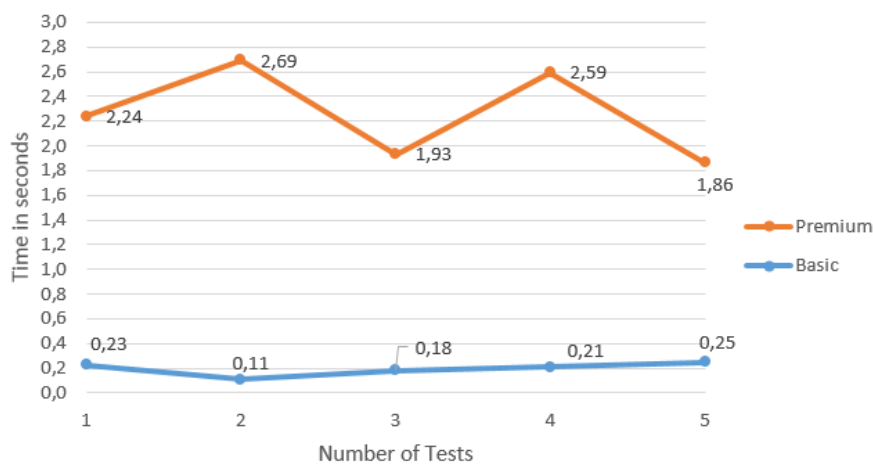


Figure 4.46: Cirros VNF Publication time results

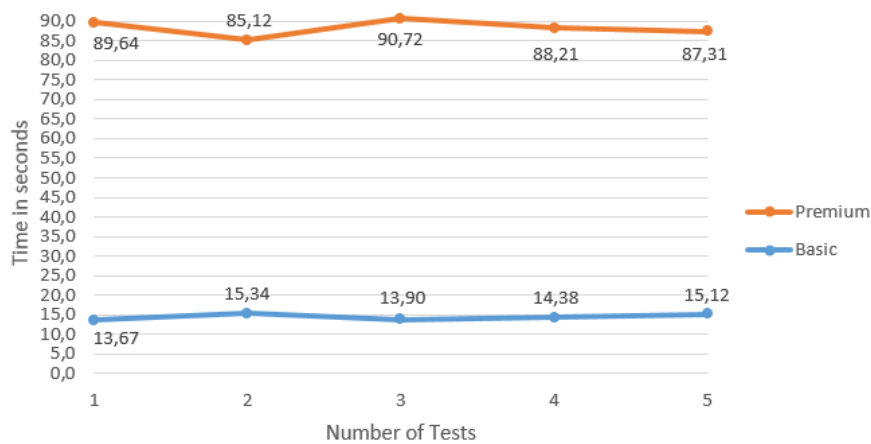


Figure 4.47: Ubuntu VNF Publication time result

There are different results in this evaluation. Publishing a VNF with Cirros in Premium mode takes approximately more 2 seconds than in Basic mode. With Cirros, advertising a VNF in Basic takes, in average, only 0.196 seconds which is very fast. During this time, it was used 48% of server CPUs. For the Premium case it only takes 2.264 seconds (with 73% CPU utilisation) which is slower because OpenStack needs to read the Cirros VNF image and VNFD as well. OpenStack has only used 28% of its CPUs during the VNF Publication.

Using Ubuntu as VNF, the amount of time to publish a VNF is quite relevant. In Basic mode, the VNF Marketplace took 14,484 seconds (with 61% of CPU utilisation) to publish an Ubuntu VNF. The reason why this happen is the necessary upload time of a long file such as Ubuntu image. The Premium mode had not such good results. This is not only because VNF Marketplace needs to read and store a long image, as has to send it and expect OpenStack to answer with a success response. The average amount of time was 88,2 seconds (with 70% of CPU utilisation) which is approximately 1 minute and 28 seconds. OpenStack used 12% of its CPUs but 62% were idle during which the system had an outstanding disk I/O request (from VNF Marketplace). This means why VNF Marketplace took an extra time to publish a VNF.

Since sellers can delete their VNFs, it were collected measurements related to VNF withdraw. The tests involved VNFs published in Basic mode and in Premium mode. For different results, it was used VNFs with Cirros and Ubuntu images. The following figures (4.48 and 4.49) illustrate the VNF withdraw results for both VNF images.

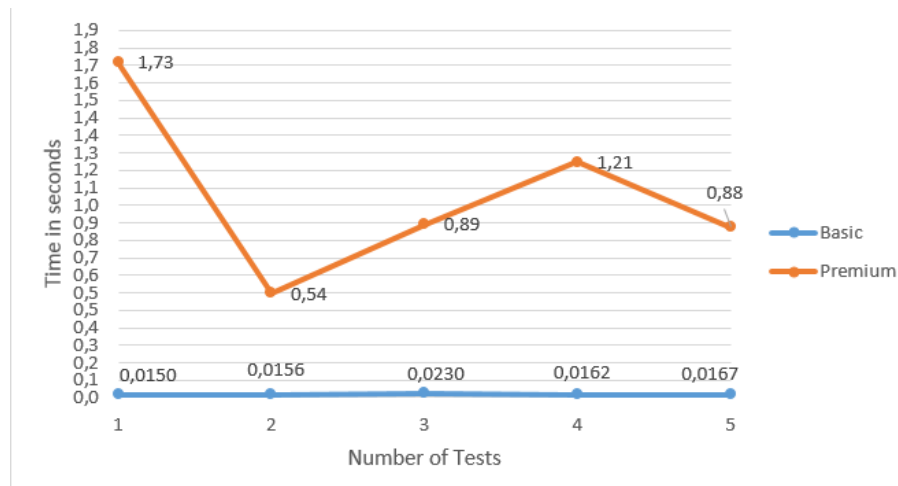


Figure 4.48: Cirros VNF Withdraw time results - Seller

The Basic mode is very fast performing all the tasks. Deleting a Cirros VNF only took 0.0017 seconds (with 27% of CPU utilisation). In Premium mode, the results, as expected, were a quite slower. The average time to delete a Cirros VNF from VNF Marketplace and OpenStack (VNFD and VNF image as well) is 1.05 seconds. Still quite fast and only used 28% of its CPUs. OpenStack used, during the elimination of Cirros VNF, 25% of its CPU.

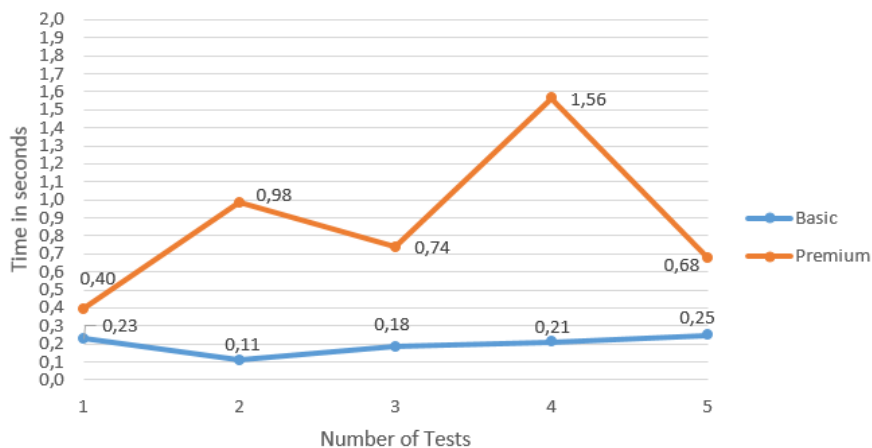


Figure 4.49: Ubuntu VNF Withdraw time result - Seller

Comparing with Cirros image, the only notorious change with Ubuntu VNF is in the Basic mode. To delete a VNF in Basic mode, the request and response lasted 0.196 seconds (with 17% CPU utilisation). The VNF withdraw in Premium mode with Ubuntu VNF is similar to Cirros VNF. The reason why this happens is because Openstack sends the response even though it has not finished deleting the VNF. Openstack had 11% of CPU utilisation during this evaluation.

After tests with seller role, it is important to evaluate the customer actions. Two tests were performed with the customer role: VNF purchase and VNF withdraw. The VNF purchase and withdraw evaluations were performed with the published VNFs of the tests above. Figure 4.50 and figure 4.51 illustrate results of VNF purchase (in Premium mode) with Cirros and Ubuntu VNF respectively. The last two figures (figure 4.52 and 4.53) show the time results but for VNF withdraw. The Basic mode is not present in this section since it had similar time results to the Premium mode. After getting the request to deploy the desired VNF, OpenStack does not wait for the conclusion of the VNF deployment. Then, it sends the response (response is quite fast) immediately to VNF Marketplace. The same does not happen with VNF withdraw because VNF Marketplace needs to delete the respective SLA agreement. Lastly, Appendix D illustrates the time results of VNF purchase with Cirros and Ubuntu and the results of CPU traces.

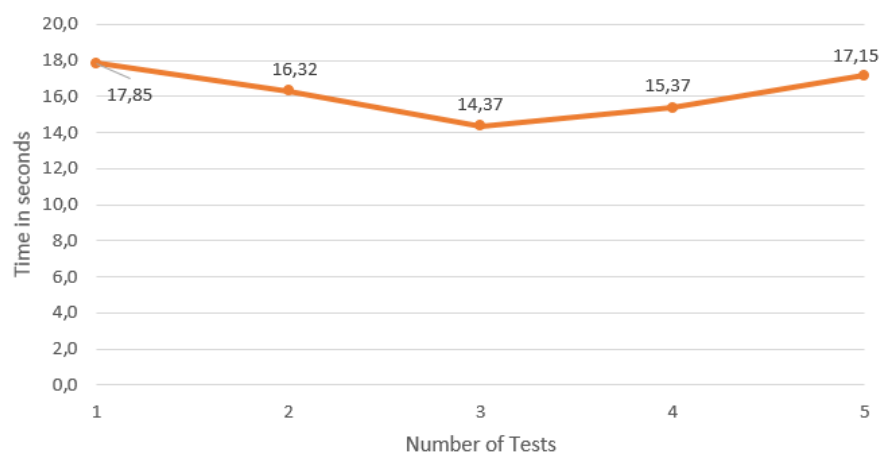


Figure 4.50: Cirros VNF Purchase time result

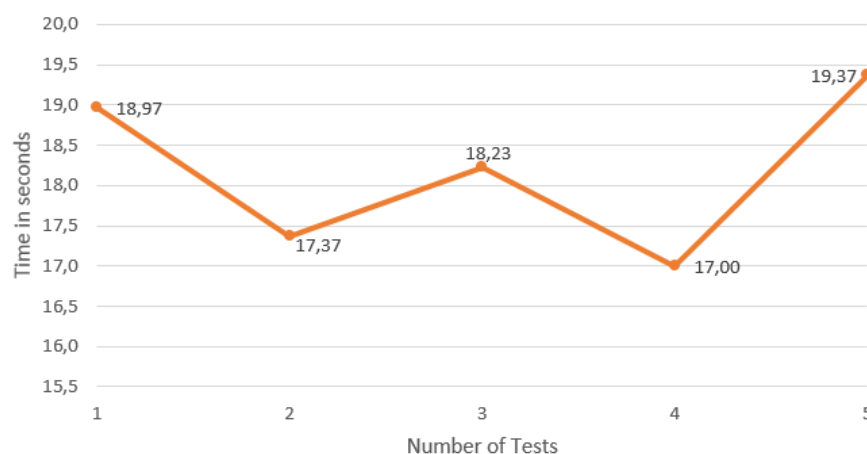


Figure 4.51: Ubuntu VNF Purchase time result

The time results for this evaluation are relative. With Ubuntu VNF, the results were slower but it was not because of the VNF image itself. All the automatic process during the VNF purchase are dependent from payments and payouts. Also the PDFs files are create with invoices and the emails are sent with them to each actor. Finally, it is created SLA templates and agreements before request OpenStack to deploy the VNF. Then, considering the fact OpenStack responds quickly to the request, the average time to purchase a Cirros VNF is 16,21 seconds and a Ubuntu VNF is 18,18 seconds. The CPU utilisation of VNF Marketplace for each VNF is the same, approximately (73%). The difference that should be referred is the CPU utilisation of OpenStack during the VNF deployment. For Ubuntu VNF, Openstack used 32% of its CPUs and, for Cirros VNF, used 20%.

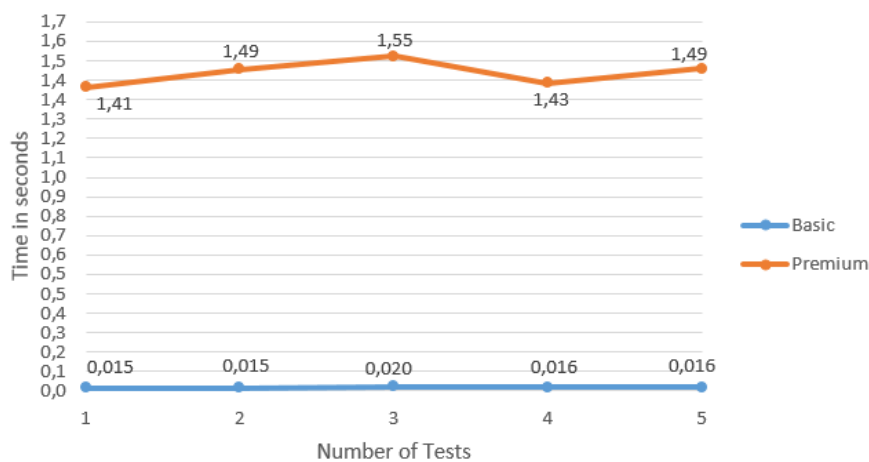


Figure 4.52: Cirros VNF Withdraw time result - Customer

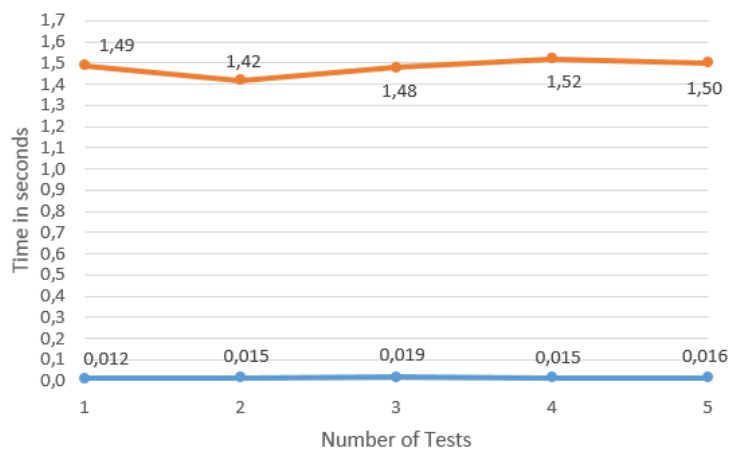


Figure 4.53: Ubuntu VNF Withdraw time result

As expected, deleting a VNF (customer role) in Basic mode is faster and, in Premium mode, it is not relevant if the VNFs have more or less images size. Even with CPU utilisation at the VNF Marketplace server, the VNF image size is not significant. Again, the interesting result in this evaluation was the CPU utilisation of OpenStack. During Ubuntu VNF withdraw, OpenStack needed 66% of its CPUs. With Cirros VNF, 39% of OpenStack's CPUs were used.

In conclusion of this section, using Basic mode has immense advantages. The time results were very fast to be performed and, with Ubuntu VNF, the wait time is very decent comparing with Premium mode. A way to avoid the waiting time in Premium mode is implementing a certification system to validate the VNF before sending it to OpenStack. With this system, the seller advertise the VNF, waits the average VNF publication time of Basic mode and the VNF is uploaded to the certification system. The CPU results were due to lack of CPU resources at the Ubuntu VM. The VM only had 2 CPUs and it was running two servers at the same time (VNF Marketplace and SLA Management).

4.3.3 OpenStack Performance

Since VNF Marketplace uses OpenStack to deploy VNFs, it was necessary evaluate OpenStack performance. This validates the efficiency of the Premium purchase mode. Figure 3.18 illustrates the topology used for this tests. Also, OpenStack had the same configuration shown in figure 3.20. The OpenStack VM configured had 8 GB RAM, 2 CPUs and 100 GB of disk. All the Devstack installation procedures were performed but Ceilometer and Gnocchi services (OpenStack Telemetry services) had bugs so it was impossible to collect VNF metrics.

For this section, it was performed two tests. The first test was designed to evaluate the time of storing VNFDs and VNF images. For this evaluation was only performed with Ubuntu image. The reason why is because, accordingly to figure 4.48, it only takes 1.05 seconds to publish a VNF into the VNF Marketplace. So it is assumed this test with Cirros VNFD and image only takes 1 second. The last evaluation executed was related to how many VNFs could OpenStack deploy. This is an important measurement since shows the capacity of providing VNFs in Premium mode. Each evaluation was performed 5 times. The CPU traces of OpenStack were already evaluated in section 4.3.2.

Taking the VNF Publication results from section 4.3.2 as reference, the average time to publish a VNF with Ubuntu image is 88,2 seconds. This result is influenced by OpenStack. To see how many time OpenStack took to store VNFD and VNF image, it was used the same mediator from VNF Marketplace server. Figure 4.54 illustrates the times of each evaluation of storing VNFD with VNF image as well.

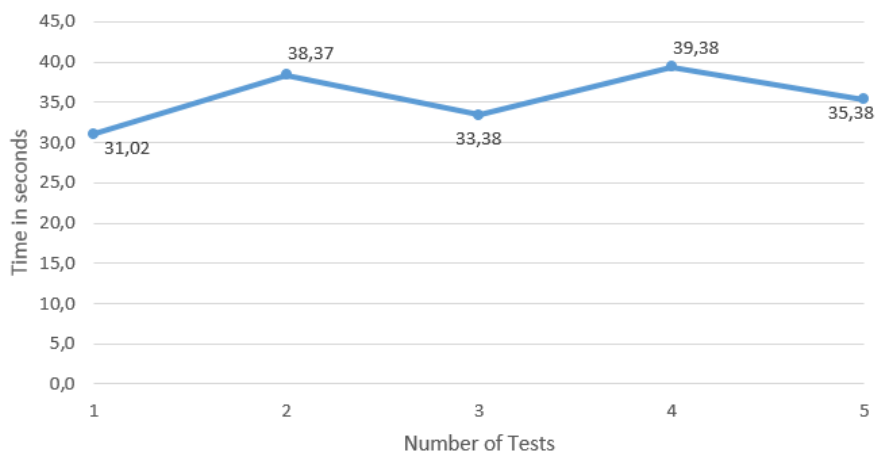


Figure 4.54: Ubuntu VNF Withdraw time result

Comparing the average time from VNF publication and the results of this evaluation, it is possible to conclude that 40% of VNF publication time is from OpenStack. This means that sending the VNF image/VNFD and waiting for Tacker and Glance storing both files, greatly affects the performance of VNF Marketplace. Since VNF Marketplace has to read the VNF image, store it in the root directory, provide it inside the request and wait for the OpenStack response, all this processes will consume relevant time to compute. A different approach for this communication is sending the VNF image asynchronously to OpenStack.

Finally, the last evaluation was performed separated from communications between OpenStack and VNF Marketplace. The results collected were to measure how many VNFs could be deployed in OpenStack. The measurements are showed in listing [D.13](#) for more detail. Figure 4.55 shows the results of this evaluation.

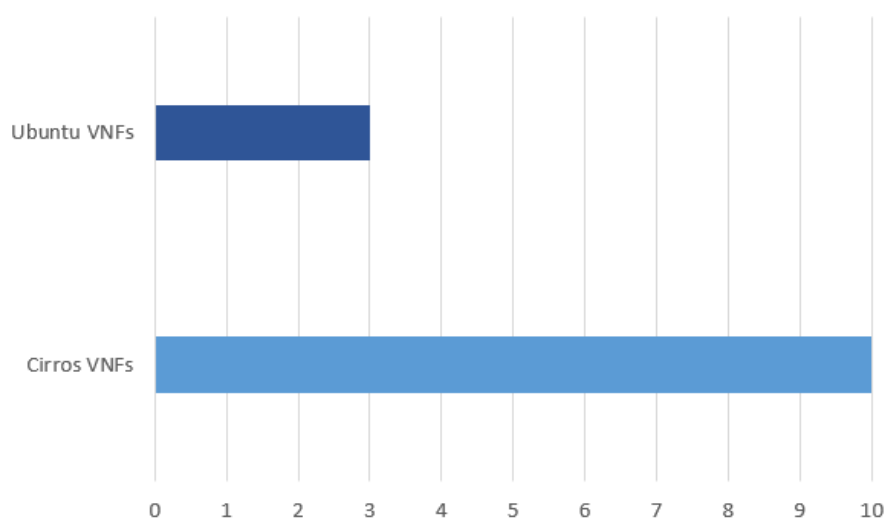


Figure 4.55: Number of VNFs

The reason why OpenStack only deployed 3 Ubuntu VNFs was because of the lack of cloud infrastructure resources. Also, during the deployment of the third VNF, the CPU reached values of 95% which is very high comparing with the CPU trace of 10 deployed Cirros VNFs. The RAM was not too much affected but, due to not enough resources from OpenStack, it was not possible to deploy more than 3. Another reason for this happening could be a bug from Tacker or other service from OpenStack but that was not validated. Finally, OpenStack could deploy more Cirros VNFs since CPU utilisation was using 36% of its total CPUs and the Openstack VM had 152 MB of free RAM. The reason why it only deployed this number was because OpenStack was configured to deploy only 10 instances per tenant.

4.3.4 Interviews

In order to collect thoughts and opinions on the VNF Marketplace system, an interview with five participants was performed. These participants are employees of Deloitte Consultores, and they were aware that their opinions were collected to evaluate VNF Marketplace. The reason why they were selected was because of their knowledge of the NFV concept. The topics evaluated were the following:

- Simplicity in purchasing VNFs as Customer;
- Simplicity in selling VNFs as Seller;
- Availability of information for the Customer and Seller;
- Diversification of payment methods (Customer);
- Intuitive registration;
- Statistics evaluation for both actors;
- Graphical interface evaluation for both actors;
- Evaluation of communication between Customer and Seller;
- Overall performance.

The topology mentioned in section 3.18 was configured to this interviews. Participants connected their machines to the switch to have access to the VNF Marketplace Dashboard. The VNFs used for this tests were Cirros and OpenWRT (vRouter). OpenWRT is described as a Linux distribution for embedded devices to route network traffic. The OpenWRT VNF was provided to perform as a virtual Router with firewall already configured. Listing D.7 illustrates the OpenWRT VNFD. OpenWRT VNF image has only 52,5 MB of disk. It was asked to each participant register with a valid email in order to receive the invoices. Appendix C illustrates the console of OpenWRT VNF validating the automatic configuration of the firewall.

To evaluate the interviews was created a template form with the topics indicated above. Each topic was ranked in a scale from 1 to 10 to facilitate the answers and results. The interviews were performed individually with the exception of three participants (multi-user environment). Finally, after each participant bought a VNF, was showed the VNF deployment in OpenStack to approve the purchase. The next figures illustrate the results for each evaluated topic.

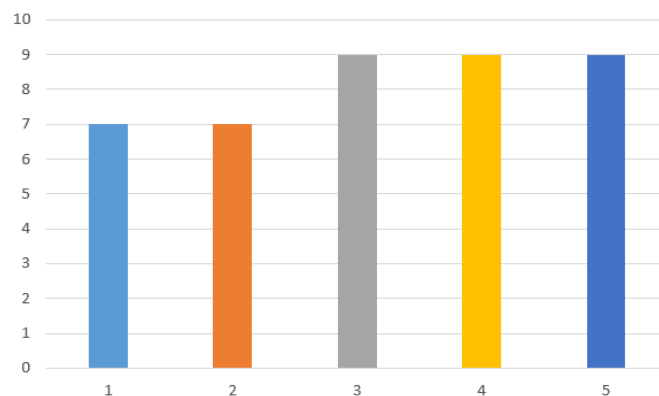


Figure 4.56: Simplicity in purchasing VNFs as Customer

Participants classified the VNF purchase with very high values. From their opinions, purchasing a VNF is very simple and intuitive. The only two negative thoughts mentioned were: the SLA pre-defined flavors and the location of the purchase mode information. The information about the purchase mode was only found by three participants. For one participant, the pre-defined SLA parameters could have more flavors with more SLOs as well. Also was mentioned that the customer could select desired SLA parameters instead of selecting a group of pre-defined SLAs (flavors).

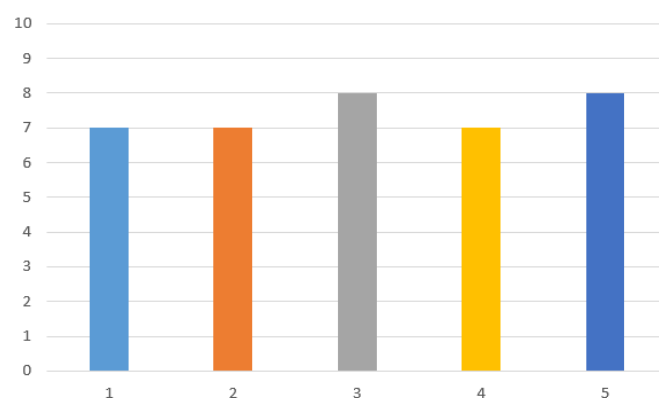


Figure 4.57: Simplicity in selling VNFs as Seller

For the second topic, every participant share the same thoughts about the VNF type and the purchase mode. The VNF type field was vague and, as suggestion, it was better use a list of pre-defined VNF types to avoid same VNF types but inserted with different names (e.g. vRouter , VRouter). The purchase mode had two thoughts. One participant wanted to selected both of purchase modes but the system only allowed to select one. For this participant, the VNF publication could have the ability of advertising one VNF with the two modes instead of publish two equal VNFs with different purchase modes. Lastly, the results were positive with an average of 7 in a scale of 1 to 10.

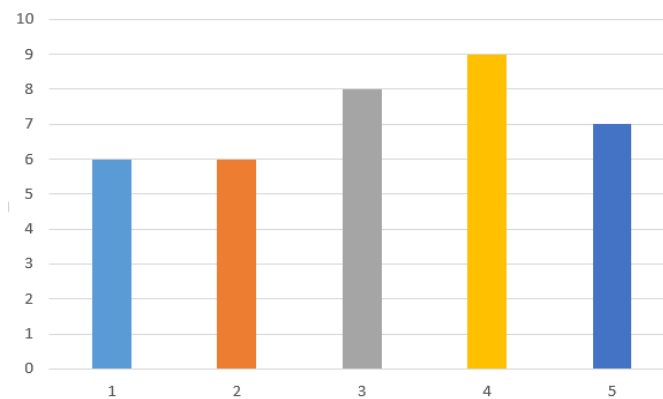


Figure 4.58: Availability of information for the Customer and Seller

As mentioned in the others topics, it is not illustrated at the Dashboard the best approach to provide information about the purchase mode. Also, it could have indications about VNFDs in VNF publication phase. Since VNFDs use TOSCA Tacker template, that information could be provided. In overall, participants had no difficult navigating the VNF Marketplace dashboard.

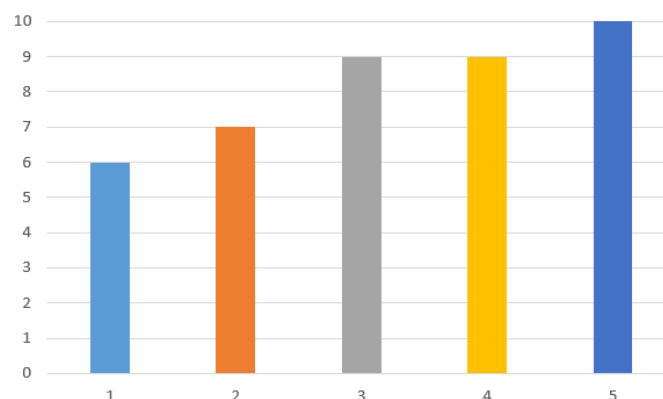


Figure 4.59: Diversification of payment methods (Customer)

The payments options were well rated by the participants. Only one participant shared that could have more options to purchase VNFs. Furthermore, another participant mention using credit card without the server having TLS is not safe for the customers. In conclusion, all participants referenced that Paypal was a good choice to use in the Marketplace because is a notorious mechanism to purchase online products and provides more security to customers and sellers as well (payouts).

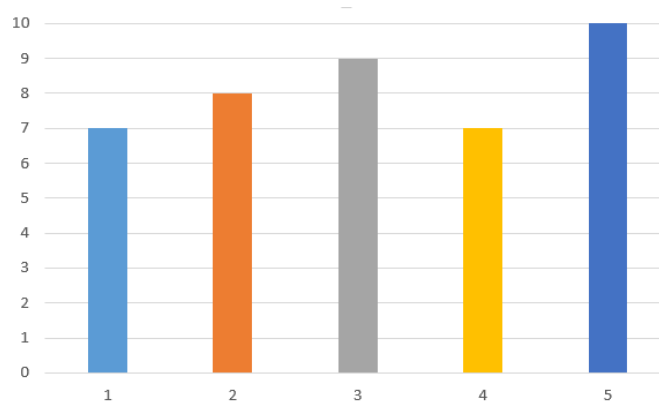


Figure 4.60: Intuitive registration

In the registration phase, participants had no difficulty performing it. They shared a very simple and fast registration. The only negative thought of two participants was the conclusion of the registration. They suggested, after the submission, the user automatically log in on the Dashboard instead of enter the credentials to log in. Finally, one participant indicated the implementation of an email verification for more security in the VNF Marketplace.

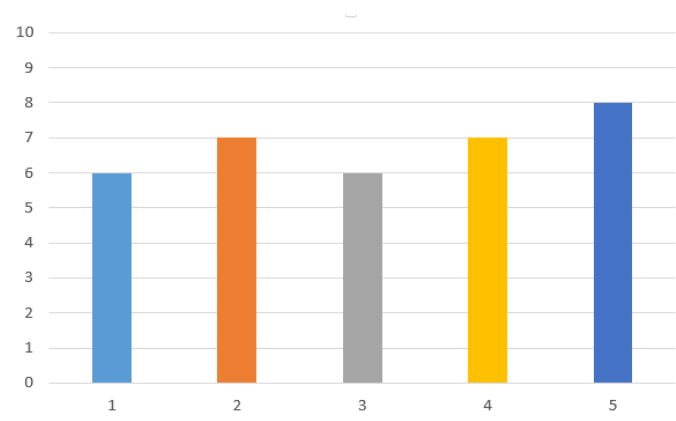


Figure 4.61: Statistics evaluation for both actors

Different thoughts were answered here. Since OpenStack does not generate VNF metrics, SLA Management did not evaluate and fulfil SLA agreements. As consequence, statistics did not have any graphical representation to detail the fulfilment of SLA agreements. This was one negative thought shared by all the participants. Two participants also mentioned that could have more graphs of VNF monitorisation. After the interview was indicated that both VNF metrics and VNF monitorisation were not being correctly generated and, because of that, it was impossible to provide statistics of them. Lastly, participants positively evaluated the available graphs such as the revenue.

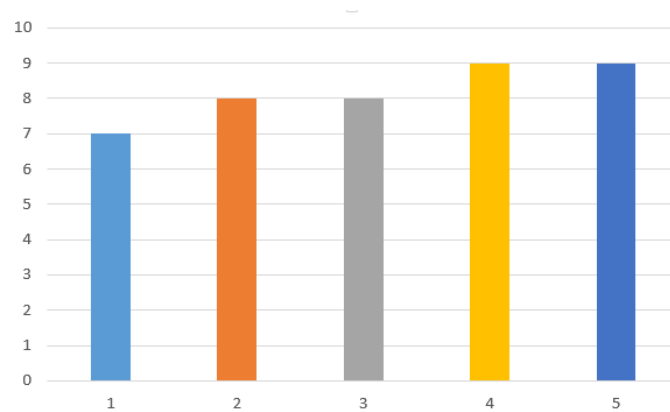


Figure 4.62: Graphical interface evaluation for both actors

All participants rated the graphical interface as the best component of VNF Marketplace. A dashboard very appealing, simple to use and, in case of an action resulted in some error, was always indicated to the participants the reason of such happening.

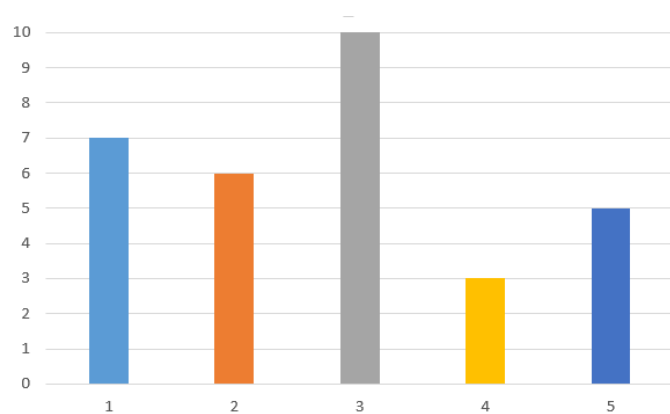


Figure 4.63: Evaluation of communication between Customer and Seller

This eight topic was quite discussed with all participants. Only one participant did not shared negative thoughts of it. The Requests functionality helped them improving the communication between each actor but the Dashboard could had more information about sellers. As customer, the seller's information that is provided is the email. Three participants mentioned using messages at the web platform give better communication among seller and customer. Finally, one participant expressed including the seller's company website for more detailed information.

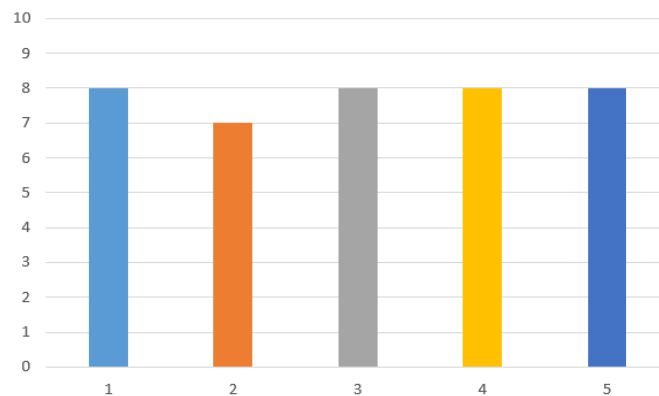


Figure 4.64: Overall performance

The last topic was well rated as well. All participants mentioned the VNF Marketplace was fast and exceeded the expectations. They recognise that this master's thesis project is not a commercial product so most of the negative thoughts can be easily avoided in near future. Also, since the server had not the best hardware components and network (all of them mentioned the server could be public), participants acknowledged that the server could have better performance if it was installed in a superior hardware environment.

In conclusion, participants mentioned that this can start a change in the telco world and, in case this platform becomes commercial, it is necessary evaluate potential VNF providers and clients as well as improve the VNF Marketplace business model.

Chapter 5

Conclusions and Future Work

5.1 Satisfaction of Objectives

In this master's thesis project, a Marketplace capable of advertising and providing VNFs has been designed and implemented. VNF Marketplace is able of, not only supply VNFs to potential clients download, but also of deploying and providing them without the need of installation and configuration on customer's premises. This is essential since gives two different options to customers choose. Comparing VNF Marketplace directly with T-NOVA, it was developed modules that improve the Marketplace concept that T-NOVA was not capable of. For instance, with T-NOVA, the way the customers buy NSs are vague and they only can choose from pre-defined services. VNF Marketplace delivers a charging system from the Billing module that really performs the payments and payouts. Section 4.3.1 mention that to install VNF Marketplace only takes nearly 15 minutes and with a simple installation. Comparing with T-NOVA, many problems were encountered installing the T-NOVA Marketplace server. It was not possible to compare the performance between them but VNF Marketplace delivers a simple and fast mode to customers download VNFs. The Premium mode delivers a better way to provide VNFs to customers but it is necessary to be integrated into a machine/VM with better hardware components.

The objectives defined for this dissertation were all fulfilled. VNF Marketplace can advertise VNFs, send them to OpenStack, purchase available VNFs and request OpenStack to deploy the purchased VNFs. VNF Marketplace allows customers to request VNFs, creates SLA agreements and provides detailed information about billing and VNFs. From section 4.3.4, we can observe that VNF Marketplace has potential to be integrated into the market but it is needed improvements. The VNF Marketplace Dashboard demonstrates to be intuitive and appealing. Finally, VNF Marketplace has an important connection with OpenStack that proves to be a key component in this master's thesis project.

5.2 Limitations

Since the Devstack installation downloads software from a frequently updated repository, it is possible that occasionally occur bugs during or after installation. Installing release versions of OpenStack (with Devstack) is not an easy task either since Tacker and Heat sometimes had incompatibilities with them. During the development phase of this thesis, installing Devstack successfully proved to be a question of luck or due to currently available web resources. Unfortunately, this affected significantly the Premium mode development.

Ceilometer and Gnocchi were never successfully installed after a large number of attempts and, due to lack of time, it was not possible to understand the reasons behind it. Without this OpenStack services, it was not possible to create VNF metrics and collect correct VNF monitorisation values. This affected SLA fulfilment and VNF statistics to customers. Another problem was finding a sufficiently large group of participants to observe and evaluate this master's thesis project. A larger sample size would give greater statistical power and more suggestions to improve the VNF Marketplace.

5.3 Future Work

There are a number of possible extensions and improvements for this master's thesis project. Since Tacker has a NFVO, network services can be operated and deployed. This means that VNF Marketplace can provide to customers another type of product using Tacker NFVO. The VNF vendors could still sell their VNFs and network operators buy them to provide as a service (network service). If VNF Marketplace adopts this new functionality, more actors could integrate this project and the NFV demands could be better fulfilled.

Others improvements can be related to the web platform itself. Providing the necessary information and statistics are keys components to achieve better results of the established business model. Security and validation are crucial elements to give comfort to each Stakeholder. The developed project did not focus on this systems but they are essential for a web platform. Section 4.3.4 mentioned some suggestions to be implemented in the VNF Marketplace as well. SLAs can be more configurable by the customers, the communication between actors can be improved with more information about each of them and the web platform could provide more information for each functionality presented in the system.

Lastly, the customers having the ability to configure VNFs is an important factor to consider. VNF Marketplace only provides the "stop/start" functionality that proves only one essential type of configuration. Scaling up/down VNFs are indispensable to increase/decrease the capacity of the VNFs and that demonstrates to be an implementation necessary to this Marketplace.

Appendix A

Use Cases

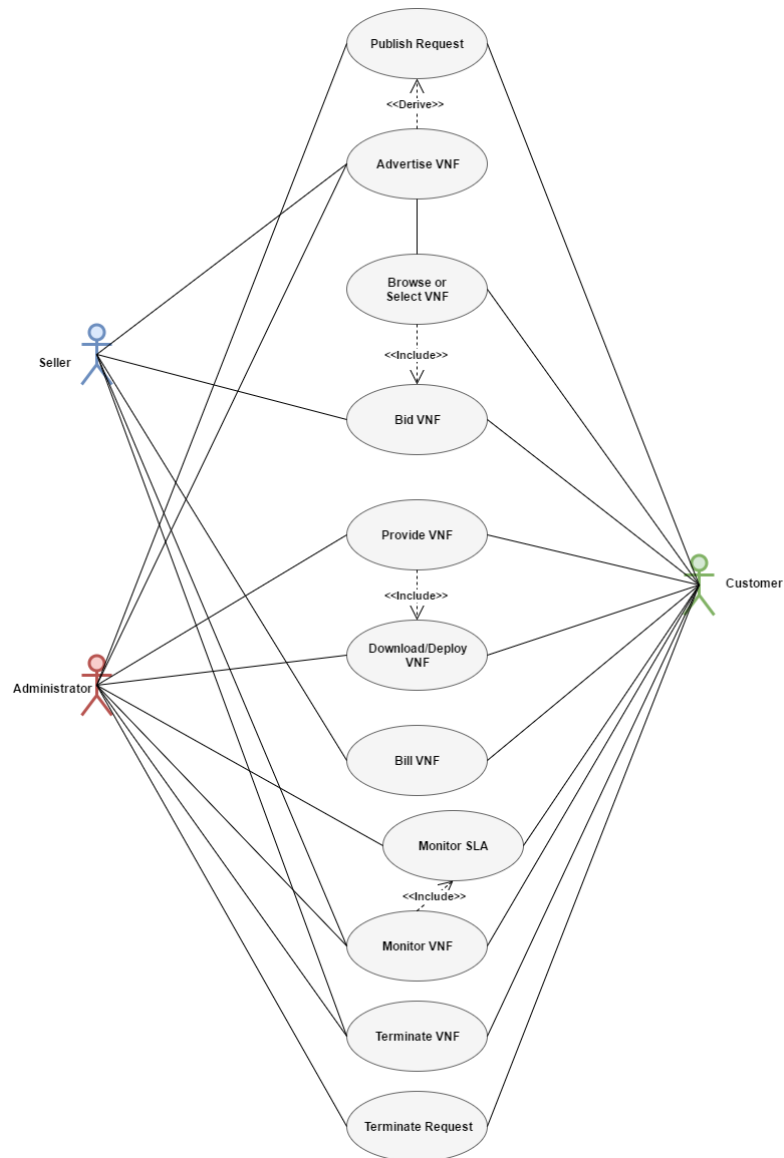


Figure A.1: VNF Marketplace Use Cases

A.1 Use Cases Details

Use Case ID	UC1
Title	Advertise VNF
Description	This use case defines how a Seller/Administrator publish and advertise a VNF
Stakeholders	Administrator and Seller
Preconditions	Seller/Administrator is logged into VNF Marketplace. Request was made and Seller/Administrator can reply to it
Postconditions	VNF is registered into VNF Catalog from VNF Marketplace
Basic Flow	<ul style="list-style-type: none"> • Seller/Administrator is authenticated to VNF Marketplace • Seller/Administrator fill VNF information, submit VNF image and VNFD/license file (depending on Purchase mode) and select billing type with price, period and currency. Seller/Administrator can do this procedure after select "Reply" on customer's request • The uploaded VNFD is certified and stored by Tacker if Seller/Administrator selects Premium Purchase mode • The same VNF is included to VNF Catalog from VNF Marketplace
Priority	High

Table A.1: Use Case 1

Use Case ID	UC2
Title	Browse or Select VNF
Description	This use case determines how the Costumer selects VNFs from the catalog and how the SLA agreement is established
Stakeholders	Customer
Preconditions	Customer is logged into VNF Marketplace. At least one VNF is present at VNF Catalog. Request was replied by the Seller
Postconditions	VNF Purchase
Basic Flow	<ul style="list-style-type: none"> • Customer is authenticated to VNF Marketplace • Customer browse through the different available VNFs with specific requirements and SLA parameters (in Premium mode and SLA parameters are pre-defined by VNF Marketplace). In case customer does not find the desired VNF, it is proceed to UC4 • Customer selects the desired VNF • Customer accepts SLA (Premium mode), pricing, other applicable conditions and selects payment option • The SLAs agreed (Premium mode) are registered in the system (SLA Control and SLA Management) • VNF Marketplace system proceeds to UC3
Priority	High

Table A.2: Use Case 2

Use Case ID	UC3
Title	Bid VNF
Description	This use case describes the procedure of billing payment and payout
Stakeholders	Customer and Seller
Preconditions	Customer already selected a VNF (UC2)
Postconditions	Customer Payment and Seller Payout
Basic Flow	<ul style="list-style-type: none"> • If payment option was Paypal, Paypal's Login is required to proceed with the payment. If not, system charges a debit payment from Customer's credit card • Payout is done asynchronously to Seller (owner of the VNF) after payment • Bid information is presented to the Customer and Seller (Emails are sent. In case of Basic mode, email contains VNF license) • VNF Marketplace system proceeds to UC5
Priority	High

Table A.3: Use Case 3

Use Case ID	UC4
Title	Publish Request
Description	This use case establishes how a Customer publish a request
Stakeholders	Customer and Administrator
Preconditions	Customer is logged into VNF Marketplace
Postconditions	Customer's is uploaded to Request module
Basic Flow	<ul style="list-style-type: none"> • Customer is authenticated to VNF Marketplace • Customer fill VNF information fields (name, type, description and Purchase mode) • The same request is included to Requests Catalog from VNF Marketplace • If request was replied by a Seller (Request's status is "Closed"), Customer can proceed to UC2.
Priority	Medium

Table A.4: Use Case 4

Use Case ID	UC5
Title	Provide VNF
Description	This use case describes the infrastructure dynamic configuration for VNF deployment. Administrator can manually configure the infrastructure and VNF
Stakeholders	Customer and Administrator
Preconditions	UC2 and UC3 already concluded
Postconditions	VNF deployed and ready for the customer
Basic Flow	<ul style="list-style-type: none"> • VNF Marketplace differentiate the Purchase mode. If is in Basic mode, the customer is notified to download the VNF image (UC6). • If is in Premium mode, VNF Marketplace checks if OpenStack is ready to communicate • VNF Marketplace sends the request order to deploy the specific VNF • Tacker maps and deploy the VNF (UC6) • Administrator can configure the VNF in case of necessity • Customer can stop/start the VNF
Priority	High

Table A.5: Use Case 5

Use Case ID	UC6
Title	Download/Deploy VNF
Description	This use case establishes the procedure of deploying a VNF at Tacker (Premium mode) or downloading the VNF (Basic mode)
Stakeholders	Customer and Administrator
Preconditions	UC2 and UC3 already concluded. VNF Marketplace knows what Purchase mode is associated with the pretended VNF (UC5)
Postconditions	VNF deployed
Basic Flow	<ul style="list-style-type: none"> • In case of Basic mode, Customer download the VNF image. • In case of Premium mode, Tacker, Heat and Nova maps into specific infrastructure (network and compute) resources, taking into account several applicable objectives (from VNFD and SLA) • Neutron establishes the virtual network service (from VNFD) • Nova gets the VNF image to be deployed • VNF is instantiated and connected to the virtual network
Priority	High

Table A.6: Use Case 6

Use Case ID	UC7
Title	Bill VNF
Description	This use case determines the billing procedure for a Customer (Premium mode only) based on SLAs and billing parameters from Seller
Stakeholders	Customer and Seller
Preconditions	UC2 and UC5 already performed
Postconditions	Bill production for the customer
Basic Flow	<ul style="list-style-type: none"> • SLA and VNF information (utilisation, CPU, etc) is collected. This is performed accordingly to the selected period time from seller • The pretended bills are generated • Customer is charged by debit credit card payment. The Seller receives the respective payout • Bill information is presented to the Customer and Seller
Priority	High

Table A.7: Use Case 7

Use Case ID	UC8
Title	Monitor VNF
Description	This use case helps Stakeholders observe overall status from VNFs. VNFs are constantly monitored. UC8 facilitate billing to Customer and Seller, detect anomalies which Administrator can intervene and provide awareness to Customer about VNF status
Stakeholders	Customer, Seller and Administrator
Preconditions	UC5 (VNF active) already performed
Postconditions	VNF Monitorisation. Allows Customer, Seller and Administrator perform actions accordingly to the monitorisation results
Basic Flow	<ul style="list-style-type: none"> • VM statistics per VNF instance are collected • Anomalies are detected • SLA parameters are collected (UC9) • Network monitoring details are collected for the connectivity service. • VNF details are presented to the Customer and Seller (Seller can only see details from billing and VNF utilisation)
Priority	High

Table A.8: Use Case 8

Use Case ID	UC9
Title	Monitor SLA
Description	This use case describes the procedure to evaluate SLA agreement between Customer and VNF Marketplace
Stakeholders	Customer and Administrator
Preconditions	UC2 (SLA agreement concluded) and UC5 (VNF active) already performed
Postconditions	SLA Monitorisation. Allows Customer and Administrator perform actions accordingly to the monitorisation results
Basic Flow	<ul style="list-style-type: none"> • VNF monitoring metrics are collected • All the terms of the SLA agreement are compared with the metrics provided to fulfill the agreement • SLA information is presented to customer • Billable SLA results are registered for billing changes (discounts) • In case of anomaly with SLA Management, Administrator can repair the necessary faults
Priority	High

Table A.9: Use Case 9

Use Case ID	UC10
Title	Terminate VNF
Description	This use case establishes the procedure to terminate a VNF. VNF can be terminated by Customer desire or the removal from the VNF Catalog (if it is not active/downloadable to any Customer)
Stakeholders	Customer, Seller and Administrator
Preconditions	UC1 or UC5 (VNF active/inactive) already performed
Postconditions	VNF termination
Basic Flow	<p>In case of the Administrator/Customer desires to terminate an active/downloadable VNF:</p> <ul style="list-style-type: none"> • Customer/Administrator is authenticated • Customer/Administrator chooses the VNF and selects to terminate it • If VNF is in Basic Mode, VNF is only deleted from Customer's VNFs. If not, VNF Marketplace sends the termination request to Tacker • VNF instances are terminated by Tacker VNFM • All billing/charging/SLA activities related to the VNF are terminated • VNF statistics (billing and monitoring data) are saved for further use <p>In case of the Seller desires to terminate an VNF (not active to any Customer):</p> <ul style="list-style-type: none"> • Seller is authenticated • Seller chooses the VNF and selects to terminate it • VNF is removed from VNF Catalog • VNF statistics (billing and monitoring data) are saved for further use
Priority	High

Table A.10: Use Case 10

Use Case ID	UC11
Title	Terminate Request
Description	This use case describes the procedure to terminate a Customer's request
Stakeholders	Customer and Administrator
Preconditions	UC1 and UC4 already performed
Postconditions	Request termination
Basic Flow	<ul style="list-style-type: none">• Administrator/Customer is authenticated• Administrator/Customer chooses the Request and selects to terminate it• Request is removed from Customer's requests
Priority	Medium

Table A.11: Use Case 11

Appendix B

Requirements Specification

To specify software requirements of each VNF Marketplace component, a table (template) is presented. The tables follows IEEE 830 standard. Requirements area from NFV requirements (section 2.1.2.1) are used in corporation with VNF Marketplace. Market Operability is introduced as complement of NFV requirements.

B.1 Authentication and Authorisation Module

ID	Use Cases	Requirements Area	Requirement Description	Category
Req1	UC1, UC2, UC4, UC10, UC11	Security	Authentication and Authorisation Module shall support authentication and authorisation (based on roles and permissions) techniques to different Stakeholders.	Functional
Req2	UC1, UC2, UC4, UC10, UC11	Security	Authentication and Authorisation Module should provide encryption (passwords, TLS and files). Only password encryption is performed	Functional
Req3		Security	Authentication and Authorisation Module shall create profiles and assign roles to each Stakeholder	Functional

Req4		Security	Authentication and Authorisation Module shall allow Administrator reassign and grant new permissions/roles to specific user	Functional
Req5		Security	Authentication and Authorisation Module shall allow Administrator update permissions/roles and create new permissions/roles if necessary	Functional

Table B.1: Authentication and Authorisation Module Software Requirements

B.2 VNF Catalog Module

ID	Use Cases	Requirements Area	Requirement Description	Category
Req6	UC1	Service Continuity and Operations	VNF Catalog Module shall store all available VNFs with VNF information, VNF image, VNFD, license file and billing information	Functional
Req7	UC1	Service Continuity and Market Operability	VNF Catalog Module shall allow Sellers upload their VNFs	Functional
Req8	UC1, UC5	Service Continuity, Market Operability and Operations	VNF Catalog Module shall allow Sellers upload VNF accordingly to Customer's request	Functional
Req9	UC1	Security and Operations	VNF Catalog Module should validate VNF images	Functional
Req10	UC1	Security and Operations	VNF Catalog Module shall provide an unique ID to each VNF advertised	Functional

Req11	UC1	Security and Operations	VNF Catalog Module should guarantee security for transmission of data (e.g. VNF image)	Functional
Req12	UC2	Service Continuity and Market Operability	VNF Catalog shall be browsable by the Customer	Functional
Req13	UC2	Market Operability	VNF Catalog Module shall provide search engine to help Customers find the desired VNF more easily	Functional
Req14	UC2, UC8	Operations	VNF Catalog Module shall allow access to authenticated and authorised Stakeholders	Functional
Req15	UC6	Operations	VNF Catalog Module shall allow downloading of the respective VNF image	Functional
Req16	UC2	Performance and Operations	VNF Catalog shall provide multi-user access	Functional
Req17	UC5, UC10	Service Continuity, Operations, Management and Orchestration	VNF Catalog Module shall allow operations such as start/stop/delete VNFs. VNF Catalog Module should allow operations such as update VNFs	Functional
Req18	UC5	Management and Orchestration	VNF Catalog Module shall know if a VNF starts correctly	Functional
Req19	UC8	Service Continuity and Market Operability	VNF Catalog Module shall provide VNF details of Customer's or Seller's VNFs	Functional
Req20	UC1	Operations, Management and Orchestration	VNF Catalog Module shall inform Tacker to store a desired VNFD	Functional

Req21	UC1, UC5, UC8	Security and Operations	VNF Catalog Module shall guarantee security for transmission of data to Tacker and Glance	Functional
Req22	UC5	Operations, Management and Orchestration	VNF Catalog Module shall inform Tacker to deploy the desired VNF	Functional
Req23	UC5	Operations, Management and Orchestration	VNF Catalog Module shall inform Tacker to deploy the desired VNF at a specific cloud infrastructure (VIM)	Functional
Req24	UC5, UC10	Operations, Management and Orchestration	VNF Catalog Module shall inform Tacker to start/stop/delete the desired VNF	Functional
Req25	UC8	Operations, Management and Orchestration	VNF Catalog Module shall request Tacker to know VNF behaviour and collect VNF details	Functional
Req26	UC8	Operations, Management and Orchestration	VNF Catalog Module shall request Tacker to know VNF behaviour and collect VNF details in a 1 minute cycle (e.g. with crontab)	Non-Functional

Table B.2: VNF Catalog Module Software Requirements

B.3 Request Module

ID	Use Cases	Requirements Area	Requirement Description	Category
Req27	UC4	Operations	Request Module shall store all available Requests with Request information	Functional
Req28	UC4	Market Operability	Request Module shall allow Customers upload their Requests	Functional

Req29	UC1, UC4	Operations and Market Operability	Request Module shall allow Sellers reply to "Open" Requests	Functional
Req30	UC4	Market Operability	Request Module shall provide search engine to help Sellers find a desired Request more easily	Functional
Req31	UC1, UC4	Operations	Request Module shall up- date Request's status af- ter Sellers replied suc- cessfully	Functional
Req32	UC11	Operations and Market Operability	Request Module shall allow Customer delete/terminate an "Open"/"Closed" Re- quest	Functional
Req33	UC4, UC11	Security and Oper- ations	Request Module shall allow access to authen- ticated and authorised Stakeholders	Functional

Table B.3: Request Module Software Requirements

B.4 Billing Module

ID	Use Cases	Requirements Area	Requirement Description	Category
Req34	UC3, UC7	Market Operability	Billing Module shall store all prices agreed between Customer, Seller and VNF Market- place	Functional
Req35	UC3, UC7	Market Operability	Billing Module shall pro- vide billing information to the dashboard	Functional
Req36	UC7	Operations, Market Operability, Man- agement and Or- chestration	Billing Module shall get VNF information from OpenStack for rating pur- poses	Functional

Req37	UC7	Operations and Market Operability	Billing Module shall get SLA information from SLA Control for rating purposes	Functional
Req38	UC7	Operations and Market Operability	Billing Module shall generate bills	Functional
Req39	UC7	Operations, Market Operability, Management and Orchestration	Billing Module shall get VNF information from OpenStack/SLA Control in 1 minute cycle	Non-Functional
Req40	UC3, UC7	Market Operability	Billing Module shall charge Customer accordingly to the generated bills or the VNF purchase	Functional
Req41	UC3, UC7	Market Operability	Billing Module shall send payout to the Seller accordingly to the generated bills or the VNF purchase	Functional
Req42	UC1	Market Operability	Billing Module shall provide different payments types (Purchase modes: License Payment and Pay-As-You-Go)	Functional
Req43	UC2	Market Operability	Billing Module shall provide different payments options (Paypal and credit card)	Functional
Req44	UC3, UC7	Market Operability	Billing Module shall provide different ways to provide billing information (dashboard and email with invoice PDF)	Functional
Req45	UC3	Market Operability	Billing Module shall send email with VNF license in case of Basic Purchase mode	Functional

Req46	UC3, UC7	Market Operability	Billing Module shall generate PDF with invoices for both Customer and Seller	Functional
Req47	UC2, UC5	Security and Operations	Request Module shall allow access to authenticated and authorised Stakeholders	Functional

Table B.4: Billing Module Software Requirements

B.5 SLA Management

ID	Use Cases	Requirements Area	Requirement Description	Category
Req48	UC2	Service Continuity, Operations and Market Operability	SLA Management shall store all the SLA agreement and templates between Customer and VNF Marketplace for each VNF	Functional
Req49	UC9	Operations, Management and Orchestration	SLA Management should communicate with OpenStack to get the VNF metrics	Functional
Req50	UC9	Operations, Management and Orchestration	SLA Management should communicate with OpenStack in a time cycle	Non-Functional
Req51	UC9	Service Continuity, Operations and Market Operability	SLA Management shall store all the information about SLA fulfillment (violations, penalties and discounts)	Functional
Req52	UC9	Operations and Market Operability	SLA Management shall be connected to SLA Control for sending essential SLA information (discounts, penalties, etc)	Functional

Req53	UC9, UC11	Service Continuity and Operations	SLA Management shall give techniques to get an agreement/template and delete it	Functional
Req54		Management and Orchestration	SLA Management shall allow Administrator manage the SLA Core	Functional

Table B.5: SLA Management Software Requirements

B.6 SLA Control

ID	Use Cases	Requirements Area	Requirement Description	Category
Req55	UC2	Service Continuity, Operations and Market Operability	SLA Control Module shall store all the SLA agreement and templates between Customer and VNF Marketplace for each VNF	Functional
Req56	UC9	Operations, Man- agement and Or- chestration	SLA Control shall com- municate with SLA Man- agement to get the es- sential SLA information (discounts, penalties, etc)	Functional
Req57	UC9	Operations, Man- agement and Or- chestration	SLA Control should communicate with SLA Management in a time cycle	Non- Functional
Req58	UC9	Service Continuity and Market Oper- ability	SLA Control shall be connected to the Dash- board to allow the Cus- tomer visualise SLA in- formation	Functional
Req59	UC9, UC11	Service Continuity and Operations	SLA Control shall give techniques to get an agreement/template and delete it	Functional

Table B.6: SLA Control Module Software Requirements

B.7 Dashboard

Dashboard Software Requirements for the three views:

ID	Use Cases	Requirements Area	Requirement Description	Category
Req60	UC1, UC2, UC4, UC10, UC11	Security	Dashboard shall provide log in feature for the different Stakeholders	Functional
Req61		Security	Dashboard shall provide Register feature for new users	Functional
Req62		Security	Dashboard shall provide log out feature for new users	Functional
Req63	UC1, UC2, UC4, UC10, UC11	Security	Dashboard shall remember authentication token for the different Stakeholders	Functional
Req64	UC1, UC2, UC4, UC10, UC11	Security	Dashboard should provide different ways to authenticate	Functional
Req65	UC1, UC2, UC4, UC10, UC11	Security	Dashboard should provide different ways to authenticate	Functional
Req66		Management and Orchestration	Dashboard should be available to the Internet	Functional
Req67		Operations and Market Operability	Dashboard shall allow Stakeholders update their profiles (name, company, etc)	Functional

Table B.7: General Dashboard Software Requirements

Dashboard Software Requirements for the Administrator view:

ID	Use Cases	Requirements Area	Requirement Description	Category
Req68		Security	Dashboard shall provide the Administrator access to every internal module of VNF Marketplace	Functional
Req69		Management and Orchestration	Dashboard shall give information to the Administrator of every internal module (billing, VNFs, Requests, SLAs, etc)	Functional
Req70		Management and Orchestration	Dashboard shall allow the Administrator perform updates/changes to the data (VNFs, users, roles, permissions and requests) present at the Database	Functional
Req70		Management and Orchestration	Dashboard shall allow the Administrator delete data (VNFs, users, roles, permissions and requests) present at the Database	Functional
Req71		Management and Orchestration	Dashboard shall allow the Administrator create data to the Database (create VNFs, Requests, Users, roles, permissions)	Functional

Table B.8: Administrator Dashboard View Software Requirements

Dashboard Software Requirements for the Customer view:

ID	Use Cases	Requirements Area	Requirement Description	Category
Req72	UC2	Service Continuity and Market Operability	Dashboard shall allow the Customer browse, search and select VNF offerings	Functional

Req73	UC2	Service Continuity and Market Operability	Dashboard shall allow the Customer visualise detailed parameters from VNF offerings	Functional
Req74	UC2	Service Continuity and Market Operability	Dashboard shall allow the Customer select among different SLA flavors (parameters) for the desired VNF	Functional
Req75	UC4	Service Continuity, Operations and Market Operability	Dashboard shall allow the Customer create requests	Functional
Req76	UC4	Service Continuity and Market Operability	Dashboard shall be able to allow the Customer buy/select a "replied" request	Functional
Req77	UC9	Service Continuity, Operations and Market Operability	Dashboard shall allow the Customer visualise SLA agreements information	Functional
Req78	UC9	Service Continuity, Operations and Market Operability	Dashboard should allow the Customer visualise SLA fulfillment information	Functional
Req79	UC2	Service Continuity, Operations and Market Operability	Dashboard shall allow the Customer select different payment options	Functional
Req80	UC11	Service Continuity and Operations	Dashboard shall allow the Customer terminate a request	Functional
Req81	UC10	Service Continuity and Operations	Dashboard shall allow the Customer terminate a VNF	Functional
Req82	UC5, UC6	Operations, Market Operability, Management and Orchestration	Dashboard shall notify the Customer if VNF started correctly (Premium mode)	Functional

Req83	UC6	Service Continuity, Operations and Market Operability	Dashboard shall allow the Customer download the VNF image	Functional
Req84	UC3, UC7	Service Continuity, Operations and Market Operability	Dashboard shall allow the Customer visualise billing information (bills, transactions, PDF with Invoices)	Functional
Req85	UC5	Management and Orchestration	Dashboard shall allow the Customer configure the deployed VNF (start/stop)	Functional
Req86	UC5	Elasticity, Management and Orchestration	Dashboard should be able to allow the Customer scale the deployed VNF (scale-up/scale-down)	Functional
Req87	UC8	Management and Orchestration	Dashboard should be able to allow the Customer visualise VNF metrics	Functional
Req88	UC8	Management and Orchestration	Dashboard shall be able to allow the Customer visualise owned VNF details (with statistics)	Functional
Req89	UC5	Management and Orchestration	Dashboard should be able to allow the Customer configure VNF metrics/parameters	Functional
Req90	UC2	Service Continuity, Portability and Market Operability	Dashboard shall allow the Customer select different NFVI-PoPs/VIMs	Functional

Table B.9: Customer Dashboard View Software Requirements

Dashboard Software Requirements for the Seller view:

ID	Use Cases	Requirements Area	Requirement Description	Category
Req91	UC1	Service Continuity and Market Operability	Dashboard shall allow the Seller provide their VNF information (Name, description, capabilities, type and version)	Functional
Req92	UC1	Service Continuity, Operations and Market Operability	Dashboard shall allow the Seller upload VNF image and VNFD/license file (accordingly to Purchase mode)	Functional
Req93	UC1	Operations and Market Operability	Dashboard shall allow the Seller visualise owned VNFs	Functional
Req94	UC3, UC7	Service Continuity, Operations and Market Operability	Dashboard shall allow the Seller visualise billing information (transactions, PDF with Invoices)	Functional
Req95	UC1	Operations and Market Operability	Dashboard shall allow the Seller visualise owned VNFs	Functional
Req96	UC10	Service Continuity and Operations	Dashboard shall allow the Seller delete a VNF (if VNF is active, it is not terminated. Only is removed from VNF Catalog)	Functional
Req97	UC3, UC5, UC7	Service Continuity, Operations and Market Operability	Dashboard shall allow the Seller visualise billing and VNF statistics	Functional
Req98	UC1	Service Continuity, Operations and Market Operability	Dashboard shall allow the Seller search, browse and reply to Customer's requests	Functional

Req99	UC5	Management and Orchestration	Dashboard should allow the Seller configure Seller's deployed VNFs	Functional
-------	-----	------------------------------	--------------------------------------------------------------------	------------

Table B.10: Seller Dashboard View Software Requirements

B.8 Analytics Module

ID	Use Cases	Requirements Area	Requirement Description	Category
Req100	UC9	Operations, Management and Orchestration	Analytics Module should communicate with all internal modules to get essential information	Functional
Req101	UC9	Operations, Management and Orchestration	Analytics module should communicate with Dashboard to send essential information	Functional

Table B.11: Analytics Module Software Requirements

B.9 OpenStack

ID	Use Cases	Requirements Area	Requirement Description	Category
Req102		Security	OpenStack shall support authentication and authorisation	Functional
Req103		Management and Orchestration	OpenStack shall allow Administrator manage roles, tenants, networks, VNFs and all others services	Functional
Req104	UC1, UC5, UC7, UC8, UC9, UC10	Operations, Management and Orchestration	OpenStack shall communicate with VNF Marketplace and SLA Management to receive requests, perform tasks and send responses	Functional

Req105	UC1	Service Continuity and Operations	OpenStack shall allow upload VNFDs	Functional
Req106		Security	OpenStack shall check and authenticate and VNFDs	Functional
Req107	UC5	Service Continuity and Operations	OpenStack shall allow deploy VNFs	Functional
Req108	UC5	Service Continuity, Operations, Management and Orchestration	OpenStack shall allow VNFs instances to be stopped/started	Functional
Req109	UC10	Service Continuity, Operations, Management and Orchestration	OpenStack shall terminate a deployed VNF	Functional
Req110	UC6	Service Continuity, Operations, Management and Orchestration	OpenStack shall grant infrastructure and resources to the VNF deployment	Functional
Req111		Operations and Management and Orchestration	OpenStack shall generate VNF statistics to Administrator check (with Horizon or CLI)	Functional
Req112	UC8	Operations and Management and Orchestration	OpenStack should generate VNF statistics to the VNF Marketplace	Functional
Req113	UC9	Operations and Management and Orchestration	OpenStack should generate VNF metrics to SLA Management	Functional

Table B.12: OpenStack Software Requirements

Appendix C

Dashboard Views, Requests and Responses

C.1 Customer

```
{'intent': 'sale', 'update_time': '2017-06-21T20:59:47Z', 'links': [{'href': 'https://api.sandbox.paypal.com/v1/payments/payment/PAY-8BB13656W99370600LFFN4PY', 'rel': 'self', 'method': 'GET'}], 'state': 'approved', 'payer': {'payment_method': 'credit_card', 'funding_instruments': [{'credit_card': {'expire_month': '6', 'expire_year': '2022', 'type': 'visa', 'first_name': 'Buyer', 'number': 'xxxxxxx4208', 'last_name': 'Test'}}]}, 'id': 'PAY-8BB13656W99370600LFFN4PY', 'create_time': '2017-06-21T20:59:43Z', 'transactions': [{'amount': {'details': {'subtotal': '11.90'}, 'currency': 'USD', 'total': '11.90'}, 'related_resources': [{'sale': {'update_time': '2017-06-21T20:59:47Z', 'links': [{'href': 'https://api.sandbox.paypal.com/v1/payments/sale/9PK89499DX319180V', 'rel': 'self', 'method': 'GET'}, {'href': 'https://api.sandbox.paypal.com/v1/payments/sale/9PK89499DX319180V/refund', 'rel': 'refund', 'method': 'POST'}, {'href': 'https://api.sandbox.paypal.com/v1/payments/payment/PAY-8BB13656W99370600LFFN4PY', 'rel': 'parent_payment', 'method': 'GET'}], 'create_time': '2017-06-21T20:59:43Z', 'state': 'completed', 'processor_response': {'cvv_code': 'M', 'avs_code': 'Y'}, 'amount': {'currency': 'USD', 'total': '11.90'}, 'id': '9PK89499DX319180V', 'fmf_details': {}}, {'parent_payment': 'PAY-8BB13656W99370600LFFN4PY'}]}, 'item_list': {'items': [{'sku': 'VNF', 'price': '11.90', 'currency': 'USD', 'quantity': '1', 'name': 'Virtual Router'}]}, 'description': 'This is the payment transaction description.'}]}
```

Payment PAY-8BB13656W99370600LFFN4PY created successfully

Figure C.1: Creation of payment with credit card - Request and Response

```
[21/Jun/2017 23:10:34] "DELETE /sla-management/sla-info/8/ HTTP/1.1" 204 0
Auth Token gAAAAABZSu7a7Q02ezGeMdvSAWWCawaFdbPtikuGFP9S2VOTM9NDTUqWLFleQsGhY5v5v
rr1V8EKHK1CRtsLPCybhXJunc2srB1Y7ISichG-iGubKpJRzdGwZ-m2Ceb9raXNt5Ugx4_oKbQKIJooh
L_3J3p_LRJLIMkiRtZEEQGAhe3ZW26cig Got from Openstack - Code: 201
VNF 93354cfe-7de7-4e40-9f89-1cfba58267be Deleted from Openstack - Code: 204
[21/Jun/2017 23:10:34] "DELETE /vnf-management/cust_vnfs/10/ HTTP/1.1" 204 0
```


Figure C.2: Delete VNF (Customer) - Request and Response


```
{
  'intent': 'sale', 'update_time': '2017-06-21T20:59:47Z', 'links': [
    {'href': 'https://api.sandbox.paypal.com/v1/payments/payment/PAY-8BB13656W99370600LFFN4PY',
      'rel': 'self', 'method': 'GET'}], 'state': 'approved', 'payer': {
    'payment_method': 'credit_card', 'funding_instruments': [
      {'credit_card': {
        'expire_month': '6', 'expire_year': '2022', 'type': 'visa',
        'first_name': 'Buyer', 'number': 'xxxxxxxxxxxx4208',
        'last_name': 'Test'}}}], 'id': 'PAY-8BB13656W99370600LFFN4PY',
    'create_time': '2017-06-21T20:59:43Z', 'transactions': [
      {'amount': {
        'details': {
          'subtotal': '11.90'}, 'currency': 'USD', 'total': '11.90'},
        'related_resources': [
          {'sale': {
            'update_time': '2017-06-21T20:59:47Z', 'links': [
              {'href': 'https://api.sandbox.paypal.com/v1/payments/sale/9PK89499DX319180V',
                'rel': 'self', 'method': 'GET'},
              {'href': 'https://api.sandbox.paypal.com/v1/payments/sale/9PK89499DX319180V/refund',
                'rel': 'refund', 'method': 'POST'},
              {'href': 'https://api.sandbox.paypal.com/v1/payments/payment/PAY-8BB13656W99370600LFFN4PY',
                'rel': 'parent_payment', 'method': 'GET'}],
            'create_time': '2017-06-21T20:59:43Z', 'state': 'completed',
            'processor_response': {
              'cvv_code': 'M', 'avs_code': 'Y'}, 'amount': {
              'currency': 'USD', 'total': '11.90'}, 'id': '9PK89499DX319180V',
              'fmf_details': {}, 'parent_payment': 'PAY-8BB13656W99370600LFFN4PY'}],
            'item_list': [
              {'sku': 'VNF', 'price': '11.90', 'currency': 'USD', 'quantity': '1',
                'name': 'Virtual Router'}], 'description': 'This is the payment transaction description.'}]
          }
        ]
      }
    ]
  }
}
Payment PAY-8BB13656W99370600LFFN4PY created successfully
```

Figure C.3: Payment successful with credit card - Request and Response

```
VNF Stopped in Openstack - Code: 202
{
  "hours_used": 0, "paymentflag": false, "owner": 33, "FlavorKey": "Flavor1", "VNFType": "VNF", "paymentseconds": 0, "VnfID": "openstack", "ba357bbs-3ceb-4d06-bb46-e8654330cdd8", "vnfdImagePath": "/media/706834de-f7b1-4932-8e68-b528f1a6f2a7.png", "vnfdTemplateName": "/media/33dec217-1c21-44db-943b-39d75e438465.yaml", "status": "STOPPED", "name": "Clirros VNF", "licenseFilePath": "/media/b0a68bfb-8b00-4325-8351-b0ea33f8598.txt", "VnfID": 33, "owner_company": "FEUP", "owner_email": "sellerTest@feup.com", "templateID": "409", "instanceID": "72623146-9ef4-4329-ba81-a9f34672200f", "serviceID": "Premlun", "VnfID": "openstack", "78918800-4e20-4469-9a1a-8c1d050f150", "client": "customerTest", "agreementID": "agreement3", "VnfID": "VNF0", "pk": 23}
[22/Jun/2017 19:56:04] "PUT /vnf-management/cust_vnfs/25/ HTTP/1.1" 202 698
```

Figure C.4: Stop VNF - Request and Response

 Buy VNF



VNF Name: Virtual Router

Owner: FEUP

VNF Description: Virtual Router

Basic Service

Price: 11.9 €

Payment Type: License Payment (available for one Month)



 PURCHASE

Figure C.5: Purchase VNF in Basic Mode


VNF Marketplace  customerTest


VNF Name: Virtual Firewall

Owner: FEUP

VNF Description: Virtual Firewall

Premium Service

VIM  VIM0

Flavor  Flavor1

For each VDU:
Monitoring Parameter: CPU Utilization
In case CPU Utilization is Greater than or equal 75% during 2 breaches count with a 360 seconds interval: 2% discount in the payment

Price: 0.6659999999999999 \$

Setup: 30 \$

Total: 0.6659999999999999 \$ + 30 \$

Payment Type: Pay-As-You-Go per Hour

Figure C.6: Purchase VNF in Premium mode - SLA Flavor 1

VNF Marketplace

customerTest

VNF Description: Virtual Firewall

Premium Service

VIM
 VIM0

Flavor
 Flavor2

For each VDU:

Monitoring Parameter: CPU Utilization
In case CPU Utilization is Greater than or equal 75% during 2 breaches count with a 360 seconds interval: 2% discount in the payment

Monitoring Parameter: CPU Idle
In case CPU Idle is Greater than or equal 85% during 2 breaches count with a 360 seconds interval: 10% discount in the payment

Price: 0.6779999999999999 \$

Setup: 30 \$

Total: 0.6779999999999999 \$ + 30 \$

Payment Type: Pay-As-You-Go per Hour

Figure C.7: Purchase VNF in Premium mode - SLA Flavor 2

VNF Marketplace

customerTest

VNF Marketplace

Premium Service

VIM
 VIM0

Flavor
 Flavor3

For each VDU:

Monitoring Parameter: CPU Utilization
In case CPU Utilization is Greater than or equal 75% during 2 breaches count with a 360 seconds interval: 2% discount in the payment

Monitoring Parameter: CPU Idle
In case CPU Idle is Greater than 85% or equal during 2 breaches count with a 360 seconds interval: 10% discount in the payment

Monitoring Parameter: Free Memory
In case Free Memory is Less than 10% (of Total Memory) or equal during 2 breaches count with a 360 seconds interval: 5% discount in the payment

Price: 0.69 \$

Setup: 30 \$

Total: 0.69 \$ + 30 \$

Payment Type: Pay-As-You-Go per Hour

Figure C.8: Purchase VNF in Premium mode - SLA Flavor 3

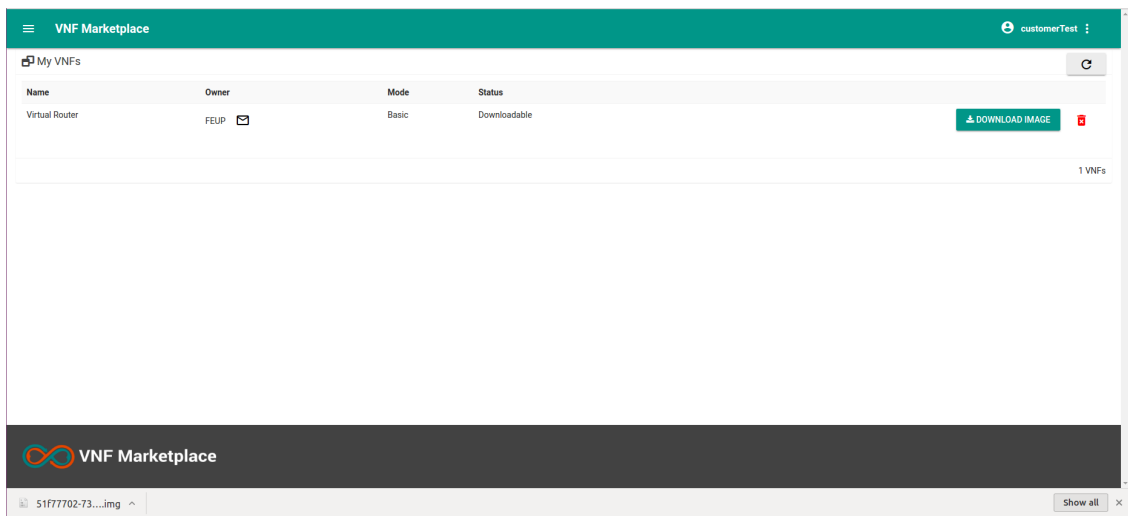


Figure C.9: Download VNF Dashboard View

C.2 Seller

```
{'items': [{'receiver': 'luispedrosousa74-buyer-1@hotmail.com', 'recipient_type': 'EMAIL', 'amount': {'currency': 'USD', 'value': '10.00'}, 'sender_item_id': 'Virtual Router', 'note': 'Thank you.'}], 'links': [{'href': 'https://api.sandbox.paypal.com/v1/payments/payouts/XZNP7P2KTSRVC', 'rel': 'self', 'method': 'GET'}], 'batch_header': {'payout_batch_id': 'XZNP7P2KTSRVC', 'batch_status': 'PENDING', 'sender_batch_header': {'sender_batch_id': 'BULTCWKFYNCZ', 'email_subject': 'You have a payment from VNF Marketplace'}}, 'sender_batch_id': 'BULTCWKFYNCZ', 'email_subject': 'You have a payment from VNF Marketplace'}]
Payout[XZNP7P2KTSRVC] created successfully
```

Figure C.10: Payout with credit card - Request and Response

```
Auth Token gAAAAABZS8Ace8XtOAZGtIWHYG0-TkD8wxX9YJXQncUD5t_560TFq_mnlafH6h7XKMMQNM6uDdzZxgV5lq4fiJa85BSwD7ejAZ6iYv01ShGHwe13K4uKAKT1-qYowm7s91dvSAsxmIYCP0anoPbdMmO_cQjqe0KKldMLDt2CHSxwNelVVuTCJQ Got from Openstack - Code: 201
VNFD 2e3cca1d-5f88-4dc9-9344-c6d8cee08742 Deleted from Openstack - Code: 204
Image d4c9fa90-ce1b-4217-93d2-670750016749 Deleted from Openstack - Code: 204
[22/Jun/2017 14:03:24] "DELETE /vnf-management/vnfs/9/ HTTP/1.1" 204 0
```

Figure C.11: Delete VNF (Seller) - Request and Response

```
<QueryDict: {}>
[OrderedDict([('pk', 8), ('name', 'Virtual Firewall'), ('file', '/media/a201031c-fdc8-4f03-a8a5-33720caa8738.img'), ('description', 'Virtual Firewall'), ('version', '1.0.2'), ('typevnf', 'vFW'), ('id_openstack', '7728df22-673c-472c-9a11-47540b1bc822'), ('vnfd_temp', '/media/a8802a52-c63c-4163-9d14-d79b9473ff9d.yaml'), ('license_file', '/media/8aadf26d-5ff9-4f2d-8dc4-d38a17ecaf15.txt'), ('id_image', '0df9099b-1956-4143-8f90-c5234bcdec76'), ('owner', 'sellerTest'), ('owner_number', 33), ('owner_company', 'FEUP'), ('owner_email', 'sellerTest@feup.com'), ('modeID', 'Premium'), ('billing', '{"model": "Pay-As-You-Go", "period": "Hour", "price": {"value": 0.6, "unit": "$"}}')]), OrderedDict([('pk', 7), ('name', 'Virtual Router'), ('file', '/media/51f77702-73e0-4712-badc-57919a7c4b24.img'), ('description', 'Virtual Router'), ('version', '1.0.0'), ('typevnf', 'vRouter'), ('id_openstack', 'ID'), ('vnfd_temp', '/media/f29841bf-2b52-41bc-8eaa-b565eed67956.yaml'), ('license_file', '/media/4834c5dd-5271-4936-9858-1a412fc19060.txt'), ('id_image', 'bcd99c3a-d6d6-4d55-848b-df6a11849a1c'), ('owner', 'sellerTest'), ('owner_number', 33), ('owner_company', 'FEUP'), ('owner_email', 'sellerTest@feup.com'), ('modeID', 'Basic'), ('billing', '{"model": "License Payment", "period": "Month", "price": {"value": 10, "unit": "\u20ac"}}')])]
```

[21/Jun/2017 21:44:04] "GET /vnf-management/vnfs/ HTTP/1.1" 200 1191

Figure C.12: Get Seller's VNFs - Request and Response

```
{'vnfID': 31, 'status': 'Closed'}
{'vnfID': 31, 'status': 'Closed', 'owner': 'customerTest', 'pk': 15, 'description': 'Request VNF', 'client': 'sellerTest', 'name': 'Request VNF', 'mode': 'Basic', 'typevnf': 'vRequest', 'client_company': 'FEUP'}
```

[22/Jun/2017 17:21:23] "PUT /vnf-management/requests/15/ HTTP/1.1" 200 192

Figure C.13: Reply to a Customer's Request - Request and Response

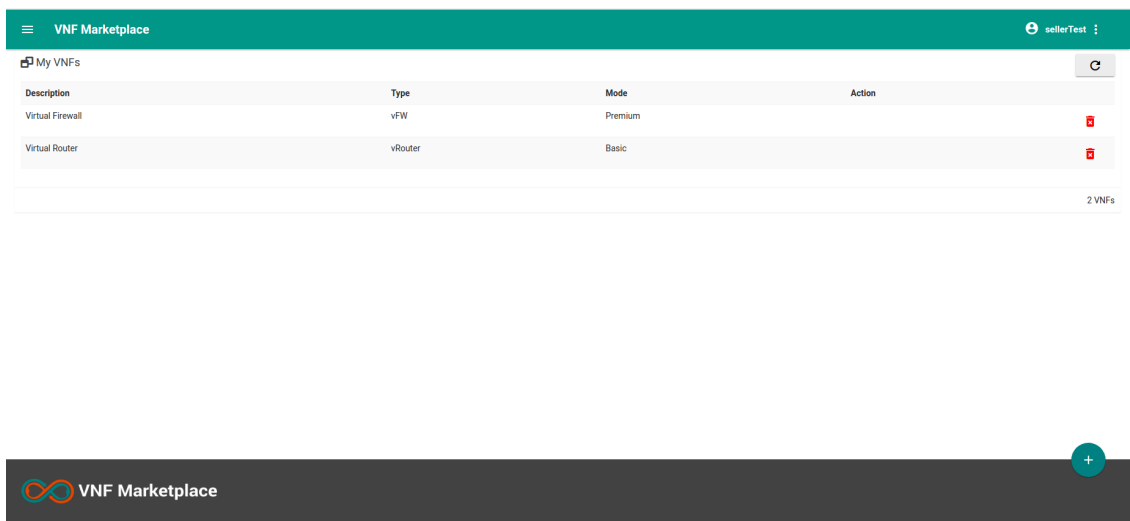


Figure C.14: Get Seller’s VNFs Dashboard View

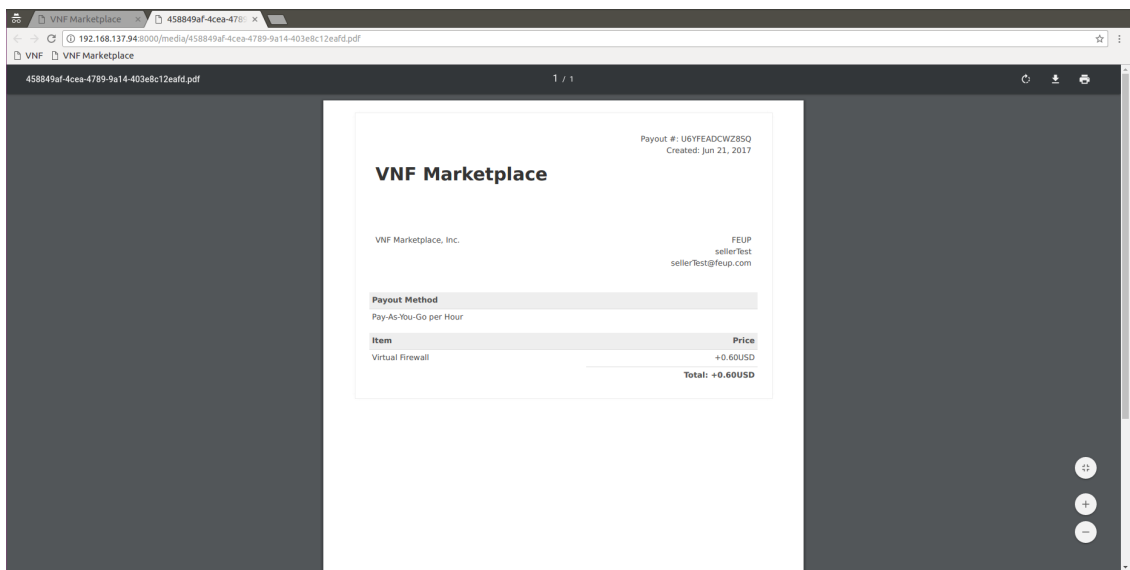


Figure C.15: Get Seller Invoice (PDF)

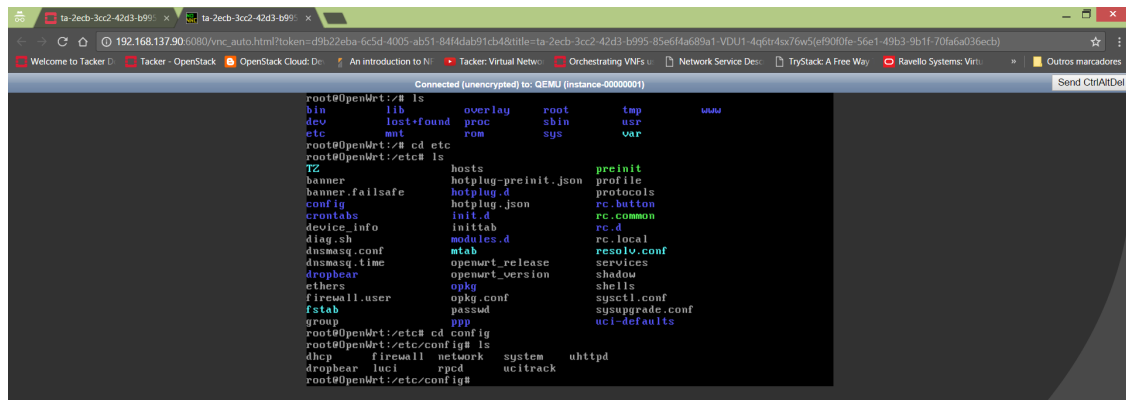


Figure C.19: OpenWRT Config Directory

```
config defaults
    option syn_flood      1
    option input          ACCEPT
    option output         ACCEPT
    option forward        REJECT
# Uncomment this line to disable ipv6 rules
#    option disable_ipv6  1

config zone
    option name           lan
    list network          'lan'
    option input          ACCEPT
    option output         ACCEPT
    option forward        ACCEPT

config zone
    option name           wan
    list network          'wan'
    list network          'wan6'
    option input          REJECT
    option output         ACCEPT
    option forward        REJECT
    option masq           1
    option mtu_fix        1
```

Figure C.20: OpenWRT with Firewall configured - Part 1

```
config forwarding
    option src                lan
    option dest               wan

# We need to accept udp packets on port 68,
# see https://dev.openwrt.org/ticket/4108
config rule
    option name               Allow-DHCP-Renew
    option src                wan
    option proto              udp
    option dest_port          68
    option target              ACCEPT
    option family              ipv4

# Allow IPv4 ping
config rule
    option name               Allow-Ping
    option src                wan
    option proto              icmp
    option icmp_type           echo-request
    option family              ipv4
    option target              ACCEPT
```

Figure C.21: OpenWRT with Firewall configured - Part 2

```
root@OpenWrt:/etc/config# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: seq=0 ttl=43 time=34.052 ms
64 bytes from 8.8.8.8: seq=1 ttl=43 time=61.887 ms
64 bytes from 8.8.8.8: seq=2 ttl=43 time=43.459 ms
64 bytes from 8.8.8.8: seq=3 ttl=43 time=57.971 ms
64 bytes from 8.8.8.8: seq=4 ttl=43 time=32.347 ms
^C
--- 8.8.8.8 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 32.347/45.943/61.887 ms
root@OpenWrt:/etc/config#
```

Figure C.22: OpenWRT Ping to Internet

Appendix D

Files & Detailed Results

D.1 *Requirements.txt* File

```
1 appdirs==1.4.3
2 asn1crypto==0.22.0
3 beautifulsoup4==4.5.3
4 braintree==3.37.2
5 bson==0.4.7
6 cffi==1.10.0
7 cryptography==1.8.1
8 Django==1.8.11
9 django-appconf==1.0.2
10 django-bower==5.2.0
11 django-common-helpers==0.9.1
12 django-compressor==2.1.1
13 django-countries==4.3
14 django-crontab==0.7.1
15 django-debug-toolbar-request-history==0.0.4
16 django-filter==1.0.2
17 django-livereload-server==0.2.3
18 django-manager-utils==0.12.0
19 django-performance-testing==0.6.1
20 django-query-builder==0.14.1
21 djangoestframework==3.6.2
22 djangoestframework-jwt==1.10.0
23 djangoestframework-yaml==1.0.3
24 fleming==0.4.4
25 html5lib==1.0b10
26 httpie==0.9.9
27 httplib2==0.10.3
28 idna==2.5
29 olefile==0.44
30 packaging==16.8
31 paypalrestsdk==1.12.0
32 pdfkit==0.6.1
33 Pillow==4.1.1
34 pycparser==2.17
35 pycrypto==2.6.1
36 Pygments==2.2.0
37 PyJWT==1.4.2
38 pymongo==3.4.0
39 pyOpenSSL==17.0.0
40 pyparsing==2.2.0
41 PyPDF2==1.26.0
42 python-dateutil==2.6.0
43 pytz==2017.2
44 PyYAML==3.12
45 rcssmin==1.0.6
46 reportlab==3.4.0
47 requests==2.13.0
48 rjsmin==1.0.12
49 six==1.10.0
50 sqlparse==0.2.3
```

```

51 tornado==4.4.3
52 webencodings==0.5.1
53 lxml2pdf==0.2b1

```

Listing D.1: VNF Marketplace Server Requirements

D.2 *Settings.py* File

```

1 """
2 Django settings for VNFMarketplace project.
3
4 Generated by 'django-admin startproject' using Django 1.10.6.
5
6 For more information on this file, see
7 https://docs.djangoproject.com/en/1.10/topics/settings/
8
9 For the full list of settings and their values, see
10 https://docs.djangoproject.com/en/1.10/ref/settings/
11 """
12
13 import os
14 import datetime
15
16 # Build paths inside the project like this: os.path.join(BASE_DIR, ...)
17 BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
18
19
20 # Quick-start development settings - unsuitable for production
21 # See https://docs.djangoproject.com/en/1.10/howto/deployment/checklist/
22
23 # SECURITY WARNING: keep the secret key used in production secret!
24 SECRET_KEY = '*****'
25
26 # SECURITY WARNING: don't run with debug turned on in production!
27 DEBUG = True
28 TEMPLATE_DEBUG = True
29 ALLOWED_HOSTS = []
30
31
32 # Application definition
33
34 INSTALLED_APPS = [
35     'django.contrib.admin',
36     'django.contrib.auth',
37     'django.contrib.contenttypes',
38     'django.contrib.sessions',
39     'django.contrib.messages',
40     'django.contrib.staticfiles',
41     'django_countries',
42     'django_crontab',
43     'rest_framework',
44     'compressor',
45     'authentication',
46     'djgobower',
47     'VNFMarketplace_app',
48     'livereload',
49     'vnf',
50     'sla',
51     'billing',
52 ]
53
54 MIDDLEWARE_CLASSES = [
55     #'VNFMarketplace.stats_middleware.RequestTimeLoggingMiddleware',
56     'django.contrib.sessions.middleware.SessionMiddleware',
57     'django.middleware.common.CommonMiddleware',
58     'django.middleware.csrf.CsrfViewMiddleware',
59     'django.contrib.auth.middleware.AuthenticationMiddleware',
60     'django.middleware.security.SecurityMiddleware',
61     'django.contrib.auth.middleware.SessionAuthenticationMiddleware',
62     'django.contrib.messages.middleware.MessageMiddleware',
63     'django.middleware.clickjacking.XFrameOptionsMiddleware',
64 ]
65
66

```

```

67 #(' * * * * ', 'billing.cron-tasks.Task_Update_Usage', '>> /tmp/task_usage.log '),
68
69 CRONTAB_COMMAND_SUFFIX = '2>&1'
70
71 CRONJOBS = [
72     ('*/1 * * * *', 'billing.cron-tasks.Task_Update_Hour', '>> /tmp/file.log ')
73 ]
74
75 REST_FRAMEWORK = {
76     'DEFAULT_AUTHENTICATION_CLASSES': (
77         'rest_framework_jwt.authentication.JSONWebTokenAuthentication',
78     ),
79     'DEFAULT_RENDERER_CLASSES': (
80         'rest_framework.renderers.JSONRenderer',
81         'rest_framework_yaml.renderers.YAMLRenderer',
82     ),
83     'DEFAULT_PARSER_CLASSES': (
84         'rest_framework.parsers.JSONParser',
85         'rest_framework.parsers.MultiPartParser',
86         'rest_framework_yaml.parsers.YAMLParser',
87         'rest_framework.parsers.FileUploadParser',
88     ),
89     'DEFAULT_FILTER_BACKENDS': (
90         'django_filters.rest_framework.DjangoFilterBackend',
91     )
92 }
93
94 JWT_AUTH = {
95     'JWT_PAYLOAD_HANDLER': 'authentication.utils.jwt_payload_handler',
96     'JWT_EXPIRATION_DELTA': datetime.timedelta(days=5),
97 }
98
99 AUTH_USER_MODEL = 'authentication.User'
100
101 ROOT_URLCONF = 'VNFMarketplace.urls'
102
103 TEMPLATE_DIRS = (
104     os.path.join(BASE_DIR, 'templates'),
105 )
106
107 WSGI_APPLICATION = 'VNFMarketplace.wsgi.application'
108
109
110 # Database
111 # https://docs.djangoproject.com/en/1.10/ref/settings/#databases
112
113 DATABASES = {
114     'default': {
115         'ENGINE': 'django.db.backends.sqlite3',
116         'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
117     }
118 }
119
120 STATICFILES_FINDERS = (
121     'django.contrib.staticfiles.finders.FileSystemFinder',
122     'django.contrib.staticfiles.finders.AppDirectoriesFinder',
123     'django_bower.finders.BowerFinder',
124     'compressor.finders.CompressorFinder'
125 )
126
127 # Password validation
128 # https://docs.djangoproject.com/en/1.10/ref/settings/#auth-password-validators
129
130 AUTH_PASSWORD_VALIDATORS = [
131     {
132         'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
133     },
134     {
135         'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
136     },
137     {
138         'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
139     },
140     {
141         'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
142     },
143 ]

```

```

144
145
146 # Internationalization
147 # https://docs.djangoproject.com/en/1.10/topics/i18n/
148
149 LANGUAGE_CODE = 'en-us'
150
151 TIME_ZONE = 'Europe/Lisbon' #'UTC'
152
153 USE_I18N = True
154
155 USE_L10N = True
156
157 USE_TZ = True
158
159
160 # Static files (CSS, JavaScript, Images)
161 # https://docs.djangoproject.com/en/1.10/howto/static-files/
162
163 STATIC_ROOT = 'staticfiles'
164 STATIC_URL = '/static/'
165 MEDIA_ROOT = os.path.join(BASE_DIR, '..', 'uploaded_media')
166 MEDIA_URL = '/media/'
167
168 STATICFILES_DIRS = (
169     os.path.join(BASE_DIR, 'static'),
170 )
171
172 CLIENT_ID_PAYPAL = '*****'
173 CLIENT_SECRET_PAYPAL = '*****'
174
175
176 SLA_URL = 'http://localhost:8080/sla-service'
177
178 SHA256_KEY = '*****'
179
180 #ACCESS_TOKEN_PAYPAL = 'access_token$sandbox$bxvm88t3cg2pxq32$31ddc008ff2a3126da66e82378891ea7'
181
182 EMAIL_USE_TLS=True
183 EMAIL_BACKEND='django.core.mail.backends.smtp.EmailBackend'
184 EMAIL_HOST='smtp.gmail.com'
185 EMAIL_HOST_USER='vnfmarketplace@gmail.com'
186 EMAIL_HOST_PASSWORD='*****'
187 EMAIL_PORT=587
188 DEFAULT_FROM_EMAIL = EMAIL_HOST_USER
189
190
191 BOWER_COMPONENTS_ROOT = os.path.join(BASE_DIR, 'static')
192 #BOWER_PATH = '/usr/bin/bower'
193
194 BOWER_INSTALLED_APPS = (
195     'angular#1.6.4',
196     'angular-ui-router',
197     'angular-cookies',
198     'ng-file-upload',
199     'restangular',
200     'lodash',
201     'file-saver',
202     'angular-animate',
203     'angular-bootstrap',
204     'angular-sanitize',
205     'braintree-web',
206     'bootstrap',
207     'angular-bootstrap-checkbox',
208     'angular-motion',
209     'angular-ui-select',
210     'angular-uuid2',
211     'chart.js',
212     'ng-file-upload',
213     'ng-dialog',
214     'angular-yamljs',
215     'roboto-fontface',
216     'font-awesome',
217     'jsplumb',
218     'normalize-css'

```

219)

Listing D.2: VNF Marketplace Settings File

D.3 Configuration.properties File

```

1  ##to copy the sla-service.war to the tomcat directory
2  ##if the path doesn't exist, it will be created
3  tomcat.directory = {sla_tomcat_directory}
4
5  ##to configure the access to the database
6  db.username = atossla
7  db.password = _atossla_
8  db.name = atossla
9  db.host = localhost
10 db.port = 3306
11 db.showSQL = true
12
13 ##for eu atos.sla a different file is generated. it will be generated at log.slaatos.fullpathFilename
14 ##another file with all the logs, hibernate, spring, etc, is generated at log.thirdpartysw.fullpathFilename
15 ##the location of this file must be specified here
16 log.slaatos.fullpathFilename = log/atosSLAfile.log
17 log.slaatos.debugLevel = ALL
18 log.thirdpartysw.fullpathFilename = log/atosSLAfullfile.log
19 log.thirdpartysw.debugLevel = INFO
20
21 #sla-enforcement
22 #configure the classes that will validate and retrieve the data from 3rd party software to monitor
23 #cron job and the poll interval is also configured
24 enforcement.constraintEvaluator.class = eu atos.sla.evaluation.constraint.simple.SimpleConstraintEvaluator
25 enforcement.metricsRetriever.class = eu atos.sla.monitoring.simple.DummyMetricsRetriever
26 enforcement.poll.interval.mseconds = 20000
27 enforcement.spawnlookup.cron = 30 * * * *
28 enforcement.notification.class = eu atos.sla.notification.DummyEnforcementNotifier
29
30 ##security
31 ##configure user and password to access to the rest services
32 ##basic security is used
33 service.basicsecurity.user = *****
34 service.basicsecurity.password = *****
35
36
37 #xml and json parsers
38 #please, set the values to eu atos.sla.parser.NullParser if no parser has to be used for json.
39 #implemented json parsers: eu atos.sla.parser.json.AgreementParser, eu atos.sla.parser.json.TemplateParser
40 #implemented xml parsers: eu atos.sla.parser.xml.AgreementParser, eu atos.sla.parser.xml.TemplateParser
41 #if eu atos.sla.parser.NullParser is set for the xml parsers, the above mentioned default parsers will be used. The
    SLA core allways accepts
42 #templates and agreements in xml format. In case of json it is possible to set the parser to null. In such a case
    nothing will be executed
43 parser.json.agreement.class = eu atos.sla.parser.json.AgreementParser
44 parser.json.template.class = eu atos.sla.parser.json.TemplateParser
45 parser.xml.agreement.class = eu atos.sla.parser.xml.AgreementParser
46 parser.xml.template.class = eu atos.sla.parser.xml.TemplateParser
47
48
49 #format for date in the template and agreement. Please check java.text.SimpleDateFormat to know the format of the
    string
50 parser.date.format = yyyy-MM-dd'T'HH:mm:ssz
51 #if no timezone is readed with the format, this will be the used timezone
52 parser.date.unmarshall.timezone = GMT
53 #timezone to be used to return the date values
54 parser.date.marshall.timezone = CET
55
56 #
57 # Converter values
58 #
59
60 # class that parses business values in a template/agreement
61 converter.businessparser.class = eu atos.sla.util.BusinessValueListParser

```

Listing D.3: SLA Management Configuration File

D.4 *Local.conf* File

```

1 [[local|localrc]]
2 HOST_IP= (INSERT IP)
3
4 ADMIN_PASSWORD=*****
5 DATABASE_PASSWORD=*****
6 RABBIT_PASSWORD=*****
7 SERVICE_PASSWORD=*****
8 SERVICE_TOKEN=*****
9
10 Q_ML2_PLUGIN_EXT_DRIVERS=port_security
11
12 enable_plugin ceilometer http://git.openstack.org/openstack/ceilometer
13
14 enable_plugin tacker https://git.openstack.org/openstack/tacker
15
16 enable_plugin heat https://git.openstack.org/openstack/heat

```

Listing D.4: Devstack Configuration File

D.5 *VNFMarkerplaceInstallation.txt* File

```

1 #Install Python
2
3 sudo apt-get update
4
5 sudo apt-get install python-pip python-dev build-essential libffi-dev libssl-dev libxml2-dev
6
7 #Install and activate test environment (optional)
8 pip install virtualenv
9 virtualenv env
10 source env/bin/activate
11
12 #Install VNF Marketplace Requirements
13 pip install -r requirements.txt #use this command inside VNF Marketplace root directory
14
15 #Run Server
16 python manage.py createsuperuser #optional!
17
18 python manage.py runserver (IP):8000

```

Listing D.5: VNF Marketplace Installation

D.6 *SLAManagementReqInstallation.txt* File

```

1 # Java JDK 8 Installation
2 sudo apt-add-repository ppa:webupd8team/java
3 sudo apt-get update
4 sudo apt-get install oracle-java8-installer
5
6 # Install MySQL
7 sudo apt-get install mysql-server
8
9 # Install Maven
10 sudo apt-get install maven

```

Listing D.6: SLA Management Requirements Installation

D.7 OpenWRT VNFD

```

1  tosca_definitions_version: tosca_simple_profile_for_nfv_1_0_0
2
3  description: OpenWRT with services
4
5  metadata:
6    template_name: OpenWRT
7
8  topology_template:
9    node_templates:
10     VDU1:
11       type: tosca.nodes.nfv.VDU.Tacker
12       capabilities:
13         nfv_compute:
14           properties:
15             num_cpus: 1
16             mem_size: 512 MB
17             disk_size: 1 GB
18       properties:
19         image: OpenWRT
20         config:
21           firewall: 1
22           package firewall
23
24         config defaults
25           option syn_flood '1'
26           option input 'ACCEPT'
27           option output 'ACCEPT'
28           option forward 'REJECT'
29
30         config zone
31           option name 'lan'
32           list network 'lan'
33           option input 'ACCEPT'
34           option output 'ACCEPT'
35           option forward 'ACCEPT'
36
37         config zone
38           option name 'wan'
39           list network 'wan'
40           list network 'wan6'
41           option input 'REJECT'
42           option output 'ACCEPT'
43           option forward 'REJECT'
44           option masq '1'
45           option mtu_fix '1'
46
47         config forwarding
48           option src 'lan'
49           option dest 'wan'
50
51         config rule
52           option name 'Allow-DHCP-Renew'
53           option src 'wan'
54           option proto 'udp'
55           option dest_port '68'
56           option target 'ACCEPT'
57           option family 'ipv4'
58
59         config rule
60           option name 'Allow-Ping'
61           option src 'wan'
62           option proto 'icmp'
63           option icmp_type 'echo-request'
64           option family 'ipv4'
65           option target 'ACCEPT'
66     mgmt_driver: openwrt
67
68     CPI:
69       type: tosca.nodes.nfv.CP.Tacker
70       properties:
71         management: true
72         anti_spoofing_protection: false
73       requirements:

```

```

74     - virtualLink:
75       node: VL1
76     - virtualBinding:
77       node: VDUI
78
79   CP2:
80     type: tosca.nodes.nfv.CP.Tacker
81     properties:
82       order: 1
83       anti_spoofing_protection: false
84     requirements:
85       - virtualLink:
86         node: VL2
87       - virtualBinding:
88         node: VDUI
89
90   CP3:
91     type: tosca.nodes.nfv.CP.Tacker
92     properties:
93       order: 2
94       anti_spoofing_protection: false
95     requirements:
96       - virtualLink:
97         node: VL3
98       - virtualBinding:
99         node: VDUI
100
101   VL1:
102     type: tosca.nodes.nfv.VL
103     properties:
104       network_name: net_mgmt
105       vendor: Tacker
106
107   VL2:
108     type: tosca.nodes.nfv.VL
109     properties:
110       network_name: private1
111       vendor: Tacker
112
113   VL3:
114     type: tosca.nodes.nfv.VL
115     properties:
116       network_name: private2
117       vendor: Tacker

```

Listing D.7: OpenWRT VNFD

D.8 Loading Static Files - CPU and RAM Results

```

1 vmstats -S m l
2 proc --memory-- --swap-- --io-- --system-- --cpu--
3 r b swpd free buff cache si so bi bo in cs us sy id wa st
4 2 0 0 4248 264 1224 0 0 554 174 263 850 10 3 87 1 0
5 2 0 0 4269 264 1224 0 0 0 0 1072 2504 19 6 76 0 0
6 1 0 0 4269 264 1224 0 0 0 0 1315 5279 30 6 64 0 0
7 3 0 0 4262 264 1224 0 0 0 0 1858 6695 44 10 47 0 0
8 0 0 0 4258 264 1224 0 0 0 20 2249 9630 61 12 27 0 0
9 2 0 0 4257 264 1224 0 0 0 0 1331 3506 25 5 69 0 0
10 0 0 0 4258 264 1224 0 0 0 0 357 675 5 2 94 0 0
11 0 0 0 4258 264 1224 0 0 0 0 604 1579 8 3 89 0 0
12 0 0 0 4258 264 1224 0 0 0 0 1609 3453 29 3 68 0 0
13 3 0 0 4254 264 1224 0 0 0 48 2365 11370 66 19 15 0 0
14 0 0 0 4254 264 1224 0 0 0 0 1133 5183 18 8 74 0 0
15 1 0 0 4254 264 1224 0 0 0 0 235 1247 5 1 94 0 0
16 0 0 0 4254 264 1224 0 0 0 0 214 910 7 2 92 0 0
17 1 0 0 4254 264 1224 0 0 0 92 288 1570 8 3 88 1 0
18 0 0 0 4254 264 1224 0 0 0 0 315 1538 8 2 91 0 0
19 5 0 0 4254 264 1224 0 0 0 0 1161 5474 25 9 66 0 0
20 0 0 0 4250 264 1224 0 0 0 0 2189 10017 61 15 24 0 0
21 1 0 0 4251 264 1224 0 0 0 0 1103 2850 15 4 81 0 0
22 0 0 0 4251 264 1224 0 0 0 0 361 769 5 2 94 0 0
23 1 0 0 4251 264 1224 0 0 0 44 655 1274 12 3 85 0 0
24 1 0 0 4251 264 1224 0 0 0 0 1067 3936 19 4 76 0 0
25 2 0 0 4247 264 1224 0 0 0 0 2280 10702 62 14 25 0 0

```


26	0	0	0	4247	264	1224	0	0	0	0	1451	4895	32	7	61	0	0
27	0	0	0	4247	264	1224	0	0	0	0	304	565	3	1	96	0	0
28	0	0	0	4248	264	1224	0	0	0	0	282	463	6	2	93	0	0
29	0	0	0	4246	264	1224	0	0	0	0	2394	7671	53	12	35	0	0
30	0	0	0	4246	264	1224	0	0	0	88	320	528	6	1	93	0	0
31	0	0	0	4247	264	1224	0	0	0	0	294	494	5	1	94	0	0
32	1	0	0	4246	264	1224	0	0	0	0	2825	9650	65	18	17	0	0
33	0	0	0	4246	264	1224	0	0	0	0	1296	3905	24	6	70	0	0
34	0	0	0	4246	264	1224	0	0	0	24	248	1581	6	2	92	0	0

Listing D.8: Loading CPU and RAM Results

D.9 Seller VNF Publication - CPU Results

1	VNF Publication – Basic with Cirros:																	
2																		
3	vmstat -S m l																	
4	procs -----memory-----swap-----io-----system-----cpu-----																	
5	r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa	st	
6	2	0	0	3804	275	1468	0	0	47	25	162	478	7	1	92	0	0	
7	0	0	0	3802	275	1470	0	0	0	84	1496	5148	21	5	74	0	0	
8	4	0	0	3719	276	1553	0	0	8	124	2395	14157	58	21	21	0	0	
9	3	0	0	3766	276	1506	0	0	0	0	1427	4829	23	7	69	0	0	
10	procs -----memory-----swap-----io-----system-----cpu-----																	
11	r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa	st	
12	0	0	0	3775	276	1496	0	0	46	26	163	480	6	1	92	0	0	
13	2	0	0	3775	276	1496	0	0	0	0	559	3397	13	5	82	0	0	
14	5	0	0	3762	276	1504	0	0	0	84	1405	8816	26	10	64	0	0	
15	0	0	0	3683	276	1588	0	0	12832	104	2533	15797	59	23	18	0	0	
16	3	0	0	3738	276	1533	0	0	0	0	1016	4184	23	7	71	0	0	
17	procs -----memory-----swap-----io-----system-----cpu-----																	
18	r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa	st	
19	0	0	0	3729	276	1537	0	0	46	27	165	487	7	1	92	0	0	
20	0	0	0	3728	276	1537	0	0	0	0	920	4455	20	6	74	0	0	
21	0	0	0	3728	276	1536	0	0	0	68	1044	3767	18	4	78	0	0	
22	4	0	0	3671	276	1594	0	0	0	100	2543	14483	49	21	31	0	0	
23	1	0	0	3697	276	1568	0	0	0	0	1430	6959	27	7	66	0	0	
24	procs -----memory-----swap-----io-----system-----cpu-----																	
25	r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa	st	
26	1	0	0	3722	276	1550	0	0	46	28	166	493	7	2	92	0	0	
27	1	0	0	3721	276	1552	0	0	0	0	1123	2389	12	4	84	0	0	
28	0	0	0	3722	276	1550	0	0	0	0	748	1940	12	3	86	0	0	
29	5	0	0	3722	276	1550	0	0	0	0	499	1816	12	2	86	0	0	
30	2	0	0	3647	276	1626	0	0	0	196	2202	11810	53	18	29	0	0	
31	2	0	0	3695	276	1578	0	0	0	0	1655	4722	29	8	63	0	0	
32	procs -----memory-----swap-----io-----system-----cpu-----																	
33	r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa	st	
34	2	0	0	3692	276	1579	0	0	46	29	168	499	7	2	92	0	0	
35	0	0	0	3690	276	1581	0	0	0	0	1607	4968	22	7	71	0	0	
36	0	0	0	3693	276	1577	0	0	0	0	1078	3938	18	6	77	0	0	
37	4	0	0	3651	276	1619	0	0	0	160	2159	12820	56	20	24	0	0	
38	0	0	0	3705	276	1565	0	0	0	0	875	4058	17	6	77	0	0	
39	0	0	0	3706	276	1564	0	0	0	0	779	3197	16	5	79	0	0	
40																		
41	VNF Publication – Premium with Cirros:																	
42																		
43	vmstat -S m l																	
44	procs -----memory-----swap-----io-----system-----cpu-----																	
45	r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa	st	
46	1	0	0	3881	273	1323	0	0	56	21	153	449	6	1	92	0	0	
47	3	0	0	3881	273	1323	0	0	0	0	578	2341	9	5	86	0	0	
48	1	0	0	3882	273	1323	0	0	0	0	710	1352	10	3	87	0	0	
49	0	0	0	3882	273	1323	0	0	0	0	429	763	8	2	90	0	0	
50	3	0	0	3810	273	1393	0	0	12832	8	2107	12612	59	19	22	0	0	
51	3	0	0	3761	273	1425	0	0	304	108	2381	12790	75	21	4	0	0	
52	0	0	0	3763	273	1440	0	0	0	0	1813	13249	55	16	28	0	0	
53	0	0	0	3840	273	1362	0	0	0	0	891	2191	11	5	85	0	0	
54	procs -----memory-----swap-----io-----system-----cpu-----																	
55	r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa	st	
56	3	0	0	3802	274	1393	0	0	53	22	156	456	6	1	92	0	0	
57	0	0	0	3800	274	1395	0	0	0	0	742	2138	12	3	85	0	0	
58	0	0	0	3800	274	1395	0	0	0	0	673	2732	14	5	81	0	0	
59	1	0	0	3801	274	1393	0	0	0	84	316	961	6	2	93	0	0	
60	0	0	0	3799	274	1395	0	0	0	0	1337	4605	21	8	71	0	0	

```

61 7 0 0 3740 274 1454 0 0 0 0 1471 9430 54 16 30 0 0
62 1 0 0 3714 274 1480 0 0 0 104 2545 14792 73 21 6 0 0
63 0 0 0 3778 274 1417 0 0 0 0 784 1310 9 4 88 0 0
64 0 0 0 3778 274 1417 0 0 0 0 1129 2913 13 4 83 0 0
65 procs -----memory-----swap-----io-----system-----cpu-----
66 r b swpd free buff cache si so bi bo in cs us sy id wa st
67 2 0 0 3865 275 1413 0 0 49 22 156 454 6 1 92 0 0
68 1 0 0 3862 275 1417 0 0 0 12 726 4237 13 4 83 0 0
69 0 0 0 3863 275 1415 0 0 0 0 562 1741 10 4 86 0 0
70 0 0 0 3863 275 1415 0 0 0 0 1006 2480 13 4 84 0 0
71 4 0 0 3811 275 1467 0 0 0 100 1830 12007 43 14 43 0 0
72 4 0 0 3764 275 1513 0 0 0 104 2454 11956 75 20 5 0 0
73 3 0 0 3782 275 1496 0 0 0 112 908 5912 23 6 71 0 0
74 1 0 0 3839 275 1439 0 0 0 0 970 2932 9 6 85 0 0
75 procs -----memory-----swap-----io-----system-----cpu-----
76 r b swpd free buff cache si so bi bo in cs us sy id wa st
77 3 0 0 3836 275 1441 0 0 48 23 157 459 6 1 92 0 0
78 0 0 0 3834 275 1443 0 0 0 0 787 4688 11 4 85 0 0
79 0 0 0 3836 275 1441 0 0 0 0 317 1891 9 2 89 0 0
80 1 0 0 3836 275 1441 0 0 0 0 239 1087 4 1 95 0 0
81 1 0 0 3835 275 1441 0 0 0 40 1084 2429 13 4 82 0 0
82 4 0 0 3735 275 1525 0 0 0 92 2383 13180 73 19 9 0 0
83 0 0 0 3735 275 1542 0 0 0 0 2087 9203 53 18 29 0 0
84 2 0 0 3812 275 1465 0 0 0 0 956 2569 14 6 80 0 0
85 procs -----memory-----swap-----io-----system-----cpu-----
86 r b swpd free buff cache si so bi bo in cs us sy id wa st
87 1 0 0 3807 275 1468 0 0 48 24 159 466 6 1 92 0 0
88 0 0 0 3807 275 1468 0 0 0 84 389 1543 6 2 92 1 0
89 0 0 0 3807 275 1468 0 0 0 0 468 921 7 1 92 0 0
90 1 0 0 3807 275 1468 0 0 0 0 348 1453 8 2 90 0 0
91 5 0 0 3784 275 1491 0 0 0 0 1398 7934 34 10 56 0 0
92 2 0 0 3774 275 1501 0 0 0 120 2582 11840 68 26 7 0 0
93 1 0 0 3822 275 1453 0 0 0 0 1977 9215 40 12 48 0 0
94 0 0 0 3836 275 1439 0 0 0 0 311 2628 9 2 89 0 0
95
96 VNF Publication – Basic with Ubuntu:
97
98 mpstat -P ALL 5
99 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
100 Average: all 62,55 0,07 25,24 1,51 0,00 1,47 0,00 0,00 0,00 9,16
101
102 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
103 Average: all 64,85 0,07 25,37 1,61 0,00 1,79 0,00 0,00 0,00 6,32
104
105 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
106 Average: all 60,45 0,07 25,93 1,03 0,00 1,78 0,00 0,00 0,00 10,74
107
108 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
109 Average: all 58,42 0,05 29,88 0,15 0,00 2,22 0,00 0,00 0,00 9,27
110
111 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
112 Average: all 61,12 0,15 26,33 0,56 0,00 2,00 0,00 0,00 0,00 9,84
113
114 VNF Publication – Premium with Ubuntu:
115
116 mpstat -P ALL 5
117 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
118 Average: all 69,74 0,01 16,42 0,47 0,00 2,13 0,00 0,00 0,00 11,23
119
120 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
121 Average: all 69,85 0,01 13,96 0,04 0,00 1,17 0,00 0,00 0,00 14,98
122
123 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
124 Average: all 69,60 0,00 17,01 0,10 0,00 0,22 0,00 0,00 0,00 13,07
125
126 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
127 Average: all 69,90 0,01 15,31 0,22 0,00 0,70 0,00 0,00 0,00 13,85
128
129 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
130 Average: all 70,22 0,02 12,99 0,01 0,00 2,01 0,00 0,00 0,00 14,74

```

Listing D.9: VNF Publication CPU Results

D.10 Seller VNF Withdraw - CPU Results

```

1 VNF Withdraw - Basic with Cirros:
2
3 vmstat -S m l
4 procs -----memory----- --swap-- --io-- --system-- --cpu-----
5 r b swpd free buff cache si so bi bo in cs us sy id wa st
6 0 0 0 3780 276 1492 0 0 46 26 162 476 6 1 92 0 0
7 2 0 0 3776 276 1496 0 0 0 12 879 4575 14 7 79 0 0
8 4 0 0 3763 276 1508 0 0 0 52 1353 4120 19 7 74 0 0
9 1 0 0 3747 276 1525 0 0 0 0 1071 3764 17 5 78 0 0
10 0 0 0 3779 276 1492 0 0 0 0 1190 2513 14 5 81 0 0
11 procs -----memory----- --swap-- --io-- --system-- --cpu-----
12 r b swpd free buff cache si so bi bo in cs us sy id wa st
13 0 0 0 3738 276 1533 0 0 46 26 163 482 7 1 92 0 0
14 0 0 0 3738 276 1533 0 0 0 4 514 2133 8 3 90 0 0
15 0 0 0 3734 276 1537 0 0 0 0 1164 4056 22 6 72 0 0
16 4 0 0 3734 276 1537 0 0 0 0 528 1868 9 3 88 0 0
17 1 0 0 3703 276 1561 0 0 0 584 1615 4813 30 9 61 0 0
18 2 0 0 3735 276 1533 0 0 0 6080 1135 2738 25 3 72 0 0
19 procs -----memory----- --swap-- --io-- --system-- --cpu-----
20 r b swpd free buff cache si so bi bo in cs us sy id wa st
21 1 0 0 3719 276 1547 0 0 46 28 165 488 7 1 92 0 0
22 2 0 0 3710 276 1555 0 0 0 0 877 2838 15 4 82 0 0
23 0 0 0 3686 276 1579 0 0 0 52 1075 7360 30 9 60 0 0
24 2 0 0 3717 276 1549 0 0 0 0 1430 4233 23 8 69 0 0
25 0 0 0 3719 276 1546 0 0 0 0 495 1589 10 2 88 0 0
26 procs -----memory----- --swap-- --io-- --system-- --cpu-----
27 r b swpd free buff cache si so bi bo in cs us sy id wa st
28 0 0 0 3698 276 1574 0 0 46 28 167 494 7 2 92 0 0
29 1 0 0 3698 276 1574 0 0 0 0 517 1810 8 3 89 0 0
30 0 0 0 3696 276 1576 0 0 0 0 827 4175 13 4 83 0 0
31 0 0 0 3677 276 1594 0 0 0 52 1486 6372 30 9 61 0 0
32 5 0 0 3698 276 1575 0 0 0 8 1147 4332 15 7 78 0 0
33 procs -----memory----- --swap-- --io-- --system-- --cpu-----
34 r b swpd free buff cache si so bi bo in cs us sy id wa st
35 2 0 0 3710 276 1560 0 0 46 28 168 500 7 2 92 0 0
36 0 0 0 3706 276 1564 0 0 0 0 1080 5312 20 5 74 0 0
37 0 0 0 3708 276 1562 0 0 0 0 905 1936 11 5 85 0 0
38 0 0 0 3681 276 1588 0 0 0 72 1443 5162 29 9 62 0 0
39 0 0 0 3710 276 1560 0 0 0 0 1307 3317 20 7 73 0 0
40
41 VNF Withdraw - Premium with Cirros:
42
43 vmstat -S m l
44 procs -----memory----- --swap-- --io-- --system-- --cpu-----
45 r b swpd free buff cache si so bi bo in cs us sy id wa st
46 1 0 0 3807 274 1390 0 0 54 23 154 449 6 1 93 0 0
47 0 0 0 3807 274 1390 0 0 0 0 565 1636 9 3 88 0 0
48 1 0 0 3807 274 1390 0 0 0 0 578 1094 9 1 90 0 0
49 2 0 0 3806 274 1392 0 0 0 0 879 2116 13 3 84 0 0
50 1 0 0 3807 274 1390 0 0 0 0 562 2151 9 3 89 0 0
51 4 0 0 3760 274 1437 0 0 0 0 1274 5913 33 9 58 0 0
52 1 0 0 3753 274 1444 0 0 0 60 1014 3983 17 5 78 0 0
53 1 0 0 3807 274 1390 0 0 0 0 403 1479 6 3 91 0 0
54 1 0 0 3808 274 1390 0 0 0 0 1198 2905 20 5 75 0 0
55 procs -----memory----- --swap-- --io-- --system-- --cpu-----
56 r b swpd free buff cache si so bi bo in cs us sy id wa st
57 2 0 0 3781 274 1417 0 0 53 24 156 457 6 1 92 0 0
58 1 0 0 3781 274 1417 0 0 0 0 424 1883 7 2 91 0 0
59 0 0 0 3781 274 1417 0 0 0 0 449 1213 8 1 91 0 0
60 1 0 0 3780 274 1417 0 0 0 36 760 1378 8 3 89 0 0
61 0 0 0 3778 274 1419 0 0 0 20 847 1695 9 3 88 0 0
62 0 0 0 3733 274 1464 0 0 0 60 1833 7463 41 12 47 0 0
63 0 0 0 3779 274 1417 0 0 0 0 701 1114 6 4 90 0 0
64 0 0 0 3779 274 1417 0 0 0 0 286 755 2 1 97 0 0
65 0 0 0 3779 274 1417 0 0 0 0 499 1558 6 4 90 0 0
66 procs -----memory----- --swap-- --io-- --system-- --cpu-----
67 r b swpd free buff cache si so bi bo in cs us sy id wa st
68 3 0 0 3840 275 1439 0 0 49 23 156 455 6 1 92 0 0
69 2 0 0 3840 275 1439 0 0 0 0 497 1335 4 3 93 0 0
70 1 0 0 3837 275 1441 0 0 0 0 1636 3924 16 4 80 0 0
71 1 0 0 3789 275 1489 0 0 0 52 1693 6708 36 15 49 0 0
72 0 0 0 3840 275 1437 0 0 0 0 730 1323 6 2 92 0 0
73 0 0 0 3840 275 1437 0 0 0 0 645 1414 5 3 92 0 0

```

```

74 2 0 0 3840 275 1437 0 0 0 0 935 2251 9 4 87 0 0
75 0 0 0 3841 275 1437 0 0 0 0 910 3582 16 4 79 0 0
76 procs -----memory-----swap-----io-----system-----cpu-----
77 r b swpd free buff cache si so bi bo in cs us sy id wa st
78 3 0 0 3812 275 1465 0 0 48 24 158 461 6 1 92 0 0
79 1 0 0 3812 275 1465 0 0 0 0 342 1807 6 2 92 0 0
80 0 0 0 3812 275 1465 0 0 0 0 547 1088 8 2 90 0 0
81 3 0 0 3812 275 1465 0 0 0 24 464 1519 8 2 90 0 0
82 0 0 0 3810 275 1467 0 0 0 0 1477 4163 25 6 69 0 0
83 0 0 0 3743 275 1533 0 0 0 52 1730 7691 42 14 45 0 0
84 0 0 0 3811 275 1465 0 0 0 0 1225 3719 19 4 77 0 0
85 procs -----memory-----swap-----io-----system-----cpu-----
86 r b swpd free buff cache si so bi bo in cs us sy id wa st
87 2 0 0 3836 275 1439 0 0 47 25 159 467 6 1 92 0 0
88 0 0 0 3836 275 1439 0 0 0 0 645 1681 10 3 87 0 0
89 0 0 0 3836 275 1439 0 0 0 0 494 1211 10 3 87 0 0
90 3 0 0 3836 275 1439 0 0 0 0 476 1880 10 4 87 0 0
91 1 0 0 3834 275 1441 0 0 0 0 1145 2498 17 4 79 0 0
92 2 0 0 3777 275 1498 0 0 0 88 1678 7758 44 13 44 0 0
93 2 0 0 3838 275 1438 0 0 0 0 978 2556 12 7 80 0 0
94
95 VNF Withdraw - Basic with Ubuntu:
96
97 mpstat -P ALL 1
98 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
99 Average: all 16,75 0,00 5,15 0,00 0,00 0,00 0,00 0,00 0,00 78,09
100
101 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
102 Average: all 18,76 0,00 5,16 0,00 0,00 0,17 0,00 0,00 0,00 75,90
103
104 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
105 Average: all 17,61 0,00 4,62 0,00 0,00 0,17 0,00 0,00 0,00 77,61
106
107 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
108 Average: all 13,68 0,00 3,47 0,06 0,00 0,00 0,00 0,00 0,00 82,79
109
110 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
111 Average: all 16,45 0,00 4,73 0,02 0,00 0,00 0,00 0,00 0,00 78,79
112
113 VNF Withdraw - Premium with Ubuntu:
114
115 mpstat -P ALL 1
116 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
117 Average: all 11,51 0,10 3,71 0,03 0,00 0,03 0,00 0,00 0,00 84,60
118
119 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
120 Average: all 15,82 0,06 4,91 0,00 0,00 0,00 0,00 0,00 0,00 79,21
121
122 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
123 Average: all 13,22 0,07 4,12 0,00 0,00 0,00 0,00 0,00 0,00 82,58
124
125 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
126 Average: all 12,93 0,00 3,99 0,01 0,00 0,01 0,00 0,00 0,00 83,06
127
128 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
129 Average: all 11,70 0,09 3,83 0,02 0,00 0,02 0,00 0,00 0,00 84,33

```

Listing D.10: VNF Withdraw (Cirros) CPU and RAM Results with seller

D.11 Customer Actions - CPU Results

```

1 Buy VNF - Basic:
2
3 mpstat -P ALL 1
4 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
5 Average: all 66,84 0,05 13,45 0,14 0,00 0,16 0,00 0,00 0,00 19,37
6
7 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
8 Average: all 63,24 0,09 13,53 0,03 0,00 0,00 0,00 0,00 0,00 23,11
9
10 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
11 Average: all 70,24 0,10 13,54 0,00 0,00 0,07 0,00 0,00 0,00 16,05
12
13 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle

```

```

14 Average:      all    63,87    0,07    13,02    0,07    0,00    0,13    0,00    0,00    0,00    22,85
15
16 Average:      CPU    %usr    %nice    %sys %iowait    %irq    %soft    %steal    %guest    %gnice    %idle
17 Average:      all    66,41    0,09    13,48    0,00    0,00    0,12    0,00    0,00    0,00    19,89
18
19 Buy VNF – Premium:
20
21 mpstat -P ALL 5
22 Average:      CPU    %usr    %nice    %sys %iowait    %irq    %soft    %steal    %guest    %gnice    %idle
23 Average:      all    74,64    0,05    14,93    0,00    0,00    0,08    0,00    0,00    0,00    10,29
24
25 Average:      CPU    %usr    %nice    %sys %iowait    %irq    %soft    %steal    %guest    %gnice    %idle
26 Average:      all    74,66    0,08    13,72    0,00    0,00    0,03    0,00    0,00    0,00    11,51
27
28 Average:      CPU    %usr    %nice    %sys %iowait    %irq    %soft    %steal    %guest    %gnice    %idle
29 Average:      all    73,20    0,07    13,92    0,00    0,00    0,00    0,00    0,00    0,00    12,81
30
31 Average:      CPU    %usr    %nice    %sys %iowait    %irq    %soft    %steal    %guest    %gnice    %idle
32 Average:      all    70,02    0,08    12,77    0,05    0,00    0,08    0,00    0,00    0,00    17,00
33
34 Average:      CPU    %usr    %nice    %sys %iowait    %irq    %soft    %steal    %guest    %gnice    %idle
35 Average:      all    72,74    0,11    13,96    0,04    0,00    0,04    0,00    0,00    0,00    13,12
36
37 Delete VNF – Cirros:
38
39 mpstat -P ALL 1
40 Average:      CPU    %usr    %nice    %sys %iowait    %irq    %soft    %steal    %guest    %gnice    %idle
41 Average:      all    52,80    0,35    11,36    0,00    0,00    0,52    0,00    0,00    0,00    34,97
42
43 Average:      CPU    %usr    %nice    %sys %iowait    %irq    %soft    %steal    %guest    %gnice    %idle
44 Average:      all    50,00    0,52    12,92    0,13    0,00    0,13    0,00    0,00    0,00    36,29
45
46 Average:      CPU    %usr    %nice    %sys %iowait    %irq    %soft    %steal    %guest    %gnice    %idle
47 Average:      all    43,11    0,31    10,98    0,00    0,00    0,31    0,00    0,00    0,00    45,28
48
49 Average:      CPU    %usr    %nice    %sys %iowait    %irq    %soft    %steal    %guest    %gnice    %idle
50 Average:      all    59,87    0,11    14,26    0,00    0,00    0,00    0,00    0,00    0,00    25,77
51
52 Average:      CPU    %usr    %nice    %sys %iowait    %irq    %soft    %steal    %guest    %gnice    %idle
53 Average:      all    47,67    0,10    11,27    0,10    0,00    0,21    0,00    0,00    0,00    40,64
54
55 Delete VNF – Ubuntu:
56
57 mpstat -P ALL 1
58 Average:      CPU    %usr    %nice    %sys %iowait    %irq    %soft    %steal    %guest    %gnice    %idle
59 Average:      all    54,52    0,17    10,17    0,00    0,00    0,17    0,00    0,00    0,00    34,96
60
61 Average:      CPU    %usr    %nice    %sys %iowait    %irq    %soft    %steal    %guest    %gnice    %idle
62 Average:      all    58,31    0,31    13,17    0,00    0,00    0,10    0,00    0,00    0,00    28,11
63
64 Average:      CPU    %usr    %nice    %sys %iowait    %irq    %soft    %steal    %guest    %gnice    %idle
65 Average:      all    54,83    0,10    13,60    0,10    0,00    0,42    0,00    0,00    0,00    30,94
66
67 Average:      CPU    %usr    %nice    %sys %iowait    %irq    %soft    %steal    %guest    %gnice    %idle
68 Average:      all    44,63    0,26    11,35    0,09    0,00    0,00    0,00    0,00    0,00    43,67
69
70 Average:      CPU    %usr    %nice    %sys %iowait    %irq    %soft    %steal    %guest    %gnice    %idle
71 Average:      all    46,89    0,35    9,34    0,00    0,00    0,17    0,00    0,00    0,00    43,25

```

Listing D.11: VNF Purchase and VNF Withdraw CPU Results with Customer

D.12 OpenStack with VNF Marketplace - CPU Results

```

1 VNF Publication (Seller) – Cirros
2
3 vmstat -S m 1
4 procs -----memory----- --swap-- --io-- --system-- -----cpu-----
5  r  b  swpd  free  buff  cache  si  so  bi  bo  in  cs  us  sy  id  wa  st
6  0  0    0   404   213   2089    0  0  34   228   815   494  14  5  78  3  0
7  21  0    0   404   213   2089    0  0  0    0  1745  3869  7  5  88  0  0
8  2  0    0   404   213   2089    0  0  0    24  1726  4073  11  3  86  1  0
9  0  0    0   365   213   2103    0  0  0   412  4459  6555  26  11  61  2  0
10 18  0    0   365   213   2103    0  0  0    68  1726  4178  11  5  84  1  0
11 procs -----memory----- --swap-- --io-- --system-- -----cpu-----

```

```

12  r b swpd free buff cache si so bi bo in cs us sy id wa st
13  1 0 0 372 213 2095 0 0 33 226 818 530 14 5 78 3 0
14  1 1 0 372 213 2095 0 0 0 132 1374 3639 10 5 84 2 0
15  1 0 0 372 213 2095 0 0 0 192 1435 3848 11 3 85 1 0
16  2 0 0 358 213 2109 0 0 0 304 4597 6558 36 9 53 2 0
17  9 0 0 358 213 2109 0 0 0 80 1498 4117 8 3 88 1 0
18  0 0 0 358 213 2109 0 0 0 88 1534 4119 8 4 87 1 0
19  13 1 0 358 213 2109 0 0 0 752 1517 4087 8 4 86 2 0
20  1 0 0 358 213 2109 0 0 0 788 1543 4148 9 4 85 1 0
21  procs -----memory-----swap-----io-----system-----cpu-----
22  r b swpd free buff cache si so bi bo in cs us sy id wa st
23  4 0 0 366 213 2099 0 0 32 220 825 625 14 5 79 3 0
24  3 0 0 366 213 2100 0 0 0 36 1672 4307 10 4 86 1 0
25  5 0 0 366 213 2100 0 0 0 216 2134 5137 31 3 66 1 0
26  8 0 0 351 213 2113 0 0 0 144 4691 6630 24 9 66 1 0
27  0 0 0 351 213 2113 0 0 0 136 1972 4608 6 4 88 3 0
28  0 0 0 351 213 2113 0 0 0 2052 1764 4520 7 5 86 2 0
29  1 0 0 351 213 2113 0 0 0 72 2115 5085 10 6 83 1 0
30  0 0 0 350 213 2113 0 0 0 248 1855 4616 7 3 89 1 0
31  procs -----memory-----swap-----io-----system-----cpu-----
32  r b swpd free buff cache si so bi bo in cs us sy id wa st
33  9 0 0 364 213 2101 0 0 31 219 827 647 14 5 79 3 0
34  2 0 0 363 213 2101 0 0 0 24 2040 4760 10 4 78 8 0
35  0 0 0 363 213 2101 0 0 0 112 2293 5143 15 4 81 1 0
36  1 0 0 349 213 2114 0 0 0 248 4058 6709 28 12 59 1 0
37  0 0 0 349 213 2114 0 0 0 0 2005 4804 9 3 89 0 0
38  0 0 0 349 213 2114 0 0 0 1944 2086 5060 9 3 80 7 0
39  3 0 0 349 213 2114 0 0 0 84 1910 5045 10 3 71 15 0
40  procs -----memory-----swap-----io-----system-----cpu-----
41  r b swpd free buff cache si so bi bo in cs us sy id wa st
42  10 0 0 360 213 2102 0 0 31 218 829 666 14 5 79 3 0
43  2 0 0 360 213 2101 0 0 0 56 1604 4474 5 3 92 1 0
44  4 0 0 360 213 2101 0 0 0 152 2319 5644 19 6 74 1 0
45  5 0 0 347 213 2115 0 0 0 196 4108 6857 17 8 74 1 0
46  2 0 0 347 213 2115 0 0 0 0 2113 4942 5 3 93 0 0
47  10 0 0 347 213 2115 0 0 0 40 2245 5244 9 2 88 1 0
48  2 0 0 347 213 2115 0 0 0 44 2255 5289 6 3 91 0 0
49  6 0 0 347 213 2115 0 0 0 40 2180 5148 7 3 90 1 0
50
51 VNF Withdraw (Seller) - Cirros
52
53 vmstat -S m 1
54 procs -----memory-----swap-----io-----system-----cpu-----
55  r b swpd free buff cache si so bi bo in cs us sy id wa st
56  0 0 0 360 213 2108 0 0 34 227 817 512 14 5 78 3 0
57  2 0 0 360 213 2108 0 0 0 40 1478 3928 10 2 88 1 0
58  0 0 0 360 213 2108 0 0 0 20 1790 4047 6 3 91 0 0
59  1 0 0 373 213 2095 0 0 0 400 2012 4749 15 6 77 2 0
60  0 0 0 373 213 2094 0 0 0 124 1820 4336 9 2 89 1 0
61  0 0 0 373 213 2094 0 0 0 40 1797 4035 7 2 91 1 0
62  1 0 0 373 213 2094 0 0 0 40 1722 4046 6 2 92 0 0
63  0 0 0 373 213 2094 0 0 0 0 1763 4012 6 3 90 0 0
64  2 0 0 373 213 2094 0 0 0 1476 1639 3970 4 1 93 2 0
65  procs -----memory-----swap-----io-----system-----cpu-----
66  r b swpd free buff cache si so bi bo in cs us sy id wa st
67  1 0 0 358 213 2109 0 0 33 226 818 539 14 5 78 3 0
68  1 0 0 357 213 2109 0 0 0 76 1986 4467 7 4 87 2 0
69  2 0 0 357 213 2109 0 0 0 0 1847 4438 11 4 85 0 0
70  6 0 0 370 213 2096 0 0 0 244 2512 5747 19 4 74 2 0
71  1 0 0 351 213 2096 0 0 0 236 2519 5941 30 11 59 0 0
72  procs -----memory-----swap-----io-----system-----cpu-----
73  r b swpd free buff cache si so bi bo in cs us sy id wa st
74  3 0 0 350 213 2113 0 0 32 220 825 632 14 5 79 3 0
75  0 1 0 350 213 2113 0 0 0 72 1690 4385 6 2 82 10 0
76  1 0 0 350 213 2113 0 0 0 616 1717 4368 6 2 87 5 0
77  1 0 0 350 213 2113 0 0 0 44 1871 4616 7 2 91 1 0
78  2 0 0 350 213 2113 0 0 0 268 1952 4645 8 2 89 1 0
79  4 0 0 350 213 2100 0 0 0 300 2321 5769 22 5 72 1 0
80  1 0 0 346 213 2100 0 0 0 1572 3048 8150 35 18 44 3 0
81  procs -----memory-----swap-----io-----system-----cpu-----
82  r b swpd free buff cache si so bi bo in cs us sy id wa st
83  1 0 0 347 213 2115 0 0 31 219 828 654 14 5 79 3 0
84  0 0 0 347 213 2115 0 0 0 228 1810 4767 7 2 90 1 0
85  1 0 0 347 213 2115 0 0 0 48 1918 4703 8 2 90 0 0
86  2 0 0 360 213 2101 0 0 0 372 2668 6221 16 7 75 3 0
87  3 0 0 360 213 2101 0 0 0 116 1722 4585 8 3 83 6 0
88  2 0 0 360 213 2101 0 0 0 4120 1958 5037 6 2 82 11 0

```

89	1	0	0	360	213	2101	0	0	0	40	1973	4814	6	2	92	1	0
90	procs	memory				swap			io			system			cpu		
91	r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa	st
92	2	0	0	345	213	2115	0	0	31	218	830	673	14	5	79	3	0
93	0	0	0	345	213	2115	0	0	0	56	2073	5193	5	3	91	1	0
94	0	0	0	345	213	2115	0	0	0	0	2196	5235	5	3	91	0	0
95	1	0	0	358	213	2102	0	0	0	388	2593	5907	17	4	78	1	0
96	3	0	0	360	213	2102	0	0	0	64	2024	4933	5	3	91	1	0
97	1	0	0	360	213	2102	0	0	0	128	1997	5059	8	3	89	1	0
98	6	0	0	360	213	2102	0	0	0	104	2117	4988	6	3	90	1	0
99																	
100	VNF Publication (Seller) - Ubuntu																
101																	
102	mpstat -P ALL 5																
103	Average:			CPU	%usr	%nice	%sys	%iowait		%irq	%soft	%steal	%guest	%gnice	%idle		
104	Average:			all	11.41	0.00	5.48	56.35		0.00	1.55	0.00	0.00	0.00	0.00	25.20	
105																	
106	Average:			CPU	%usr	%nice	%sys	%iowait		%irq	%soft	%steal	%guest	%gnice	%idle		
107	Average:			all	10.64	0.00	4.49	71.85		0.00	0.84	0.00	0.00	0.00	0.00	12.19	
108																	
109	Average:			CPU	%usr	%nice	%sys	%iowait		%irq	%soft	%steal	%guest	%gnice	%idle		
110	Average:			all	10.98	0.00	4.67	68.13		0.00	1.02	0.00	0.00	0.00	0.00	15.19	
111																	
112	Average:			CPU	%usr	%nice	%sys	%iowait		%irq	%soft	%steal	%guest	%gnice	%idle		
113	Average:			all	11.63	0.00	5.68	55.88		0.00	1.57	0.00	0.00	0.00	0.00	13.04	
114																	
115	Average:			CPU	%usr	%nice	%sys	%iowait		%irq	%soft	%steal	%guest	%gnice	%idle		
116	Average:			all	15.73	0.00	5.31	59.12		0.00	1.12	0.00	0.00	0.00	0.00	18.71	
117																	
118	VNF Withdraw (Seller) - Ubuntu																
119																	
120	mpstat -P ALL 1																
121	Average:			CPU	%usr	%nice	%sys	%iowait		%irq	%soft	%steal	%guest	%gnice	%idle		
122	Average:			all	11.10	0.00	4.17	1.42		0.00	0.89	0.00	0.00	0.00	0.00	82.42	
123																	
124	Average:			CPU	%usr	%nice	%sys	%iowait		%irq	%soft	%steal	%guest	%gnice	%idle		
125	Average:			all	11.21	0.00	4.61	2.45		0.00	0.47	0.00	0.00	0.00	0.00	81.25	
126																	
127	Average:			CPU	%usr	%nice	%sys	%iowait		%irq	%soft	%steal	%guest	%gnice	%idle		
128	Average:			all	11.03	0.00	4.32	1.34		0.00	0.90	0.00	0.00	0.00	0.00	82.40	
129																	
130	Average:			CPU	%usr	%nice	%sys	%iowait		%irq	%soft	%steal	%guest	%gnice	%idle		
131	Average:			all	11.15	0.00	4.21	1.95		0.00	0.78	0.00	0.00	0.00	0.00	81.91	
132																	
133	Average:			CPU	%usr	%nice	%sys	%iowait		%irq	%soft	%steal	%guest	%gnice	%idle		
134	Average:			all	11.09	0.00	4.20	2.10		0.00	0.61	0.00	0.00	0.00	0.00	81.98	
135																	
136	VNF Purchase (Customer) - Cirros																
137																	
138	mpstat -P ALL 5																
139	Average:			CPU	%usr	%nice	%sys	%iowait		%irq	%soft	%steal	%guest	%gnice	%idle		
140	Average:			all	19.63	0.00	5.77	2.87		0.00	0.80	0.00	0.00	0.00	0.00	70.93	
141																	
142	Average:			CPU	%usr	%nice	%sys	%iowait		%irq	%soft	%steal	%guest	%gnice	%idle		
143	Average:			all	17.96	0.00	4.30	1.64		0.00	0.79	0.00	0.00	0.00	0.00	75.31	
144																	
145	Average:			CPU	%usr	%nice	%sys	%iowait		%irq	%soft	%steal	%guest	%gnice	%idle		
146	Average:			all	23.13	0.00	6.62	2.72		0.00	1.09	0.00	0.00	0.00	0.00	66.44	
147																	
148	Average:			CPU	%usr	%nice	%sys	%iowait		%irq	%soft	%steal	%guest	%gnice	%idle		
149	Average:			all	19.73	0.00	5.95	1.54		0.00	0.69	0.00	0.00	0.00	0.00	72.09	
150																	
151	Average:			CPU	%usr	%nice	%sys	%iowait		%irq	%soft	%steal	%guest	%gnice	%idle		
152	Average:			all	19.07	0.00	5.00	1.54		0.00	1.17	0.00	0.00	0.00	0.00	73.22	
153																	
154	VNF Withdraw (Customer) - Cirros																
155																	
156	mpstat -P ALL 1																
157	Average:			CPU	%usr	%nice	%sys	%iowait		%irq	%soft	%steal	%guest	%gnice	%idle		
158	Average:			all	35.69	0.00	9.71	8.96		0.00	1.36	0.00	0.00	0.00	0.00	44.28	
159																	
160	Average:			CPU	%usr	%nice	%sys	%iowait		%irq	%soft	%steal	%guest	%gnice	%idle		
161	Average:			all	39.10	0.00	12.18	3.98		0.00	0.98	0.00	0.00	0.00	0.00	43.76	
162																	
163	Average:			CPU	%usr	%nice	%sys	%iowait		%irq	%soft	%steal	%guest	%gnice	%idle		
164	Average:			all	32.00	0.00	7.35	3.53		0.00	1.41	0.00	0.00	0.00	0.00	55.71	
165																	

```

166 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
167 Average: all 38.82 0.00 7.76 2.57 0.00 1.64 0.00 0.00 0.00 49.21
168
169 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
170 Average: all 47.57 0.00 11.57 1.50 0.00 1.04 0.00 0.00 0.00 38.31
171
172 VNF Purchase (Customer) - Ubuntu
173
174 mpstat -P ALL 5
175 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
176 Average: all 30.17 0.00 5.31 3.64 0.00 0.92 0.00 0.00 0.00 59.96
177
178 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
179 Average: all 66.09 0.08 13.49 0.08 0.00 0.03 0.00 0.00 0.00 20.24
180
181 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
182 Average: all 25.51 0.00 6.15 3.44 0.00 0.93 0.00 0.00 0.00 63.97
183
184 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
185 Average: all 13.97 0.00 3.99 2.22 0.00 0.68 0.00 0.00 0.00 79.15
186
187 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
188 Average: all 27.58 0.00 6.34 2.42 0.00 1.05 0.00 0.00 0.00 62.61
189
190 VNF Withdraw (Customer) - Ubuntu
191
192 mpstat -P ALL 1
193 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
194 Average: all 66.95 0.00 7.08 2.80 0.00 0.86 0.00 0.00 0.00 22.32
195
196 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
197 Average: all 60.13 0.00 6.29 2.24 0.00 1.16 0.00 0.00 0.00 30.17
198
199 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
200 Average: all 71.03 0.00 5.84 3.02 0.00 0.64 0.00 0.00 0.00 19.46
201
202 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
203 Average: all 65.30 0.00 7.47 1.13 0.00 0.97 0.00 0.00 0.00 25.13
204
205 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
206 Average: all 68.97 0.00 5.27 1.04 0.00 0.78 0.00 0.00 0.00 23.94

```

Listing D.12: Stakeholders Actions with CPU Results

D.13 OpenStack - CPU and RAM Results

```

1 0 Cirros VNFs:
2 free -m
3
4 Mem: total used free shared buff/cache available
5 Swap: 7983 5104 2595 38 283 2563
6
7 mpstat -P ALL 1 20
8 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
9 Average: all 12.33 0.00 5.41 1.05 0.00 0.46 0.00 0.00 0.00 80.76
10
11
12 10 Cirros VNFs:
13 free -m
14
15 Mem: total used free shared buff/cache available
16 Swap: 7983 7656 152 16 174 75
17
18 mpstat -P ALL 1 20
19 Average: CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle
20 Average: all 36.40 0.00 10.98 13.77 0.00 0.95 0.00 0.00 0.00 37.90
21
22
23 0 Ubuntu VNFs:
24 free -m
25
26 Mem: total used free shared buff/cache available
27 Swap: 7983 5210 2366 71 406 2412
28 Swap: 8188 793 7395

```



```

29
30 mpstat -P ALL 1 20
31 Average:  CPU    %usr    %nice    %sys %iowait    %irq    %soft    %steal    %guest    %gnice    %idle
32 Average:   all    14.44     0.00     5.88     1.40     0.00     0.62     0.00     0.00     0.00    77.66
33
34
35 3 Ubuntu VNFs:
36
37 free -m
38
39          total        used        free      shared  buff/cache   available
40 Mem:           7983         5527         2061           64           394           2102
41 Swap:           8188           803         7385
42
43 mpstat -P ALL 1 20
44 Average:  CPU    %usr    %nice    %sys %iowait    %irq    %soft    %steal    %guest    %gnice    %idle
45 Average:   all    94.49     0.00     5.11     0.00     0.00     0.35     0.00     0.00     0.00     0.05
46
47 4 Ubuntu VNFs Log:
48 PENDING_CREATE CREATE VNF UUID assigned.
49 ERROR CREATE Infra Instance ID created: dea03cf3-6df9-46bf-87dc-994e78a861f2 and Mgmt URL set: None

```

Listing D.13: Number of VNFs with CPU and RAM Results

D.14 VNF Marketplace installation times

Command	VM1	VM2	VM3	VM4	VM5
sudo apt-get update	7.484s	4.981s	5.430s	5.819s	6.219s
sudo apt-get install (python libraries)	6.056s	6.001s	9.616s	6.012s	6.105s
pip install virtualenv	4.175s	4.252s	3.403s	4.083s	3.728s
virtualenv env	4.490s	3.332s	2.600s	3.371s	4.001s
source env/bin/activate	0.003s	0.003s	0.002s	0.003s	0.003s
pip install -r require- ments.txt	63.110s	53.233s	55.880s	62.339s	57.901s
python manage.py cre- atesuperuser	16.234s	18.116s	10.431s	12.810s	11.112s
python manage.py run- server (IP):8000	2.993s	2.664s	2.783s	2.873s	2.671s

Table D.1: VNF Marketplace detailed installation time measurements

D.15 SLA Management installation times

Installation Phase	VM1	VM2	VM3	VM4	VM5
Install Java JDK	107.100s	112.947s	129.367s	101.423s	123.098s
Install MySQL	31.791s	19.908s	43.293s	26.665s	38.328s
Install Maven	18.881s	19.039s	19.286s	18.604s	18.513s
Create MySQL Database	26.891s	30.012s	25.253s	25.724s	23.765s
Compiling	180.932s	201.810s	166.634s	220.808s	171.162s
Run Server	23.281s	26.089s	20.333s	39.463s	22.112s

Table D.2: SLA Management installation time measurements

D.16 Time Results

Action	Test 1	Test 2	Test 3	Test 4	Test 5
Loading Static Files	0,591029s	0,693091s	0,583872s	0,621092s	0,653844s
Register	0,072312s	0,062901s	0,059201s	0,058391s	0,070192s
Log in	0,15729s	0,14300s	0,16100s	0,15200s	0,15900s
Settings	0,026317s	0,012561s	0,011819s	0,023s	0,018s
Get Seller's VNFs	0,029764s	0,0229s	0,0207s	0,02671s	0,0289s
Get Customer's VNFs	0,007697s	0,007289s	0,006721s	0,007389s	0,007192s
Get VNF Catalog	0,0337928s	0,031829s	0,035943s	0,032431s	0,035893s
Seller Create VNF in Basic (Cirros)	0,228551s	0,111884s	0,184543s	0,21s	0,247s
Seller Create VNF in Basic (Ubuntu)	13,67492s	15,33731s	13,90100s	14,38219s	15,12312s
Seller Delete VNF in Basic (Cirros)	0,015015s	0,01557s	0,022991s	0,016235s	0,01671s
Seller Delete VNF in Basic (Ubuntu)	0,228551s	0,111884s	0,184543s	0,21s	0,247s

Seller Create VNF in Premium (Cirros)	2,239243s	2,692757s	1,932131s	2,594530s	1,863271s
Seller Create VNF in Premium (Ubuntu)	89,640077s	85,123214s	90,723814s	88,213210s	87,312321s
Seller Delete VNF in Premium (Cirros)	1,726237s	0,536780s	0,893123s	1,213820s	0,879012s
Seller Delete VNF in Premium (Ubuntu)	0,39545s	0,98381s	0,73821s	1,56217s	0,67812s
Customer Buy VNF in Basic (Cirros)	17,847636s	16,321720s	14,367218s	15,372180s	17,152632s
Customer Buy VNF in Basic (Ubuntu)	15,123636s	17,991020s	13,743811s	15,748381s	17,003192s
Customer Delete VNF in Basic (Cirros)	0,015454s	0,014513s	0,019843s	0,016322s	0,015981s
Customer Delete VNF in Basic (Ubuntu)	0,011920	0,014989	0,018773s	0,015113s	0,015761s
Customer Buy VNF in Premium (Cirros)	17,847636s	16,321720s	14,367218s	15,372180s	17,152632s
Customer Buy VNF in Premium (Ubuntu)	18,965085s	17,368721s	18,231730s	16,998312s	19,372123s
Customer Delete VNF in Premium (Cirros)	1,406080s	1,489410s	1,551601s	1,427612s	1,491832
Customer Delete VNF in Premium (Ubuntu)	1,489180s	1,416732s	1,479182s	1,518730s	1,499984s
Upload VNF image and VNFD into OpenStack	31,020130s	38,372130s	33,381912s	39,382112s	35,381928s

Customer Delete VNF in Premium (Ubuntu)	0,018763s	0,017833s	0,019821s	0,016152s	0,017642s
-----------------------------------------------	-----------	-----------	-----------	-----------	-----------

Table D.3: Time Measurements of Requests-Responses

References

- [1] Anton A. Huurdeman. *The Worldwide History of Telecommunications*. Wiley-IEEE Press, First edition, July 31 2003.
- [2] Raj Jain and Subharthi Paul. Network virtualization and software defined networking for cloud computing: a survey. *IEEE Communications Magazine*, vol.51, pages 24–31, November 11 2013.
- [3] ETSI GS NFV. NFV White Paper, October 2012. URL: http://portal.etsi.org/NFV/NFV_White_Paper.pdf.
- [4] Dawn Bushaus. NFV: Bridging the chasm. Technical report, TM Forum, February 29 2016.
- [5] Huawei. White Paper - Huawei Observation to NFV, 2014. URL: http://www.huawei.com/ilink/en/download/HW_399662.
- [6] Muhammad Abid Huseni Saboowala and Sudhir Modali. *Designing Networks and Services for the Cloud: Delivering business-grade cloud applications and services*. Cisco Press, First edition, May 16 2013.
- [7] ETSI GS NFV. NFV White Paper 2, October 2013. URL: http://portal.etsi.org/NFV/NFV_White_Paper2.pdf.
- [8] ETSI GS NFV. Network Virtualisation Functions (NFV); Virtualisation Requirements, October 2013. URL: http://www.etsi.org/deliver/etsi_gs/nfv/001_099/004/01.01.01_60/gs_nfv004v010101p.pdf.
- [9] ETSI GS NFV. Network Functions Virtualisation (NFV); Virtual Network Functions Architecture, December 2014. URL: http://www.etsi.org/deliver/etsi_gs/NFV-SWA/001_099/001/01.01.01_60/gs_NFV-SWA001v010101p.pdf.
- [10] ETSI GS NFV. Network Functions virtualisation (nfv); terminology for main concepts in nfv, December 2014.
- [11] ETSI GS NFV. Network Functions Virtualisation (NFV); Use Cases, October 2013. URL: http://www.etsi.org/deliver/etsi_gs/NFV/001_099/001/01.01.01_60/gs_NFV001v010101p.pdf.
- [12] ETSI GS NFV. Network Virtualisation Functions (NFV); Management and Orchestration, December 2014. URL: http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf.
- [13] Mehmet Ersue. ETSI NFV Management and Orchestration - An Overview. Technical report, IETF, 2013.

- [14] ETSI GS NFV. Network Virtualisation Functions (NFV); Architectural Framework, October 2013. URL: http://www.etsi.org/deliver/etsi_gs/nfv/001_099/002/01.01.01_60/gs_nfv002v010101p.pdf.
- [15] Openstack. Openstack Training Courses.
- [16] Cody Bunch Kevin Jackson and Egle Sigler. *OpenStack Cloud Computing Cookbook*. Packt Publishing, Third edition, August 2015.
- [17] OASIS TOSCA. Topology and Orchestration Specification for Cloud Applications (TOSCA) Primer Version 1.0, December 31 2013. URL: <http://docs.oasis-open.org/tosca/tosca-primer/v1.0/cnd01/tosca-primer-v1.0-cnd01.pdf>.
- [18] TM Forum. IG1133C Federated Catalogs for (Automated) Procurement – Orange Case Study, R15.0.1 Framework Exploratory Report, September 2015.
- [19] Malwarebytes Labs. State of Malware Report, 2017.
- [20] G. Xilouris, E. Trouva, F. Lobillo, J.M. Soares, J. Carapinha, M.J. McGrath, G. Gardikis, P. Paglierani, E. Pallis, L. Zuccaro, Y. Rebahi, A. Kourtis. T-NOVA: A Marketplace for Virtualized Network Functions. *2014, European Conference on Networks and Communications (EuCNC)*, June 23-26 2014.
- [21] J. Carapinha (Editor), A. Ramos, F. Lobillo (ATOS), T. Pliakas (CLDST), A. Pietrabissa, D. Macone, F. D. Priscoli, M. Panfili (CRAT), Y. Rebahi (Fraunhofer), J. F. Riera (i2CAT), M. McGrath (INTEL), P. Comi (Italtel), P. Papadimitriou (LUH), E. Trouva, G. Xilouris (NCSRD), A. Gamelas (PTInS), D. Christofi, G. Dimosthenous (PTL), G. Gardikis (SPH), A. Bourdena, E. Markakis, E. Pallis (TEIC), M. Sidibe (VIO), A. Cimmino (ZHAW). Deliverable D2.1: System Use Cases and Requirements Version 1.0, 2014.
- [22] Kourtis Akis. T-NOVA: Developing a platform for NFaaS. T-NOVA Presentation: fp7tnova.
- [23] G. Xilouris (NCSRD - Editor), E. Trouva (NCSRD), A. Ramos, F. Lobillo (ATOS), P. Neves, J. Bonnet, J. Monteiro, A. Gamelas (PTIN), B. Coffano, M. D. Girolamo (HP), J. F. Riera, E. Escalona (i2CAT), M. McGrath, V. Riccobene (INTEL), D. Christofi, G. Dimosthenous (PTL), G. Gardikis (SPH), M. Sidibè (VIO), P. Comi (ITALTEL), A. Bourdena, E. Markakis, E. Pallis (TEIC), P. Papadimitriou (LUH), A. Pietrabissa, D. Macone, F. D. Priscoli, M. Panfili (CRAT), Y. Rebahi (FRAUNHOFER), A. Cimmino, P. Harsh (ZHAW). Deliverable D2.22: Overall System Architecture and Interfaces Version 1.0, September 30 2015.
- [24] Oliver Spatscheck Yang Xu V. Gopalakrishnan, C. Wang and David Applegate. Toward High-Performance and Scalable Network Functions Virtualization. *IEEE Internet Computing, Volume: 20*, pages 10–20, December 12 2016.
- [25] N. Herbaut (Viotech - Editor), A. Ramos, J. Melián (ATOS), E. Figini, P. Comi (Italtel), Y. Rehabi (Fokus). Deliverable D5.1: Network Function Store Version 1.0, October 30 2015.
- [26] Paolo Comi (Italtel - Editor), A. Ramos, J. Melián (ATOS), T. Pliakas (CLDST), S. Hohberg, Y. Rebahi (Fraunhofer), M. D. Girolamo (HP), E. Figini, F. Andreotti (Italtel), A. Abujoda, P. Papadimitriou (LUH), E. Kafetzakis, G. Xilouris (NCSRD), J. Bonnet (PTIN), D. Christofi, G. Dimosthenous (PTL), A. Bourdena, E. Markakis, E. Pallis (TEIC), N. Herbaut, M. Sidibe (VIO). Deliverable D2.42: Specification of the Network Function framework and t-nova marketplace version 1.0, October 30 2015.

- [27] Dr. Piyush Harsh. Cyclops - Versatile Billing Engine (DI4R), 2016.