8-21-2015 12:00 AM

Elastic Highly Available Cloud Computing

Hassan Hawilo
The University of Western Ontario

Supervisor
Prof. Abdallah Shami
The University of Western Ontario

Graduate Program in Electrical and Computer Engineering
A thesis submitted in partial fulfillment of the requirements for the degree in Master of Engineering Science

ELASTIC HIGHLY AVAILABLE CLOUD COMPUTING

(Thesis format: Integrated Article)

by

Hassan Hawilo

Graduate Program in Electrical & Computer Engineering

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Engineering Science

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

Abstract

High availability and elasticity are two the cloud computing services technical features. Elasticity is a key feature of cloud computing where provisioning of resources is closely tied to the runtime demand. High availability assures that cloud applications are resilient to failures. Existing cloud solutions focus on providing both features at the level of the virtual resource through virtual machines by managing their restart, addition, and removal as needed. These existing solutions map applications to a specific design, which is not suitable for many applications especially virtualized telecommunication applications that are required to meet carrier grade standards. Carrier grade applications typically rely on the underlying platform to manage their availability by monitoring heartbeats, executing recoveries, and attempting repairs to bring the system back to normal. Migrating such applications to the cloud can be particularly challenging, especially if the elasticity policies target the application only, without considering the underlying platform contributing to its high availability (HA). In this thesis, a Network Function Virtualization (NFV) framework is introduced; the challenges and requirements of its use in mobile networks are discussed. In particular, an architecture for NFV framework entities in the virtual environment is proposed. In order to reduce signaling traffic congestion and achieve better performance, a criterion to bundle multiple functions of virtualized evolved packet-core in a single physical device or a group of adjacent devices is proposed. The analysis shows that the proposed grouping can reduce the network control traffic by 70 percent. Moreover, a comprehensive framework for the elasticity of highly available applications that considers the elastic deployment of the platform and the HA placement of the application's components is proposed. The approach is applied to an internet protocol multimedia subsystem (IMS) application and demonstrate how, within a matter of seconds, the IMS application can be scaled up while maintaining its HA status.

Keywords: mobile networking, network function virtualization, mobile core network, virtualized evolved packet core, cloud computing, high availability, cloud application elasticity, highly available cloud application, telecommunication cloud.

Co-authorship statement

This thesis contains the following manuscripts that have been accepted and published.

1) H. Hawilo, A. Kanso, A. Shami, “Towards an Elasticity Framework for Legacy Highly Available Applications in the Cloud” 10th IEEE SERVICES 2015.

2) H. Hawilo, A. Shami, M. Mirahmadi, R. Asal, "NFV: state of the art, challenges, and implementation in next generation mobile networks (vEPC)," Network, IEEE , vol. 28, no.6, pp.18,26, Nov.-Dec. 2014.

The following coauthors provided experimental and technical support for the studies listed above:

A. Shami provided technical expertise, opinion and perspective, based on his expertise and experience as a professor. He contributed to the work done in Chapter 2 and 3.

A. Kanso provided technical expertise, opinion and perspective, based on his expertise and experience as a researcher at Ericsson Company. He contributed to the work done in Chapter 3.

M. Mirahmadi provided technical expertise, opinion and perspective, based on his expertise and experience as a researcher at IBM Company. He contributed to the work done in Chapter 2.

R. Asal provided technical expertise, opinion and perspective, based on his expertise and experience as a researcher at British Telecom Company. He contributed to the work done in Chapter 2.

Acknowledgments

This thesis is the result of a hard work and very busy research program, which would not have been possible without the support of a many of people.

I am very thankful to Prof. A. Shami, who provided thorough and helpful support with a close guidance throughout my masters' research program. He has been my supervisor and friend.

I am very thankful to Dr. A. Kanso for providing his expertise that was essential for the success of this work. He always made sure to have a close consultation and support throughout my masters' research program.

Many thanks to my friends and colleagues for the words of support and help during my masters' research program.

To my family, particularly my parents, thank you for your love, support, and unwavering belief in me. Without you, I would not be the person I am today.

Above all, I would like to thank my wife Manar Jammal for her love and constant support, for all the late nights and early mornings, and for keeping me sane over the past few months (years) . Thank you for being my muse, editor, proofreader, and sounding board, but most of all, thank you for being my best friend.

This work is partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC-STPGP 447230) and Ericsson Research.

Table of Contents

Abstract i

Co-authorship statement ii

Acknowledgments iii

List of Figures viii

List of Tables x

List of Abbreviations xi

Chapter 1 1

1. Introduction 1

1.1 High Availability 1

1.1.1 Mean Time to Failure (MTTF) 2

1.1.2 Mean Time to Repair (MTTR) 3

1.1.3 Mean Time Between Failures (MTBF) 3

1.2 Achieving High Availability 4

1.2.1 Fault Tolerance 4

1.2.2 Redundancy 5

1.2.3 Logical Entities of Highly Available System 6

1.2.4 Redundancy Models by SAForum: 7

1.2.4.1 2N Redundancy Model 7

1.2.4.2 N+M Redundancy Model 8

1.2.4.3 N-way Redundancy Model 9

1.2.4.4 N-Way Active Redundancy Model 10

1.2.4.5 No-Redundancy Redundancy Model 10

1.3 HA with Virtualization 11

1.4 Elasticity in Cloud 12

1.5 Problem Formulation 13

1.6 Research Contributions 14

Bibliography 15

Chapter 2 18

2. NFV: State of the Art, Challenges and Implementation in Next Generation Mobile Networks (vEPC) 18

2.1 Introduction 18

2.2 Network Function Virtualization 19

2.2.1 Openness of Platforms 20

2.2.2 Scalability and Flexibility 21

2.2.3 Operation Performance Improvement 21

2.2.4 Improve Development Cycle 22

2.2.5 Reduced CAPEX and OPEX 22

2.3 NFV and SDN 23

2.4 NFV Framework 24

2.5 Proposed Placement of Framework Entities 27

2.6 NFV Ecosystem 29

2.7 NFV Challenges and Requirements 29

2.7.1 Security 29

2.7.2 Computing Performance 30

2.7.3 Interconnection of VNFs 31

2.7.4 Portability 32

2.7.5 Operation and Management 32

2.7.6 Co-existence with Legacy Networks 33

2.7.7 Carrier-Grade Service Assurance 33

2.8 Use Cases and Services 35

2.8.1 NFVI as a Service (NFVIaas) 35

2.8.2 VNF as a Service (VNFaaS) 35

2.8.3 Virtual Network Platform as a Service (VNPaaS) 35

2.8.4 Fixed Access Network Functions Virtualization 36

2.8.5 Content Delivery Networks Virtualization 36

2.8.6 Home Environment Virtualization 37

2.8.7 Mobile Network Virtualization 37

2.9 Virtualization of the Evolved Packet Core (EPC) 38

2.10 Grouping EPC Entities in the NFV Environment 39

2.10.1 Segment One 40

2.10.2 Segment Two 41

2.10.3 Segment Three 42

2.10.4 Segment Four 42

2.11 Quantitative Analysis 44

2.12 Chapter Contribution 47

Bibliography 47

Chapter 3 51

3. Towards an Elasticity Framework for Legacy Highly Available Applications in the Cloud 51

3.1 Introduction 51

3.2 High Availability Middleware and Scheduling 54

3.2.1 OpenSAF Cluster 56

3.2.2 High Availability Scheduling 58

3.3 Elasticity Framework 58

3.3.1 Application Design and Elasticity Requirement Specification 59

3.3.2 Elastic HA-Scheduling 60

3.3.2.1 Identifying the Constraints 61
3.3.2.2 Maximizing the Availability of the Application 63
3.3.2.3 Optimizing the Placement for Performance and Other Factors.. 63
3.3.3 Automated Elastic Multi-Level Deployment 65
3.3.3.1 Infrastructure Elasticity 65
3.3.3.2 Platform Elasticity 66
3.3.3.3 Application Elasticity 67
3.4 Framework Workflow and Implementation 68
3.5 Test-bed and Case Study 69
3.6 Literature Review 72
3.7 Chapter Contribution 74
Bibliography 74
Chapter 4 79
4. Conclusion and Future Work 79
4.1 Conclusion 79
4.2 Future Work 80
Curriculum Vitae 82

List of Figures

Fig. 1.1 Maximum allowable downtime for different availability levels [1]..... 2

Fig. 1.2 Availability in terms of MTTF, MTTR, and MTBF 2

Fig. 1.3 MTTR, MTTF, MTBF [1]. 3

Fig. 1.4 2N redundancy model [6]. 7

Fig. 1.5 N+1 Redundancy Model [6]. 8

Fig. 1.6 N-way redundancy model [6]. 9

Fig. 1.7 N-way active redundancy model [6]. 10

Fig. 1.8 Vertical Scaling vs Horizontal Scaling. 13

Fig. 2.1 Network function virtualization concept. 20

Fig. 2.2 Service provider revenues vs traffic [2]. 23

Fig. 2.3 NFV differs from SDN. 24

Fig. 2.4 Virtualization Layout 25

Fig. 2.5 The cloud services 26

Fig. 2.6 NFV framework 27

Fig. 2.7 NFV Framework Entities Proposed Placements. 28

Fig. 2.8 Networking in virtualization environment. 31

Fig. 2.9 Base station virtualization evolution. 38

Fig. 2.10 vEPC Entities Grouping. 41

Fig. 2.11 Sequence diagram for user equipment attachment process to LTE network. ... 45

Fig. 3.1 The different perspectives of the cloud levels. 52

Fig. 3.2 Application components deployed in different datacenters. 53

Fig. 3.3 Overview of the elasticity framework. 59

Fig. 3.4 Snapshot of the application description interface. 60

Fig. 3.5 The cloud infrastructure hierarchical overview. 61

Fig. 3.6 The orbital distance of a given component. 62

Fig. 3.7 HA elastic scheduling algorithm. 64

Fig. 3.8 Elasticity framework workflow. 68

Fig. 3.9 Installation duration results. 71

Fig. 3.10 Configuration duration results. 71

List of Tables

Table 2.1 NFV challenges and solutions. 34

Table 2.2 Grouping of EPC entities in NFV environment..... 44

Table 2.3 Signaling traffic before and after grouping..... 46

Table 2.4 Traffic profile and planning parameters [25]..... 46

List of Abbreviations

AIS	Application Interface Specification
AMF	Availability Management Framework
API	Application Program Interface
ATCA	Advanced Telecommunications Computing Architecture
CAPEX	Capital Expenditure
CDN	Content Distribution Network
CMS	Configuration Management System
COTS	Commercial-of-the-Shelf
CPU	Central Processing Unit
CSCF	Call Session Control Function
CSI	Component Service Instance
DC	Data Center
DevOps	Development and Operations
DPDK	Data Plane Development Kit
DPI	Deep Packet Investigation
EMS	Element management System
ETSI	European Telecommunications Standards Institute
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
GTP	GPRS Tunneling Protocol
HA	High Availability
HLR	Home Location Register

HSS	Home Subscriber Server
IaaS	Infrastructure as a Service
ICT	Information and Communication Technology
IMM	Information Model Management
IMS	IP multi-media Subsystem
Inc.	Incorporation
IPSec	Internet Protocol Security
IPTV	Internet Protocol Television
IT	Information Technology
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
LTE	Long-Term Evolution
M2M	Machine-to-Machine
MAC	Media Access Control
MDS	Message Distribution system
MME	Mobility Management Entity
MTBF	Mean Time Between Failure
MTTF	Mean Time to Fail
MTTR	Mean Time to Repair
NaaS	Network as a Service
NFV	Network Function Virtualization
NFVI	Network Function Virtualization Infrastructure
NIC	Network Interface Controller
OCS	Online Charging System
OFCS	Offline Charging System

OPEX	Operational Expenditure
OS	Operating System
PaaS	Platform as a Service
PCRF	Policy and Charging Rules Function
PDCP	Packet Data Convergence Protocol
PGW	Packet Gateway
QoE	Quality of Experience
QoS	Quality of Service
RLC	Radio Link Control
SaaS	Software as a Service
SAForum	Service Availability Forum
SCTP	Stream Control Transmission Protocol
SDN	Software Defined Networking
SG	Service Group
SGSN	Serving GPRS Support Node
SGW	Serving Gateway
SI	Service Instance
SMF	Software Management Framework
SR-IoV	Single Root I/O Virtualization
SU	Service Unit
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDR	Unified Data Repository
UE	User Equipment
UML	Unified Modeling Language

vEPC	Virtualized Evolved Packet Core
VM	Virtual Machine
VNF	Virtualized Network Function

Chapter 1

1. Introduction

With the rapid increase in the number of smart connected devices such as smart phone, tablets and wireless sensors, our society is greatly dependent on computer-based systems. These systems provide services that shaped our communities and how we interact in different field such as healthcare, financial services, and social networks. The need for providing these services continuously to the users is very critical in today's society. Moreover, users are expecting and eager to have these service accessible anytime and anywhere. In order to provide such services upon which people can rely, application service designers must have a clear view of all the potential causes that may bring down a system and the possible solutions to address these challenges. In particular, the costs of such solutions in terms of computing resources requirements must be known. Users Dependability on a service defines its quality of service attributes such as reliability and availability. For instance, reliability is defined as time to fail from an initial referenced instant, whereas availability is the probability of obtaining a service at an instance of time. Such services require complex computer systems with high level of availability, typically 99.999% (five nines) of the time, which amounts to slightly over five minutes of downtime over a year time interval of continuous operation. This poses a significant challenges on service provides that need to minimize the capital and operational investment expenditures in order to increase their return-on investments.

1.1 High Availability

The availability is a measure presenting the percentage of time in which the system is able to provide its services successfully during a time interval. For instance, an availability of 100% indicates that a system was healthy with no downtime in a specific time interval. System availability depends on how frequent it fails and how quick it recovers from failure. Conventionally highly available (HA) systems must sustain at least 99.999% (five-nines) of availability [1]. Five-nine systems allow a maximum 5 minutes and 15 seconds of downtime caused by planned and unplanned outages in a 1 year of

continuous operation. Fig.1.1. illustrates the maximum allowable downtime of a system against of availability nines.

Years of continuous operations	1	2	3
Availability	Maximum allowable downtime		
99.0000% (2–9s)	3 d 15 h 36 min 0 s	7 d 7 h 12 min 0 s	10 d 22 h 48 min 0 s
99.9000% (3–9s)	8 h 45 min 15 s	17 h 31 min 12 s	1 d 2 h 16 min 48 s
99.9900% (4–9s)	52 min 34 s	1 h 45 min 7 s	2 h 37 min 41 s
99.9990% (5–9s)	5 min 15 s	10 min 31 s	15 min 46 s
99.9999% (6–9s)	32 s	1 min 3 s	1 min 35 s

Fig. 1.1 Maximum allowable downtime for different availability levels [1].

Availability is expressed as a probability representing the portion of which the system is healthy and can deliver its services as intended. Availability can be expressed by the mean time to failure (MTTF) and mean time to repair (MTTR) as shown in Fig.1.2.

$$\text{Availability} = \text{MTTF/MTBF} = \text{MTTF}/(\text{MTTF} + \text{MTTR})$$

Fig. 1.2 Availability in terms of MTTF, MTTR, and MTBF

1.1.1 Mean Time to Failure (MTTF)

MTTF attribute is defined as the expected time for the system to encounter a service failure. It is also referred to as the uptime of a system in which it is operating and providing successful service.

1.1.2 Mean Time to Repair (MTTR)

MTTR attribute is defined as the expected time for the system to recover from a failure and return to a healthy state. It is also referred to as the downtime of a system in which it is neither operating nor providing a service.

1.1.3 Mean Time Between Failures (MTBF)

MTBF attribute is defined as the expected time for the system to have two consecutive failures. It is only defined in repairable systems.

Fig.1.3 illustrates MTTF, MTTR, and MTBF.

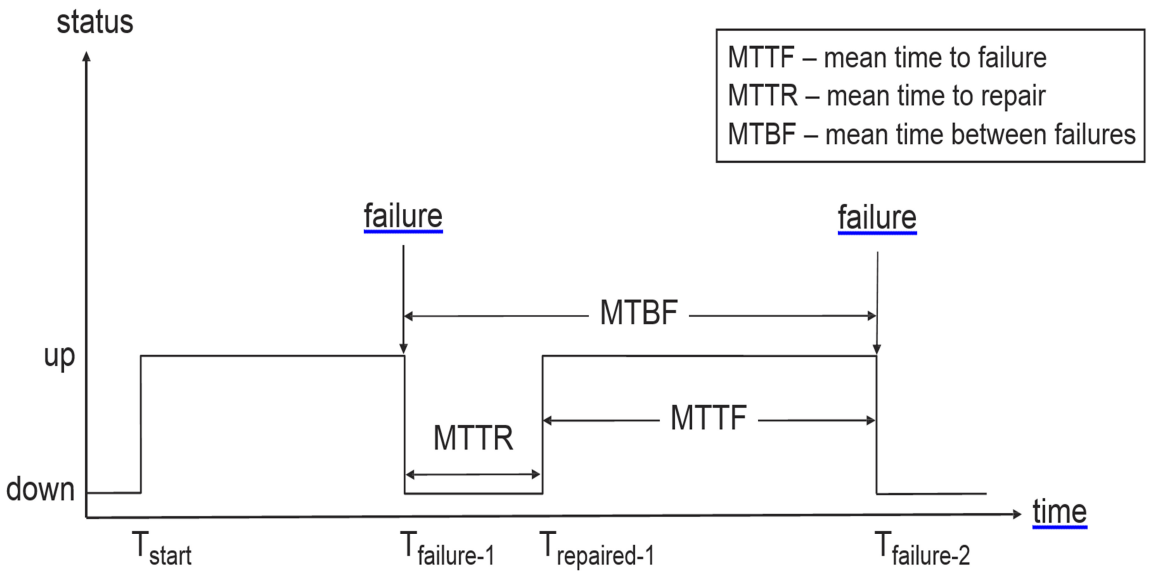

Fig. 1.3 MTTR, MTTF, MTBF [1].

In order to relate availability to the service time of a service, it can be expressed in terms of service uptime and service outage as follows:

Service availability = service uptime / (service uptime + service outage).

1.2 Achieving High Availability

High availability of an application can be achieved by a set of fault tolerance practices and protective redundancy of application components. HA is not about preventing failures of application component from happening but minimizing recovery time of component failures to assure overall service continuity [2].

1.2.1 Fault Tolerance

Fault tolerance is a failure avoidance approach aiming to mediate and wrap faults from causing service failure. Despite all the fault prevention methods employed by the application developer, there still a probability that a fault can cause a system failure. The main objective of fault tolerance is to ensure that a system can tolerate the possibility of fault occurrence by applying error detection and system component recovery. It can detect and handle faults before causing application service failure. Fault tolerance is considered to manage recovery of unplanned events and outages of the underlying system. The four major faults tolerance phases for achieving service availability [3] are:

1) Error Detection: to successfully sustain a highly available system, the fault occurrence must be first identified.
2) Damage Confinement and Assessment: the failure damage level is evaluated and restrained as much as possible. System state information is communicated between components to limit the scope and propagation of the error.
3) Error Recovery: is the process of eliminating the system failure cause and transforming the system to a healthy state.
4) Fault Treatment and Service Continuation: The key concern in this process is the perception to users that the intended service continues to be functioning as if nothing has happened.

In ideal case, the highly available application should provide its services regardless of what faults it encounters. Application requirements and resource constraints defined the necessity level for implementing multiple fault tolerance phases. This poses a trade-off between application availability and resources needed to offer the protection. The system

failure response shapes how it reacts when a failure is encountered. The system failure response can be defined as follow:

1) Fail-Operational: The service operates normally in the presence of faults with no degradation of service functionality and performance. This type of failure response assures the highest level of service availability but on the expense of additional system resources to cover all the conceivable failures.
2) Fail-Soft (graceful degradation): The service operates in a degraded functionality and performance. The main objective of this system failure response type is keeping the mission critical functionality of the system functioning normally.
3) Fail-Safe: The service maintains it functionalities for the current operation and halts its intended operation. This type of system failure response mainly used to secure safe condition for the users being served.
4) Fail-Stop: The service is immediately stop when an error is detected. This type of system failure response also known as Fail-Fast response. It used in situation where error propagation is suspected.

1.2.2 Redundancy

The major fault tolerance approach is to have a redundancy system components protection. Redundant system components translate into additional resources held at idle state in a normal operation of the system. The replicated system components protect the service from potential failures, giving the feel of uninterrupted service even when there are failures in the underlying system. The main two aspect related to redundancy that system designers and administrator must consider are (1) what should be replicated; and (2) in what state these replica stand to achieve the desired service availability.

Resources replication can be defined in different forms such as hardware, software, communication, and information. Hardware replication is the most common form of redundancy used in most current HA systems. Running multiple replicated copies of software on different hardware can tolerate hardware faults but if the fault lies in the software, which is a design faults, a replica of the same software and data would propagate the same fault all over the system. For instance, this kind of incident occurred

recently with one of the pioneer information technology (IT) companies; Apple incorporated (Inc.). March 11, 2015, Apple store experienced a complete outage for eleven hours [4]. This outage was caused by a release of faulty update that propagated along all their system components [5]. The best practice of redundant system components protection is applying a redundancy model based on the functionality of the system component. It is necessary to differentiate the role that each system component entitles. A component is considered to have an active role if it is serving current users. The redundant component takes the standby role if it is capable to take over the active role and sustains the service provided by the system. The active and standby components must communicate health conditions to successfully failover the workload up on fault occurrence. Service availability forum defined five types of redundancy models in the SAForum specifications [6].

1.2.3 Logical Entities of Highly Available System

Service availability forum (SAForum) defined the availability management framework (AMF) logical entities as follow [6]:

1) AMF Node: is also a logical entity where the HA middleware and HA application component are executed. The AMF manages its different states and defines its operations.
2) Component: is the logical entity that represents a set of resources to the AMF. It encapsulates specific application functionality. The resources can be a set of hardware resources, software resources, or a combination of the two. In addition, it presents the smallest logical entity on which the AMF performs error detection and isolation, recovery, and repair.
3) Component Service Instance (CSI): represents the workload that the AMF can dynamically assign to a component. High availability (HA) states are assigned to a component on behalf of its component service instances.
4) Service Unit (SU): is a logical entity that combines a set of components to provide a higher-level service. Most AMF administrative operations are applied to service units and not the components. A service unit can contain different components, but a particular component can only belong to one service unit. The service units

present the unit of redundancy from the AMF perspective. It is the smallest logical entity that can be instantiated in a redundancy manner.

5) Service Instances (SI): is aggregated CSIs in one logical entity. AMF assigns the SIs as workload to the SUs.
6) Service Group (SG): is a logical entity combining one or more SUs to form a protective group for a specific service.

1.2.4 Redundancy Models by SAForum:

The redundancy models defined the layout of the components states such as active and standby.

1.2.4.1 2N Redundancy Model

In a 2N redundancy model, at most one SU has the active HA state for all the workloads assigned as SIs. In addition, only one SU has the standby HA state for all the workloads assigned as SIs. Other SUs in the SG are configured to be spare units.

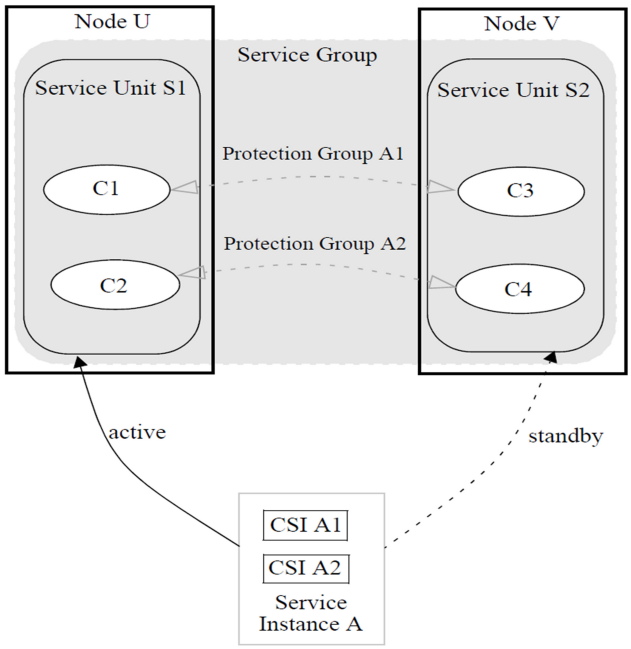

Fig. 1.4 2N redundancy model [6].

1.2.4.2 N+M Redundancy Model

The N+M redundancy model is an extension to the 2N redundancy model by permitting two or more SUs to have active and standby HA states. N presents the number of SUs having the active HA state. M presents the number of SUs having the standby HA state. This redundancy model mandates that the SU can have a strict HA state. For instance if one SU is assigned active HA state for a particular SI, it cannot be assigned standby HA state for another SI. The most common N+M redundancy model is N+1. Fig. 1.5 illustrates the N+1 redundancy model.

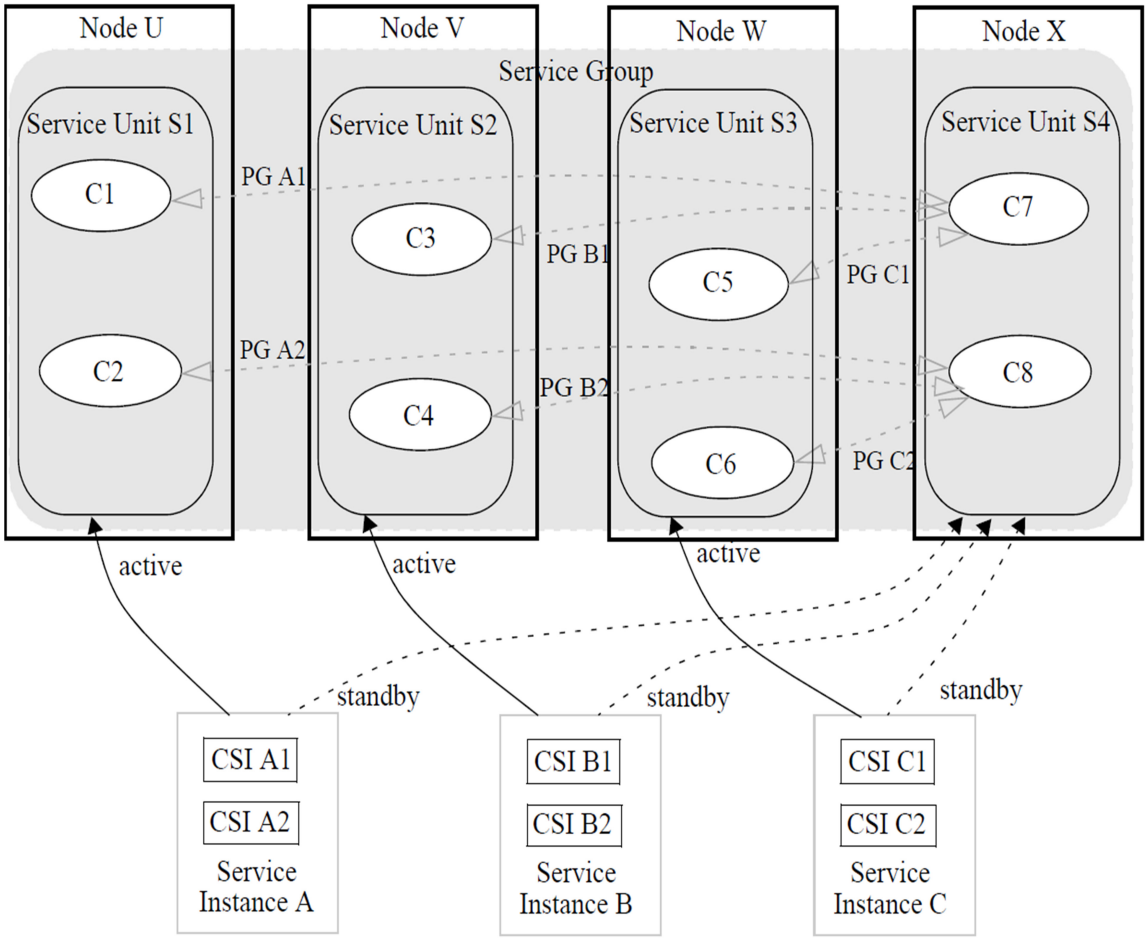

Fig. 1.5 N+1 Redundancy Model [6].

1.2.4.3 N-way Redundancy Model

The N-way redundancy model slightly differs from the N+M redundancy Model. It allows the SUs to have simultaneously active and standby assignments for different SIs. The advantage of this redundancy model is that all SUs can have active HA state while providing protection standby HA state.

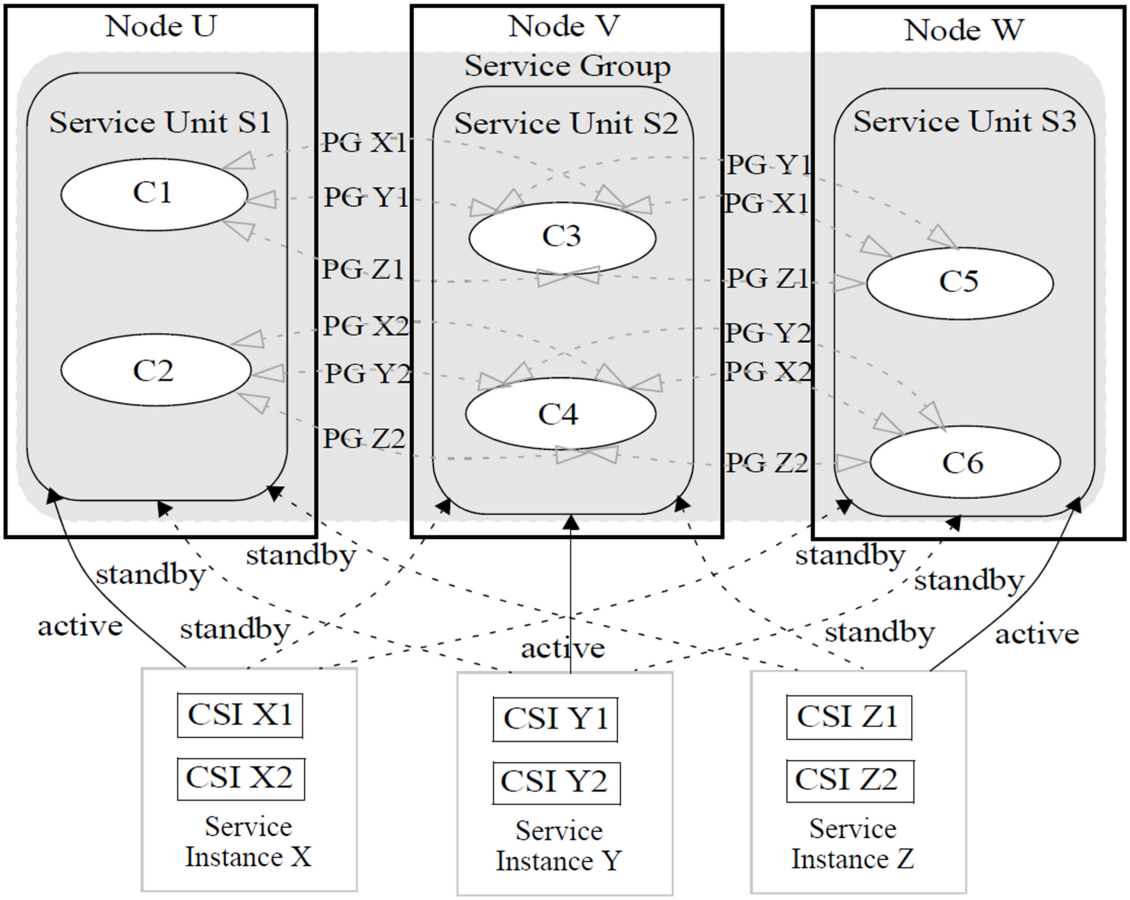

Fig. 1.6 N-way redundancy model [6].

1.2.4.4 N-Way Active Redundancy Model

The N-way active redundancy model differs from the previous redundancy models, as it does not support standby HA state. It allows the SI to have a several active SUs.

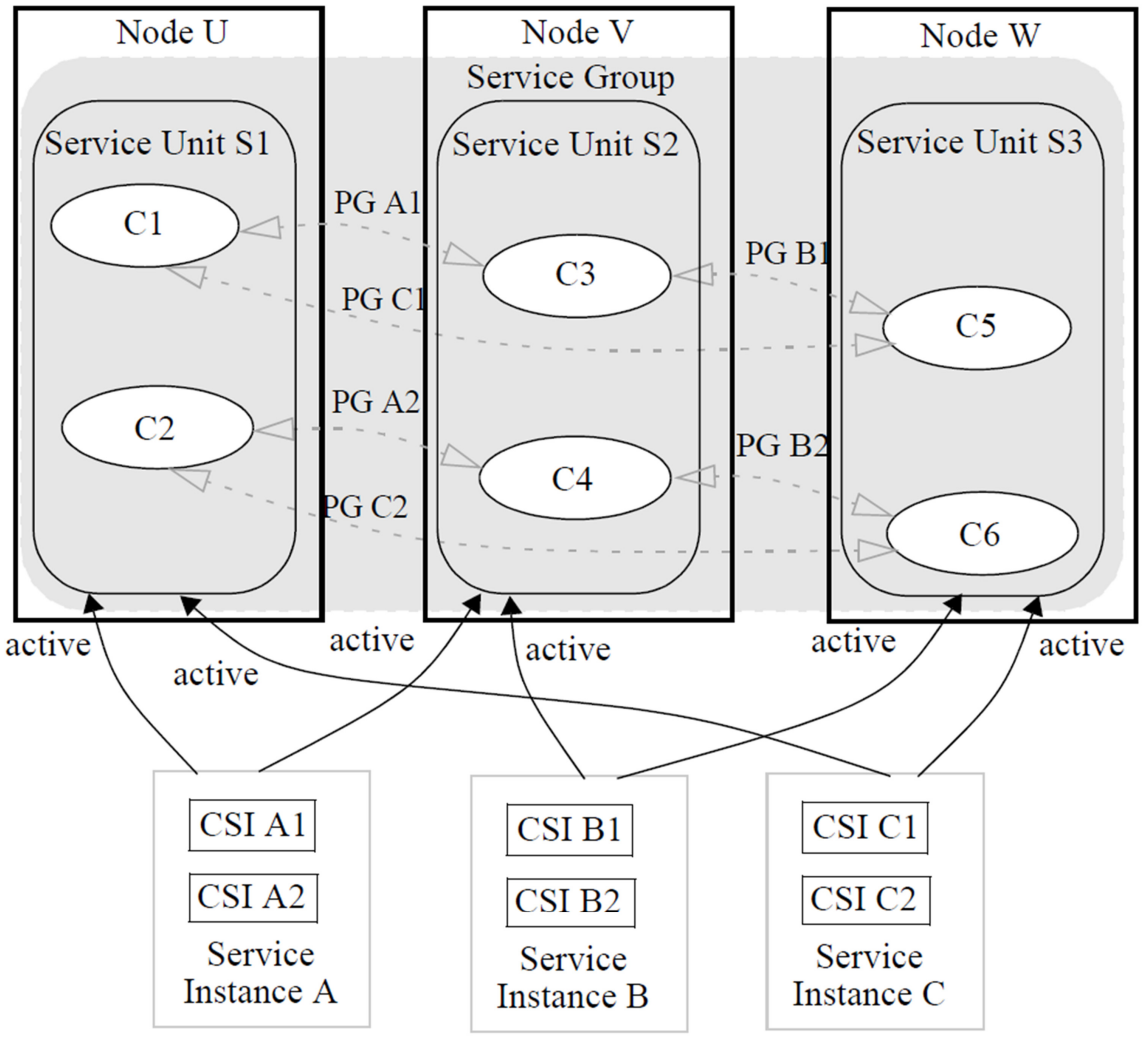

Fig. 1.7 N-way active redundancy model [6].

1.2.4.5 No-Redundancy Redundancy Model

The no-redundancy redundancy model is the simplest redundancy model. It only allows the SI to have at most one active HA state service unit and vice versa. This redundancy model is typically used for non-critical components with no severe impact on the overall system.

1.3 HA with Virtualization

Could computing is the paradigm of providing data, software and hardware services on-demand through different means of connectivity [7]. Mainly cloud computing utilizes various virtualization technologies to provide its services. Datacenters packing large number of servers, are the core engines of cloud computing that provide the main resource pools for many information and communication technology (ICT) companies. ICT companies rent cloud resources to provide their own distinct services ranging from business-critical processes and scientific computing [8], to social networking [9] and online gaming [10][11]. With large scale datacenters infrastructure composed from wide range of devices, resource failures are expected to happen [12][13]. Failures occurring during peak service periods, such as flashcrowds [14][15], and just before the outcome of specific application results are generated, leads to a significant low quality of experience (QoE). This low QoE results in revenue losses or customer departure [16].

Cloud service providers have contended many HA techniques for masking any infrastructure resource failure, but these techniques proven to be costly and difficult to manage when implemented on a large scale. Furthermore, current cloud service providers provide a limited exposure of HA management techniques to the users. Virtualization-based HA techniques Active/Active and Active/Standby are the most adopted technique in large datacenters and commercial datacenter products [17]. The adoption of these techniques is shown adequate to achieve HA of 99.99% (1 hour downtime per year) [17][18], and it is reflected in various cloud service providers service level agreements such as Microsoft Azure, Amazon web services, and google cloud [18][19][20]. HA of 99.99% significantly affect the revenue of service providers. For instance, a few seconds delay in generating the webpage response to a customer can lead to fewer sales. One-second delay in generating Amazon page response had a huge cost on amazon sales with 1.6 billion dollar losses [21]. As for companies built over advertisement eco-systems, one-second delay shows a devastating reduction in site traffic up to 20% [21]. All the HA techniques provided by the cloud service providers are at the level of the virtual machines (VMs) which also suffer from the same problem as replicating the hardware with a replicated copy of the software problem, discussed in previous section.

1.4 Elasticity in Cloud

Elasticity is one of the core attributes of cloud computing paradigm. The elasticity term is heavily used by the cloud providers' advertisements as a functionality to enhance cloud hosted application workload response.

Elasticity is the ability of a system to adapt its resources according to changes in the workload in an autonomic manner, such that the available resources are provisioned and released to match the current demand in real-time. Elasticity differs from scalability of the system where resources have the ability to be increased but not meant to match the workload [22]. To achieve elasticity in cloud different aspects have to be considered.

1) Automated Up and Down Scaling Configuration:
   Elasticity of resource is meant to match the workload in real-time. To provide the real-time aspect of elasticity the scaling process must assure no manual configuration is involved in the process.
2) Elasticity Dimensions:
   Application service consists of various tiers (1) application software tier (2) application middleware tier (3) system resources tier. Scaling of an application should be defined according which tiers it should scale. For instance, an application can be scaled on the resources tier while maintaining the same number of software instance. This kind of scaling is known as vertical resource scaling
3) Scalability Bounds:
   In a multi-tier application such as web-application, scaling should be considered at the component level. Moreover resource bounds should be well defined to make accurate decision about when (number of requests) and how much (vertical or horizontal scaling) resources should be allocated.

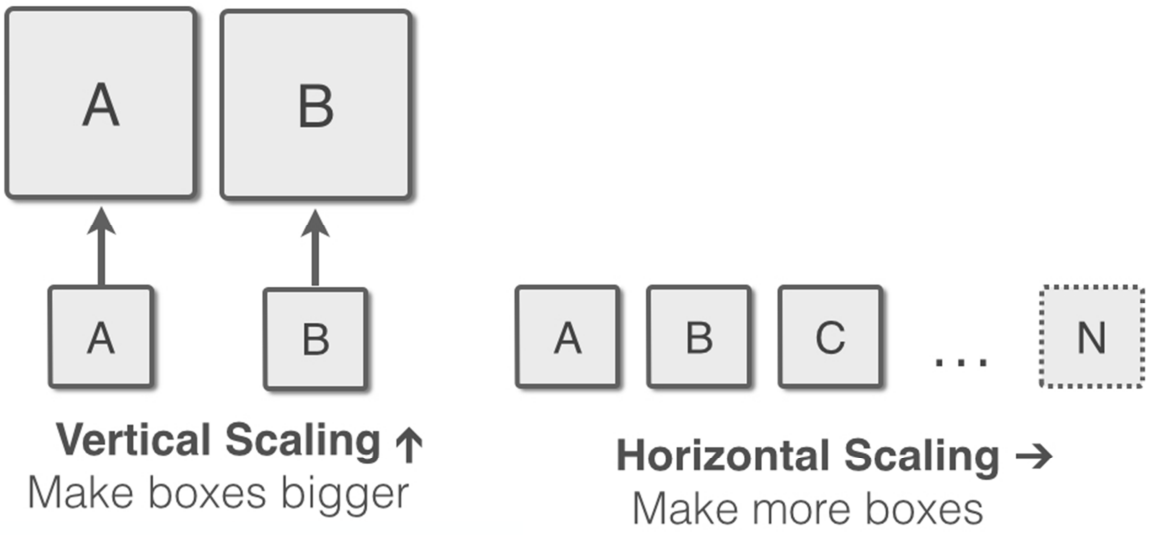

Fig. 1.8 Vertical Scaling vs Horizontal Scaling.

1.5 Problem Formulation

Virtualization has become the de-facto in the information technology industry. With the evolution and success of the modern data centers hosting various private and public cloud applications, virtualization has attracted other industries especially the telecommunication industry. As a leading step towards virtualization in the telecommunication field, a group of leading telecommunication service providers has introduced the network function virtualization (NFV) technology. NFV aims to waive the telecommunication applications' dependency on proprietary hardware and introduce them using commercial off-the-shelf (COTS) equipment. This shift in the telecommunication paradigm introduces various challenges to the cloud-computing era since carrier grade applications have very strict requirements in terms of performance, reliability, and availability. NFV applications require HA to be more than 99.999% (5 nines); achieving this kind of availability with the current provided cloud services is not applicable. Moreover, using the aged HA approaches such as pre-allocated redundant resources will not migrate the benefits of cloud computing to the telecommunication industry. Telecommunication service providers are looking for reducing their capital and operation investments while enhancing their quality of service (QoS) and experience by using the cloud computing features such as elastic scaling of application, resource consolidation,

and pay-as-you-go services. Therefore, in order to prepare the cloud to meet the requirements of carrier grade applications, we conducted a research to investigate and provide a solution that satisfies these requirements without sacrificing the advantages of the cloud-computing era.

1.6 Research Contributions

The work described in the subsequent chapters introduces several research contributions.

In particular:

1) Chapter 2 defines various challenges that encounter the NFV application development and deployment along with a set of practices and solutions for these challenges.

2) Chapter 2 proposes a placement for the ETSI NFV framework's entities in the current cloud computing stack.

3) Chapter 2 defines a grouping criterion for the virtualized evolved packet core (vEPC) entities. The entities' grouping criterion is conducted to suit vEPC for cloud deployment while maximizing performance and minimizing the signaling traffic between its entities.

4) Chapter 3 defines an elasticity framework for highly available application in cloud. This framework migrates the SAForum specification to the cloud environment to satisfy the carrier grade requirements.

5) Chapter 3 presents the implemented prototype of the framework using various technologies such as DevOps tools and proprietary developed software.

6) Chapter 3 defines the constraints and an algorithm for elastic placement of HA application components to achieve the desired availability while satisfying the functional and non-functional requirements.

Bibliography

[1] Service Availability Forum, “Application Interface Specification,” http://devel.opensaf.org / SAI-Overview -B.05.03.AL.pdf, 2011.

[2] P. S. Weygant “Clusters for High Availability: A Primer of HP Solutions,” Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001.

[3] P. A. Lee, T. Anderson “Fault Tolerance: Principles and Practice,” Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1990.

[4] CNBC,”http://www.cnbc.com/2015/03/11/some-apple-services-suffering-outages.html,” 2015.

[5] Apple, “http://appleinsider.com/articles/15/03/11/apple-issues-apology-as-itunes-app-store-outage-persists-for-7-hours,” 2015.

[6] Service Availability Forum, “Availability Management Framework,” http://devel.opensaf.org/SAI-AIS-AMF-B.04.01.AL.pdf, 2011.

[7] M.A. Sharkh, M. Jammal, A. Shami, A. Ouda, “Resource allocation in a network-based cloud computing environment: design challenges”, IEEE Communications Magazine, vol. 51, no. 11, pp. 46–52, November 2013.

[8] D. Talia, “Clouds for scalable big data analytics.” IEEE Computer, vol. 46, no. 5, 2013.

[9] Y. Chen, S. Alspaugh, R. H. Katz, “Interactive analytical processing in big data systems: A cross-industry study of mapreduce workloads,” PVLDB, vol. 5, no. 12, 2012.

[10] E. Deelman, G. Singh, M. Livny, G. B. Berriman, J. Good, “The cost of doing science on the cloud: the montage example,” in SC, 2008.

[11] V. Nae, A. Iosup, R. Prodan, “Dynamic resource provisioning in massively multiplayer online games,” TPDS, vol. 22, no. 3, 2011.

[12] F. Cappello, A. Geist, B. Gropp, L. V. Kale´, B. Kramer, M. Snir, “Toward exascale resilience,” IJHPCA, 2009.

[13] B. Schroeder, E. Pinheiro, W.-D. Weber, “DRAM errors in the wild: a large-scale field study,” in SIGMETRICS, 2009.

[14] I. Ari, B. Hong, E. L. Miller, S. A. Brandt, D. D. E. Long, “Managing flash crowds on the internet,” in MASCOTS, 2003.

[15] P. Bod´ık, A. Fox, M. J. Franklin, M. I. Jordan, D. A. Patterson, “Characterizing, modeling, and generating workload spikes for stateful services,” in SoCC, 2010.

[16] B. Javadi, D. Kondo, A. Iosup, D. H. J. Epema, “The failure trace archive: Enabling the comparison of failure measurements and models of distributed systems,” JPDC, vol. 73, no. 8, 2013.

[17] VMWare Inc., “Protecting Mission-Critical Workloads with VMware Fault Tolerance,”www.vmware.com/files/pdf/ resources/ft_virtualization_wp.pdf, 2009.

[18] Amazon, “Building Fault-Tolerant Applications on AWS,” http://media.amazonwebservices.com/AWS_Building_Fault_Tolerant_Applications.pdf, 2011.

[19] Google, “Google Cloud SQL now Generally Available with an SLA, 500GB databases, and encryption,” http://googlecloudplatform.blogspot.ca/2014/02/googlecloud-sql-now-generally-available.html, 2014.

[20] Microsoft, “Manage the availability of virtual machines Understand planned versus unplanned maintenance,” http://azure.microsoft.com/enus/documentation/articles/virtual-machines-manage-availability/ , 2014.

[21] CDNetworks," How Much is Each Second Worth in Ecommerce?," http://www.cdnetworks.com/blog/how-much-is-each-second-worth-in-ecommerce-check-out-these-15-stats/, 2013.

[22] L. Duboc, D. Rosenblum, T. Wicks, "A Framework for Characterization and Analysis of Software System Scalability," European Software Engineering Conference and ACM SIGSOFT Symposium on The Foundations of Software Engineering (ESEC-FSE '07), ACM, pp. 375–384 ,2007.

Chapter 2

2. NFV: State of the Art, Challenges and Implementation in Next Generation Mobile Networks (vEPC)

As mobile network users look forward to the connectivity speeds of 5G networks, network providers are facing challenges in complying with projected demands without substantial financial investments. Network function virtualization (NFV) is introduced as a new methodology that offers a way out of these bottlenecks. NFV is poised to change the core structure of telecommunications infrastructure to be more cost-efficient. In this chapter, Network Function Virtualization (NFV) is introduced with a discussion about the challenges and requirements of its use in mobile networks. In particular, an architecture for NFV framework entities in the virtual environment is proposed. Moreover, in order to reduce signaling traffic congestion and obtain better performance, this chapter proposes to bundle multiple functions of virtualized evolved packet-core in a single physical device or a group of adjacent devices.

2.1 Introduction

The demand for reducing capital expenditures (CAPEX) and operating expenditures (OPEX) has pushed information technology (IT) specialists toward contemplating designs to achieve more effective capital investments with higher return on capital. Toward this goal, the virtualization technology has emerged as a way to decouple software applications from the underlying hardware and enable software to run in a virtualized environment. In a virtual environment, hardware is emulated, and the operating system (OS) runs over the emulated hardware as if it is running on its own bare-metal resources. Using this procedure, multiple virtual machines can share available resources and run simultaneously on a single physical machine [1].

The demand for broadband network connectivity has been increasing dramatically in the last decade. It gains additional momentum with the increase in the number of Internet-connected mobile devices, ranging from smartphones, tablets, and laptops to sensor networks, and machine-to-machine (M2M) connectivity. This increasing demand

is pushing network service providers to invest in their infrastructure to keep up with the demand, although studies show that the return on such investments is minimal [2]. Network expenditures depend highly on the infrastructure on which the network relies. The high cost of any network-improvement upgrade or new service release narrows the revenue margin of the service provider. Network operating challenges are not limited to the cost of expensive hardware devices, but also include increasing energy costs and the competitive market for highly qualified personnel with the skills necessary to design, integrate, and operate an increasingly complex hardware-based infrastructure. In addition, managing network infrastructure is another major concern of service providers. These issues do not affect revenue only, but they also increase time-to-market and limit innovation in the telecommunications industry. Therefore, network operators seek to minimize or even eliminate their dependency on proprietary hardware.

To achieve these targets successfully, a group of seven telecom operators has formed an industry specifications group for Network Function Virtualization (NFV) under the European Telecommunications Standards Institute (ETSI). They revealed their solution in October 2012 [3]. More recently, several telecom-equipment providers and IT specialists joined the group.

2.2 Network Function Virtualization

The substantial dependence of networks on their underlying hardware and the existence of various specialized hardware appliances, for example firewalls, deep packet inspection (DPI) equipment, and routers, in the network infrastructure have escalated the challenges facing network service providers. Furthermore, the reduced life cycles of these types of hardware, due to fast pace of innovation tends to multiply CAPEX and OPEX investments [3]. Network function virtualization technology was developed to take advantage of the evolution of IT virtualization. It separates network functions from the underlying proprietary hardware appliances. NFV is the concept of transferring network functions from dedicated hardware appliances to software-based applications running on commercial off-the-shelf (COTS) equipment.

These applications are executed and consolidated on standard IT platforms like high-volume servers, switches, and storage. Through NFV, network functions can be instantiated in various locations such as datacenters, network nodes, and end-user premises as the network requires [3]. Fig 2.1 illustrates the migration of network proprietary hardware to a software-based application on COTS equipment.

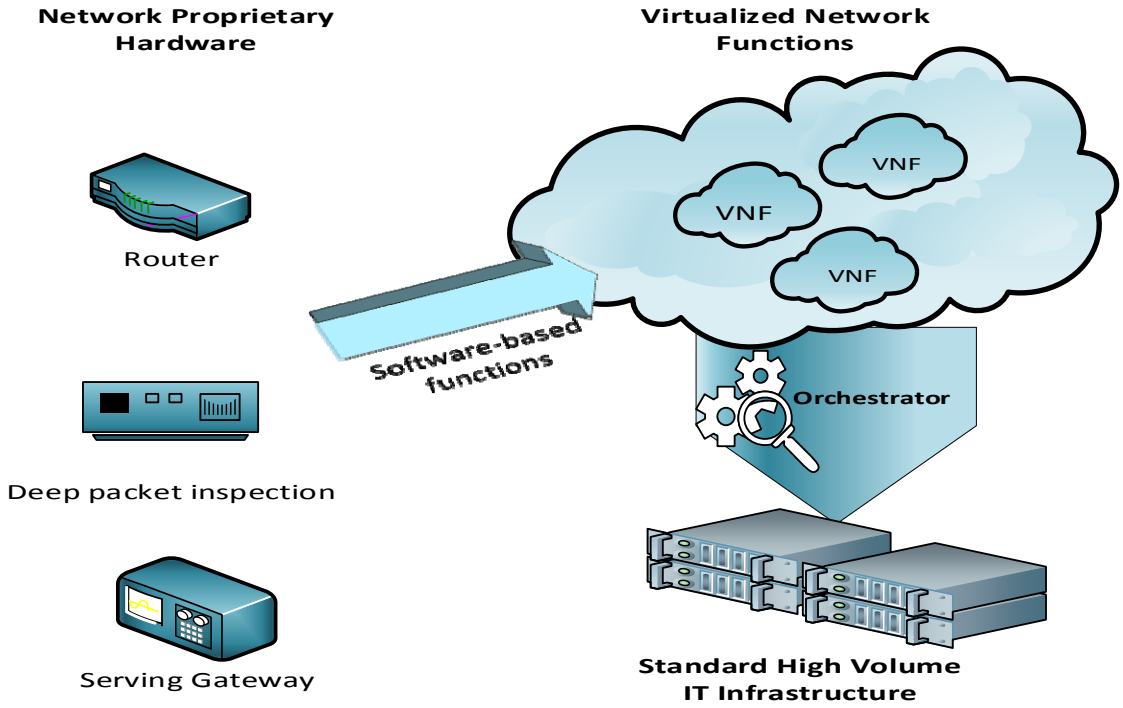

Fig. 2.1 Network function virtualization concept.

NFV is poised revolutionize the telecommunication era from research to industrial implementations. Telecommunication equipment vendors have to reform their doctrine to compete in the new software based telecommunication area. NFV has been foreseen to lead the future of telecommunication. As NFV decouple the network functions from underlying hardware, it offers many benefits to the telecommunication systems. Some of these benefits are listed below.

2.2.1 Openness of Platforms

As network functions are virtualized to be executed on standard IT infrastructure in a software manner, this will grant the telecommunication industry an opportunity to

embrace a standardized form for the interfaces and the network functions themselves. By such act, service providers could avoid the vendor lock-in in their network equipment that have sustained for years. Vendor Locked-in has overwhelmed their budgets, and degraded their network performance. Adopting openness concept assures that service providers benefit from multi-vendor network functions that serve their needs [4]. They merge all the expertise from different vendors since there is no single vendor who masters all the network functionalities. Furthermore, openness of the platforms would facilitate new revenue trends and opportunities for better contribution and innovation from IT and software designing companies, startup companies and academia. Also open source projects could be initiated to improve network functions performance [3].

2.2.2 Scalability and Flexibility

Telecommunication service providers design their networks infrastructure according to network traffic demand in the peak hours to be able to handle that traffic and maintain their targeted quality of service (QoS). Most of the time these equipment would not be efficiently utilized and yet they have to be managed and maintained functional 24/7 to provide the services. Furthermore, proprietary telecommunication equipment occupies space and consumes energy to operate and maintain the functional terms, *e.g.*, temperature of the equipment. NFV addresses these concerns and allow service providers to easily scale their infrastructure in real-time since they are deployed in virtualized environment [5]. Virtualized network functions (VNFs) are dynamically scaled to fulfill the traffic demands. VNFs resources could be scaled up in specific locations where higher demands are needed. NFV also allows resources sharing between lightly utilized VNF and higher demanded VNFs. However, service providers will benefit from downscaling their VNF resources during off peak hours. This down scaling allows service providers to benefit from assigning these resources to serve other tasks or can be easily switched off [3].

2.2.3 Operation Performance Improvement

Virtualization in the IT era has made a great progress and advanced orchestration mechanisms from which NFV will benefit. VFNs are installed and initiated automatically

by the orchestrator of the environment, which senses the network traffic on real-time basis. Intelligent automated resource allocations or instantiation of new VNFs can be used to improve the network service performance in a real-time basis. These orchestration mechanisms can be also used to improve network resiliency and limit service interruption by automatic network failure and fault recovery.

2.2.4 Improve Development Cycle

Any upgrade from developing to releasing new services has been an ordeal in telecommunication industry. It requires lengthy time to develop the software then porting it to specific hardware, besides the required procedures for testing and quality assurance. In addition, the newly introduced service should assure compatibility with the legacy equipment. NFV waives all of these concerns and provide the possibility of having the production, testing and development environment running on the same infrastructure. This reduces the development cycle since the hardware development and porting part is eliminated from the cycle. This elimination leads to reduced time to market and lower CAPEX and OPEX investments during development cycle.

2.2.5 Reduced CAPEX and OPEX

Service providers are implementing and investing in their network infrastructures to satisfy the network traffic demands as connectivity devices like smartphone, tablets, and laptops are increasing dramatically. Although, they are supporting high network traffic but their revenues are steady over time with slight increment. NFV implementation reduces the CAPEX and OPEX; therefore revenue will increases. CAPEX is reduced by decoupling the network functions from proprietary hardware and consolidating on commercial off-the-shelf (COST) hardware which lead to lower hardware footprint. Furthermore, OPEX is reduced by implementing the automated robust orchestrators. In addition, the consolidation of VNFs on COST servers reduce the power and maintenance cost. NFV allows faster and flexible releases of new services and support multi-tenancy therefore new revenue trends can be introduced to the market.

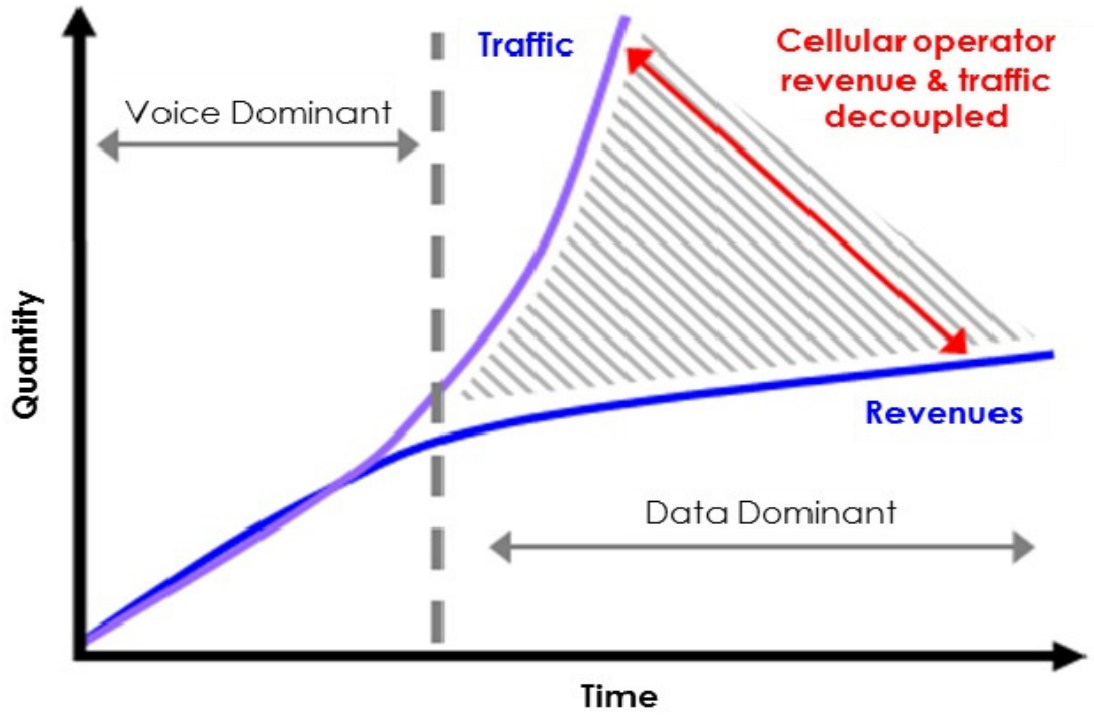

Fig. 2.2 Service provider revenues vs traffic [2].

2.3 NFV and SDN

Software defined network (SDN) and NFV are two different independent technologies; however, they are complementary to each other. SDN technology started as a project targeting the network functionalities at Stanford University in 2007. SDN decouples the control plane from the data plane in the network. The control plane is responsible for network management such as path computation. It is centralized and implemented in the SDN controller, which provides the necessary computational power to achieve the goal [6]. Data plane is responsible for transporting the traffic. It is implemented in open flow switches. They are programmable switches that provide application programming interface (API) to be managed and controlled by the SDN controller. SDN will serve NFV by providing the programmable connectivity between VNFs; these programmable connections can be managed by the orchestrator of the VNFs that mimic the role of the SDN controller [6]. On the other hand, NFV can serve SDN by implementing its network functions as software application on COSTs servers. It can virtualize the SDN controller

to run on cloud, which could be migrated to the best-fit location based on to the network needs. Fig.2.3. lists how NFV and SDN differ and complement each other.

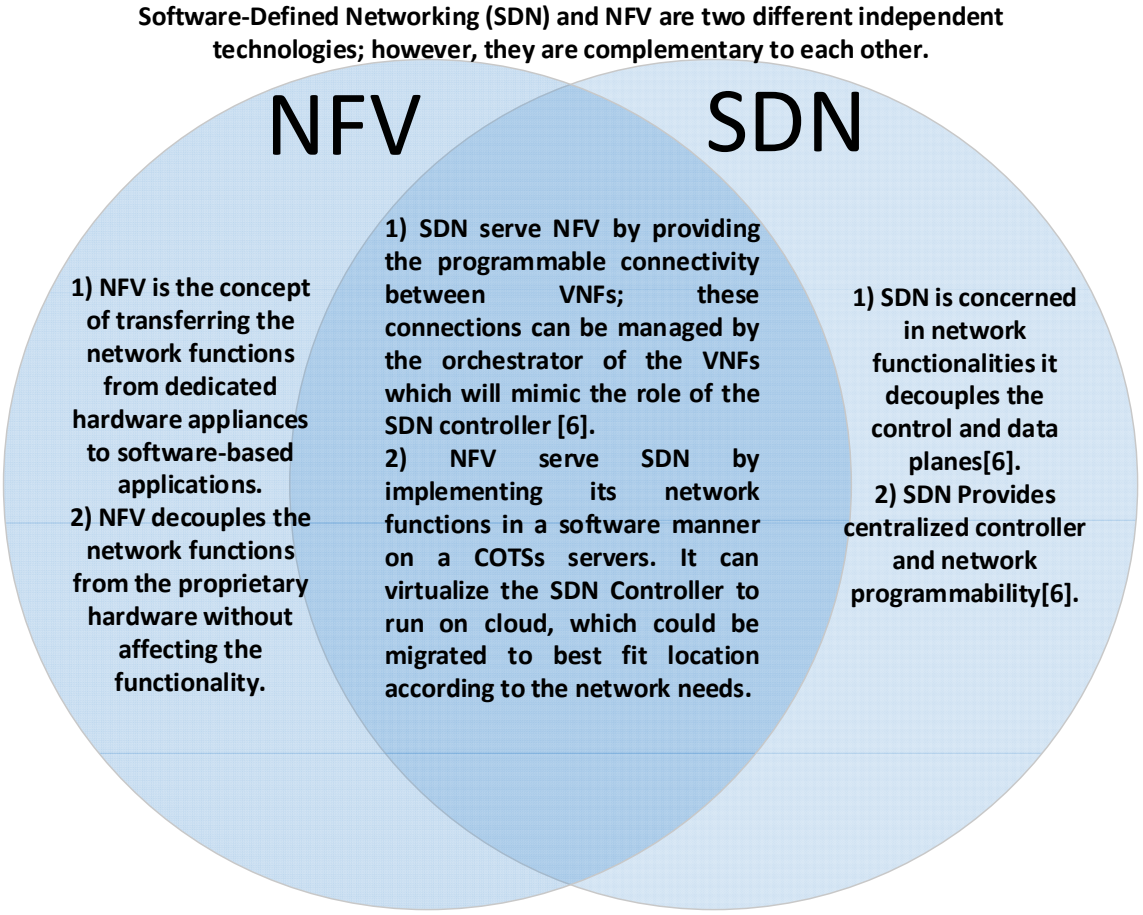

Fig. 2.3 NFV differs from SDN.

2.4 NFV Framework

The basic components of virtualized platforms where NFV is deployed, are listed below:

1) Physical server: The physical server is the bare-metal machine that has all the physical resources such as CPU, storage, and random access memory (RAM).
2) Hypervisor: The hypervisor, or virtual machine monitor, is the software that runs and manages the physical resources. It provides the virtual environment where the guest virtual machines are executed.

3) The guest virtual machine: A piece of software that emulates the architecture and functionalities of a physical platform on which the desired application is executed.

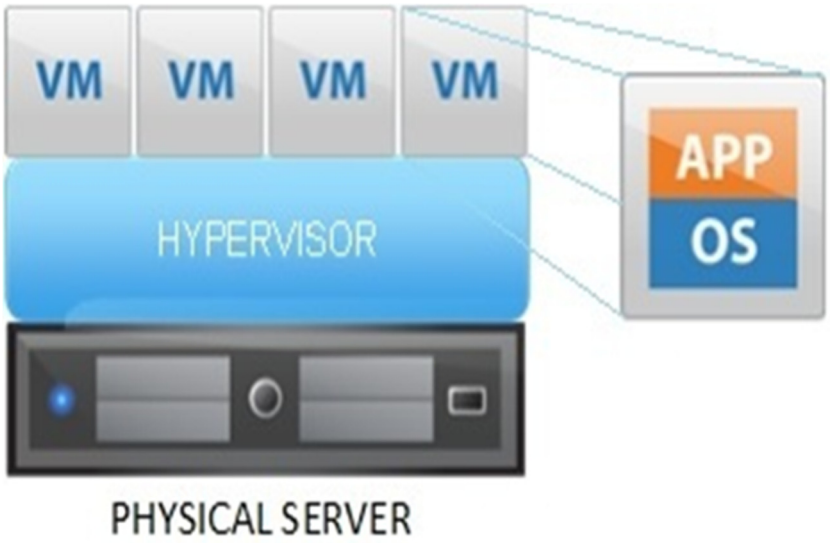

Fig. 2.4 Virtualization Layout.

Virtual machines (VMs) are deployed on high-volume servers that can be located in datacenters, at network nodes, or in end-user facilities. Moreover, most VMs provide on-demand computing resources using cloud enviroments. Cloud-computing services may be offered in various formats [4]; as shown in Fig. 2.5:

1) Infrastructure as a service (IaaS) is referred to as hardware as a service (HaaS). The service provider offers the computing resources (CPU, storage and RAM) to the user without any specific operating system or application preinstalled on the resources by the service.
2) Platform as a service (PaaS) is a computing environment that enables the development and implementation of applications without the hassle of managing the underlying software and hardware.
3) Software as a service (SaaS), referred to as software on demand is a service of providing a specific application (like office, management, or CAD software) to the end-user over the web.

4) Network as a service (NaaS): There is no standard definition NaaS. NaaS is often considered to be provided under IaaS or as a standalone service that provides the connectivity between network nodes, datacenters, virtual machines, and end-user premises.

Fig. 2.5 The cloud services.

The NFV technology takes advantage of the infrastructure and networking services (IaaS and NaaS) to form the network function virtualization infrastructure (NFVI) [5].

To achieve the objectives promised by NFV such as flexibility in assigning virtual network functions (VNFs) to hardware, rapid service innovation, enhanced operational efficiency, reduced power usage, and open standard interfaces between VNFs, each VNF should run on a framework that includes dynamic initiation and orchestration of VNF instances. In addition, the framework should also manage the NFVI hosting environment on IT virtualization technologies to meet all VNF requirements regarding data, resource allocation, dependencies, availability, and other attributes. The ETSI NFV group [3] has defined the NFV architectural framework at the functional level using functional entities and reference points, without any indication of a specific implementation. The proposed framework is shown in Fig.2.6.

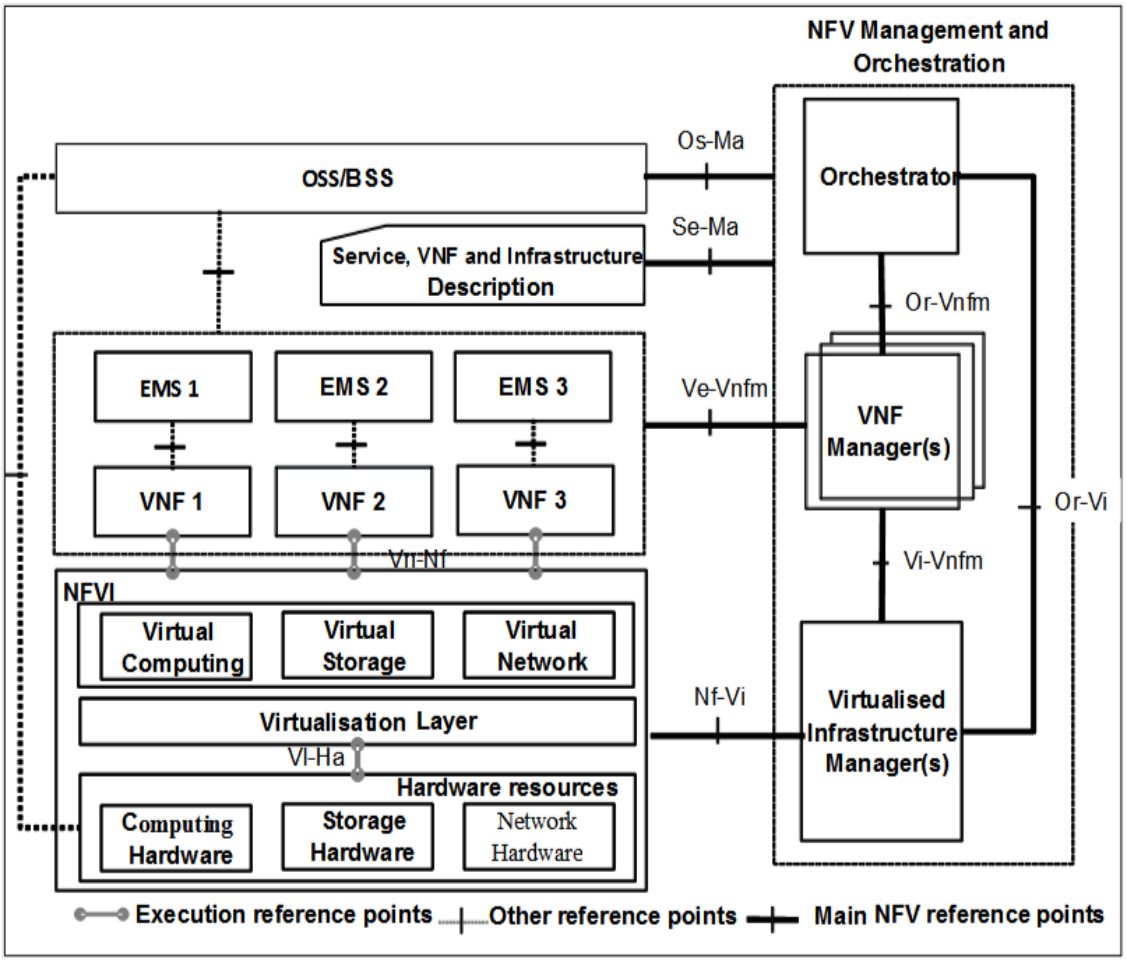

Fig. 2.6 NFV framework [5].

2.5 Proposed Placement of Framework Entities

The proposed placement is based on mapping the NFV framework entities to best fit into the virtual environment. The virtual resources manager, the VNF manager, and the orchestrator have been grouped at the hypervisor level. Since the virtual environment will not only host VNFs, but also other IT applications, this grouping leads to a centralized controller. The infrastructure that provides NFVI as a service provide cloud services simultaneously on the same hardware resources. Essentially, the hypervisor manages and orchestrates the physical and logical resources of the virtualized environment. It is aware of the virtual machines that are using the underlying hardware and manages resource scheduling and decisions such as migration, resource scaling, and fault and failure

recovery, more efficiently to meet the specified quality-of-service requirements of VMs (VNFs and APPs) [5].

The virtualization layer consists of a cross-platform virtual resource manager that runs on top of the hypervisor to ensure the portability and flexibility of VNF independently of the hypervisor. OpenStack, Eucalyptus, oVirt, OpenNebula, and Nimbula are examples of cross-platform virtual layers [7]. The virtual machine hosts VNF and its element-management system (EMS). Each VNF instance has its private EMS to reduce complexity when migrating an existing VNF or initiating a new one. Operations and business support systems with VNF infrastructure description entities are deployed in a centralized form which provides uniformity of VNF software images and minimize database fragmentation. The proposed placements are illustrated in Fig.2.7.

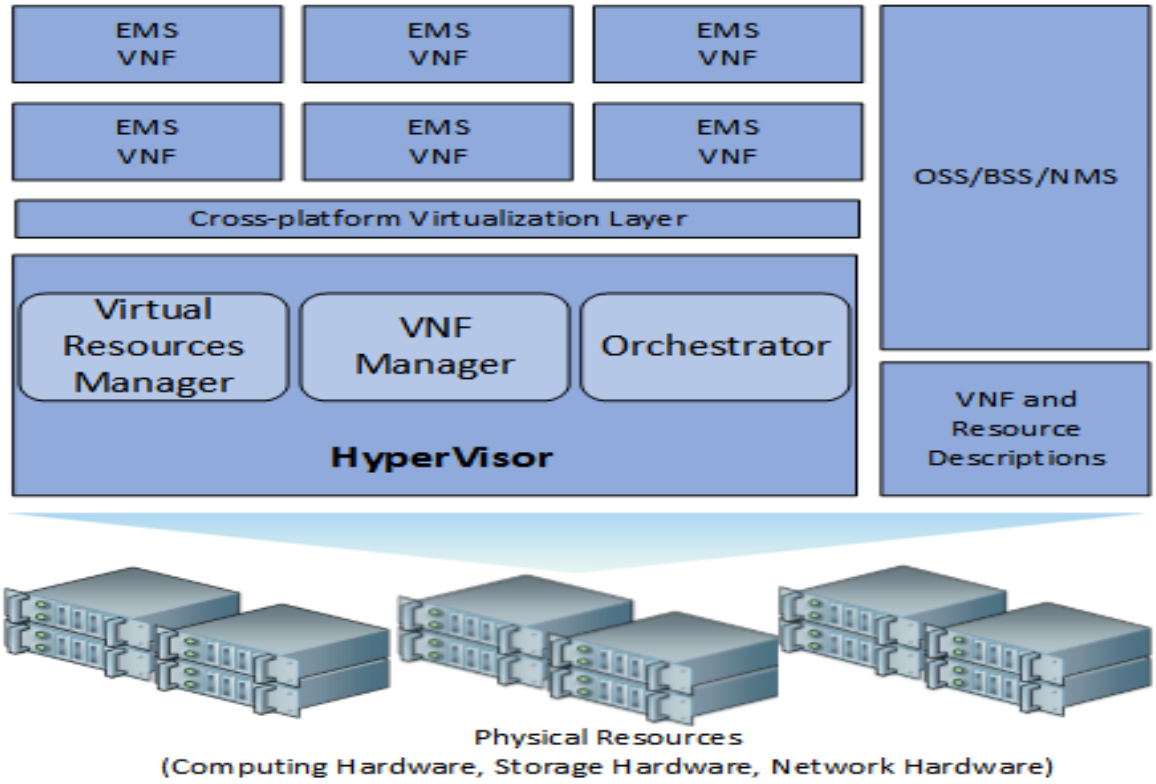

Fig. 2.7 NFV Framework Entities Proposed Placements.

2.6 NFV Ecosystem

Service providers have shown keen interest in NFV. Observing this interest, telecom equipment vendors and IT companies have started to investigate different aspects of NFV realization. Leading telecommunication equipment vendors like Ericsson, Nokia, Alcatel-Lucent, and Huawei have already started to adopt and upgrade their equipment to support NFV [8]. Moreover, leading IT companies that provide carrier grade software like Wind River, 6wind, Qosmos, and HP have been working closely with Intel to optimize their software on Intel processors in order to achieve higher packet processing computations that enable NFV and SDN on COTS platforms [9]. Intel has released the Data Plane Development Kit (DPDK) and has scheduled the release of signal processing development kit in its software development roadmap to extend and speed up NFV and SDN adoption [10]. Service providers started experimenting with these NFV products and put the devices under heavy testing to ensure that they will meet the expectations as carrier-grade products [11].

2.7 NFV Challenges and Requirements

Although NFV is a promising solution for telecommunications service providers, it faces certain challenges that could degrade its performance and hinder its implementation in the telecommunications industry. In this section, some of the NFV requirements and challenges, and proposed solutions are discussed. Table 2.1 summarizes this section.

2.7.1 Security

Security is an important aspect of the telecommunications industry. NFV should obtain a security level close to that of a proprietary hosting environment for network functions. The best way to achieve this security level is by dividing it according to functional domains. Security in general can be defined according to the following functional domains:

1) Virtualization environment domain (hypervisor)
2) Computing domain
3) Infrastructure domain (networking)

4) Application domain.

Security attacks are expected to increase when implementing network functions in a virtualization environment. A protected hypervisor should be used to prevent any unauthorized access or data leakage. Moreover, other processes such as data communication and VM migration should run in a secure environment [12]. NFV uses APIs to provide programmable orchestration and interaction with its infrastructure. These APIs introduce a higher security threat on VNFs [13].

2.7.2 Computing Performance

The virtual environment underlying hardware server characteristics such as processor architecture, clock rate, cache memory size, memory bandwidth, and speed has a profound impact on VNF performance. VNF software design also plays a major role in VNF performance. VNF software can achieve high performance using the following techniques:

1) A high-demand VNF should be implemented using multi-threading techniques and in a distributed and scalable fashion, in order to execute it on multiple cores or different hosts.
2) Software instances should have independent memory structures to avoid operating-system deadlocks.
3) VNF should implement its own network stack and avoid networking stacks implementation in the operating system, which consume large amounts of computing resources.
4) Direct access to input/output interfaces should be used whenever possible to reduce latency and increase data throughput.
5) Processor affinity techniques should be used to take advantage of cache memories.

Implementing these techniques in VNF software may require a different approach from the automated resource allocation within a given pool of servers, which is currently used in IT environments.

2.7.3 Interconnection of VNFs

Unlike the classical approach of interconnecting network functions by a direct connection or through Layer 2 (L2) switches, a virtualized environment uses different approaches. In a virtualized environment, virtual machines can be connected in different scenarios [14]:

1) If two VNFs are on the same physical server and on the same local-access network (LAN), they are connected through the same Vswitch.
2) If two VNFs are on the same physical server but on different LANs, the connection passes through the first Vswitch to the network interface controller (NIC), then to the external switch, and back again to the same NIC. This NIC forwards the connection to the Vswitch of the second LAN and then to the VNF.

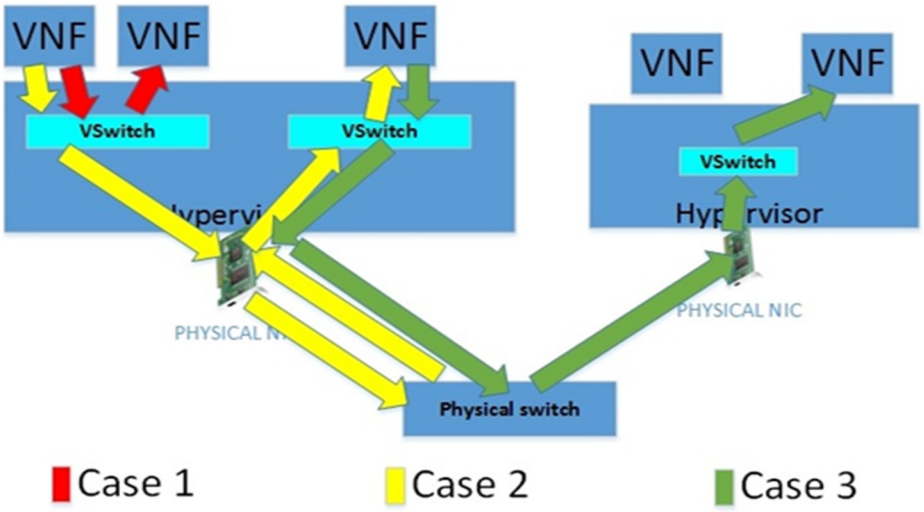

Fig. 2.8 Networking in virtualization environment.

3) If two VNFs are on different servers, the connection passes through the first Vswitch to the NIC and then to an external switch. This switch forwards the connection to the NIC of the desired server. Finally, this NIC forwards it to its internal Vswitch and then to the destination VNF.

Some NICs provide direct access from the virtual machine. These NICs are single-root I/O virtualization (SR-IOV) compliant. They offer faster and higher throughput to virtual machines. Each connectivity technique has its own advantages in terms of performance, flexibility, and isolation. Virtual interfaces managed by the

hypervisor have lower performance compared to virtual interfaces offered by SR-IOV-compliant NICs. However, virtual interfaces provided by the hypervisor are simpler to configure, and support VM live migration in a simpler way. The correct choice depends on the VNF workloads. Fig.2.8. illustrates the VNF interconnection cases.

2.7.4 Portability

Virtualized network functions can be deployed in different ways. Each way has its own advantages and drawbacks. Virtualized network functions that are executed directly on bare-metal ensure predictable performance because mappings of software instances to hardware are predictable. This kind of deployment sacrifices resource isolation and makes software-instance security difficult to achieve because multiple software appliances are executed as processes on the same operating system. In addition, the designed software would be OS-dependent.

Deploying virtual network functions through a virtual environment improves portability and ensures that hardware resources are viewed uniformly by the VNF. This deployment also enables each VNF to be executed on its specific operating system while remaining unaware of the underlying operating system. In addition, VNF resource isolation is ensured because VNFs are executed on independent VMs managed by the hypervisor layer, which guarantees no unexpected interactions between them. Strict mapping of resources should be used to guarantee resource isolation.

2.7.5 Operation and Management

Virtual network functions should be implemented as simple drag-and-drop operations in the orchestration management system. To make this a reality, both VNFs and computing infrastructure should be described using standard templates that enable automated management.

The orchestration management system is responsible for providing and managing the NFV environment through north- and south-bound interactions. Northbound interactions are used to manage and provide access to the VNFs. Moreover, VNFs could use them for information or request queries such as asking for more computing resources. Southbound

interactions are used to interact with the NFVI and request information from other framework entities. In addition, they are used to request information about policies, VNF software images, VNF descriptions, or network forwarding graphs.

2.7.6 Co-existence with Legacy Networks

Virtual network functions should be able to coexist with legacy network equipment. It means that a) it should be able to interact with legacy management systems with minimal effects on existing networks, b) the network forwarding graph should not be affected by the existence of one or more VNFs, and c) a secured transition should be ensured between VNF instances and physical functions, without any service interruption or performance impacts [15].

2.7.7 Carrier-Grade Service Assurance

Carrier-grade service is a service in which hardware, software, and system components ensure high availability and reliability. For NFV to meet carrier-grade service requirements, it should provide resilience to failure, service continuity, and service assurance. Resilience to failure is provided by implementing an automated on-demand mechanism in the NFV framework to reconstitute the VNF after a failure. VNF reconstitution should not have any impact on the system to ensure stable service. Service assurance is provided by the NFV orchestrator, which is monitoring network-function performance and scale resources almost in real time [16].

Challenge	Description	Solutions and Requirements
Security	Virtualization security risks according to functional domains: 1) Virtualization environment domain (Hypervisor): – Unauthorized access or data leakage. 2) Computing domain: – Shared computing resources: CPU, memory…etc. 3) Infrastructure domain (networking): – Shared logical-networking layer (Vswitches). – Shared physical NICs.	Security implementations according to functional domains: 1) Virtualization environment domain (Hypervisor): – Isolation of the served virtual-machine space, with access provided only with authentication controls. 2) Computing domain: – Secured threads. – Private and shared memory allocations should be erased before their re-allocation. – Data should be used and stored in an encrypted manner by which exclusive access is provided only to the VNF. 3) Infrastructure domain (networking): – Usage of secured networking techniques (TLS, IPSec, or SSH).
Computing performance	The virtualized network function should provide comparable performance to network functions running on proprietary hardware equipment.	VNF software could achieve high performance using the following techniques: – Multithreading to be executed over multiple cores, or could be scaled over different hosts. – Independent memory structures to avoid operating-system deadlocks. – VNF should implement its own network stack. – Direct access to input/output interfaces. – Processor affinity techniques should be implemented.
VNF interconnection	Virtualized environment has different approaches from classical network function interconnection.	VNFs should take advantage of accelerated Vswitches and use NICs that are single-root I/O virtualization (SR-IOV) compliant.
Portability	VNFs should be decoupled from any underlying hardware and software. VNFs should be deployable on different virtual environments to take advantage of virtualization techniques like live migrations.	The VNF development should be based on a cross-platform virtual resource manager that ensure its portability.
Operation and management Existence with legacy networks Carrier-grade service assurance	VNFs should be easy to manage and migrate with existing legacy systems without losing the specification of a carrier-grade service.	To achieve the desired operation and management performance, a standard template of NFV framework entities should be well-defined. It should be able to interact with legacy management systems with minimal effects on existing networks. The NFV orchestrator must monitor network function performance almost in real time.

Table 2.1 NFV challenges and solutions.

2.8 Use Cases and Services

The Network Function Virtualization technology in principle considers all network functions for virtualization through well-defined standards. Most likely, NFV services will be provided in a similar way to IT virtualization service models. NFV service models include NFVI as a Service (NFVIaaS), VNF as a Service (VNFaaS), and Virtual Network Platform as a Service (VNPaaS). Service providers will choose between these service models to serve their network-connectivity needs and use cases. Some of the use cases will include, for example, fixed-access network function virtualization, content-delivery network virtualization, and home environment virtualization [17].

2.8.1 NFVI as a Service (NFVIaas)

The NFV Infrastructure can be considered as providing the required infrastructure for an environment in which virtualized network functions can be executed. The NFVIaaS should provide computing and networking capabilities comparable to IaaS and NaaS in cloud computing services. Compatible is assured with any IT application since it provides a standard IT computing resources.

2.8.2 VNF as a Service (VNFaaS)

In VNFaaS the VNF is an application provided by the service provider to the service consumer. The consumer will not be in charge of managing, controlling the NFVI, and VNF instances. Consumers of the VNFaaS do not have to develop or own the VNF application rather they can obtain them on an expense basis from the Service Provider as needed. The Service Provider allocates resources of NFVI and manages the VNF in order to deliver the desired network functionality with the required QoS.

2.8.3 Virtual Network Platform as a Service (VNPaaS)

A Platform as a Service provides a development environment tool for the consumers to develop their own specific VNFs. VNPaaS is the service where the consumer will not worry about the infrastructure and compatibility of the development environment. The consumer will use his developed VNF and the provided VNF to build his own virtual networks.

2.8.4 Fixed Access Network Functions Virtualization

The Virtualization in the fixed access network will mainly target the network functions in the fiber-to-distribution nodes. They are deployed in street cabinets, customer premises, or underground. The hardware footprint of these network functions should be in compact form. Furthermore, the hardware should run with a very low power consumption and assure durability with minimum failure. Implementing the fix access network function with NFV technology will expunge the challenges. NFV offers lower hardware complexity, footprint, and energy consumption. Moreover, NFV will permit rapid service upgrades and deployment to meet all technology revolutions that fix access network is sighting.

VNFs are implemented mainly in Layer 2 (L2) and Layer 3 (L3) network functions in the fixed access network. As for layer 1 functions will be considered for virtualization later. Layer 1 has the signal processing computations that need to be delivered in real-time manner to avoid latency issues. VNFs most probably will be seen at first in L2 and L3 network functions of Multiple Dwelling Units, digital subscriber line access multiplexer, and cable modem termination system.

2.8.5 Content Delivery Networks Virtualization

Content delivery networks (CDNs) are systems used by service providers to deliver diverse of data contents to the end user. They have been deployed in various geographic areas to assure fast and reliable data content delivery. CDNs have been used for various applications for example web caching, data storage and download, live streaming media, and on-demand media services. Video data traffic has been increasing dramatically with the increase of mobile devices and streaming media services like Internet protocol television (IPTV) and on-demand video streaming. This increase in Video traffic had forced service providers to initiate multiple CDNs to serve their customers with the desired quality of service. This multiple deployment of CDNs has led to a complex management of the distributed sites and inefficient utilization of resources during off-peak hours. Virtualization will target all CDN functions and will bring all the advantages of NFV to the service provider. Multiple CDNs management will be easier with NFV

automated management and orchestration. Efficient resource utilization and lower footprint is achieved through NFV automated resource allocations. In addition, Service resilience is achieved through NFV automated VNF deployment.

2.8.6 Home Environment Virtualization

As the home environment services like internet, VoIP, and on-demand video streaming are increasing ,more client equipment are needed to be installed at the customer home. Service providers have to install a residential gateway to provide all these services to the customer. Furthermore, when a new service is going to be released the customers have to upgrade their home equipment to have access to these services. Virtualizing the home environment network functions such as the residential gateway and digital box will provide a great deal for the service providers and customers. Virtualizing the home environment function and implementing these functions on the cloud introduces a network bandwidth challenge. Higher bandwidth connection should be provided between the customer home and the service provider network. With video streaming in high-definition, VoIP, and internet access the network bandwidth should be almost 1 Gigabit per second with low latency to assure the service quality. This requirement would be achieved in the near future since some service providers have started offering these kind of high-speed connections.

2.8.7 Mobile Network Virtualization

Mobile network connectivity demand is rapidly increasing with the growing number of mobile devices and applications that need to be always connected. Service operators must continually upgrade and enhance their infrastructure, for example by providing enough mobile base stations and network cores to achieve the desired data throughput, latency, and quality of service. Virtualization of mobile networks targets the mobile-network base station and mobile core network. Service providers have been showing interest in virtualizing mobile base stations so that they can consolidate as many network functions as possible in a standard hardware as needed to serve different mobile network technologies with a single virtualized mobile base station. Virtualizing the mobile base station is a challenging process because it hosts signal-processing functions in its physical

layer. Therefore, virtualization first is considered for implementation in the higher network stack layers. Considering eNodeB, which is the fourth-generation network (LTE) base station, virtualization will be implemented in layer 3 and then in layer 2 [17]. Layer 3 hosts the functionalities of the control and data plane that connects to the mobile core network. Layer 2 hosts the packet data convergence protocol (PDCP), radio link control (RLC), and media access control (MAC) network functions. Virtualizing layer 2 and 3 of the base station provide the opportunity to offer a centralized computing infrastructure for multiple base stations, which lead to lower-cost base stations because only baseband signal processing should be implemented on-site. Furthermore, service providers will benefit from sharing their remote base-station infrastructure to achieve better area coverage with minimum CAPEX and OPEX investment. There are also some efforts to centralize the L1 functionalities of several base stations [18]. They will be able to upgrade VNFs to support multiple telecommunications technologies and adapt them for new releases.

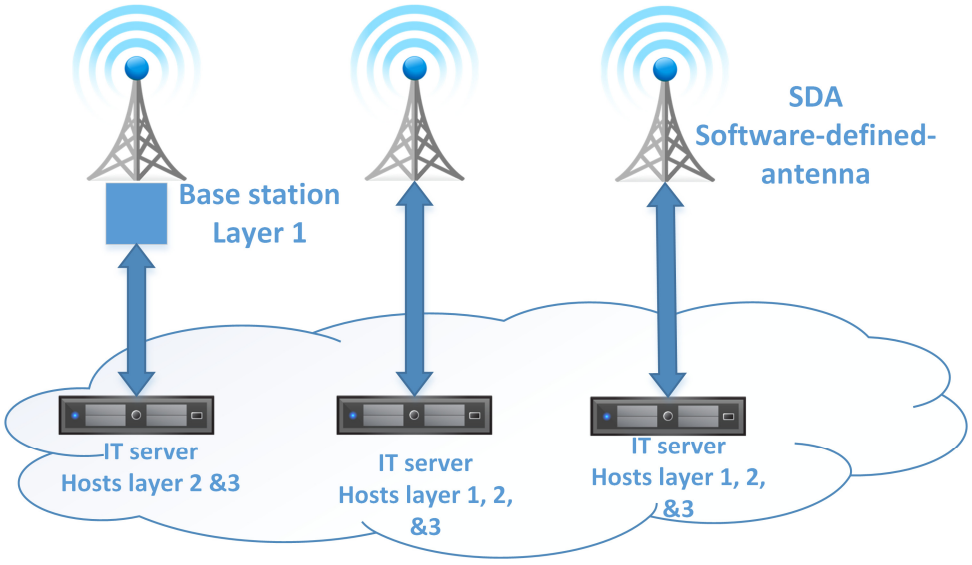

Fig. 2.9 Base station virtualization evolution.

2.9 Virtualization of the Evolved Packet Core (EPC)

The mobile core network is the most important part of the network in many access technologies. Virtualizing the functionalities within the core is the main target for NFV. The most recent core network is the evolved packet core (EPC) network. EPC has been

introduced in release 8 as a simplified all-IP core network architecture. It is designed to permit mobile broadband services by combining leading-edge IP infrastructure and mobility. Moreover, EPC is designed to support a variety of access technologies [19]. The rapid increase in connectivity demand has led service providers to undertake more CAPEX and OPEX investments beyond financial sense in their mobile core network infrastructure. From this point onward, it is becoming essential to have a flexible, robust, and easily manageable network; a network that could be scaled on-demand in real time and would be easily manageable. Virtualizing EPC offers all these benefits to service providers.

The basic EPC entities to support IP connectivity in LTE are the following:

1) The mobility management entity (MME) is the main control-plane entity in the LTE network.
2) The serving gateway (SGW) is responsible for routing and forwarding user data packets from and to the base station.
3) The packet data network gateway (PDN-GW) (PGW) ensures connectivity between the user data plane and external networks.
4) The Home Subscriber Server (HSS) is the central user information database.
5) The policy and charging rules function (PCRF) is responsible for passing and deciding the policies and charging in real time for each service and user.

2.10 Grouping EPC Entities in the NFV Environment

Implementing a virtualized EPC (vEPC) is the prime objective of the telecommunication equipment vendors. Since EPC encompasses multiple functionalities, instantiation of VNF in cloud has a tremendous effect on the performance and hence, VNFs are grouped together based on their interactions and workload. Generally, it is beneficial to instantiate each group in one physical server, or one local network depending on the workload.

A vEPC entity grouping approach that can improve performance is proposed. The approach is based on analyzing the interconnections and functionalities of vEPC entities to achieve less control-signaling traffic and less congestion in the data plane. The

proposed approach maintains the two EPC principles of flat architecture and decoupling of the control and data planes. The grouping approach divides the entities into four segments. These are listed below and summarized in Table 2.2 and illustrated in Fig.2.10.

Since understanding the LTE framework is necessary to truly grasp the benefit of the proposed grouping. Please refer to [19] for a comprehensive study of the LTE architecture.

2.10.1 Segment One

In the proposed grouping, MME is migrated with the HSS front-end (HSS FE). The HSS front-end is an application that implements all the logical functionality of HSS but does not contain the user information database. By implementing the HSS FE with the MME, authentication and authorization processes are carried out internally, without any data transactions through the network. The HSS FE ensures that all the interactions with MME happen as if the MME was accessing the complete HSS database. The HSS FE issues a query for user information data from the user data repository (UDR), which is the central user information database and stores these data temporarily in cache memory. After querying for user information, the HSS FE act as a complete user database and performs all authentication and authorization processes with the MME entity. Fig.2.11 shows the process for attaching user equipment to the LTE network. This grouping minimizes the number of network transactions that must be performed to authenticate a user because the HSS FE obtains all the required information in one query [20]. Furthermore, communication between the UDR and the HSS FE occurs through the lightweight directory access protocol (LDAP), not the diameter signalling protocol. LDAP is an application protocol used to exchange and manage distributed directory information services over IP networks. LDAP is a more efficient protocol than the diameter protocol for database information querying [20]. It is also faster and requires fewer resources than the diameter protocol [20-21]. It uses Transport Layer Security (TLS) or Secure Sockets Layer (SSL) to secure information exchanges, while diameter signaling uses an Internet Protocol Security (IPsec) connection for information exchanges. TLS/SSL requires fewer computational resources than IPsec and needs less initiation and resumption time.

Furthermore, TLS and SSL are application-layer security protocols that provide better flexibility on a virtualized platform [22].

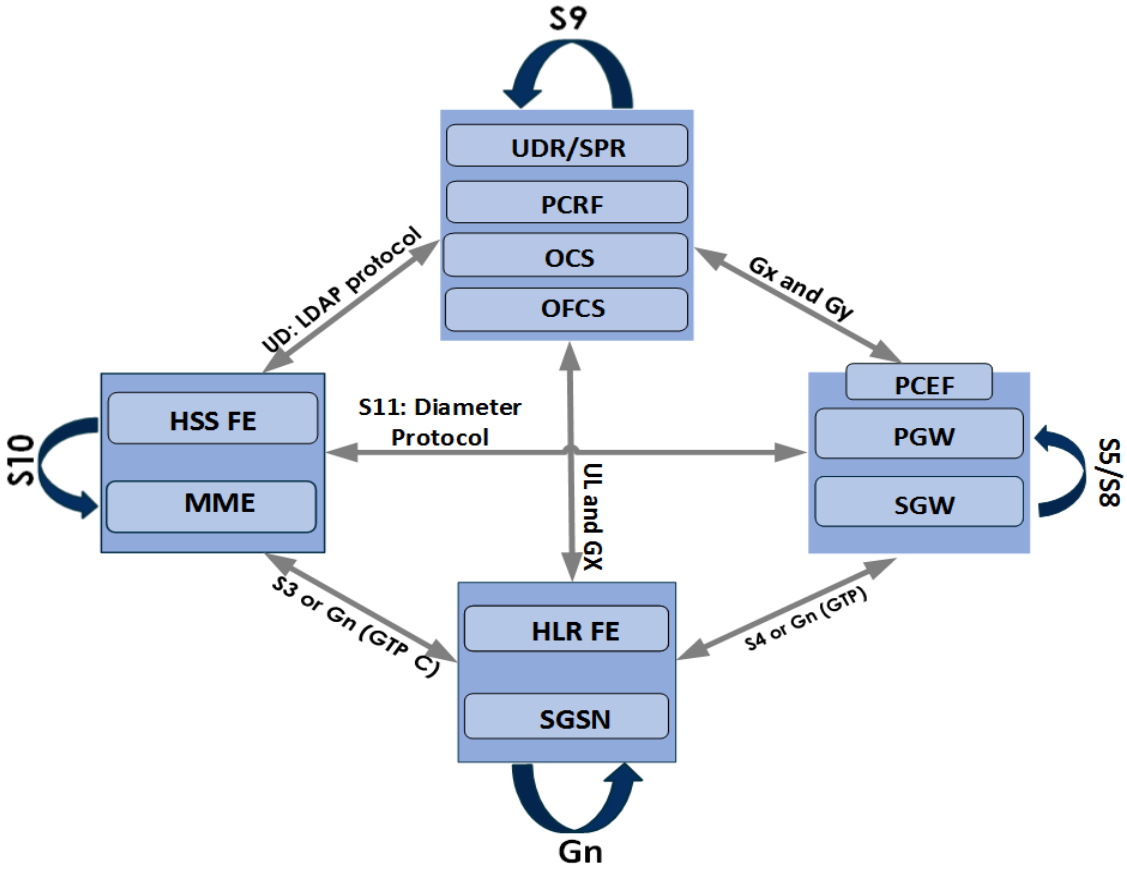

Fig. 2.10 vEPC Entities Grouping.

2.10.2 Segment Two

In the proposed grouping, the serving general packet radio service (GPRS) support node (SGSN) is migrated with the home location register front end (HLR FE). The SGSN is a serving entity, which has almost the same functionality as a combined MME and SGW. The SGSN is a network function entity existing in the GPRS core network, which permits mobile networks (2G, 3G) to transmit IP packets to external networks. It takes charge of delivering data packets to and from mobile base stations. The SGSN has user data-plane functions such as managing packet routing and transfers. Furthermore, it has control plane functions such as mobility management, logical link management, and

authentication and charging functions. The SGSN is assumed in the proposed approach as almost all service providers support 2G and 3G networks besides their 4G networks. The SGSN is not combined with any EPC entities because the SGSN has a control and data plane, which contradicts the EPC architectural decoupling principle. The HLR is the database that conserves the user information in a Global System for Mobile (GSM) core network. The HLR FE is combined with the SGSN for almost the same reasons that combine the MME with the HSS FE. Moreover, this combination enables a unified database and supports the combination of the existing SGSN with the Gn interface to the EPC system. Gn is an interface that is based on the GPRS tunneling protocol (GTP).

2.10.3 Segment Three

In the proposed grouping, the PGW is migrated with the SGW. This merging of the two data-plane entities follows the flat architecture principle to minimize the number of data-plane processing nodes. Implementing the two entities in one VM or VNF will benefit from centralized processing in the data plane and helps to overcom the processing and network bottlenecks. In this segment, user data are not routed or transferred to the PGW after being served by the SGW. Instead, the segment has direct access to the PGW, which routes it to external networks. Centralized processing in the virtualized environment enables applications to apply the CPU affinity procedure, leading to an efficient use of CPU cache memory. In addition, this merge avoids unnecessary routing through Vswitches, which are a major bottleneck in virtual environments. Higher VNF data throughput could also be achieved using direct network interface access, in order to meet the required latencies and quality of service of the PGW and the SGW. This migration leads to a better data monitoring and charging in addition to the elimination of signaling-transaction traffic between the SGW and the PGW. All signaling transactions are carried out internally.

2.10.4 Segment Four

In the proposed grouping, the UDR, the PCRF, the on-line charging system (OCS), and the off-line charging system (OFCS) are migrated. Having the UDR migrated with the PCRF leads to an efficient way of generating the policy function from user information

because the PCRF requests user information to generate the required policies for each established bearer. This approach prevents information exchange from overwhelming the network node, minimizes the latency of policy-function generation, and speeds policy enforcement to the PGW. As for the OCS and OFCS, the OCS is used to charge network users in a real-time manner, as in a pre-paid credit system, whereas the OFCS is used to charge users after the session is ended, as in billing services known as “pay as you go”. The OCS and the OFCS interact with the PCRF and the PCEF to gather information about the session and enforce charging policies to the PGW, such as terminating the communication session when the credit limit has been exceeded. In addition, this segment groups all the entities that need to interact with the OSS/BSS. Limiting fragmentation of OSS/BSS interactions leads to more efficient control over network services.

In this grouping approach, all segments are connected almost entirely through the GPRS Tunneling Protocol (GTP), not through diameter protocol interfaces. Even though the diameter protocol is an enhanced signaling protocol in the control plane of the EPC entities, it relies on the Stream Control Transmission Protocol (SCTP) and the Transmission Control Protocol (TCP) in its transport layer. SCTP and TCP are known to downgrade network performance when small amounts of data are being exchanged [23]. These network downgrades are due to the control packets, such as acknowledgment packets, which are sent to set up the connection. When the packets are small, in general, they require more computational resources to transfer the same amount of data when they are larger. The GTP relies on the User Datagram Protocol (UDP) in its transport layer, which has satisfactory performance on small-packet data-exchange connections [23]. Because control-signaling packets are small, using an approach that maintains interfaces on GTP leads to better computing performance and use of network resources. Although the proposed grouping introduces benefits in terms of minimizing control signaling traffic to avoid congestion on the networking infrastructure, it requires much computational power because most transactions are carried out internally.

Groups	Entities	Benefits
Segment one	– HSS front-end (HSS FE) – Mobility Management Entity (MME)	– Interactions between HSS and MME occur locally. – Fewer networking transactions through Vswitches. – Network transactions use the LDAP protocol, which is an efficient protocol for database information querying.
Segment two	– Home location register front end (HLR FE) – Serving general packet radio service support node (SGSN)	– Supports combining existing SGSN with the Gn interface to the EPC system – Interactions between HLR and SGSN occur locally. – Fewer networking transactions through Vswitches. – Network transactions use the LDAP protocol, which is an efficient protocol for database information querying.
Segment Three	– Packet data network gateway (PGW) – Policy and charging enforcement function (PCEF) – Serving gateway (SGW)	– Minimizes the number of data-plane processing nodes (flat architecture principle) – Helps to overcome data-forwarding and network bottlenecks – better data monitoring and charging
Segment Four	– User data repository (UDR). – On-line charging system (OCS). – Off-line charging system (OFCS). – Policy and charging rules function (PCRF)	– Unified user database; less fragmentation. – PCRF interacts locally with UDR to generate policies. – Local interaction between OCS and PCRF – Central interaction point for OSS/BSS – Fewer networking transactions through Vswitches.

Table 2.2 Grouping of EPC entities in NFV environment.

2.11 Quantitative Analysis

The main intent of the proposed grouping is reducing control-signaling traffic in the EPC, which is expected to increase exponentially by 2015 [24]. To illustrate the proposed grouping benefits, the signaling traffic generated in [25] were applied, and the reduction in the required bandwidth is estimated. In [25], the signaling traffic was generated for 15 eNBs connected to the EPC entities where traffic profile and planning parameters are shown in Table 2.3. However, the total signaling transaction traffic between the MME and the HSS in [25] was 1,039,430 transactions, and the average number of transactions

per second between the MME and the HSS was 6.2 transactions per subscriber. Using the proposed grouping of MME and HSS FE, the number of transactions decreased from 6.2 to one transaction(s) per subscriber.

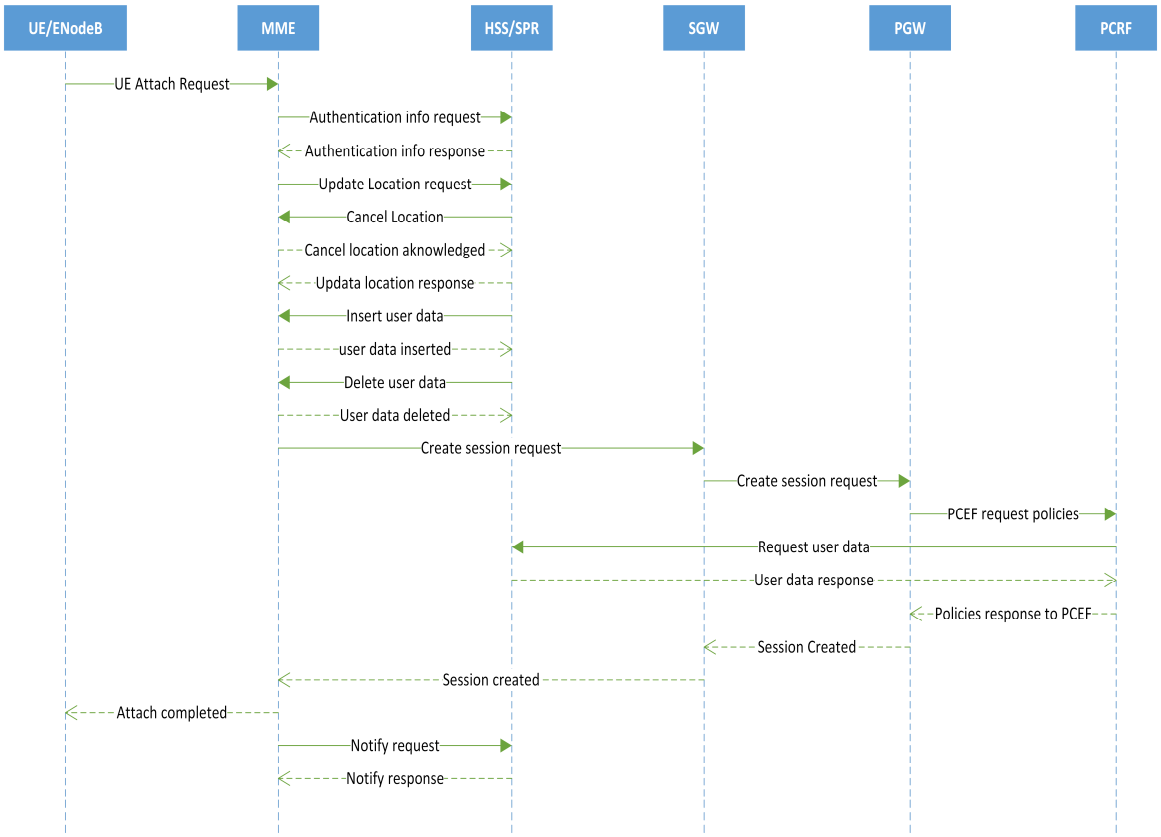

Fig. 2.11 Sequence diagram for user equipment attachment process to LTE

The reason behind this reduction in the number of transactions is the combination of all the user information in one query from the UDR [25]. Consequently, total transaction traffic was reduced to 173,239 transactions per second. In [25], signaling transaction traffic between the SGW and the PGW was 56,559 transactions. Using the proposed grouping of the SGW and the PGW, signaling transactions over the network were eliminated. Moreover, total signaling-transaction traffic for PCRF was 37,706, with an average of two transactions per bearer in [25]. As a result, the number of PCRF and UDR signaling transactions was 18,853. Because 80 percent of users had pre-paid accounts, 30,164 transactions between the PCRF and the OCS were generated. Using the proposed grouping, these transactions were eliminated because the PCRF, OCS, and UDR are

implemented in the same segment. Signaling traffic takes place between these entities and the PCEF, which is implemented with the PGW in a different segment. These results are illustrated in Table 2.4.

Traffic Profile	
Registered Subscribers	167,650
Subscribers Attached to The Network	150,878
Busy Hour Session Attempts	64,940,898
Simultaneous Evolved Packet System Bearers (EPSB)	18,853
Planning Parameters	
Mean Session Time	180 sec
Handover Ratio	0.4
Dense Area Attached Subscriber Ratio	0.9
Average EPSB session duration	900 sec
Busy Hour Traffic Ratio	0.15
Retransmission factor	0.25
Pre-paid Accounts	80%

Table 2.4 Traffic profile and planning parameters [25].

Transactions between Core Elements	Signaling (transactions per sec) [25]	After Grouping
MME, eNBs , and S-GW	175,332	175,332
S-GW and P-GW	56,559	0 (Internal transactions)
MME and HSS	1,039,430	173,239 to UDR
PCRF and P-GW	37,706	37,706
PCRF and UDR	18853	0 (Internal transactions)
PCRF and OCS	30164	0 (Internal transactions)
Total Traffic	1358044	386277

Table 2.3 Signaling traffic before and after grouping.

2.12 Chapter Contribution

In this chapter, network function virtualization is introduced with a discussion about the challenges and requirements of its use in mobile networks. In particular, an architecture for NFV framework entities in the virtual environment is proposed. Moreover, a grouping criterion is defined to bundle multiple functions of the virtualized evolved packet-core in purpose of reducing the signaling traffic between the vEPC entities. The analysis shows that the proposed grouping can reduce the network control signaling traffic by 70 percent.

Bibliography

[1] M.A. Sharkh, M. Jammal, A. Shami, A. Ouda, "Resource allocation in a network-based cloud computing environment: design challenges", IEEE Communications Magazine, vol. 51, no. 11, pp. 46–52, November 2013.

[2] Cisco, "Cisco Visual Networking Index Global Mobile Data Traffic Forecast Update 2012-2017", http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/ white_ paper_c11-520862.pdf, 2012.

[3] ETSI, "Network Function Virtualization: An Introduction, Benefits, Enablers, Challenges, & Call for Action," portal.etsi.org/NFV/NFV_White_Paper.pdf, 2012.

[4] ETSI," Network Function Virtualization: Network Operator Perspectives on Industry Progress", portal.etsi.org/NFV/NFV_White_Paper2.pdf, 2013.

[5] ETSI, "Network Function Virtualization: Architectural Framework", http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.01.01_60/gs_NFV002v010101p.pdf, 2013.

[6] M. Jammal, T. Singh, A. Shami, R. Asal, Y. Li, "Software defined networking: State of the art and research challenges," Computer Networks, Elsevier, July 2014.

[7] 6WIND, "6WIND and Red Hat Bring High-Performance Data Plane Processing to NFV", http://www.6windblog.com/6wind-and-red-hat-bring-high-performance-data-

plane-processing-to-fv/?utm_source=rss&utm_medium=rss&utm_campaign=6wind-and-red-hat-bring-high-performance-data-plane-processing-to-nfv, 2013.

[8] Ericsson, "Telefonica and Ericsson partner to virtualize networks," http://www.ericsson.com/news/1763979 , press release February 2014.

[9] Heavy Reading, "Implementation of SDN & NFV in the WAN," https://networkbuilders.intel.com/docs/HR-Intel-SDN-WP.pdf, 2013.

[10] Intel, "Intel Data Plane Development Kit (Intel DPDK) Overview Packet Processing on Intel Architecture," http://www.intel.com/content/dam/www/public/us/en/documents/presentation/dpdk-packet-processing-ia-overview-presentation.pdf ,2012.

[11] AT&T, "AT&T Vision Alignment Challenge Technology Survey, AT&T Domain 2.0 Vision White Paper", http://www.att.com/Common/about_us/pdf/AT&T%20Doma in%202.0%20Vision%20White%20Paper.pdf ,November 2013.

[12] V. Ashktorab, T. Seyed, "Security threats and countermeasures in cloud computing", International Journal of Applications and Innovation in Engineering and Management (IJAIEM), Vol. 1, No. 2, pp. 234-245 , 2012.

[13] Cloud Security Alliance, "The Notorious Nine Cloud Computing Top Threats in 2013", https://downloads.cloudsecurityalliance.org/initiatives/top_threats/The_Notorious_Nine_Cloud_Computing_Top_Threats_in_2013.pdf, 2013.

[14] H. Tseng, L. Lee, J. Hu, T. Liu, J. Chang, W. Huang,"Network virtualization with cloud virtual switch", Proceedings, 2011 IEEE 17th International Conference on Parallel and Distributed Systems (ICPADS), pp. 998–1003, 7–9 December 2011.

[15] T. Taleb and A. Ksentini, "Follow Me Cloud: Interworking Federated Clouds & Distributed Mobile Networks", in IEEE Network Magazine, Vol. 27, No. 5, Sep./Oct. 2013. pp. 12 – 19

[16] T. Taleb, M. Corici, C. Parada, A. Jamakovic, S. Ruffino, G. Karagiannis, M. Karimzadeh, and T. Magedanz, "EASE: EPC as a Service to Ease Mobile Core Network," in IEEE Network Magazine.

[17] ETSI, “Network Function Virtualization: Use Cases”, http://www.etsi.org/deliver/etsi_gs/NFV/001_099/001/01.01.01_60/gs_NFV001v010101p.pdf, 2013.

[18] T. Taleb, “Towards Carrier Cloud: Potential, Challenges, & Solutions,” in IEEE Wireless Communications Magazine, Vol. 21, No. 3, Jun. 2014. pp. 80-91.

[19] ETSI, “Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS);LTE;Network architecture”, 3GPP TS 23.002 version 11.6.0, Release 11, 2013.

[20] Bhadrapur, P., “HSS Front-End Implementation for a Large-Scale Common HLR/HSS”, M.S. thesis, Department of Information Technology, Uppsala University, Uppsala, Sweden, 2012.

[21] Wang, X., Schulzrinne, H., Kandlur, D., Verma, D., “Measurement and analysis of LDAP performance”, IEEE/ACM Transactions on Networking, vol. 16, no. 1, pp. 232–243, 2008.

[22] Arshad, F.A., Modelo-Howard, G., Bagchi, S., “To cloud or not to cloud: a study of trade-offs between in-house and outsourced virtual private networks”, 2012 20th IEEE International Conference on Network Protocols (ICNP), pp. 1–6, 30 October–2 November 2012.

[23] G. Wang, T.S.E. Ng, “The impact of virtualization on network performance of the Amazon EC2 data center”, Proceedings, 2010 IEEE INFOCOM, pp. 1–9, 14–19 March 2010.

[24] Nokia Siemens Networks, “Signaling is Growing 50% Faster than Data Traffic” http://nsn.com/system/files/document/signaling_whitepaper_online_version_final.pdf, 2012.

[25] D. Dababneh, “LTE Traffic Generation and Evolved Packet Core (EPC) Network Planning”, M.S. thesis, Department of Systems and Computer Engineering, Carleton University, Ottawa, ON, 2013.

Chapter 3

3. Towards an Elasticity Framework for Legacy Highly Available Applications in the Cloud

Elasticity is a key characteristic of cloud computing where provisioning of resources can be directly proportional to the runtime demand. Legacy highly available applications typically rely on the underlying platform to manage their availability by monitoring heartbeats, executing recoveries, and attempting repairs to bring the system back to normal. Migrating such applications to the cloud can be particularly challenging, especially if the elasticity policies target the application only, without considering the underlying platform contributing to its high availability (HA). A comprehensive framework for the elasticity of highly available applications that considers the elastic deployment of the platform and the HA placement of the application's components is discussed in this chapter.

3.1 Introduction

The promise of having simplified information technology (IT) infrastructure and an on-demand provisioning model is a key feature that enabled the wide spread adoption of cloud computing by the enterprise [1]. From the perspective of the cloud provider offering infrastructure as a service (IaaS), elasticity is both a cloud feature and a service. It is a cloud feature that allows the infrastructure to absorb the addition or removal of physical resources in an intuitive manner. It is a cloud service offered to the cloud tenants, allowing the virtual resources allocated to their applications to grow and shrink in proportion to the runtime demand. On the other hand, from a cloud tenant perspective the elasticity service instituted by the provider becomes a feature of their cloud deployed application(s).

Elasticity spans across multiple layers of the cloud, as shown in Fig. 3.1. Hence, a comprehensive elasticity solution must consider all the cloud layers. The proposed solution is oriented toward achieving elasticity for the revenue generating and business critical applications hosted in the cloud. Such applications require high availability (HA)

to the magnitude of five nines (99.999%) [2], undergoing less than six minutes of downtime per year, including maintenance and upgrades. However, major cloud providers (e.g., Amazon AWS) offer a service level agreement that guarantees a lower HA level (99.95%), which leaves room for several hours of outage per year. Such outages can entail millions of dollars in direct monetary losses, in addition to the reputation damage [3].

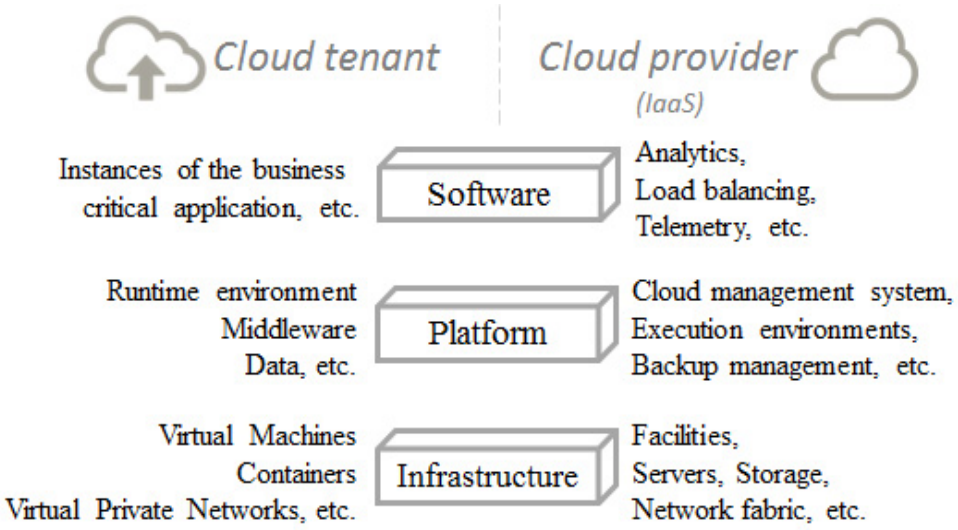

Fig. 3.1 The different perspectives of the cloud levels.

In fact, the cloud tenants that thrive at maintaining the high availability of their applications [4] are the ones that leverage the HA enabling features of the cloud provider (e.g., elastic load balancing). In addition, they add their own application specific components [5] that complement the HA solution. Such applications are cloud-fitted applications composed of stateless components that can be deployed behind redundant load balancers while the system state is maintained in a replicated and distributed storage. Another approach garnering attention revolves around the construction of applications in a platform as a service (PaaS) environment, which can in turn manage scaling and data replication. Nevertheless, a large number of business critical applications are neither conceived for the cloud nor cloud-fitted [6]. In that sense, they are considered legacy applications for the cloud. Such applications have typically been deployed in a datacenter (DC) and their high availability is maintained by specialized HA clustering solutions

(henceforth, referred to as HA middleware [7, 8]). This middleware is responsible for monitoring the application's components and reacting to their failures. Such solutions can ensure an availability level of fives nines of the applications due to the fast recovery and frequent heartbeat monitoring. However, because HA middleware is not conceived for the cloud but rather for static deployments within the datacenter, efficiency comes at a hefty price of rigidity and complexity. When deployed in a virtual DC of interconnected VMs that can grow and shrink on demand, a static middleware deployment cannot cope with such dynamic changes, which destabilizes the HA status of the application. Another important factor, and often neglected in elastic deployments is the dynamic HA-aware scheduling for the addition and removal of the VMs hosting the application's components. The scheduling solution should consider the non-functional requirements such as HA by ensuring that the added components are not placed in the same failure domain, yet the functional requirements of the application's components must not be violated, such as delay tolerance of inter-component communications. For instance, deploying in different servers, racks and datacenters, the replicated components can certainly protect against larger catastrophe scopes, as shown in Fig.3.2.

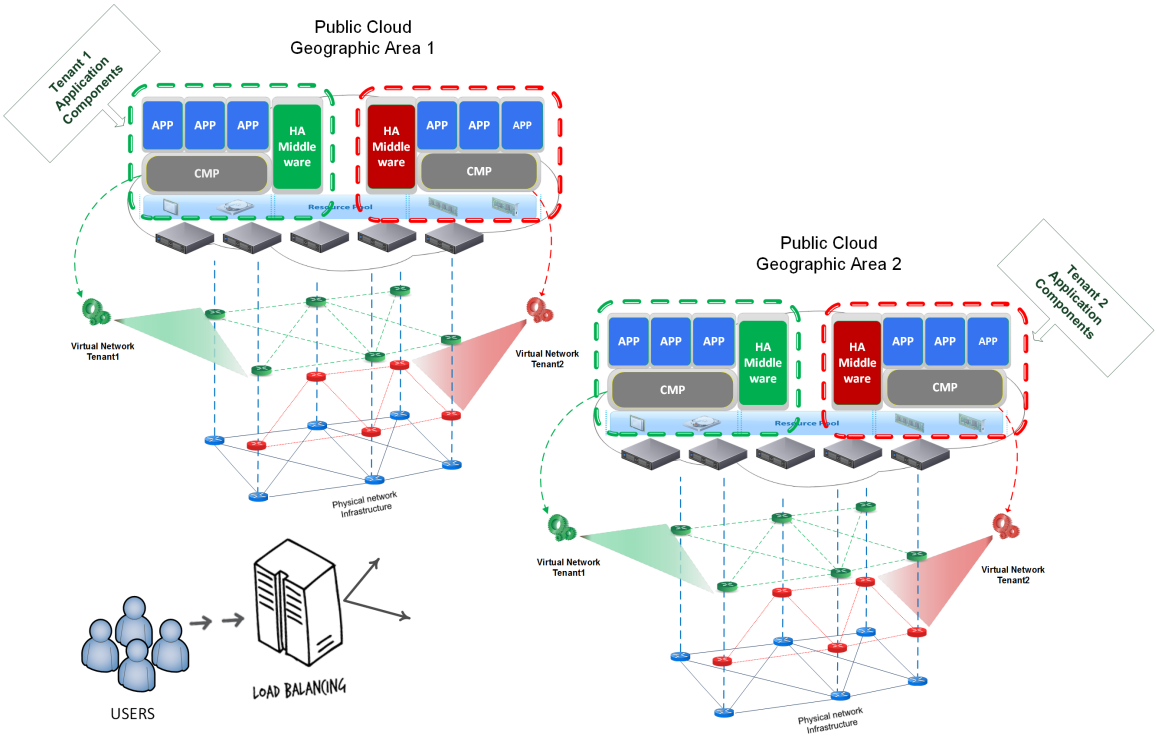

Fig. 3.2 Application components deployed in different datacenters.

Nonetheless, it should take into consideration the functional (e.g., collocation dependencies for shared libraries, delay tolerance among dependent components, etc.) and non-functional requirements such as HA.

A comprehensive elasticity solution should address the HA-aware scheduling of the added/removed VMs, the dynamic deployment of the middleware managing the availability of the applications, and the runtime addition/removal of the application instances without service interruption. This ought to be handled in an intuitive and user-friendly manner to leverage the simplicity model affiliated with the cloud. This chapter proposes a framework that considers all three aspects while retaining a higher level of abstraction by delineating a domain specific language for the HA and elasticity requirement specification.

3.2 High Availability Middleware and Scheduling

With the wide variety of smart devices like smart mobile phones, tablets and laptops information technology (IT) has become a necessity in our daily activities such as social connectivity, medical services, and other uses. Furthermore, IT has become an essential asset in the enterprises business models after shifting from legacy to online service delivery paradigm. Consequently, the high availability of IT system becomes a crucial aspect to maintain always-on and always-available services for all users. Most of software and system engineers have used proprietary platforms during their development and implementation phases to achieve the desired high availability level. These systems have suffered from vendor lock-in and platform dependency. In order to provide HA systems with reduced lock-in to a specific vendor different parties have agreed on the need for standards that define HA system interfaces.

A group of telecommunication and IT leading companies formed a consortium called “Service Availability Forum” (SAForum) in order to create standards for high availability systems. The SAForum has defined standards to leverage HA systems on commercial off-the-shelf (COTS) equipment. The SAForum standards enables HA systems deployment on standard IT platforms of different architectures, such as x86, ARM, and ATCA, which leads to facilitating the portability and interoperability of the

HA application across various standard compliant platforms [7]. More specifically, the SAForum delineates standards and guidelines for the design of an HA middleware that manages the availability of the services provided by an application. It attains the desired application's availability through the management of the redundant components, and by seamlessly swapping the faulty component workload to a redundant and healthy one upon detecting a failure. The SAForum middleware offers several essential application interface specification (AIS) services including: (1) the availability management framework (AMF) responsible for monitoring the application's components and orchestrating their recoveries [9], and (2) the software management framework (SMF) that is responsible for carrying software upgrades supporting the automated rolling upgrade that allows the incremental upgrade of the applications components. Moreover, it minimizes the downtime by synchronizing with the AMF [10]. In turn, the AMF leverages the redundant replicas of a given component by dynamically switching over the workloads to the upgraded replicas while the older-version replica is upgraded simultaneously. The applications that integrate with the SAForum middleware can also benefit from other services such as distributed messaging, check pointing, log and other essential for distributed HA applications.

The SAForum specification has succeeded to find its way in different industries, with various implementations appearing in the market. One of the well-matured and highly adopted implementation is the OpenSAF project. It is an open source HA middleware project launched in 2007 by Motorola. Then a multi-industry consortium was formed to maintain and manage further development of the project. Leading companies such as Ericsson, Emerson Network Power, and Nokia Siemens Networks have enunciate support for this initiative as founding members of OpenSAF. OpenSAF is a SAForum compliant project devoted to implement the SAForum application interface specification standards. With openness and standard AIS, OpenSAF has prevented any vendor-lock and has established a fast pace adoption as HA operating environment to be utilized by multi-industry demanding HA solution. The OpenSAF project is considered the de facto implementation of the SAForum standards [8].

3.2.1 OpenSAF Cluster

OpenSAF system architecture defines two node types in the HA cluster; the System controller and the payload nodes [8]. The system controller node hosts different OpenSAF services centralized functions and acts as a management entry point for the whole cluster [8]. The payload node hosts the OpenSAF services node purview functions conjointly with HA application's components, which intended to it. Despite that, HA application's components are mainly entitled to run on the payload nodes, they also could be configured to run on the controller nodes.

OpenSAF mainly is developed with C language, and it is persistent with linux standard base APIs usage. With this practice in OpenSAF development, it supports portability across various linux distributions. Consequently, various hardware architecture hosting these operating systems are supported such as IT standard servers and advanced telecommunications computing architecture (ATCA) equipment. OpenSAF deployment process consists of several steps that must be followed for a successful admission of a newly added node to the OpenSAF cluster. The OpenSAF installation and configuration are analyzed and grouped in six steps allowing concurrent execution.

- Step One: OpenSAF code version is selected and downloaded by cloning the project from mercurial repository, distributed version control system.
- Step Two: Defines the configuration of the installations and code compilation. In this step, system admin should install all the prerequisites packages and copy the OpenSAF files to the designated location on the system.
- Step Three: The system administrator defines the services of OpenSAF to be installed and the procedure of their configuration. For instance, the administrator defines the protocol (TCP/IP or TIPC) to be used for the message distribution service (MDS). MDS is a nonstandard service that provides the inter-process communication infrastructure within different OpenSAF nodes and services.
- Step Four: The system administrator manages the system users and files permissions configurations. In this step, the appropriate privileges are granted by modifying the sudoers file.

- Step Five: The node specific configuration should be applied, such as specifying the node slot ID and IP address, which should be used during communication between the nodes.
- Step Six: The information model management (IMM) configurations should be configured and modified to reflect the desired OpenSAF cluster architecture. Finally, the nodes are rebooted to apply the permanent configuration and start the OpenSAF Services.

Applying all these steps on each node is a challenging process. The system integrator deploying OpenSAF has to synchronize the configuration files between the cluster nodes and assign a unique name, slot ID, and so forth, for each node while verifying that conflicts have been nullified. Reducing the time and the complexity of deploying OpenSAF, and eliminating human error is essential in a dynamic cloud setting. The SMF framework, while it is efficient for the upgrade of the applications, but it cannot upgrade the middleware itself [10].

Development and operation (DevOps) tools have existed in the market to ease and facilitate the automated deployment and configuration of software applications. Puppet labs [11] and Chef [12] are examples of highly adopted IT configuration management systems (CMSs). They are used to distribute and apply configuration resources to the computing nodes (servers or VMs). The configuration resources define what packages and software needed to be installed on the system. Furthermore, they define the configuration attributes of the installed software and manage the users' access rights on the system. These configuration resources highly depend on the package management system used by the operating system of the computing node. Deploying OpenSAF through the CMS is a challenging process. Such CMS tools apply the system changes based on static manifests or cookbooks that are not meant for dynamic deployments, particularly where the configuration attributes such as IP address and node names/IDs are assigned at runtime. Furthermore, OpenSAF does not follow any package management system standards. It is provided as a source code and part of its configurations are applied during code compilation process. To facilitate the automated deployment of OpenSAF using CMSs, an additional module or plug-in should be developed. However, the

development of additional module or plug-in does not eliminated the human factor of the process since the CMSs lack the domain specific intelligence for automated dynamic configuration generation. This issue hinders the elastic deployment of OpenSAF HA cluster and requires additional system components to facilitate it.

3.2.2 High Availability Scheduling

Cloud applications typically have a multi-tier architecture serving a broad range of users. The placement of the application's components may have an enormous impact on its availability; for example, the redundant instances of a database must be placed as far apart as possible in different availability zones to avoid losing multiple instances in a single zone failure. Nevertheless, in terms of delay tolerance those databases would be serving requests from dependent components with a constrained latency. Hence, situating the database in such a way to maximize the availability irrespective of its dependent components may yield suboptimal results. In previous work, HA-aware scheduling for cloud applications [13] has been proposed to resolve this issue. Jammal et al. [13] present various patterns used by the scheduler to place the redundant deployments. These practices target the elimination of single point of failure caused at the level of VM, cluster, or datacenter by utilizing geographically distributed datacenters to deploy new components. Yet a similar issue is experienced when implementing the elasticity framework whereby not only the added components, but also those removed (when scaling down) need to be cautiously selected, as these choices impact the availability of the application. Moreover, the solution space is more constrained in this environment because, rather than being an initial deployment, it is a variation of an existing deployment. Therefore, a need exists for an elastic HA-aware scheduler that is defined and integrated with the proposed elasticity framework.

3.3 Elasticity Framework

The proposed approach targets elasticity from the cloud tenant perspective. In order to achieve elasticity for the tenants' highly available applications, all three levels (1) the infrastructure, (2) the platform, and (3) the application software have to be elastic in response to the variation of the runtime workload. In reality, the visibility and control

associated with each of these are decoupled. Therefore, the proposed elasticity framework relies on different entities to define a comprehensive elastic HA solution. The framework requires visibility of the cloud infrastructure in terms of the different availability zone, and the communication latency between zones, as well as the ability to monitor the runtime workload. Hence, it can be managed either by the cloud provider or by the tenant if the cloud provider exposes this information. An overview of the proposed elasticity framework ecosystem is illustrated in Fig3.3.

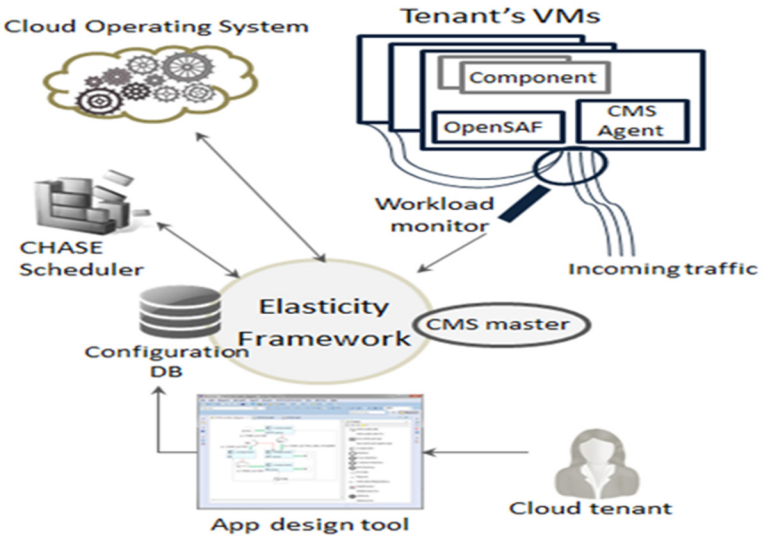

Fig. 3.3 Overview of the elasticity framework.

3.3.1 Application Design and Elasticity Requirement Specification

In an application centric approach, users should embed the elasticity and HA requirements at the application specifications. Therefore, a domain specific language is derived based on the unified modelling language (UML) component diagram, which allows the description of the application in terms of components and interfaces. A component can provide or require an interface from another component. In order to express the deployment and HA requirements of the application, the component diagram is extended to include more interfaces (e.g., the proxy interface) and dependencies (e.g., the colocation dependencies) [14]. In addition, the specifications of HA-oriented requirements such as the redundancy models and the number of replicas of a given component are allowed.

This high-level information will later be transformed into a middleware specific language (based on the extensible markup language) known as the IMM configuration and serve as guidelines for the HA middleware to instantiate, monitor, and react to failures. Furthermore, the UML based language is extended to enable the specifications of elasticity attributes at the interface level (as shown in Fig. 3.4.). The values of these attributes are extracted by the elasticity framework, and aid configuration of the monitoring and telemetry components in order to trigger the proper elasticity action. Moreover, the collocation and other forms of dependencies dictate which components are installed in the same VMs, and the number of needed VMs. The next step would be scheduling the VMs for placement.

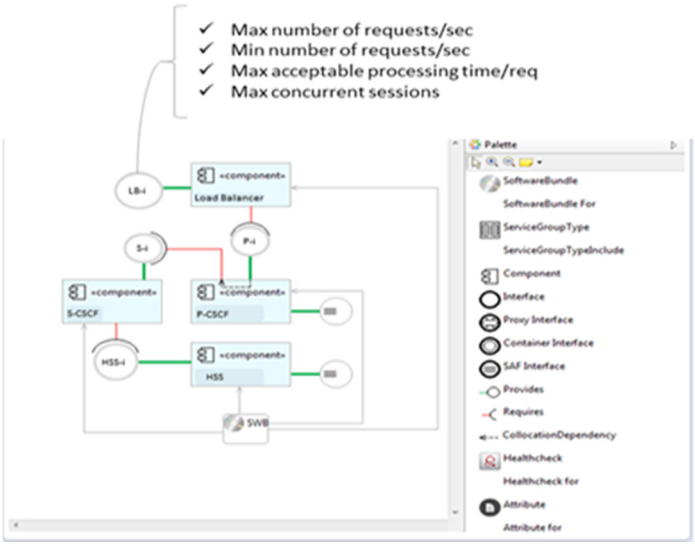

Fig. 3.4 Snapshot of the application description interface.

3.3.2 Elastic HA-Scheduling

The cloud infrastructure can be viewed from a hierarchical perspective as an aggregation of several datacenters (DCs), each hosting a set of racks composed of servers where the VMs are placed (as illustrated in Fig. 3.5). In previous work [13], an HA-aware scheduling that can place the VMs hosting interdependent components of an application in an HA optimized manner is persented. The scheduling approach only targeted the initial HA placement and not the subsequent runtime changes induced by a fluctuating workload. Hence, the previous approach is extended to include the concept of elastic

scheduling. The elastic scheduling concept is based on three main steps, its algorithm pseudo code presented in Fig.3.7.

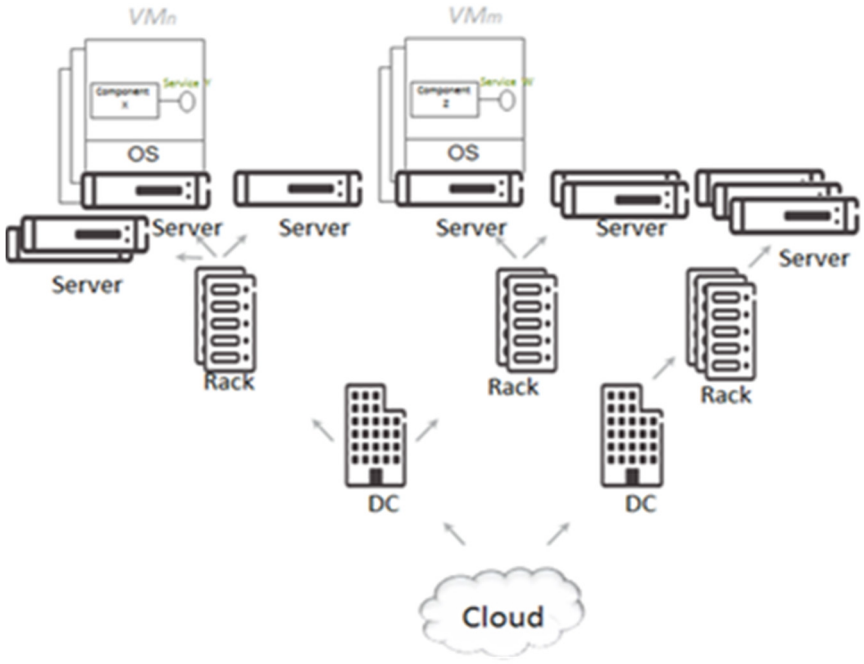

Fig. 3.5 The cloud infrastructure hierarchical overview.

3.3.2.1 Identifying the Constraints

This step includes three sub-steps:

1) *Identifying the minimum number of instances* of a given component type that needs to be added or removed in response to the workload changes. This calculation is based on the information provided by the application design phase presented in the previous section.
2) *Identifying the anchors* for the components to be added or removed. The anchors are defined by a functional dependency that exists between components; dependencies may introduce distance limitations between the component and its anchor. For example, if an extra instance of a database is needed, then it will be

anchored by (1) the other instances of the database that require synchronization, and by (2) the dependent components on the database.

3) *Identifying the orbital area* that is defined by the region where the newly added component can be placed. This area is bounded by the delay tolerance between the components. For example, when adding a new instance of the database it cannot be placed too far apart from its peers nor its dependents. The same applies when removing an instance, where the scheduler should make sure that the dependents connected to that instance can reestablish the connection to the sponsor without violating any delay constraints. Fig. 3.6 illustrates the conceptualization of the anchors in relation to the orbital area. A component can have multiple peers and multiple dependents; hence, the orbital distance must be carefully calculated. Distance unit depends on the cloud specifications: it can be the number of hops (between switches/routers), or an availability zone that determines the delay incurred by firewalls and load balancing.

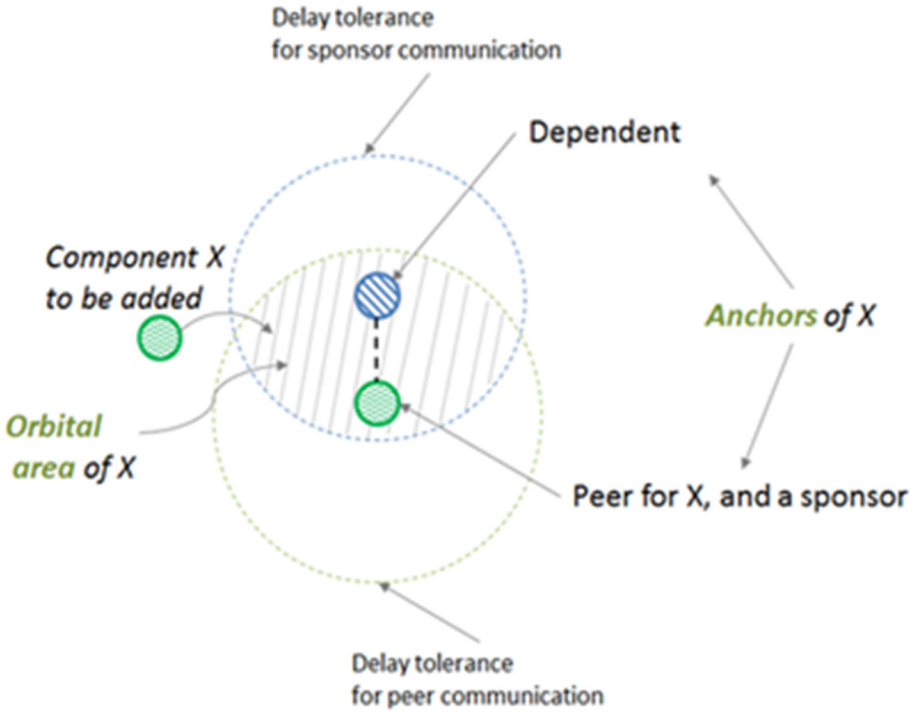

Fig. 3.6 The orbital distance of a given component.

3.3.2.2 Maximizing the Availability of the Application

First, the scheduler filters out the cloud regions that cannot be used to host the added components. Next, it selects the placement that maximizes the availability of the application in general. This is based on two main criteria.

1) *Minimize the frequency of failure* based on the mean time to failure (MTTF) of the software components, the VMs, the hypervisors, the computing servers, the racks, the datacenter facility, and the inter/intra-datacenter connectivity, the scheduler will select the placements that maximize the MTTF of the newly added components, and the average MTTF of the application.
2) Minimize the impact of failure: decreasing the failure rate depends not only on maximizing the MTTF, but also on two other factors considered by the scheduler. (1) It should minimize the mean time to recover (MTTR) and (2) favor lightly saturated (e.g., datacenter, or availability zone) regions over higher ones. The MTTR is determined by the outages caused due to failures.

3.3.2.3 Optimizing the Placement for Performance and Other Factors

Optimizing the placement for performance implies that the scheduler must incorporate the intelligence to consider several other factors alongside HA. Such factors include:

1) Assessing the workload proximity where the added components are placed in regions that are close to where the surge in the workload occurred (if the surge is regional).
2) Considering data proximity where the added components would be placed next to the data in case the application is data-driven. In this latter case, the computing components often communicate with databases or storage agents. Finally, other factors, such as legal agreements forbidding the proximity of the tenants' data in certain geographic regions, might influence the placement. In summary, it is not feasible to have an HA-centric scheduler that is agnostic to the other factors that could affect the placement of the components; instead, an HA-aware scheduler is more suitable.

Algorithm 1 HA Elastic Scheduling Algorithm

INPUT: $= (ClusterInfo, CloudStruct, WorkLoad, Comp)$

```
1: begin:
2: [Dependents,Spons,Peers,Cap,Resrc] = getClusterMemberInfo(Comp)
3: while ScaleApp do
4:     ScalingFactor = 1
5:     if WorkLoad > (ScalingFactor * PeersCount * Cap) then
6:         ScalingFactor++
7:     end if
8:     if WorkLoad < (ScalingFactor * PeersCount * Cap) then
9:         if WorkLoad < ((ScalingFactor − 1) * PeersCount * Cap) then
10:            ScalingFactor−−
11:        else
12:            Break
13:        end if
14:    end if
15: end while
16: PotentialServers = []
17: Servers = getServers(CloudStruct)
18: for i=1 to Servers.size do
19:    if delay(Dependents.Hosts,Servers[i])<tolerance(Comp.type, Dependents)
    then
20:        if delay(Spons.Hosts,Servers[i])<tolerance(Comp.type, Spons)
    then
21:            if Servers[i].AvailableResources >= Comp.type.Resrc) then
22:                PotentialServers = PotentialServers + Servers[i]
23:            end if
24:        end if
25:    end if
26: end for
27: PlacementAvail = []
28: for i=1 to PotentialServers.size do
29:    PlacementAvail += PotentialServers[i].MTTF/(PotentialServers[i].MTTF+
    PotentialServers[i].MTTR)
30: end for
31: for i=1 to abs(ScalingFactor) do
32:    if ScalingFactor > 0 then
33:        NewComp = Comp(Comp.type)
34:        PlacementIndex = MaxAvailability(PlacementAvail)
35:        NewComp.Scheduled = PotentialServers[PlacementIndex]
36:        PotentialServer− = PotentialServers[PlacementIndex]
37:    else
38:        if ScalingFactor < 0 then
39:            PotentialDropComp = getComp(Comp.type)
40:            DropCompServers = getServres(PotentialDropComp)
41:            CompsAvail = CompAvail(MttfWeight, MttrWeight, PotentialServers)
42:            CompIndex = minAvailability(CompsAvail)
43:            DropComp.Schedule(CompIndex)
44:        end if
45:    end if
46: end for
47: end
```

Fig. 3.7 HA elastic scheduling algorithm.

3.3.3 Automated Elastic Multi-Level Deployment

The deployment and removal of the VMs and their components necessitates changes in the infrastructure, platform, and application from the tenant's perspective.

3.3.3.1 Infrastructure Elasticity

Elastic cloud environment offers the ability to dynamically scale actual amount of resources used by a user over time, without previous knowledge about the future resources demands. To achieve high performance of a cloud application with minimum cost, the application should utilize the on-demand resource allocation service provided by the cloud operating system. The resource allocation can be assigned in a vertical or horizontal scaling manner. The vertical scaling of resources is the process where more computation resources such as virtual central processing unit (vCPU), memory, and storage capacity are assigned for the same VM. This kind of scaling increases the performance of application's component, but it is limited to the physical machine resources' capacity. Moreover, this can increase the risk of service interruption when failure occurs on the machine. As to horizontal scaling, it is the process where a new instance of the application's component is instantiated. This kind of scaling is the preferable solution in cloud environment because increasing the number of instantiated component instances maximizes the application reliability and performance.

Infrastructure horizontal elasticity is achieved with the help of the cloud operating system. The cloud operating system processes the requests from the elasticity framework to add/remove VMs for a particular tenant based on the recommendation of HA scheduler. The cloud operating system will add/remove the VMs and facilitate their connectivity. The VMs are spawned from special images that yield HA-enabled VMs. These HA-enabled VMs are equipped with an installation of the HA middleware that is neither configured nor instantiated, and collaborate with an agent of the configuration management system (CMS).

3.3.3.2 Platform Elasticity

Platform elasticity in the context of this work entails the elasticity of the HA middleware, which is the dynamic addition and removal of the middleware cluster nodes inside the tenants vDC. While the HA middleware namely OpenSAF, is capable of the deployment/removal of the applications throughout the HA cluster, it is unable to install and configure itself on the newly added VMs. Nevertheless, the tenant application's components rely on the middleware to manage their HA and other aspects such as reliable messaging. Therefore, it is essential that the HA middleware cluster grows and shrink in a synchronized manner with the tenant's vDC. To aid this function a CMS is used, namely Puppet, to perform this task. Nevertheless, the CMS handles the orchestration and deployment based on static manifests grouped into modules; the manifests include the classes' definitions and declarations. A class declaration contains the puppet code that performs a given functionality.

This code is defined in a declarative way, which makes the CMS versatile and enables the platform to manage itself independently. For instance, the manifest can include a class that ensures a given package is installed and instantiated. This information is read by the Puppet master and then pushed to the puppet agent as a set of instructions to be performed. The puppet agent will perform the required action according to the environment where it is deployed. For instance, according to a given Linux distribution it can figure out how to fetch and deploy the package. The manifest can retain its prior configuration if the Linux distribution changes. However, due to the static nature of this manifest, an extension to the CMS is proposed via the addition of more agents. These agents are obliged to dynamically change the content of the manifest to reflect the changes in the system (for example, the need to add or remove more instances of OpenSAF). Moreover, the OpenSAF cluster configuration is included in a specific OpenSAF IMM configuration file; when the HA cluster grows or shrinks, this configuration file must be regenerated or modified to reflect this change. In summary, a supplemental agent that acts as the OpenSAF configuration generator is required to assist modification. As soon as the new configuration files are generated, the CMS is

summoned to replace the old ones. As a result, several agents are defined that complement the CMS with the dynamic ability to scale up/down the HA middleware.

1) ***Request listener agent:*** is an agent that listens to cluster node addition or removal requests. Once it receives the request, it analyzes whether the added node should be a controller or a payload. Then it forwards this information to the configuration generation agent.
2) ***Configuration generation agent:*** receives instructions from the request listeners, and generates a new middleware configuration to reflect the needed change by either adding or removing the nodes description from the middleware configuration file.
3) ***Change applier agent:*** will dynamically modify the puppet manifest files to reflect the changes in the system that puppet needs to enforce.
4) ***Change enforcer agent:*** will make sure that the puppet agents apply the changes across the VMs in a consistent manner.

3.3.3.3 Application Elasticity

Application elasticity is achieved with the software management framework (SMF) [10] of the OpenSAF middleware. SMF is conceived for the runtime upgrade of HA applications. Nevertheless, it requires an upgrade campaign file that serves as a roadmap for the upgrade [15]. An upgrade can be performed in a single step, or in a rolling manner where nodes are upgraded sequentially. Once a request for an upgrade is issued, the upgrade campaign generator agent reads the information specified in the application design file, and accordingly generates an upgrade campaign that satisfies the requested change. This upgrade campaign is then forwarded to SMF to execute the upgrade.

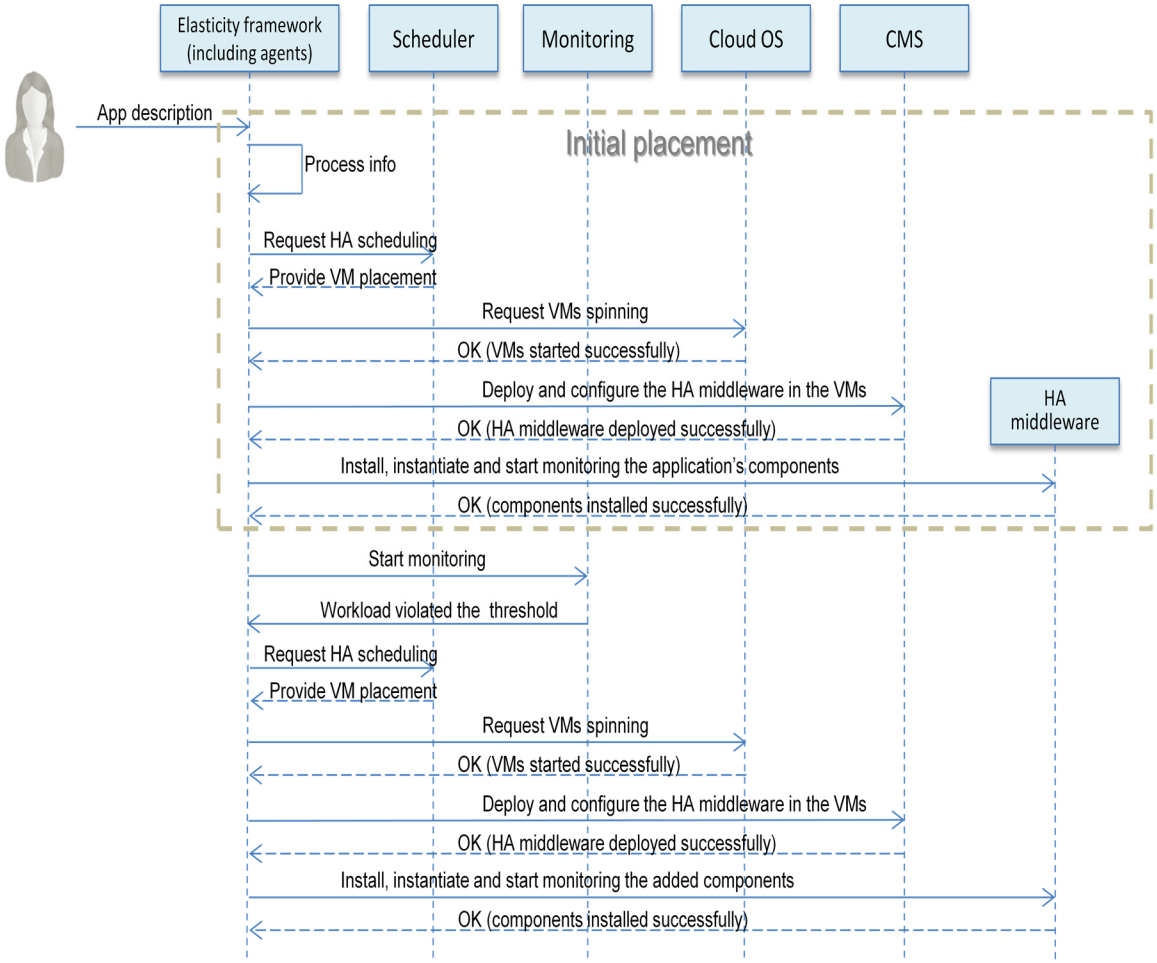

Fig. 3.8 Elasticity framework workflow.

3.4 Framework Workflow and Implementation

The workflow in the elasticity ecosystem with the different interaction between the various elements is illustrated in Fig. 3.8. These interactions begin with the cloud tenant providing a description of the HA application. Subsequently, this description is interpreted by the elasticity framework, which analyzes the required number of VMs and their deployment constraints. Thereafter, this info is delivered to the HA-aware scheduler that determines the optimal HA placement. Next, the elasticity framework instructs the cloud operating system to spin the HA-enabled VMs according to the placement recommendation provided by the scheduler. Once completed, the elasticity framework agents create a middleware configuration, modify the manifests, and instruct the CMS to deploy OpenSAF on these VMs. Thereafter, the agents create an upgrade campaign and

send it to OpenSAF to install the tenant's applications. Finally, the monitoring facility will be instructed to report on the events that can trigger an elasticity action. Once a threshold is violated e.g., the workload exceeds its limitation, requiring new components to be instantiated on new VMs, the whole process will be repeated with one exception: this time the scheduler will be constrained by the initial placement. Except for the change applier agent that is implemented using Ruby scripts, and the proposed elasticity framework agents are implemented mainly using Java. Third party tools are used for monitoring [16, 17] and load balancing [18, 19].

3.5 Test-bed and Case Study

To evaluate the applicability and performance of the framework prototype, an IP multimedia subsystem (IMS) application is chosen as a case study. IMS has been defined by the third generation partnership project (3GPP) group as a service delivery platform of 3G networks and later became the standard service delivery platform for 4G networks as LTE the 3GPP technology dominated the field. With the rise of the network function virtualization (NFV) technology, network operators intend to cloudify their core network entities such as the evolved packet core (EPC) and IMS in purpose of achieving low cost, robust and high performance network operations. In order to achieve a flourishing migration to the cloud, the virtualized network functions (VNFs) have to meet the carrier grade requirements. As a leading step, an open source project, ClearWater, attempts to adopt and migrate the IMS core to the cloud. ClearWater redesigned IMS with the cloud architecture in mind, which resulted in the modifying the standard entities of the IMS core. This modification of the IMS entities had violated the 3GPP control and data plane decoupling principle and limited IMS from exploiting the software defined networking (SDN) paradigm. In the test-bed, OpenIMSCore is evaluated for cloud deployment with the proposed framework. OpenIMSCore is an open source project developed by FOKUS. It implements the 3GPP standard specification of IMS basic entities and enables instantiation of each entity separately. The basic entities of IMS are analyed, which include the Call Session Control Functions (CSCFs) and Home Subscriber Server (HSS) [20]. The CSCF is comprised of three types: the proxy-CSCF (P-CSCF), the interrogating-CSCF (I-CSCF), and the serving-CSCF (S-CSCF). P-CSCF is the proxy

server for the outbound/inbound SIP traffic between the user equipment (UE) and IMS core. I-CSCF is the interrogator server that interacts with the HSS to obtain the user-relevant S-CSCF to process the SIP initiation request. S-CSCF is the principal server that manages the session control service for the UEs.

In the testing environment, eight virtual nodes are implemented with resources of two vCPU and two GB RAM. These nodes where hosted on physical servers of the following characteristics:

1) 32 Giga Byte (GB) random access memory (RAM).

2) 2 hard disk drives connected with RAID 0 configuration.

3) I7-4700 CPU (4 Cores, 8 threads).

4) Ethernet network card of 1Gbps bandwidth.

5) Internet connectivity of 10 Mbps.

Different testing scenarios were undertaken to evaluate the installation time in different stages. In the experiments, the installation phase and the configuration phase after the installation are distinguished. Fig. 3.9. illustrates how long it takes to scale up the OpenSAF cluster with the HSS component that uses the MySQL database [21].

Concurrent installation from 1 to 6 VMs were executed in parallel. Note that these durations exclude the VM creating time, since durations differ from one hypervisor to another (e.g., starting containers are much faster than starting VMs) and from one cloud OS to another. Fig. 3.10. shows the duration of the configuration phase. It is observed that the duration could be measured in seconds, which is significantly smaller than the installation phase results.

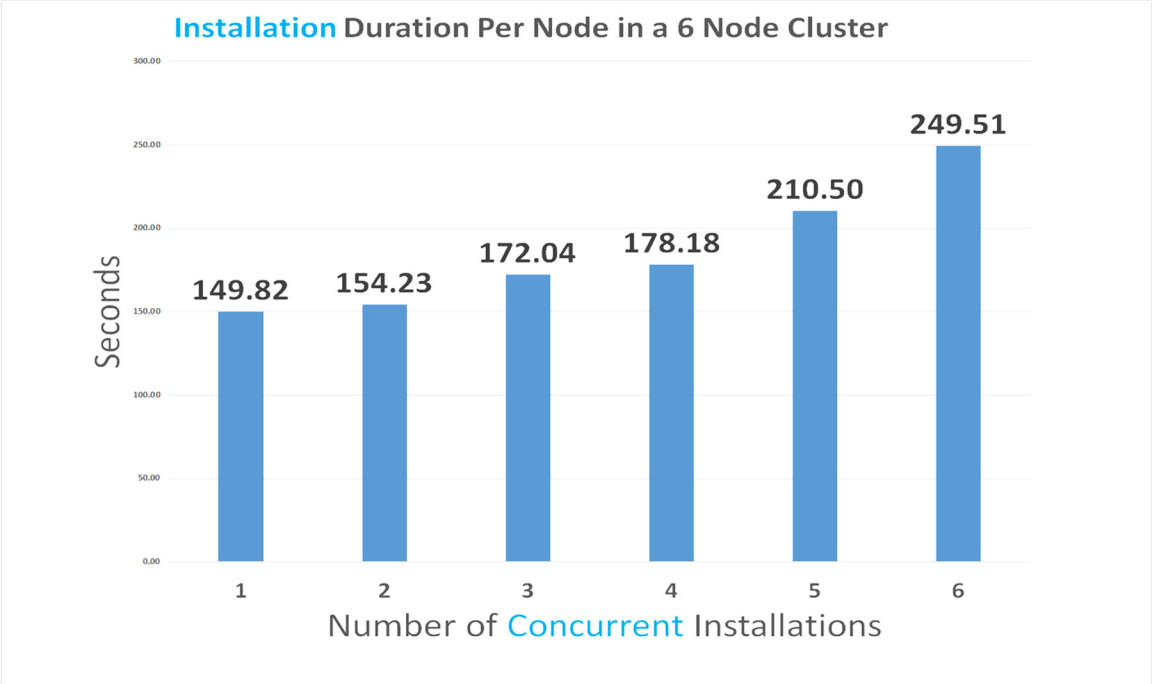

Fig. 3.9 Installation duration results.

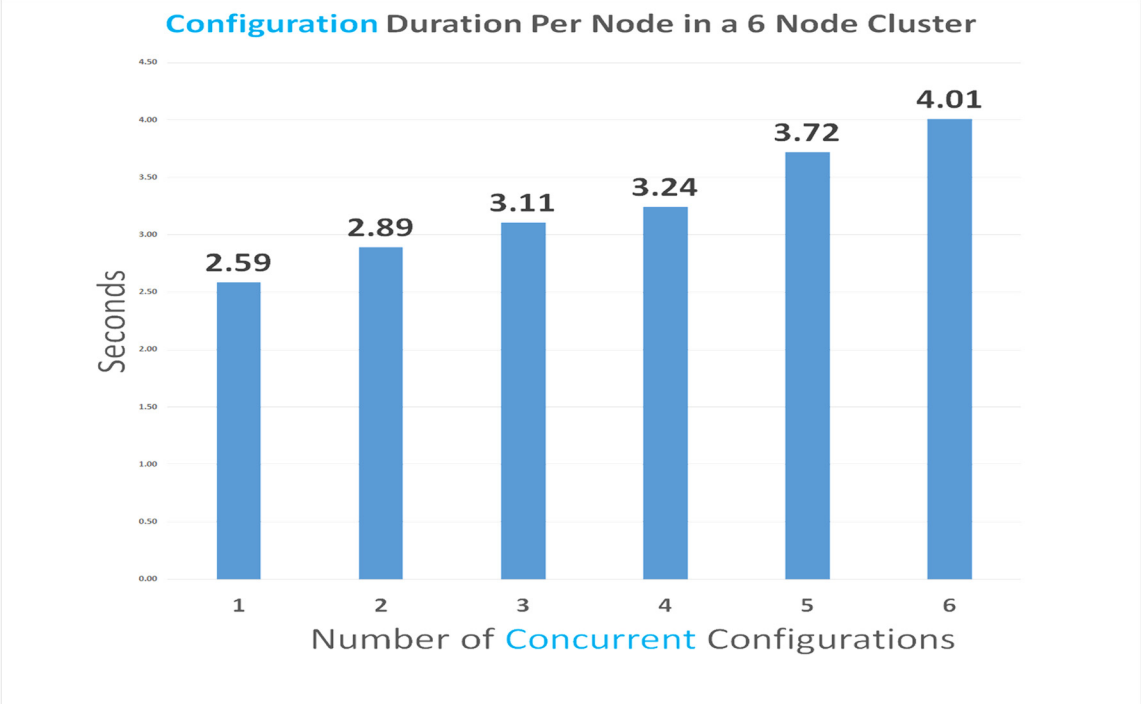

Fig. 3.10 Configuration duration results.

Hence, it can be concluded that using VM images that already include the installation of the software would considerably improve the responsiveness of the system to the increasing workload. Using this approach would require either: (1) preparing different VM images for each of the application components, and then selecting the proper image based on the component that needs to be scaled; or (2) preparing a larger VM image that contains all the components, while only instantiating the components to be scaled. Lighter images come with the price of managing a higher number of images in the image repository where, for instance, the upgrade of the common components would trigger the upgrade of all the images. On the other hand, having a heavier single image that can be used for the scalability of any component adds more runtime overhead.

3.6 Literature Review

Amazon and Google have addressed availability by providing the capability of deploying application's components to distinct availability zones, thereby creating a multi-site solution. The anticipated goal is to create an autonomous copy of each application in multiple availability zones [22, 23] and establishing elastic load balancing that can stretch on demand. Microsoft's Azure service, on the other hand, has not exposed the availability zones for the users; instead, it has defined availability sets. Each virtual machine hosting application belongs to an availability set, and each availability set is assigned an update domain (UD) and a fault domain (FD). UDs are assigned to indicate which VMs and physical machines can be rebooted simultaneously. FDs define the group of virtual machines that share a common power source and a network switch [24]. Nevertheless, these solutions do not target the monitoring and recovery of the application's components inside the VMs, nor do they provide solutions for scaling up the HA middleware inside the tenant's vDC. OpenStack proposes Heat [18] for the management and orchestration of the software. Heat can be considered as a potential replacement for Puppet in the solution, since using Heat would still require the same set of agents and the proposed elastic scheduling. However, using Heat would bind the proposed solution to OpenStack software, while Puppet can be used inter-changeably with other cloud operating systems.

Tools and frameworks for elastic high performance application have been addressed in associated literature. Rajan et al. [25] proposes the Work Queue framework

to manage the master-slave elastic applications development. This framework enables adding the worker components, which are defined as “slave” entities to be replicated at runtime. When a new worker component is instantiated, it communicates with the master node to synchronize the task execution. In [26], Fito et al. proposes the Cloud Hosting Provider (CHP), a web provider that can elastically allocate public cloud resources in support of overwhelmed local machines. The main objective of CHP is avoiding the SLA violation. Marshall et al. [27] recommends Elastic Site, a platform that elastically extends clusters by allocating additional cloud resources. It is developed to request additional computing based on available resources on Nimbus private cloud and Amazon EC2. Elastic Site initially assigns the available resources on the local nodes before escalating to request virtual resources provided by Amazon EC2. Virtualization based HA has been widely investigated. In [28, 29], HA is maintained using the concept of failover within the workload of the component. While deploying an application in a virtualized environment, each application’s component is encapsulated in a virtual machine. This allows virtualization managers to capture snapshots of the system state, and enables checkpoint and rollback mechanisms to be the primary recovery criteria in a virtualization environment. The recovery process based on this checkpoint and rollback can be achieved using various approaches. Stop-Resume approach is the first category of VM checkpoint, and is discussed in [30] where the VM is arrested completely to save its state in persistent storage, and then resumed on another server. This approach incurs a large system downtime during the checkpoint. In order to eliminate the downtime of a virtual machine, a live migration is proposed [29][31,32,33]. In live migration, the VM computing resources state image is transferred in real time (with zero downtime) to different locations initiated on the newly assigned server.

All the aforementioned virtualization based HA solutions and elastic frameworks address the cloud applications availability at the level of the virtual machine, as well as the allocation of virtual resources. Moreover, some of these solutions are a proprietary remedy for specific types of applications and clouds. In summary, all of the discussed HA solutions guarantee the application’s availability level to achieve 99.95 percent. This achievement is reflected in the service level agreements of the cloud service providers.

For instance, Microsoft's and Google's maximum guaranteed availability is 99.95% in their cloud platforms.

The proposed solution offers an elastic HA-aware framework that differ from the discussed frameworks and HA solutions. The proposed framework migrates the SAForum services to cloud environments to ensure the 5 nines (99.999%) and higher availability for cloud application. In addition, the prototype framework implemented tool can be integrated with different cloud management systems to ease the development, deployment, and manageability of the HA cloud applications.

3.7 Chapter Contribution

This chapter proposed a comprehensive framework for the elasticity of highly available applications that considers the elastic deployment of the platform and the HA placement of the application's components. The proposed approach allows the use of robust and standards-based HA middleware (OpenSAF) solution in a dynamic cloud setting by defining elastic HA-aware scheduling constraints and extending configuration management tools (Puppet) via a set of agents that dynamically generate configurations, modify manifests, and enforce changes in a transparent manner. The approach is applied to an internet protocol multimedia subsystem (IMS) application and demonstrates how within a matter of seconds, the IMS application can be scaled up while maintaining its HA status.

Bibliography

[1] M. A. Sharkh, M. Jammal, A. Shami, and A. Ouda, "Resource allocation in a network based cloud computing environment: design challenges," IEEE Communications Magazine, vol. 51, no. 11, pp. 46-52, Nov. 2013.

[2] Oracle, "Database High Availability Architecture and Best Practices," http://docs.oracle.com/cd/B14117_01/server.101/b10726/hadesign.htm, Chapter 2, 2004. [Feb. 18, 2015]

[3] Continuity Software, “2014 Service Availability Benchmark Survey,” http://www.continuitysoftware.com/wp-content/uploads/2014/05/2014-SA-Survey-Report.pdf , 2014. [Feb. 18, 2015]

[4] NetFlix, “Fault Tolerance in a High Volume, Distributed System,” http://techblog.netflix.com/2012/02/fault-tolerance-in-high-volume.html , 2012. [Feb. 18, 2015]

[5] NetFlix, “Netflix Shares Cloud Load Balancing And Failover Tool: Eureka!” http://techblog.netflix.com/search?q=Eureka,2012. [Feb. 18, 2015]

[6] CIO, “How to Evaluate Moving Legacy Mission-Critical Apps to the Cloud,” http://www.cio.com/article/2383841/enterprise-architecture/ho w-to-evaluate-moving-legacy-mission-critical-apps-to-the-cloud.html, 2013. [Feb. 18, 2015]

[7] Service Availability Forum, “Application Interface Specification,” http://devel.opensaf.org/SAI-Overview-B.05.03.AL.pdf, 2011. [Feb. 18, 2015]

[8] OpenSAF, “OpenSAF Overview, “http://sourceforge.net/p/opensaf /doc umentation/ci/default/tree/OpenSAF_Overview_PR.odt?format=raw, Release 4.4 Programmer's Reference, 2014. [Feb. 18, 2015]

[9] Service Availability Forum, “Availability Management Framework,” http://devel.opensaf.org/SAI-AIS-AMF-B.04.01.AL.pdf , 2011. [Feb. 18, 2015]

[10] Service Availability Forum, “Software Management Framework,” http://devel.opensaf.org/SAI-AIS-SMF-A.01.02.AL.pdf , 2011. [Feb. 18, 2015]

[11] Puppet Labs, “Overview of Puppet's Architecture,” https://docs.puppetlabs.com/puppet/3.6/reference/architecture.html, 2015. [Feb. 18, 2015]

[12] Chef, “An Overview of Chef,” https://docs.chef.io/chef_overview.html , 2015. [Feb. 18, 2015]

[13] M. Jammal, A. Kanso, and A. Shami, "High Availability-Aware Optimization Digest for Applications Deployment in Cloud, " to appear in Proc. IEEE International Conference on Communication, 2015.

[14] M. Turenne, A. Kanso, A. Gherbi, and S. Razzook, "A tool chain for generating the description files of highly available software," Proc. ACM/IEEE international conference on Automated software engineering (ASE '14) , 2014, pp. 867-870..

[15] S. Kohzadi, "Automatic generation of upgrade campaign specifications Setareh Kohzadi," M.S. thesis, Computer Science and Software Engineering, Concordia University, Montreal, QC Canada, 2013.

[16] Nagios, "Nagios XI Documentation," http://library.nagios.com/library products/nagiosxi/documentation/ . [Feb. 18, 2015]

[17] Zabbix, "The Enterprise-class Monitoring Solution for Everyone," http://www.zabbix.com/. [Feb. 18, 2015]

[18] HAProxy, "The Reliable, High Performance TCP/HTTP Load Balancer," http://www.haproxy.org/ . [Feb. 18, 2015]

[19] OpenStack, "OpenStack Neutron/LBaaS," https://wiki.openstack.org/wik i/ Neutron/LBaaS . [Feb. 18, 2015]

[20] ETSI, "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; IP Multimedia Subsystem (IMS); Stage 2," 3GPP TS 23.228 version 12.7.0 Release 12, 2015.

[21] OpenIMS Core, "FHoSS HSS Database," http://www.openimscore.org /docs /FHoSS/using.html . [Feb. 18, 2015]

[22] Amazon, "Building Fault-Tolerant Applications on AWS," http://media.amazonwebservices.com/AWS_Building_Fault_Tolerant_Applications .pdf , 2011 . [Feb. 18, 2015]

[23] Google, “Google Cloud SQL now Generally Available with an SLA, 500GB databases, and encryption,” http://googlecloudplatform.blogspot .ca/2014/02/google-cloud-sql-now-generally-available.html , 2014. [Feb. 18, 2015]

[24] Microsoft, “Manage the availability of virtual machines Understand planned versus unplanned maintenance,” http://azure.microsoft.com/en-us/documentation/articles/virtual-machines-manage-availability/ , 2014. [Feb. 18, 2015]

[25] D. Rajan, A. Canino, J. A. Izaguirre, and D. Thain, “Converting a high performance application to an elastic cloud application,” Proc. International Conference on Cloud Computing Technology and Science (CLOUDCOM), 2011, pp. 383-390.

[26] J. O. Fito, I. G. Presa, J. G. Fernandez, “Sla-driven elastic cloud hosting provider,” Proc. IEEE Euromicro Conference on Parallel, Distributed and Network-based Processing, 2010, pp. 111-118.

[27] P. Marshall, K. Keahey, T. Freeman, “Elastic site: Using clouds to elastically extend site resources,” Proc. IEEE Internaltional Conference on Cluster, Cloud and Grid Computing, 2010, pp. 43-52.

[28] D. Singh, J. Singh, A. Chhabra, "High Availability of Clouds: Failover Strategies for Cloud Computing Using Integrated Checkpointing Algorithms," Proc. International Conference on Communication Systems and Network Technologies (CSNT), 2012, pp. 698-703.

[29] A. Stanik, M. Hoger, O. Kao, "Failover Pattern with a Self-Healing Mechanism for High Availability Cloud Solutions," Proc. International Conference on Cloud Computing and Big Data (CloudCom-Asia), 2013, pp. 23-29.

[30] M. Zhao, R. Figueiredo, “Experimental study of virtual machine migration in support of reservation of cluster resources,” Proc. International workshop on Virtualization technology in distributed computing, 2007, p. 5.

[31] D. Huang, D. Ye, D., Q. He, J. Chen, K. Ye, "Virt-LM: a benchmark for live migration of virtual machine," In ACM SIGSOFT Software Engineering Notes, vol. 36, no. 5, pp. 307-316. .

[32] K. Ye, J. Che, X. Jiang, J. Chen, and X. Li, "vTestkit: A Performance Benchmarking Framework for Virtualization Environments," Proc. of fifth ChinaGrid Annual Conference, 2010, pp. 130-136.

[33] H. Jin, L. Deng, S. Wu, X. Shi, and X. Pan, "Live virtual machine migration with adaptive memory compression," Proc. IEEE International Conference on Cluster Computing, 2009, pp. 1-10.

Chapter 4

4. Conclusion and Future Work

4.1 Conclusion

The NFV aims to revolutionize the telecommunication industry by decoupling network functions from the underlying proprietary hardware. It provides all the benefits of IT virtualization platforms. Academic researchers and network engineers are exploiting virtual environments to simplify and enhance NFV in order to find its way smoothly into the telecommunications industry. Besides all the advantages brought by NFV to the telecommunications industry, it faces technical challenges that might hinder its progress. Therefore, IT organizations, network enterprises, telecommunication equipment vendors, and academic researchers should be aware of these challenges and explore new approaches to overcome them.

In the thesis, a grouping criterion for virtualized network functions to enhance performance and minimize the transaction of vEPC entities occurring on the physical network has been presented. The analysis shows that the presented grouping can reduce the network control traffic by 70 percent and suit vEPC entities to the virtualization environment.

Furthermore, an elasticity framework is presented. The elasticity framework considers a holistic approach that targets elasticity from the cloud tenant's perspective of the infrastructure, HA platform, and application level. The proposed approach allows the use of robust and standards-based HA middleware solutions in a dynamic cloud setting by defining elastic HA-aware scheduling and extending configuration management tools via a set of agents that dynamically generate configurations, modify manifests, and enforce changes in a transparent manner. Experimental results show that avoiding the runtime installation of the software on the newly added VMs by using VM (or container) images that already include the software installation would significantly improve the response time of scaling up the system. The implementation of the elasticity framework

will ease the migration of VNF to the cloud while satisfying the carrier grade requirement.

4.2 Future Work

Elastic highly available application in cloud environment is achieved through a set of integrated components. In this thesis, a Framework for achieving elastic highly available applications in the cloud environment is presented. The proposed framework integrated different component to provide the core functionality for managing and scaling the HA application in the cloud environment.

As a future work, the proposed framework can be extended to integrate further components to enhance the performance, reliability and availability of the HA applications in the cloud environment. The extension components can be added to the framework as plugin applications. The following specific areas are suggested for the enhancement of the framework:

1) Workload Monitoring and Provisioning: the framework can be extended to include workload monitoring and provisioning components for achieving the desired QoS without sacrificing the HA state of the application. The workload-provisioning component can include various artificial intelligence techniques to predict and allocate the required resources. The provisioning component is fed with data from the monitoring component that collects the data from the proposed framework. By integrating the workload and provisioning components into the proposed framework, cloud service providers can satisfy the service level agreement's objectives more efficiently.
2) Automated Application Components Deployment: development and operations (DevOps) is an emerging paradigm that integrates the development and operation concepts. DevOps is essential in the cloud computing era for enabling fast-orchestrated application deployment and release management. The framework proposed in this thesis is a solution that manages and provides elastic HA middle for applications in the cloud environment. Application components that utilize the framework meanwhile have to be deployed by the user. For providing a complete

automated elastic HA cloud application, the application components have to be managed by a DevOps entity. This DevOps entity has to be researched and designed for the integration with the proposed framework's software management entity. Having a DevOps entity embedded in the proposed framework will assure the ease of HA software deployment in cloud environments and yields a lower response time for the application scaling process.

Curriculum Vitae

Name: Hassan Hawilo

Post-secondary Education and Degrees:

Beirut Arab University
Beirut, Lebanon
2007 – 2012 B.A.

The University of Western Ontario
London, Ontario, Canada
2014 - 2015 M.A.

Honours and Awards:

Top Three Engineering Students' Award.
60% tuition fees waiver scholarship.
2007 - 2012

First place award in the IEEE ComSoc Lebanon Chapter Second Student Competition Event at AUB.
2012

Graduate Student Award for Excellence in Research, Electrical and Computer Engineering Department, Western University, ON Canada.
2015

Related Work Experience

Industrial Maintenance and Automation Supervisor
SUKOMI – AVERDA Group.
2012 – 2013

Teaching Assistance.
The University of Western Ontario
2014 – 2015

Research Assistance.
The University of Western Ontario (OC2 Lab) in collaboration with Ericsson Research.
2014 - 2015

Publications:

H. Hawilo, A. Shami, M. Mirahmadi, R. Asal, "NFV: state of the art, challenges, and implementation in next generation mobile networks (vEPC)," Network, IEEE , vol. 28, no.6, pp.18,26, Nov.-Dec. 2014.

H. Hawilo, A. Kanso, A. Shami, “Towards an Elasticity Framework for Legacy Highly Available Applications in the Cloud” 10th IEEE SERVICES, 2015.