Universidade de Aveiro
2014

Departamento de Engenharia Electrónica, Telecomunicações e Informática

João Monteiro Soares

Integração do paradigma de Cloud Computing com a Infraestrutura de Rede do Operador

Integration of the Cloud Computing paradigm with the Operator Network's Infrastructure

Universidade de Aveiro
2014

Departamento de Engenharia Electrónica, Telecomunicações e Informática

João Monteiro Soares

Integração do paradigma de Cloud Computing com a Infraestrutura de Rede do Operador

Integration of the Cloud Computing paradigm with the Operator Network's Infrastructure

Tese apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Doutor em Engenharia Informática, realizada sob a orientação científica da Professora Doutora Susana Sargento, Professora Associada do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor Pedro Neves, Consultor Tecnológico da Portugal Telecom Inovação e Sistemas S.A.

O trabalho desenvolvido no decorrer desta Tese foi realizado com o apoio da Bolsa de Doutoramento em Empresa, com referência SFRH/BDE/51102/2010, financiada ao abrigo do programa POPH – QREN, Formação Avançada, comparticipada pelo Fundo Social Europeu (FSE) e por fundos do Ministério da Ciência, Tecnologia e Ensino Superior (MCTES) através da Fundação para a Ciência e Tecnologia (FCT) e cofinanciada pela Portugal Telecom Inovação e Sistemas S.A..

Dedico este trabalho aos meus pais pelo apoio e grande referência que são.

o júri / the jury

presidente / president

Prof. Doutor Artur Manuel Soares da Silva
Professor Catedrático da Universidade de Aveiro

vogais / examiners commitee

Prof. Doutor Ricardo Morla
Professor Auxiliar do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Prof. Doutor Paulo Simões
Professor Auxiliar do Departamento de Informática da Universidade de Coimbra

Prof. Doutor Pedro Sousa
Professor Auxiliar do Departamento de Informática da Universidade do Minho

Prof. Doutor André Zúquete
Professor Auxiliar do Departamento de Engenharia Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Prof.ª Doutora Susana Isabel Barreto de Miranda Sargento
Professora Associada do Departamento de Engenharia Electrónica, Telecomunicações e Informática da Universidade de Aveiro (Orientadora)

agradecimentos / acknowledgements

À Professora Susana Sargento por me ter dado esta oportunidade e por me ter guiado, mais uma vez, numa importante fase da minha vida académica. Ao Mestre Jorge Carapinha por ter partilhado comigo o seu vasto conhecimento no domínio das redes de operador de telecomunicações e pelas inúmeras discussões no âmbito da temática deste trabalho. Ao Doutor Pedro Neves pelos conselhos, disponibilidade, ajuda, e partilha de conhecimento. A vós um muito obrigado!

A todos os meus colegas que de uma forma ou de outra me apoiaram neste percurso, em especial ao Márcio Melo, Romeu Monteiro, Bruno Parreira, Miguel Dias, Carlos Gonçalves, Paulo Tavares, João Aparício, Rafael Gomes e Gonçalo Pardal.

To Hareesh Puthalath, for the discussions, sharing of knowledge, confidence and incentives.

Um especial agradecimento também ao Nuno Coutinho, André Cardote, Tiago Condeixa, Ricardo Matos, Lucas Guardalben e André Reis pela camaradagem durante todo este percurso.

À Sofia pelo apoio incondicional, bem como ao Vasco, Nuno, Piri.

Aos meus pais, pelo constante apoio, confiança, e enorme referência que são. A vós, um especial obrigado!

A todos aqueles que acreditaram, incentivaram e ajudaram, o meu mais sincero obrigado!

palavras-chave

Cloud Computing, *Cloud Networking*, Operador de Rede, Redes Definidas por Software, Virtualização, Infraestruturas Virtuais, Gestão de Recursos, Formulação Matemática, *NP-Hard*, Heurísticas, Programação Linear Inteira, Virtualização de Funções de Rede, Função de Serviço, Encadeamento de Funções de Serviço, Orientação de tráfego.

resumo

A proliferação do acesso à Internet permite aos utilizadores usar serviços disponibilizados diretamente através da Internet, o que se traduz numa mudança de paradigma na forma de usar aplicações e na forma de comunicar, popularizando desta forma o conceito denominado de *cloud computing*. *Cloud computing* traz consigo requisitos a dois níveis: ao nível da própria *cloud*, geralmente dependente de centros de dados centralizados, onde as tecnologias de informação e recursos de rede têm que ser capazes de garantir as exigências destes serviços; e ao nível do acesso, ou seja, dependendo do serviço que esteja a ser consumido, são necessários diferentes níveis de qualidade de serviço na rede de acesso, um domínio do operador de rede. Em síntese, existe uma clara dependência da *cloud* na rede. No entanto, o papel que a rede tem vindo a desempenhar neste âmbito é reduzido, sendo principalmente um fornecedor de conectividade (*best-effort*) tanto no dominio da *cloud* como no da rede de acesso.

O trabalho desenvolvido nesta Tese permite uma integração efetiva dos domínios de *cloud* e operador de rede, dando assim à *cloud* o efetivo suporte da rede. Para tal, apresentamos uma plataforma e um conjunto de mecanismos associados para gestão e controlo integrado de domínios *cloud computing* e operador de rede por forma a fornecer serviços fim-a-fim. Além disso, elaboramos um estudo aprofundado sobre o mapeamento de recursos virtuais neste ambiente integrado. O estudo centra-se na maximização da incorporação de recursos virtuais na infraestrutura física por meio de estratégias de mapeamento ótimas (considerando a alocação inicial de recursos, bem como adaptações ao longo do tempo), enquanto que se minimizam os custos associados ao consumo de energia. Este estudo é feito para cenários de apenas um domínio e para cenários com múltiplos domínios. Além disso, exploramos como o operador de rede pode aproveitar o referido ambiente integrado para suportar funções de rede tradicionais. Neste sentido, estudamos como as funções de rede virtualizadas devem ser modeladas e geridas num ambiente *cloud* e estendemos a plataforma de acordo com este conceito.

No âmbito desta Tese foi feita uma avaliação extensa das soluções propostas, avaliando os seus benefícios. Implementámos provas de conceito por forma a demonstrar as mais-valias, viabilidade e fácil implantação das soluções propostas. Além disso, a avaliação das estratégias de mapeamento foi realizada através de ferramentas de simulação e de programação linear inteira, mostrando que é possível reduzir o consumo de energia da infraestrutura física, sem comprometer a aceitação de recursos virtuais. Este aspeto pode ser melhorado através da adaptação de recursos virtuais ao longo do tempo. No entanto, deve-se ter em mente os custos associados aos processos de adaptação. Os custos podem ser minimizados, mas isso implica uma redução na aceitação de recursos virtuais. Esta compensação foi também um tema abordado nesta Tese.

keywords

Cloud Computing, Cloud Networking, Network Operator, Software Defined Networking, Virtualization, Virtual Infrastructure, Resource Management, Mathematical Formulation, NP-Hard, Heuristics, Integer Linear Programming, Network Functions Virtualization, Service Functions, Service Function Chaining, Traffic Steering.

abstract

The proliferation of Internet access allows that users have the possibility to use services available directly through the Internet, which translates in a change of the paradigm of using applications and in the way of communicating, popularizing in this way the so-called cloud computing paradigm. Cloud computing brings with it requirements at two different levels: at the cloud level, usually relying in centralized data centers, where information technology and network resources must be able to guarantee the demand of such services; and at the access level, i.e., depending on the service being consumed, different quality of service is required in the access network, which is a Network Operator (NO) domain. In summary, there is an obvious network dependency. However, the network has been playing a relatively minor role, mostly as a provider of (best-effort) connectivity within the cloud and in the access network.

The work developed in this Thesis enables for the effective integration of cloud and NO domains, allowing the required network support for cloud. We propose a framework and a set of associated mechanisms for the integrated management and control of cloud computing and NO domains to provide end-to-end services. Moreover, we elaborate a thorough study on the embedding of virtual resources in this integrated environment. The study focuses on maximizing the host of virtual resources on the physical infrastructure through optimal embedding strategies (considering the initial allocation of resources as well as adaptations through time), while at the same time minimizing the costs associated to energy consumption, in single and multiple domains. Furthermore, we explore how the NO can take advantage of the integrated environment to host traditional network functions. In this sense, we study how virtual network Service Functions (SFs) should be modelled and managed in a cloud environment and enhance the framework accordingly.
A thorough evaluation of the proposed solutions was performed in the scope of this Thesis, assessing their benefits. We implemented proof of concepts to prove the added value, feasibility and easy deployment characteristics of the proposed framework. Furthermore, the embedding strategies evaluation has been performed through simulation and Integer Linear Programming (ILP) solving tools, and it showed that it is possible to reduce the physical infrastructure energy consumption without jeopardizing the virtual resources acceptance. This fact can be further increased by allowing virtual resource adaptation through time. However, one should have in mind the costs associated to adaptation processes. The costs can be minimized, but the virtual resource acceptance can be also reduced. This tradeoff has also been subject of the work in this Thesis.

Table of Contents

Index of Figures

Index of Tables

Acronyms

4WARD	Architecture and Design for the Future Internet
ANDSF	Access Network Discovery and Selection Function
API	Application Programming Interface
ASON	Automatically Switched Optical Network
ATM	Asynchronous Transfer Mode
BFS	Breadth First Search
BSS	Business Support System
CAPEX	CAPital EXpenditure
CDN	Content Delivery Network
CE	Customer Equipment
CE4Cloud	Carrier Ethernet for Cloud
CEAS	Cross-Entropy Ant System
CIS	Cluster Index Server
CMS	Cloud Management System
CNMS	Cloud Network Management System
CP	Cloud Provider
CPE	Customer Premises Equipment
CPU	Central Processing Unit
CRUD	Create, Read, Update, Delete
CSPF	Constrained Shortest Path First
DC	Data Center
DPDK	Data Plane Development Kit
DPI	Deep Packet Inspection
DRE	Distributed Real-time Embedded
DSL	Digital Subscriber Line
D-Vine	Deterministic Node Mapping with Slipttable Link Mapping using Multi-Commodity Flow Constraint
GEYSERS	GEneralised architecture for dYnamic infrastructure SERviceS
EMS	Element Management System
EPC	Evolved Packet Core
EPCB	Extended Power Consumption-Based
ETSI	European Telecommunications Standards Institute
EU	European Union
E-VPN	Ethernet VPN
FTTH	Fiber-to-the-home
GMPLS	Generalized Multi-Protocol Label Switching
GPRS	General Packet Radio Service
GRE	Generic Routing Encapsulation
HDSPA	High-Speed Downlink Packet Access
HSS	Home Subscriber Server
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure-as-a-Service
ICF	Interdomain Coordination Framework
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
ILP	Integer Linear Programming
IMS	IP Multimedia Subsystem
IP	Internet Protocol
IPS	Intrusion Prevention System
IPsec	Internet Protocol Security

IT	Information Technology
ISG	Industry Specification Group
KVM	Kernel-based Virtual Machine
L2	Layer 2
L3	Layer 3
LBaaS	Load-Balancer-as-a-Service
LISP	Locator/ID Separation Protocol
LNP	Link Negotiation Protocol
LTE	Long Term Evolution
MAC	Media Access Control Address
MB	Megabyte
MCN	Mobile Cloud Networking
MEF	Metro Ethernet Forum
MILP	Mixed Integer Linear Programming
MME	Mobility Management Entitiy
MOP	Multi-Objective Problem
MPLS	Multi Protocol Label Switching
MSBN	Multi-Service Broadband Networks
MsC	Master of Science
NA	Network Activator
NaaS	Network-as-a-Service
NAT	Network Address Translation
NBI	Northbound Interfaces
NIC	Network Interface Controller
NIST	National Institute of Standards and Technology
NF	Network Function
NFV	Network Functions Virtualization
NFVI	Network Functions Virtualization Infrastructure
NO	Network Operator
NOMS	Network Operator Management System
NP	Non-deterministic Polynomial-time
NVGRE	Network Virtualization using Generic Routing Encapsulation
OCCI	Open Cloud Computing Interface
OCNI	Open Cloud Networking Interface
OGF	Open Grid Forum
ONF	Open Networking Foundation
OPEX	OPerational EXpenditure
OSS	Operation Support System
OVSDB	Open vSwitch Database
PaaS	Platform-as-a-Service
PCB	Power Consumption-Based
PCRF	Policy and Charging Rules Function
PE	Provider Edge
PGW	Public Data Network Gateway
PoC	Proof of Concept
PoP	Point-of-Presence
PT	Portugal Telecom
PTInS	Portugal Telecom Inovação e Sistemas
QEMU	Quick EMUlator
RAM	Random-Access Memory
RAN	Radio Access Network
REST	REpresentational State Transfer
RSVP	Resource Reservation Protocol
R-Vine	Randomized Node Mapping with Splittable Link Mapping using Multi-Commodity Flow Constraint

SaaS	Software-as-a-Service
SAIL	Scalable & Adaptive Internet soLutions
SDK	Software Development Kit
SDN	Software Defined Networking
SF	Service Function
SFE	Service Function Endpoint
SFaaS	Service Function-as-a-Service
SFC	Service Function Chaining
SGW	Serving Gateway
SLL	Service provider Logical Link
QoS	Quality of Service
QoE	Quality of Experience
TLL	Tenant Logical Link
T-NOVA	Network Functions as-a-Service over Virtualized Infrastructures
UNIFY	UNIFYing Cloud and Carrier Networks
VI	Virtual Infrastructure
VIE	Virtual Infrastructure Embedding
VIM	Virtual Infrastructure Manager
VLAN	Virtual Local Area Network
VM	Virtual Machine
VN	Virtual Network
vNIC	Virtual Network Interface Controller
VNF	Virtual Network Function
VPN	Virtual Private Network
VPNaaS	Virtual Private Network as-a-Service
VXLAN	Virtual Extensible Local Area Network
WAN	Wide Area Network
WG	Working Group
WOC	Wide are network Optimization Controller

1. Introduction

This chapter introduces today's challenges in cloud computing and Network Operator (NO) environments and the research context that motivates this Thesis and its approach. Furthermore, we point out the main objectives that we pursue throughout the work developed in this Thesis in order to address the identified challenges. We then present the contributions of this work as the result of exploring the proposed concepts. This chapter ends with an overview of the Thesis structure.

1.1. Research Scope

In the last years we have witnessed a shift from the traditional industry to an economy based on information computerization. This shift and the constant evolution of technology have redefined industries, politics and cultures, shaping what is today the modern society [1].

Today, the Internet is part of everyday and everyon's life. Until very recently the Internet users were running applications and querying data that was stored in their personal devices. A main advantage of this model lies on the fact that it is possible, at least most of the times, to use applications and have access to data without being connected to a company network, university network, or generically, to the Internet, i.e., in offline mode. However, the offline access is limited in the sharing of information, access to new applications, services and content, and presents high costs in licenses of software and infrastructure with the capability to run and host the applications, services or content.

The constant evolution of access technologies, whether wired – Cable Dial-up access, Digital Subscriber Line (DSL), Fiber-to-the-home (FTTH) - or wireless – General Packet Radio Service (GPRS), High-Speed Downlink Packet Access (HDSPA), Evolved High-Speed Packet Access (HSPA+), Long Term Evolution (LTE), WiFi - is making the access to Internet ubiquitous, faster and with higher quality. The proliferation of Internet access, by some called democratization, allows that users have the possibility to use services available directly through the Internet, which translates in a change of the paradigm of using applications and in the way of communicating, popularizing in this way the so-called cloud computing paradigm [2].

Another key-enabler behind cloud computing, apart from the broadband Internet access, is virtualization. Virtualization abstracts the details of physical elements and enables the logical detachment between execution environment and infrastructure. In the early 2000s, the virtualization of the popular x86 architecture [3] paved the way to a major revolution in the Information Technology (IT) industry, as it opened new possibilities on how resources can be provisioned, controlled, managed and ultimately used.

Initially, the scope of these changes was circumscribed to the IT domain. The emergence of cloud computing was the most visible outcome of this evolutionary process. Soon after cloud computing emerged as a novel delivery model for IT services, it became clear that networks would not be immune to this major paradigm shift. Although network virtualization has been around for quite a long time [4], the virtualization

of the x86 architecture, and subsequently, the emergence of cloud computing brought a new dimension to what virtualization can bring to networking, both in terms of challenges and opportunities [5].

This shift of paradigm has led to a tremendous explosion of connected devices and applications in the last years, which has been pushing network technologies and architectures to the limit, revealing a series of limitations in the development and management of new services in the network, on the end-user side, due to the fact that a large part of these devices is mobile and there are well inherent challenges to keep these devices permanently connected; on the other side, due to the dynamics that new paradigms such as cloud computing are bringing to the network [6].

The range of cloud services can go from a simple storage or computing service to a video on-demand or gaming service. Each service brings requirements at two different levels: at the cloud level, usually relying in centralized Data Centers (DCs), where IT and network resources must be able to guarantee the demand of such services; and at the access level, i.e., depending on the service being consumed different Quality of Service (QoS) is required in the access network, which is a Telco[1] domain. In summary, there is an obvious network dependency, whether on the access side or on the service side [7].

However, the network has been playing a relatively minor role, mostly as a provider of (best-effort) connectivity within the cloud and in the access network. Furthermore, these trends have led to an exponential increase of traffic volume in the networks and of the associated costs to support this traffic. However, while these latter factors increase exponentially, in the Telco sector the number of customers and revenues has not followed the same behaviour – see Figure 1-1. The number of Telco customers and revenue has indeed increased but with a trend to stabilize. This is one of the major challenges that Telcos are faced against and that has pushed them to evolve and find new business opportunities (whether new business models or new business segments).

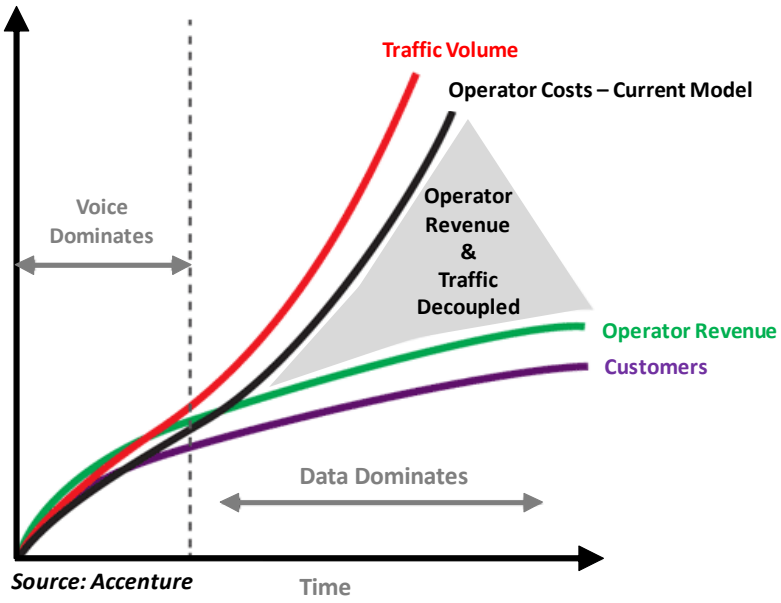


Figure 1-1: Decoupling of network traffic and operator revenue – adapted from [8]

To provide assured levels of performance to cloud services, the research community argues that cloud and network resources need to be provisioned, managed, controlled and monitored in an integrated way [9]. Still, the relationship and inter-dependency between cloud and network technologies goes beyond the need for integration. The emergence of the cloud concept, its ongoing evolution and the opportunities that it brings has led many businesses to adapt in order to get the most out of it. One can say that the Telco sector is today one of the most active business sector exploring the opportunities offered by the cloud [10].

In summary, the relation and inter-dependency between cloud and network can be analysed from two distinct perspectives:

[1] The terms Telco and Network Operator (NO) are used interchangeably throughout this Thesis.

- The network ***supporting*** the cloud: As highlighted before, in a cloud environment communication, end points are Virtual Machines (VMs) that can be hosted in different physical locations, according to varying conditions. In addition, network capacity requirements are no longer static, but are likely to change as the associated computing and storage resources expand and contract. This poses a whole new set of challenges to the network, including the DC and the Wide Area Network (WAN) segments.
- The network ***using*** the cloud: As virtualization technologies reach maturity and are able to provide carrier-grade performance and reliability, it becomes feasible to consolidate multiple network equipment types, traditionally running on specialized hardware platforms, onto industry standard hardware, which minimizes costs, reduces time-to-market and facilitates open innovation.

These two challenges correspond to two major threads of research in this area: cloud networking [11] and Network Functions (NFs) Virtualization (NFV) [12]. They are also two major motivation factors of this Thesis, which will be further detailed.

1.2. Motivation

Considering the growth of services relying on cloud computing, there are several reasons for the effective integration of network and cloud. First, the establishment of Service Level Agreements (SLA) that cover the access network is essential to encourage customers, particularly enterprises, to adopt cloud services. Today, the lack of reliability and performance guarantees is one of the main obstacles against the widespread use of cloud services. It is clear that these SLAs can only be implemented through network integration. Just like the WAN component of enterprise networks is usually based on reliable managed network services such as NO's Virtual Private Networks (VPNs), rather than the public Internet, there is no reason to believe that future enterprise cloud services will require a lesser degree of reliability and performance guarantees from the network. However, the cloud and access network are two different entities, i.e., a user may request a cloud service which inherently has specific access requirements that are not provisioned in the access network. In some cases, when the typical best effort Internet model is not enough, it can be done the purchase of a Telco service that meets the cloud service requirements, connecting the user and the cloud hosting the service. Obviously, the network service is static and cannot follow the dynamicity of the cloud (increase or decrease of QoS requirements). The most common Telco service currently addressing similar issues is the VPN, where the user specifies the "points" he wants to connect and the network requirements (e.g. bandwidth, latency) between them. A "full-blown" network virtualization is also able to address the subject; however, its wide deployment has not yet taken place. Moreover, current network management for VPNs as well as "full-blown" Virtual Networks (VNs) are static in nature, and the latter mainly addresses the problem of subdividing a physical infrastructure into partitions that can be managed independently.

Secondly, it is essential that cloud properties such as elasticity and self-provisioning be also extended to network resources [7]. Quite often, expanding or reducing cloud resource capacity, or provisioning new cloud resources, requires a corresponding reconfiguration of network resources, e.g. bandwidth allocated into the network. Today, by contrast, reconfiguration of network services is supposed to be relatively infrequent and usually involves a significant amount of manual effort.

Thirdly, the dynamism of the cloud will often require live migration of resources (e.g. from a local enterprise DC to the cloud, or between two different sites of the cloud service provider) without interrupting the operating system and any noticeable impact on the running application. This requires addressing to remain unchanged after migration and all relevant QoS, security and traffic policies applied on network equipment (e.g. routers, switches, firewalls) to be adapted appropriately in real time.

The United States National Institute of Standards and Technology (NIST) is one of the most cited organizations when it comes to cloud computing definition, recommendations and open issues [2] [7]. NIST clearly points out the access network dependency and the need for dynamic network configuration features as two major concerns in the cloud [7].

For the reasons stated above, it is clear that next generation of cloud services must handle network and IT resources in an integrated way, both within cloud domains (DCs) as well as within NOs. In fact, Telcos are

considered to be in a good position to evolve and become end-to-end cloud players, being able to provide service-centric end-to-end propositions – see Figure 1-2.

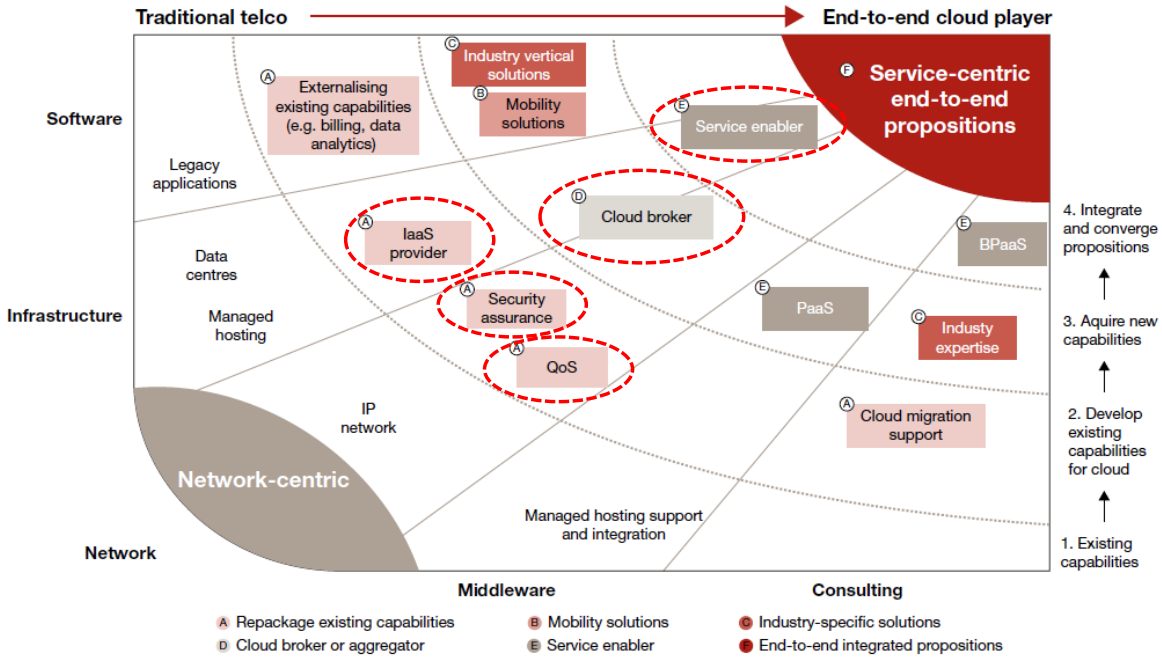


Figure 1-2: From a traditional Telco to an End-to-end cloud player – adapted from [10]

By the time the work of this Thesis started, no real attempt to merge operator networking and cloud resources in a common framework has actually taken place. Technologies treat each component as black boxes, detached from each other: the network often views the DC as a black box, having neither control nor visibility; the DC virtualization is totally handled by the DC servers and hypervisors, and the process is invisible to the underlying networks. DCs, networks, and the end users are not interworking together, compromising seamless end-to-end service delivery. Note that:

- Today's cloud traffic balancing and congestion avoidance is purely DC based, and network conditions are not taken into account.
- There is no synchronization between the virtualization functions, QoS requirements in DCs, and the access network QoS.
- There is lack of automation in end-to-end network configurations, including Quality of Experience (QoE) mechanisms (monitoring and management) which ensure the application/service layer parameters to be kept within certain threshold with the purpose of keeping the user experience at a certain level.

This is a major drawback which certainly will disrupt the leverage of cloud services: more "heavy" services appear, which are not able to be attended in a best-effort way. The Telco sector is an example, as it seems eager to "cloudify" some of its solutions. The momentum around the NFV working group within the European Telecommunications Standards Institute (ETSI) [13], which has brought together most leading NOs, is to a certain extent an example of how serious the subject is being addressed. Complete network systems such as the Evolved Packet Core (EPC) and Internet Protocol (IP) Multimedia Subsystem (IMS) are already virtualized and prepared to be deployed in the cloud and provided "as-a-service". These systems are composed of a set of network Service Functions (SFs), such as the Mobility Management Entitiy (MME), Home Subscriber Server (HSS), and others. However, these services have very strict requirements both in terms of computing and network resources that cannot be met in current cloud offers. The refered network requirements are related to the ability to specify QoS, and also to the need for greater network flexibility (e.g. traffic steering and SF Chaining (SFC)).

Taking into account the abovementioned factors, it is not feasible to have a static network (as today) that does not have dynamic capacity. The network architecture needs to evolve to be able to provide current and next generation networking services, both from a performance and implementation point of

view. In other words, it needs to be more service oriented. The concept of Network as a Service (NaaS) is not new, but we can say that the cloud computing "boom" has re-leveraged it. We argue that, in a near future, the NaaS and Infrastructure-as-a-Service (IaaS) concepts will tend to merge, i.e. cloud and network resources will be jointly provisioned.

Furthermore, the cloud model relies on the ability of large DCs to achieve scalability, which has proven to reduce costs [14]. However, this is not the best solution for every problem. In particular when:

- The access to a shared resource/content/service requires low latency access.
- Many individuals or organizations in a certain geographical area need to access the same resource/content/service, it makes sense for that object to be located, moved or cached near the users, rather than being repeatedly fetched across the entire network (which also negatively impacts the network).

Even though it is possible to provide network services able to attend services and users demands, at a certain point the network itself would not stand due to the centralized nature of today's DCs. This suggests a new business opportunity for the Telcos. Telcos know the network, operate at a large scale and have the ability to get "near the user"; therefore, the opportunity to gain a strong participation in the cloud business arises.

This line of reasoning has brought the concept of cloud networking. Cloud networking has no standard definition, but we can say that it goes beyond classical networks to redefine scalability, administration and management processes. Some key considerations of cloud networking encompass scalability, guaranteed performance, self-healing and extensible management. It is considered not only as the combination of network services with today's cloud (big DCs), but alsoas an evolution of the cloud, where its concept is extended to smaller cloud DCs (which we also refer to as Points-of-Presence (PoPs)) distributed through the network, in strategic places, where some are located in network edges closer to users. This vision meets one of the main motivations behind this Thesis, which is to define a unified management framework for computing and communication, where the Telco can provide simultaneously the network and computational resources in a unified approach.

Given the research scope and the main motivation of this Thesis, the next section provides an insight on the plan to address the previous mentioned challenges and the goals that we pursue in the evolved work.

1.3. Approach & Objectives

Today, the need for an effective integration of network within the cloud computing paradigm is clear. While on one side the integration can be confined to the current cloud models, i.e. to DC domains, on the other hand the NO domain must come into the play. The latter case is a clear opportunity for the NOs to also profit from the businesses supported by cloud computing.

From a high-level perspective, this Thesis covers the research areas depicted in Figure 1-3: cloud computing, carrier-grade services, and NFV. It defines a framework for the integrated management and control of cloud computing and NO domains to provide end-to-end services. We envision to pursue the concept of cloud networking, which is based on the vision of a unified management framework for computing and communication, where one can provide simultaneously the network and computational resources in a unified approach, optimizing overall resource allocations by considering network and computing resources as a unified whole. The control and management of this approach will enable computing and communication services to be provisioned together to meet the resource needs of services with a wide range of requirements, where resource allocation and service migration can exploit dynamic network resource availability as well as compute and storage capacity. It will be possible to manage demand for network resources instantaneously, to meet the changing service needs, enabling fast and efficient responses in a dynamic environment. Furthermore, the framework is enhanced with the capability to manage and compose virtual SFs.

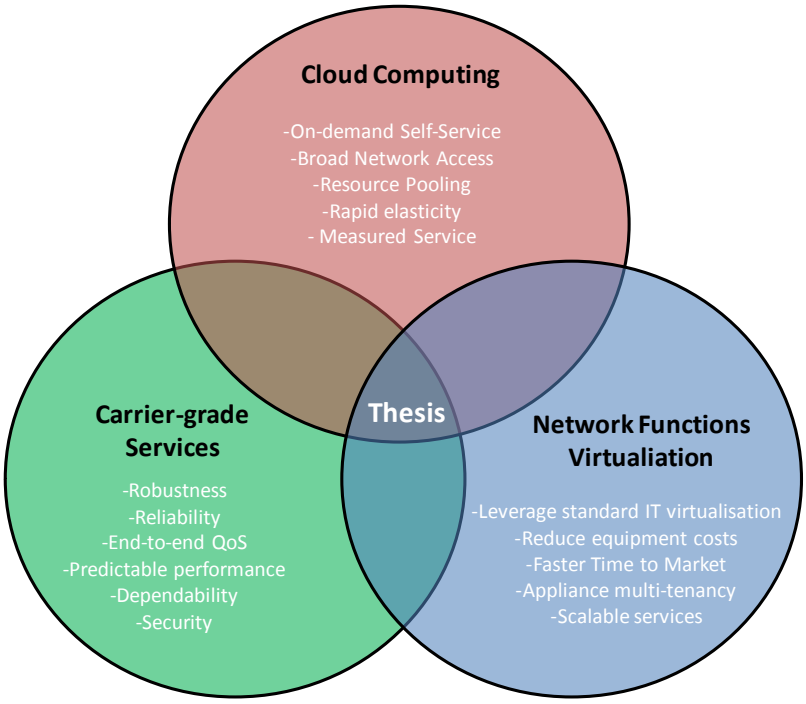


Figure 1-3: High-level Research Objectives of this Thesis

In order to create such framework, we identify the main research topics and objectives that we address in the scope of this Thesis work:

Research Objective 1: How should cloud computing and NO domains be integrated?

- Certain services require end-to-end QoS from the cloud. Thus, we envision a framework with the necessary mechanisms capable of providing true end-to-end QoS in a cloud environment. In order to accomplish the end-to-end QoS we have highlighted the need to have a more distributed cloud environment. In this vision cloud resources are geographically distributed through the network in a more fine-grained fashion than traditional centralized DCs. Note that we are not looking to abandon today's centralized DCs as they are equally important. Centralized DCs will still be used for storing and processing information that does not necessarily have high access demands. DCs have higher scalability capacities and can also take advantage of its location, e.g. place a DC in a low-priced power location. We envision a distributed cloud as a complement to today's centralized DC model.

Research Objective 2: How should the NO evolve to accommodate cloud properties?

- In order to expand the cloud to the network domain, it is require that network connectivity services are able to be provisioned in a cloud-based way, in other words, in a self-service on-demand manner. An evolution of the NO should consider both short-term (i.e. relying in more legacy technologies) as well as long-term solutions (i.e. relying on emerging technologies like Software Defined Networking (SDN) that are more prone to provide the mentioned feature).

Research Objective 3: How should cloud and network resources be embedded in the best way?

- When managing any cloud environment the embedding of virtual resources is crucial. Embedding service requests in an optimal way is the provider's ultimate goal. In these situations, guaranteeing the request SLA and minimizing the embedding costs (e.g. minimize resources used to fulfil the request, minimize energy consumption of the physical

infrastructure) are two major concerns. In this sense, Virtual Infrastructure (VI)[2] Embedding (VIE) mechanisms that take into account the refered aspects should be available.

Research Objective 4: How should the cloud computing and NO integration evolve in order to fulfil NFV requirements?

- Moving traditional network SFs to a cloud environment requires adaptation to take the most out of the cloud. SF management and composition is fundamental. On the other hand, there is also the need to enhance current cloud service models to be able to properly accommodate virtual SFs. In this Thesis we explore how SFs can be modelled in a cloud environment, enhance current cloud environments with respect to network flexibility, and study how to propose a framework that is able to manage and compose SFs.

1.4. Contributions

As mentioned in the previous section, the work developed in this Thesis is not confined to a very narrow research topic. On the contrary, our work comprises the definition, implementation and evaluation of a comprehensive set of research aspects in the integration of the cloud computing paradigm with the NO. As the result of the evolved research work, we published the main scientific achievements as summarized in Table 1-1.

Table 1-1: Publication Contribution

Type	Year	Title	Venue
Conferences	2011	Building Virtual Private Clouds with Network-aware Cloud [15]	Advanced Engineering Computing and Applications in Sciences
	2012	Resource Allocation in the Network Operator's Cloud: A Virtualization Approach [16]	Institute of Electrical and Electronics Engineers (IEEE) ISCC
		Negotiating On-Demand Connectivity between Clouds and Wide Area Networks [17]	IEEE CloudNet
	2014	SDN Framework for Connectivity Services [18]	IEEE ICC
		Optimizing the Embedding of Cloud Network Virtual Infrastructures [19]	IEEE ICT
		Cloud4NFV: A Platform for Virtual Network Functions [20]	IEEE CloudNet
		Optimizing the Embedding of Virtualized Cloud Network Infrastructures across Multiple Domains [21]	IEEE ICC (Submitted)

[2] A VI is a collection of virtualized resources (compute, memory, storage, network) organized in a specific way.

Journals	2014	Optimizing the Embedding of Virtualized Cloud Network Infrastructures [22]	IEEE Transactions on Network and Service Management (Submitted)
		Towards a Telco Cloud Environment for Service Functions [23]	IEEE Communications Magazine (Minor revisions received)
		Re-Optimizing the Embedding of Virtualized Cloud Network Infrastructures [24]	IEEE Transactions Network and Service Management (Submitted)
Book Chapter	2013	The Cloud inside the operator's network: Resource allocation [25]	Communication Infrastructures for Cloud Computing @ IGI Global

One of the first and most significant contributions of this Thesis is the definition of a framework architecture for the joint management of cloud and network resources. Such framework enables the integrated view and lifecycle management of both network and cloud resources. A description and initial evaluation of this concept was published in [15].

After the definition of the management framework, we started looking to the challenges inherent to the integrated mapping of resources. We particularly addressed the resource allocation problem through joint virtualization of network and cloud resources, by proposing a heuristic algorithm to allocate cloud and network resources in an integrated way. This work was published in [16].

In the meantime, challenges in the actual control and resource enforcement started to arise, namely those related to the ability of provisioning and connecting resources across different administrative domains. In this sense, we defined a protocol for dynamic negotiation of connectivity services between domains, which was published in [17], in order to leverage the automation of on-demand connectivity.

With the ongoing and promising advances in the networking area in order to build more flexible and dynamic networks, namely with the SDN concept, we dedicated a considerable effort on exploring the opportunities that SDN could bring. This effort led to the definition of a complete and modular SDN framework targeted to connectivity services that was published in [18].

Maintaining the focus on the very complex resource allocation topic, we devoted a lot of effort on optimal approaches (initially in [16] we focused on heuristic approaches). This effort led to the definition of several optimal strategies using Integer Linear Programming (ILP), which were published in [19], [21], [22], [24] and [25]. [25] presented a first optimal strategy that was also used to better assess the performance of the heuristic approach [16]. Later, [19] and [22] further elaborated on the definition of other optimal embedding strategies that started to look not only to the performance in terms of acceptance, but also in terms of energy consumption aspects. Inevitably, the eagerness to improve the performance and the scope of the embedding strategies led to the work in [24], where new strategies were presented that consider re-optimization processes. In [21], we address the resource management across multiple domains.

The rise of the NFV concept, the enormous momentum around it, and its intimate relationship to the topic of this Thesis, led to an inevitable approach to this concept. With some of the challenges inherent to NFV already addressed in our previous research (i.e. integrated management of cloud and network resources, and the dynamic and on-demand establishment of connectivity services in the WAN and across multiple domains), we devoted some effort in defining the relation between Virtual NFs[3] (VNFs) and

[3] The term Service Function (SF) and Network Function (NF) are used interchangeably throughout this Thesis. Moreover, VNF refers to a virtalized NF, while NFV refers to the concept of virtualizing a function.

infrastructure resources as well as on the composition of VNFs. This research was published in [20] and a longer version was submitted to [23].

This Thesis is supported by a selection of the most relevant publications made during the research work. The selected papers and articles were annexed at the end of this document, starting with the work that introduces the integrated (Resource) Management Architecture along with a Proof of Concept (PoC) framework [15] in Annex Paper A. Then, Annex Paper B refers to the definition of a protocol for the on-demand negotiation and establishment of connectivity services between different network administrative domains, which was published in [17]. Given the extended research in the resource management area, articles, [22] and [24] were also selected to be part of the Annex, in Annex Paper D and Annex Paper E respectively. Furthermore, Annex Paper G refers to the definition of an SDN framework for connectivity services, which was published in article [18], due to its significant importance in achieving a more dynamic management of the operator's network. Finally, one of the works that directly addresses the NFV concept was also considered to be part of the Annex, namely the definition of the platform for VNFs [23] in Annex Paper H.

Finally, the previously mentioned work performed in the scope of this Thesis was also an important contribution in the following research projects:

- P2051, Opportunities and Challenges for Operator in the Mobile Cloud, Eurescom (January 2010 – December 2010) [26]
- SAIL, Scalable & Adaptive Internet Solutions, European Union (EU) FP7-ICT-2009-5-257448, working in the area of cloud networking (August 2010 – January 2013) [27].
- Mobile Cloud Networking (MCN): Mobile Network, Compute, and Storage as One Service On-Demand, EU FP7-ICT-2011-8, working in the area of cloud networking and NFV (November 2012 – October 2014) [28].
- CloudAnchor, working in the research and development of integration and interoperability between cloud computing service providers (Jan. 2013 – Oct. 2014) [29].
- T-NOVA, Network Functions as-a-Service over Virtualized Infrastructures, EU FP7-ICT-619520, working in the area of cloud networking and NFV (January 2014 – October 2014) [30].

Important contributions were also made in the conception of new services, which were considered of high value not only to the research community but also to Portugal Telecom (PT) Inovação e Sistemas (PTInS) and the PT Group (see section 6.2):

- Definition of the VPN as-a-Service (VPNaaS) concept;
- Definition of the SF as-a-Service (SFaaS) concept.

This work had the contribution of several Master of Science (MsC) Thesis, and the author has collaborated to the work developed in those MsC Thesis (in portuguese):

- "Criação e Reconfiguração de Redes Virtuais na Perspectiva do Operador" [31];
- "Integração da Cloud com Rede na Perspectiva do Operador" [32];
- "Demonstrador de uma rede com tecnologia OpenFlow" [33];
- "Integração da Cloud com a rede do Operador" [34];
- "Demonstrador de uma Rede de Operador com Tecnologia OpenFlow e Serviços na Cloud" [35].

1.5. Document Outline

The presented Thesis is organized as follows:

- Chapter 2 provides an overview of the reference technologies used in the scope of this Thesis, focusing on cloud computing, cloud networking, Network Virtualization, SDN and NFV aspects. The reference scenarios for the integration of cloud computing with the NO are also presented. The chapter also highlights the recent trends and evolvements that have taken place in academy, industry, standardization bodies and the cloud market.
- Chapter 3 endorses the cloud computing and NO integration (research objectives 1 and 2 – section 1.3) by presenting an integrated architecture. Special attention is given to integrated resource management and inter-domain challenges. With respect to the integrated resource

management, an architectural solution is presented. For the inter-domain case, a protocol for the on-demand establishment of connectivity across domains is presented. Further, this chapter also explores the need for dynamic connectivity services by proposing two solutions, one based on a legacy network approach, and another based on an SDN approach. Finally, a PoC that brings together all the presented solutions in this chapter is presented.

- Chapter 4 addresses the VI resource management topic (research objective 2 – section 1.3), both in single domain and multi domain scenarios, by presenting a set of embedding strategies that cover both heuristic and optimal approaches. This chapter not only addresses the first embedding process, but it also considers adaptation/re-optimization processes.
- Chapter 5 looks to how the cloud and NO integration should evolve to fulfill NFV requirements (research objective 4 – section 1.3). It presents a platform for the management of SFs that lays part of its foundation upon the principles and mechanisms presented in chapter 3. Further, it elaborates on the SF virtualization process and by presenting a set of data models for the definition and chaining of SFs.
- Chapter 6 presents the conclusions of this Thesis and the envisaged future work.
- Annex contains a selection of the most relevant papers and articles, representing the major contributions of the work evolved and in which this Thesis is supported.

2. State-of-the-Art

This chapter presents the related work on the key research areas within the scope of this Thesis. The importance of the network role in the cloud service delivery, and the possibilities that the cloud brings to the network itself (Network Functions (NFs) Virtualization (NFV)), are increasingly evident. Recent trends and evolvements show that academy, industry, standardization bodies and the cloud market itself are very active on this subject. First, we provide an overview of the cloud computing concept, focusing on its main pillars, management platforms at the infrastructure level, and most relevant research directions. Then, we provide an overview on virtualization, giving special attention to the branch of network virtualization and the virtualization technologies available. Afterwards, we provide an overview of the Software Defined Networking (SDN) concept followed by the NFV one. In both cases the approach is similar, starting by presenting the main pillars of each concept. Then, for each case, details are provided in terms of architecture, most prominent platforms, and most relevant research directions. Then, we survey the available solutions in terms of virtual resource management, namely in terms of allocation and adaptation. Later we review the work that has been carried out in Future Internet Research Projects related to both cloud and networking areas. Following, an insight on the standardization and research group activities is provided. Finally, this chapter ends with a summary of the reviewed work.

2.1. Cloud Computing

2.1.1. The Concept

The concept of cloud computing has no standard or official definition. Several definitions can be found in the literature being the National Institute of Standards and Technology (NIST) definition the most mentioned one: "*Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.*" [2].

Even without definition, there is a common understanding: cloud computing is not a revolutionary concept but rather an evolutionary one, as it has grown from concepts and frameworks such as Grid Computing, Utility Computing and Software-as-a-Service (SaaS). In this sense, cloud computing is not seen as a technology but instead as a combination of technologies, as it pulls together Grid Computing resource's sharing functionality, with utility computing's business model, along with the SaaS's subscription for services on-demand, and others. Cloud computing takes computational resources and applications to a level where they can be provisioned as a service, just like water and electricity. Cloud computing main

characteristics include: on-demand self-service, broad network access, resource pooling, rapid elasticity, and pay-as-you-go.

In a cloud environment applications and data do not need to be necessarily installed or stored in the user's device, being available by the cloud through dedicated service providers, also refered to as Cloud Providers (CPs). The CP is responsible, for example, for guaranteeing the storage, maintenance and backup of the user's data, while the user limits itself to access the platform provided and paying only for what he needs - a pay-as-you-go model.

The main cloud service categories are: *Infrastructure-as-a-Service* (IaaS) - provides computational infrastructure; *Platform-as-a-Service* (PaaS) - provides a layer of encapsulated software over computational resources through which applications and services can be developed, tested and stored; and *Software-as-a-Service* (SaaS) - provides software through a front-end web [36]. More recently another model is gaining momentum in this environment as a missing piece, the Network as-a-Service (NaaS) model. In its simplest form, the NaaS model uses Virtual Networks (VNs) [37] to offer networking resources on-demand. NaaS may then be seen as a special case of IaaS, where the network is the infrastructural resource being offered. However, it is also legit to consider the case where network Service Functionss (SFs) (e.g. firewall, Deep Packet Inspection (DPI), etc) are provided as a service. Note that our work is focused on IaaS and NaaS aspects.

There are three main deployment models: public cloud, private cloud, and hybrid cloud (see Figure 2-1) [2]. Public clouds are implemented by third-party providers and made available to any client through the Internet. This sort of cloud is characterized by its large scale capacity and ability to reduce risks and the costs associated to the provisioning of a service by granting a temporary "extension" of the infrastructure (e.g. Amazon AWS [38]). Private clouds are always developed for the use of a single client, usually enterprises. This type of cloud is developed and managed by the Information Technology (IT) department of the enterprise itself (internal private cloud) or by an external CP (external private cloud). Hybrid clouds are the combination of both public and private models previously described. The ability to increase a private cloud with the resources of a public cloud can be used to maintain the Quality of Service (QoS) and scalability of the cloud towards quick data variations. There are other deviations from these three models, such as the community cloud model. This variant of a private cloud is shared by several companies or organizations that have shared concerns (e.g.: mission, security, policy).

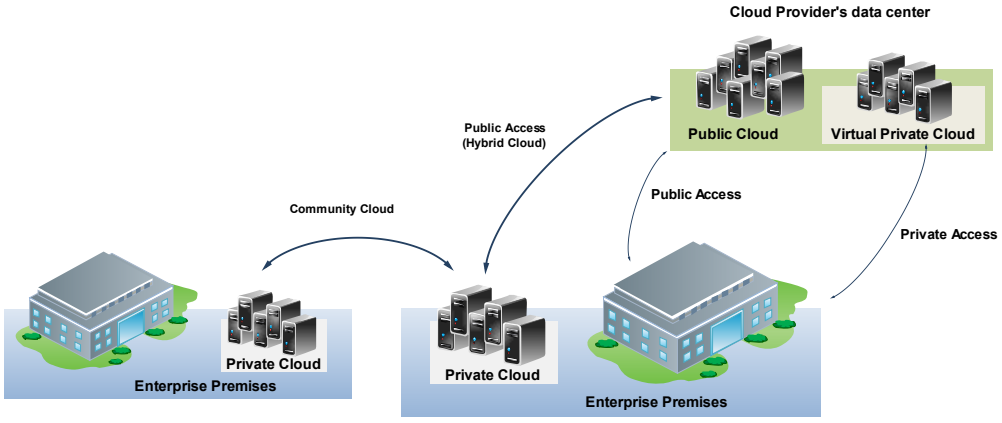


Figure 2-1: Cloud Models

2.1.2. Platforms

Cloud computing builds upon advances on virtualization and distributed computing to support cost-efficient usage of computing and storage resources, emphasizing dynamic scaling and on-demand services. In this section we present some of the frameworks that were designed for helping to manage cloud environments at the infrastructure level. The focus is towards platforms at the infrastructure level, since they are within the scope of this Thesis. We first look to OpenStack [39] and OpenNebula [40] as open source reference platforms, and then refer to some industry proprietary ones.

2.1.2.1. OpenStack

OpenStack [39] is a collection of open source technologies delivering a massively scalable cloud operating system. It aims to produce the ubiquitous open source cloud computing platform for public and private clouds. OpenStack is considered the reference initiative in its domain.

A particularity of this platform is the fact that it consists of a series of interrelated projects delivering various components for the overall solution. Furthermore, these components are built over the premise that no component is shared and the communication is all done through messages. On one hand, this allows components to be distributed; on the other hand it gives (certain) development independence.

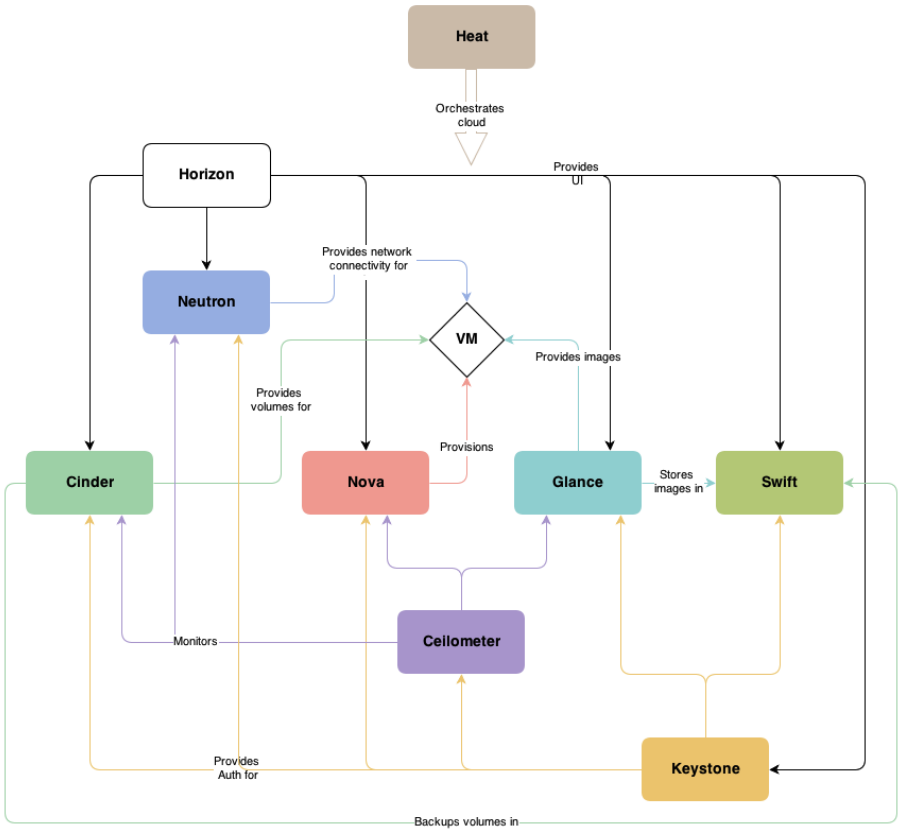


Figure 2-2: OpenStack conceptual architecture [41]

Currently, there are ten projects as part of OpenStack's official releases. These projects are:

- Compute (code-name *Nova*) – service that controls all activities needed to assure the lifecycle management of compute instances, i.e. Virtual Machines (VMs). It has no virtualization technologies and relies on external technologies, e.g.: libvirt Application Programming Interfaces (APIs) [42], Kernel-based VM (KVM) [43], XEN [44], VMWare [45], etc.
- Networking (code-name *Neutron*) - service that allows to create VNs. It provides a set of basic network service abstractions (e.g. Layer 2 (L2) network, Layer 3 (L3) routing) and some more advanced service abstractions (e.g. Firewall as-a-Service, Load Balancer as-a-Service).
- Object Storage (code-name *Swift*) – service that provides a distributed storage system.
- Block Storage (code-name *Cinder*) – service that provides storage blocks (also known as "volumes") to be used by compute instances.
- Identity (code-name *Keystone*) – service that acts as the central point for identification and authentication.
- Image service (code-name *Glance*) – service that is responsible for storing and making available compute instance images (e.g. Operating System images). Images can be stored in its own file system or within *Swift* or Amazon S3 [38].

- Metering (code-name *Ceilometer*) – monitoring service that allows to aggregate performance and utilization information about the different OpenStack services
- Orchestration (code-name *Heat*) - service oriented to templates that allows to automatically provision a virtual infrastructure. It has auto-scaling features, which requires integration with the Metering service.
- Database service (code-name *Trove*) – service that provides multiple database instances as needed. It has the goal of allowing users to quickly and easily utilize the features of a relational database without the burden of handling complex administrative tasks.
- Dashboard (code-name *Horizon*) – tool that provides the graphical web front-end for all OpenStack services.

Figure 2-2 provides an overview of the overall OpenStack conceptual architecture, focusing on the relation and interaction between the different services.

Currently, the Networking project is one of OpenStack's projects with more activity. It still lacks important features like network QoS or finer grained network control (e.g. traffic steering, SF Chaining (SFC)). One can say that its activity has also been driven by the possibilities that SDN approaches bring and by NFV related requirements. More projects are under development in OpenStack; however they are not yet part of official releases and are considered to be in an incubation stage. Those in incubation are not considered of relevance to this Thesis.

2.1.2.2. OpenNebula

OpenNebula [40] started as a research project to later become an open source project developing a standard solution for building and managing virtualized enterprise Data Centers (DCs), i.e. private clouds, and hybrid clouds. OpenNebula is modular to allow its integration with tools and services in the virtualization, cloud environment and DC management. Its primary use is to manage a private cloud inside DCs or inside clusters, but it also supports public clouds, by providing cloud interfaces in order to expose its functionalities for VM, storage and network management. Currently it is in the 4.8 version.

Although the OpenNebula initiative started earlier than OpenStack, the latter has gained more traction by the industry.

2.1.2.3. Industry Proprietary Platforms

The industry has showed two different approaches towards cloud management platforms. The first one, and the most expected one, was to build complete proprietary solutions (in a closed innovation sense). This is the case of VMWare with the vCloud solution [45], Microsoft with Azure [46], and others. The second one, and the most recent, is that of building proprietary solutions over open source software/platforms (in an open innovation sense). In this latter case, the most noticeable example is the one of OpenStack. OpenStack has been used by many industry players to build their customized solutions, e.g. RedHat [47], HP [48], Alcatel Lucent (with CloudBand) [49], and others.

2.1.3. Research directions

Focusing on the effective network integration in cloud computing, this subsection refers to some of the most noticeable research works proposed in this sense.

AT&T and the University of Massachusetts released one of the first studies highlighting the need for more comprehensive control over network resources and security on the cloud service delivery, especially in the enterprise market sector [50]. Also, [51] highlights the critical impact of network performance on applications and presents an extension of a platform presented in [52] that extends the traditional cloud paradigm to also effectively consider network resource provisioning. This work is majorly focused on the modeling aspects of how the end-user can describe a complete Virtual Infrastructure (VI), and the solution relies on the *Virtual eXecution Description Language* (VXDL) [53].

Verizon performed an initial study on the extension of Virtual Private Networks (VPNs) for private clouds and an Internet Engineering Task Force (IETF) Internet-Draft was released [54], however no further progress was carried out by this initiative. In [55] the authors propose a VPN architecture for cloud environments

that can accommodate a large scale number of connections. The approach is focused on the VPN connection setup, while other aspects like QoS are not actively considered. We believe that this solution is very tight to the actual networking technologies, something we believe to be unnecessary. In other words, we believe that a more broad approach will be more suitable for cloud environments.

On a survey perspective, [56] also outlines the key business drivers and requirements of cloud networking, followed by a review of some existing cloud networking solutions and their limitations. Its focus is on the various feature optimizations and technology solutions to evolve carrier cloud networking, such as: end-to-end optimized L2 forwarding using Ethernet VPN (E-VPN) and fabric path; Internet Protocol (IP) virtual overlay solution using Virtual Extensible Local Area Network (VXLAN) and Locator/ID Separation Protocol (LISP); and optimized L3 routing with centralized Provider Edge (PE)/virtual Customer Equipment (CE) solution or distributed PE solution.

On a content distribution perspective, the authors of [57] propose a content distribution network cloud architecture that has into account not only QoS criteria (e.g. round trip time, network hops, loss rate, etc) but also Quality of Experience (QoE) aspects. The authors also propose an algorithm for the placement of content. This approach improves the placement of content taking into account network information; however it does not effectively interact with the Network Operator (NO) (e.g. to reserve bandwidth). Although this is a suitable approach for the distribution of certain type of content, it is not enough for many other cases, since it cannot effectively guarantee QoS or QoE.

While some research works have looked on how to adapt traditional network technologies to the cloud environment, others have looked further ahead, trying to understand how other network approaches like SDN improve the integration of cloud and network. In this sense and in order to improve security, management and energy saving in cloud computing networks, [58] proposes a cloud computing network architecture based on OpenFlow, Autonomic and Identifier Locator split technologies. On contrary to traditional network architectures, this work proposes to bring together traditional aggregation layer and service layer (e.g. firewall, load balancing, and others) by using OpenFlow enabled devices.

IBM published [59], in which they presented CloudNaaS, a cloud networking platform for enterprise applications that improves the control over cloud NFs, such as, the ability to ensure security, performance guarantees or isolation, and to flexibly interpose middleboxes in application deployments. The solution relies on an SDN network control layer that relies on OpenFlow to bring the refered flexibility to the DC network.

The works presented so far are of extreme value, as they look to the actual network technologies that can be used for an effective integration of cloud and network. However, cloud is also related to service abstractions. In this sense, [60] presents a PaaS model for networking, in which the network is abstracted by a single router representation. We consider this approach to be one of the first steps towards building suitable network abstractions that are simple and relatively technology agnostic. Furthermore, [61] presents an open cloud interface extension to merge cloud computing and networks into one modeling and representation framework to extend clouds beyond DCs and enable cloud networking to set up distributed cloud infrastructures. The proposed solution is an extension of the standard interface in cloud computing Open Cloud Computing Interface (OCCI) [62], and is named Open Cloud Networking Interface (OCNI).

The work in [63] was one of the first to propose an architecture for the deployment of clouds over virtualized networks for providing traffic isolation, improved security, and others, thus leveraging the concept of NaaS within cloud environments. In the same line of reasoning as [63], [64] and [65] investigate the application of SOA in network virtualization for composing network and cloud services. They propose a service-oriented framework for composing network and cloud services.

Furthermore, [66] proposes an infrastructure and architectural approach for end-to-end service orchestration based on the orchestrated planning and operation of Optical DC networks and Wireless Access networks. It also proposes a formulation based on a multi-objective Non Linear Programming model that considers energy efficient VI planning over the converged wireless, optical network interconnecting DCs with mobile devices, taking a holistic view of the infrastructure.

In the same line as [66], [67] proposes an architecture for self-establishing an end-to-end Service Level Agreements (SLA) between a cloud user and a CP in a cloud networking environment, focusing on QoS parameters for NaaS and IaaS services. The NaaS services refered in this work are deployed over the operator's network. Relying also on network virtualization, [68] presents a virtualization oriented architecture that allows infrastructure and service providers to achieve service delivery independently and

transparently to end users based on virtualized network control planes. The contribution is majorly focused on the VN control plane.

Another aspect that has been subject of research is the inter-DC connectivity topic. The work in [69] presents one of the first proposals on the topic by exploring the on-demand bandwidth reservation for inter-DC communication. Further, [70] reviews various switching, routing, and optical transport technologies, and their applicability in addressing the networking needs of large-scale multi-tenant DCs. It also reviews technologies for inter-DC connectivity, such as IP/Multi Protocol Label Switching (MPLS) VPNs.

In [71], the authors present a survey on VN mapping algorithms that can eventually be used in cloud infrastructure networks. Furthermore, they present the *Totally Virtualized Cloud Infrastructure* (TVCI) architecture, which considers multiple DC domains and a backbone network domain. The architecture considers the provisioning of full VNs in the backbone network for the interconnection of DCs. However, we believe the assumption of provisioning complete VNs over the backbone network to connect DCs to be an excessive complex solution for the problem.

The authors of [72] address network energy efficiency at the architectural and service levels, and propose a unified network architecture for intra-DC and inter-DC connectivity based on hybrid optical switching. The solution relies on a specific network technology, and it also assumes that a single entity owns both DC and core network domains. These considerations might be in fact valid in certain cases, but do not cover the general case.

Moreover, [73] defines a network abstraction to incorporate the physical and virtual data plane within a DC. It also addresses the interconnection among multiple DCs by relying on OpenFlow switches on each DC. However, the inter-DC connection relies on an overlay network solution, i.e. there is no active participation of the NO, and therefore, there are no connectivity guarantees.

In [74], the authors present a SDN controller that enhances networking of distributed cloud resources and provides authorized customers with the ability to control and configure networks. It interconnects VMs acquired from distributed heterogeneous resources and services from multiple providers using a generic gateway. The cloud networking gateways are managed by the SDN controller that handles allocation and configuration of the gateways according to connectivity requirements. However, just like in the case of [73], the inter-DC solution also relies on overlay networks. This work also proposes an algorithm for request splitting based on resource pricing, trying to minimize the pricing cost of the request overall provisioning.

Building on the theme of request splitting of [74], we now look to the cloud brokerage topic (a topic further elaborated in section 2.5). The work in [75] explores the concept of multi-site cloud brokerage and shows the benefits of network-aware cloud brokering mechanisms. However, there is no active intervention on the network infrastructure between the multiple sites. The proposed solution only assesses externally the QoS levels and performs action according to them. In [76] the authors present a hybrid cloud architectural framework for controlling and managing integrated computing services in on- and off-premise cloud environments. The framework allows creation, modification, and management of integrated hybrid cloud services; however, the network between the multiple clouds is, as in [75], not considered.

More recently, some works have argued for the need to go beyond traditional cloud approaches, which are focused on big centralized DCs. Although some already foresee such an evolution, there have been few strong reasons to actually follow it. However, one can say that recently the telecommunications sector has been pushing for this to happen in a short term period. The authors of [77] discuss the recent trends the mobile telecommunications market is experiencing, showcasing some of the emerging consumer products and services that are facilitating such trends. In this line of reasoning they discuss the challenges that these trends present to mobile NO, and demonstrate the possibility and benefits of extending cloud computing beyond DCs toward the mobile end user, providing end-to-end mobile connectivity as a cloud service. The work in [78] also presents a cloud computing model, refered as Edge Cloud, which addresses edge computing specific issues by augmenting the traditional DC cloud model with service nodes placed at the network edges. The authors present a prototype that extends the OpenStack cloud platform to be able to manage edge nodes.

The research around the effective integration of cloud and network has also been covered in wider research initiatives. European Union (EU)-funded projects such as SAIL [27], GEYSERS [79], and UNIFY [80] are examples of the strong research work carried out by the European research community. Details about these initiatives are provided in section 2.7.

2.1.4. Remarks

One can say that there has been a considerable amount of valuable research done in order to effectively integrate network as a true cloud resource. However we believe that there are still aspects that need to be addressed in order to effectively accomplish this integration. On one side, we believe that there is still lack of flexibility in the integration of cloud and NO domains. Most of the approaches assume the scenario where a single entity controls all domains. Moreover, to effectively integrate multiple domains there is the need for technology agnostic solutions that are able to be framed in different technology environments. Most of the existing research work presented has focused on the technology dependent aspects and not so much on the agnostic ones. Chapter 3 presents solutions for these aspects.

Furthermore, we also consider the flexibility in terms of network control in cloud environments still to be limited. This aspect will become more evident ahead in this document when the NFV topic is addressed in chapter 5.

2.2. Virtualization

Virtualization abstracts the details of physical elements and enables the logical detachment between execution environment and infrastructure. In the sixties, IBM introduced the VM concept [81], describing how to apply virtualization to computers to have a set of simulators/emulators with the same physical hardware. Later in the early 2000s, the virtualization of the popular x86 architecture [3] paved the way to a major revolution in the IT industry, as it opened new possibilities on how resources can be provisioned, controlled, managed and ultimately used.

Initially, the scope of these changes was circumscribed to the IT domain. The emergence of cloud computing was the most visible outcome of this evolutionary process. Soon after cloud computing emerged as a novel delivery model for IT services and it became clear that networks would not be immune to this major paradigm shift.

2.2.1. Network Virtualization

Although network virtualization has been around for quite a long time [4], the emergence of cloud computing brought a new dimension to what virtualization can bring to networking, both in terms of challenges and opportunities.

Network virtualization is a concept where several network instances can co-exist on a common physical network infrastructure. Each VN should allow full administrative control and costumization. In this sense, network virtualization should not be confused with technologies such as VPNs, which only provide traffic isolation.

Two fundamentals of network virtualization are the concepts of virtual link and virtual node. Both are abstract entities and technology independent as both can be realized by a wire range of technologies.

- Link virtualization (virtual link): has the basic purpose to divide, share and isolate resources of physical links providing the traditional functionality of a physical link (i.e. bit transport between two connected endpoints). There is a wide range of technology options, especially for wired link virtualization: Ethernet Virtual Local Area Networks (VLANs) (IEEE 802.1Q [82], IEEE 802.1ad [83]), VXLAN [84], Network Virtualization using Generic Routing Encapsulation (NVGRE) [85], and others.
- Node virtualization (virtual node): has the basic purpose to divide, share and isolate resources of physical nodes providing the traditional functionality of a physical node (i.e. to host a certain function or set of functions). Router virtualization is probably the most notable case, as modern routers are built on very powerful hardware and software that allows resources to be “sliced”. However, the concept transverses other NFs, as we will show ahead in chapter 5.

2.2.2. Virtualization Technologies

Virtualization technologies can be separated in two major groups: compute/IT virtualization technologies and network virtualization technologies.

2.2.2.1. Compute/IT virtualization

On the compute/IT side, all solutions rely on the term hypervisor. A hypervisor is a piece of software, firmware or even hardware that creates and runs VMs. There are two major types of hypervisors (although the distinction between both is not always clear): native and hosted. Native hypervisors run directly on the host's hardware, while hosted hypervisors run within a conventional operating-system environment. In the first case, one can say that VMs run at a first level above the hardware, while on the latter one they run on a third level. KVM [43], XEN [44], VMWare ESX/ESXi [45] are some examples of native hypervisors. Quick EMUlator (QEMU) [86], Virtualbox [87], LinuX Container (LXC) [88] are examples of hosted hypervisors.

2.2.2.2. Network Virtualization

On the network side, there are many technologies, from Asynchronous Transfer Mode (ATM) [89], MPLS [90], and VPN [91], to Overlay Networks [92], Active Networks [93] that in some way emulate network virtualization. However, only more recent tecnologies like SDN [94] have actually been able to realize it.

ATM and MPLS implement only one aspect of network virtualization, virtual links. However, the former is limited by not supporting dynamic configuration of virtual relay nodes inside the network, which is essential for network virtualization. Although it is a robust technology, ATM also suffers from scalability problems. On the other hand, MPLS has a similar objective as ATM and addresses the scalability problem of ATM by using label swapping instead of cell switching. Furthermore, MPLS uses a combination of IP routing algorithms and Resource ReSerVation Protocol (RSVP) for reservation of resources. These characteristics naturally make MPLS a more suitable technology to be used in network virtualization.

The VPN concept (IP/MPLS/VPNs[4]) can be seen as the first virtualization solution recognized by the networking community. First, VPNs were used to build multiple virtual IP (layer 3) networks over a common large scale network infrastructure. Later in time, it was extended to layer 2 technologies, such as Ethernet, through services like Virtual Private Wire Services (VPWS) and Virtual Private LAN Services (VPLS). However, VPNs provide an elusive network virtualization, namely at the node level, as they are tight to the network protocol and merely perform addressing separation.

Overlay networks usually rely on a set of software routers deployed at the edges of the Internet in order to allow different forwarding mechanisms other than those of the Internet. However, the agility provided by overlay networks is only at the edges of the current Internet, and link virtualization is not effectively integrated in the architecture of the overlay networks.

Active networks are composed of execution environments (identical to Unix shells that can execute active packets), where nodes operating systems are capable of supporting one or more execution environments. While overlay networks are implemented at the application layer, active networks are implemented at an (extended) network layer.

SDN is an architecture that aims to bring dynamic, manageable, cost-efficient, and adaptable features to the network. To achieve this, SDN decouples the control plane from the data plane and takes the network intelligence to a logically centralized layer making it easier to program the network. The SDN topic is further detailed in section 2.3.

[4] The term VPN is used throughtout this Thesis to refer to an IP/MPLS VPN.

2.3. Software Defined Networking (SDN)

2.3.1. The Concept

Networks need to keep up with the agility and rapid evolution that we see in cloud-based applications today, and this requires a fresh technological approach. SDN [94] brings new capabilities in terms of network automation and programmability that facilitates the integration with the cloud. Making use of the SDN feedback loop, network control plane decisions can be made not only based on traffic engineering rules, but also in response to dynamic conditions (e.g. network performance, application use trends, user behaviour, congestion events, network malfunction). This allows the network to become more dynamic and resources to be more efficiently used, for example in terms of QoS [95].

SDN is based on three fundamental ideas: decoupling of the control and data planes; abstraction of the network infrastructure resources; and programmability of the network via open APIs. From the point of view of the control plane, one of the potential benefits offered by the decoupling from the data plane is the possibility to get a global perspective of the network resources and make decisions with much greater flexibility and speed compared to traditional networks. This aspect becomes especially (but not exclusively) important in the cloud DC environment, where creation, migration and disposal of VMs occur on a very frequent basis. In other words, in SDN the network intelligence is logically centralized. This, however, does not imply that the control of a network will rely on a single SDN controller. Multiple controllers can be considered depending on the scenario (e.g. DCs may rely in a single controller, while a NO will most certainly rely on multiple ones). The way these controllers will interact among each other is still a subject under investigation.

Another advantage of SDN is a more granular end-to-end view of services with the ability to apply comprehensive and wide ranging policies, thus enabling a better QoS and QoE while improving efficiency [94].

Network resource abstraction is another important SDN characteristic. For the network manager, the use of a standard interface between the controller and the network elements creates an abstraction layer above the network physical substrate. The independence from the specific characteristics of the network infrastructure reduces vendor lock-in allowing the control of network elements from different vendors transparently. The network manager only needs to worry about the supported API versions of the network devices, which need to be consistent within the network infrastructure. It should also be noted that the abstraction layer favours the creation of a programmable and automated network environment which increases network reliability and security.

The network elements present in the network infrastructure are also impacted by SDN. Since these elements no longer need to comprise all the network intelligence, they can be replaced by simpler elements which only have to guarantee connectivity. This brings some advantages from the maintenance and acquisition costs point of view while reducing the number of causes for failures.

Finally, easy and agile adaptation of the network, following requirements raised by applications' dynamics and business requirements, is another major advantage of SDN. Programmability is the key enabler here. Most importantly, open APIs guarantee independence from specific vendors or proprietary solutions.

2.3.2. Reference Architecture

The SDN architecture is divided in three layers, similar to the architecture found in computers (they are also divided in three layers, hardware, operating system and applications). Figure 2-3 shows the SDN three layers' architecture.

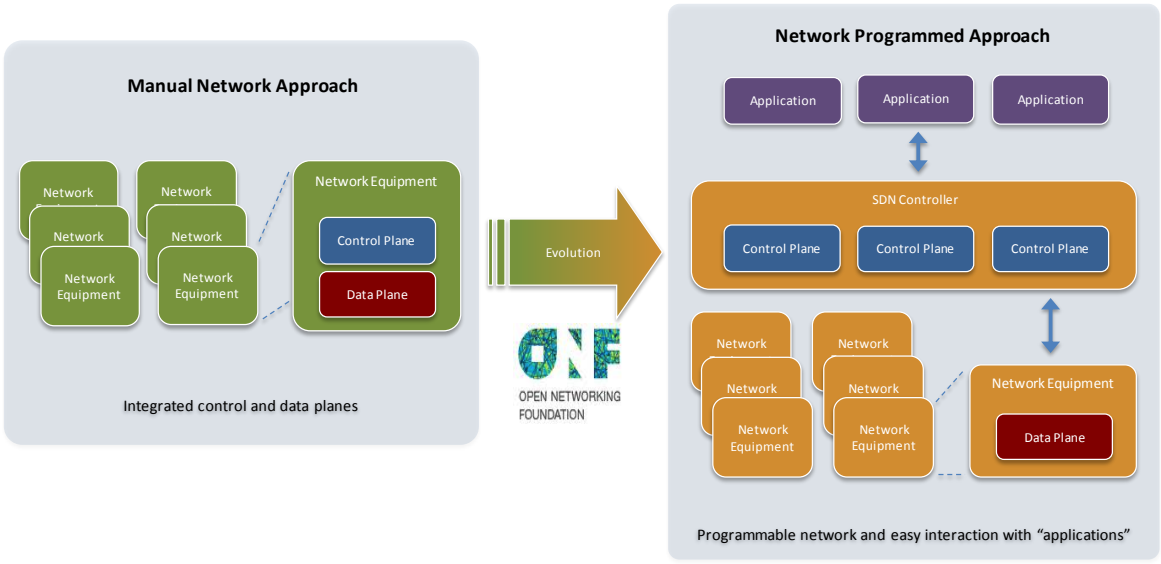


Figure 2-3: Software-Defined Network Architecture

The bottom layer, which can be seen as corresponding to the hardware layer found in computers, includes the software or hardware based network devices that perform the data forwarding functions. The Control Layer represents the network operating system; the software located in this layer uses the lower layer resources to build L2 to L7 network services. This layer provides an important abstraction of the network infrastructure for the upper layer, which frees the Application Layer from the implementation details of the managed services. The upper layer is where services that are network related translate their requirements into abstracted network resources for an optimal service delivery. This architecture allows the coexistence of different VN infrastructures, which can be optimized for delivery of specific services; the VN infrastructure is realized inside the applications.

Communication between layers is accomplished with each layer using the lower layer API. The control layer plays a pivotal role in the architecture and defines the two basic interfaces – Northbound (with the Application layer) and Southbound (with the Infrastructure layer). As to the former, a standard API is still missing, but relevant initiatives in this area (e.g. Open Networking Foundation (ONF) Northbound Interfaces (NBI) Working Group (WG) [96], OpenDaylight Consortium [97]) have been looking at this specific issue and the first results are expected soon. With regard to the southbound interface, OpenFlow [98] is the only standard defined so far, although there are other alternatives. In SDN, "OpenFlow is the first standard communications interface defined between the control and forwarding layers of an SDN architecture" [94]. OpenFlow is a protocol that "specifies basic primitives that can be used by an external software application to program the forwarding plane of network devices" [94]. Although OpenFlow is sometimes mistaken as a synonym to SDN, it is not the only solution. Cisco Open Network Environment (Cisco ONE) [99] is another approach to SDN, although it is also compatible with OpenFlow. Moreover, Cisco developed a Software Development Kit (SDK), onePK [100], which allows users to develop applications that can program the network using an API. Other solutions use overlay technologies to bridge virtual networking devices running on commodity hardware, for example VMWare. VMWare SDN solution uses VXLAN, an overlay technology developed to overcome the 4k VLAN limitation, and a distributed virtual switch, which is basically a cloud enabled version of L2-L7 network services [101].

2.3.3. Platforms

With respect to platforms, OpenDaylight is today the most prominent one. Other platforms were already in place when OpenDaylight was launched, such as Floodlight [102], NOX [103], FlowVisor [104] and others. However, the model adopted by OpenDaylight (one can say that it is similar to the one applied to OpenStack) seems to have been the key differentiating factor. More recently, a new initiative has been

announced by ON.LAB, the ONOS [105]. ONOS is an open source network operating system that will be released soon and is generating a lot of expectation.

2.3.3.1. OpenDaylight

OpenDaylight is a collaborative project under the Linux Foundation whose members are some of the biggest companies regarding networking. It is building a network platform designed to foster primarily the adoption of SDN, but also the one of NFV, by enabling control and programmability over a network infrastructure. The platform follows a versatile modular approach with the goal of providing a generic open source SDN and NFV controller framework, which can be used to by Enterprise IT providers, Telcos, and CPs. In this sense, three editions are available: Base Edition; Virtualization Edition; Service Provider Edition.

From an architectural viewpoint OpenDaylight consists, just like SDN, of three layers (see Figure 2-4): Network Appplications and Orchestration, Controller Platform, Physical and VN Devices.

- Network Applications and Orchestration: consists of business and network logic applications that control and monitor network behavior. In addition, it also hosts more complex solution orchestration applications needed for cloud and NFV thread services together, and engineer network traffic in accordance with the needs of those environments.
- Controller Platform: the framework in which the SDN abstractions can manifest, providing a set of common APIs to the application layer (commonly refered to as the northbound interface) while implementing one or more protocols for command and control of the physical hardware within the network (typically refered to as the southbound interface).
- Physical and VN Devices: the physical and virtual devices, switches, routers, etc., that make up the connective fabric between all endpoints within the network.

One of OpenDaylight's most recent and prominent applications is the one that integrates with the OpenStack Networking service (*Neutron*).

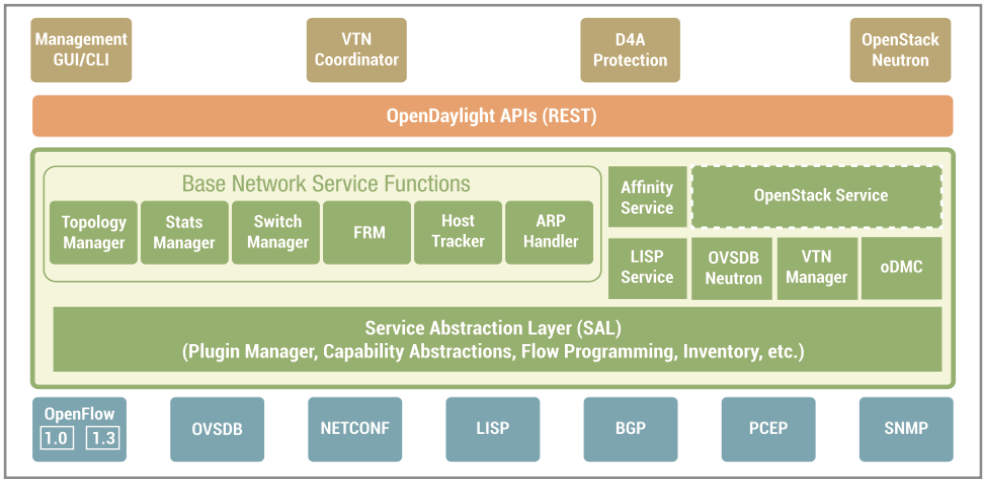


Figure 2-4: OpenDaylight Architecture [106]

2.3.3.2. Industry Proprietary Platforms

Currently, most network related companies are looking into SDN, and some of them already have commercial offers available. Major hardware vendors like Cisco, Juniper, NEC and others already have their own SDN products, for example OpenFlow enabled switches and controllers or other proprietary solutions like the onePK [100].

The available platforms are mainly oriented for DC management. Alcatel-Lucent, through its subsidiary Nuage Networks, launched the Virtualized Services Platform, an SDN solution that uses overlay technologies to enable the coexistence with legacy devices and allows the virtualization of a complete

network infrastructure [107]. Moreover, VMware recently launched a similar product, VMware NSX [108], after acquiring Nicira, a company founded by Martin Casado, the creator of OpenFlow.

2.3.4. Research directions

There are several research works that tackle SDN and OpenFlow related topics, some of which have already been refered in section 2.1.3. In this section we provide further insight on the SDN research directions. [109] provides a good historical survey from active network, to intermediate efforts to separate control and data plane, until more recent solutions like OpenFlow, SDN, and network operating systems.

The work in [110] proposes an efficient method to locate the link failures in the network. Through the implementation of a monitoring function in the network switches, the scalability limitation is overcome, which allows a fast recovery of the system. The work in [111] proposes a framework that uses OpenFlow to handle the transient link failure. This still uses the legacy routing protocols, shifting for the OpenFlow to tackle the problem until the system is recovered. These works focus on resolving the fault management: detection and recovery. However, the processes are neither integrated in a single framework nor only based in a fully SDN/OpenFlow network.

In [112] the authors propose a network management and control system framework that allows operators to run their networks in a hybrid mode, i.e. composed by the legacy network protocols and a SDN controller for OpenFlow networks. With this approach, it automates and simplifies network management while increasing the dynamic control of individual flows. Although this is an improvement, this architecture is still hindered by legacy management tools and protocols, limiting the potential of a complete SDN solution. Furthermore, [113] proposes a different approach of QoS architectures for rerouting capability, using non-shortest paths for lossless and lossy QoS flows. In their latter work [114], the authors propose a framework for dynamic rerouting of QoS flows to stream scalable coded videos. This work aims to resolve the optimization problem of the flows concerning the QoS requirements. Although it is similar to the framework here presented, it is limited to video streaming services and with limited expandability due to its single module architecture. Moreover, [115] proposes an integrated OpenFlow framework capable of running with different OpenFlow versions simultaneously and specifying QoS parameters.

Based on the fact that the emergence of the SDN paradigm provides a new opportunity to closely integrate application provisioning in the cloud with the network through programmable interfaces and automation, [116] describes the architecture and implementation of an SDN controller platform (an extension of Floodlight), named Meridian, that supports a service-level model for networking in clouds. The network service model described is mainly focused on managing the connectivity properties of cloud applications like web service applications within cloud DCs. The SDN controller was integrated both with OpenStack *Neutron* as well as with IBM Smartcloud Provisioning [117].

The authors of [118] argue that the programmability and extensibility of SDN to the data plane should be extended to allow network owners to add their custom NFs while keeping the programmability of existing SDN. Furthermore, they elaborate on the ability to deploy user-defined actions/policies within network devices, rather than on external devices (e.g. Firewall).

The work in [119] presents a SDN-based policy enforcement layer for efficient middlebox-specific “traffic steering”. This can be seen as a significant step toward addressing the concerns surrounding the ability of SDN to integrate with existing infrastructure and support L4–L7 capabilities.

Considering SDN as a new network paradigm, there are also broader initiatives like OpenDaylight (presented in section 2.3.3.1) and the O3 (O Three) Project in Japan. The O3 project was announced with the participation of NEC, NTT, Fujitsu and Hitachi. This SDN project aims to create the first Wide Area Network (WAN) completely based on SDN in which it is expected to reduce by 90% the time necessary for the planning and deployment of WANs [120].

2.3.5. Remarks

The development of SDN approaches will be an ongoing process, on the one side due to the novelty of the concept, and on the other one, due to its inherent ease of extensibility. Furthermore, being SDN a wide topic, it is not surprising that the current related work focus on specific topics (e.g. link failure, path computation) and not on an overall solution. Although the former approach is without doubt essential, the

latter perspective also needs to be taken into account. In this sense, we first focus our work on a complete and modular approach targeted at WAN connectivity services (chapter 3). Later, we narrow our work to a more specific functionality, the SFC and traffic steering (chapter 5).

2.4. Network Functions Virtualization (NFV)

2.4.1. The Concept

The principle of NFV aims to transform network architectures by implementing NFs in software that can run on industry standard hardware (see Figure 2-5). Furthermore, it aims to transform traditional network operations, as software can easily be moved to, or instantiated in, various locations (e.g. DCs, network nodes, end-user premises) without the need to include new equipment. NFV can bring many benefits, from improving operational efficiency and reducing power usage to shorter deployment/upgrade intervals and near-optimal network resource usage.

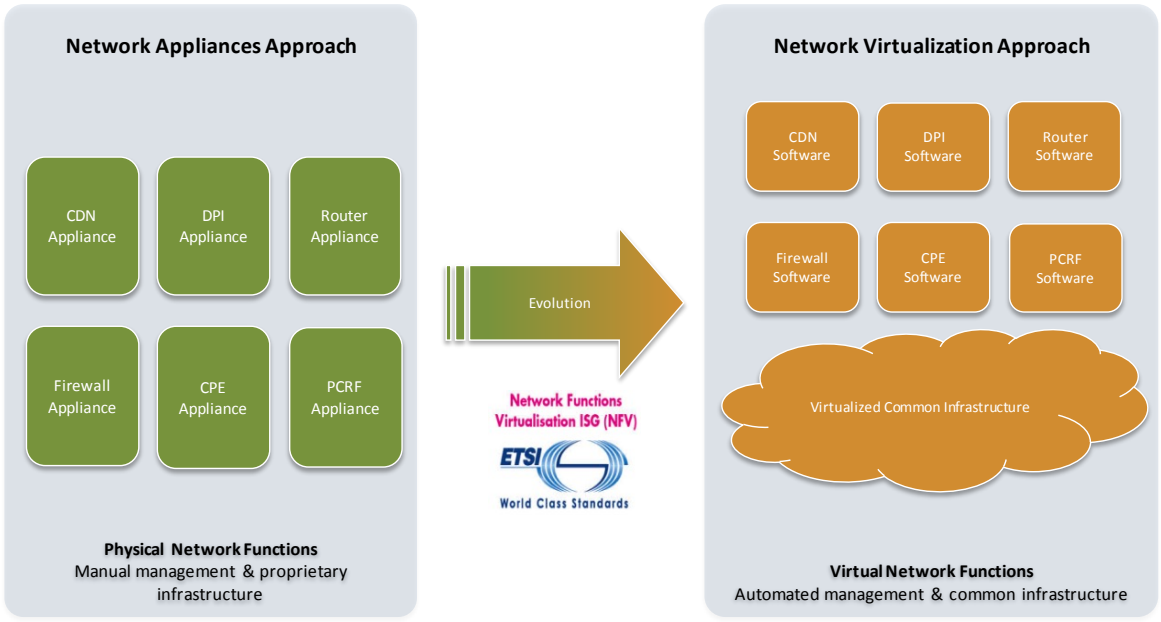


Figure 2-5: Network Functions Virtualization

Although we are facing a change there are still many questions that need to be answered, such as "what to virtualize" and "how to virtualize".

2.4.1.1. What to virtualize

The question can be analyzed from two different angles – on the one hand, the control plane or the data plane, or both, make sense to be virtualized; on the other hand, from a use case perspective, which NFs are the most obvious candidates to be virtualized, from the point of view of potential gains in terms of cost reduction and operational simplicity.

Moving simple software from a physical host to a virtual one is not the most complex achievement. However, we are not talking about simple software functions, especially when referring to NFs where software has close dependencies to hardware. Moreover, NFV is also related to making functions elastic and able to scale up and down, in other words cloud-enabled. Under these circumstances, some of the current NFs may require profound re-architecting work.

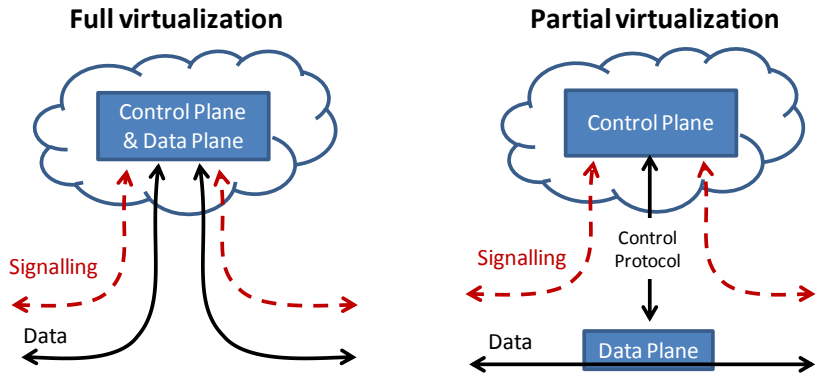


Figure 2-6: Full vs Partial virtualization

Network functions are about control and user/data plane functionalities. Some can have just control functionalities (e.g. Policy and Charging Rules Function (PCRF)), and others can have both (e.g. Packet Data Network Gateway (PGW), Serving Gateway (SGW)). When moving a NF to a virtualized architecture, there are two possible general approaches (see Figure 2-6):

- Full virtualization approach - all control and user plane functional entities are moved to virtual resources (i.e. VMs).
- Partial virtualization approach - only control plane functional entities are virtualized, while the user traffic is forwarded and handled by physical hardware. In this approach SDN can play a decisive role.

It is legitimate to consider control plane entities to be more suitable candidates to be virtualized in a short term due to the fewer requirements in terms of throughput capacity when compared to user plane entities. Nevertheless, this does not mean that user plane entities should not be virtualized. In fact, this is already happening with systems like the Evolved Packet Core (EPC) [121] and Internet Protocol (IP) Multimedia Subsystem (IMS) [122].

Deciding which NFs to virtualize depends: a) if from a technical perspective the specific function is in fact able to be virtualized (e.g. if there are hardware or performance limitations); b) if the process and effort of doing so brings real value to the NO, both from an economic and an operational perspective. The answer to these questions is leading the way to the first NFV deployments.

European Telecommunications Standards Institute (ETSI) has identified and started exploring use-cases believed to be of commercial and technical interest [123], e.g. virtualization of Mobile Core Network and IMS, virtualization of Mobile base station, virtualization of the home environment, and virtualization of Content Delivery Networks (CDNs). Figure 2-7 also presents a very interesting analysis with respect to the relation between automation gains versus cost gains in NFV, where the Customer Premises Equipment (CPE) is positioned as the optimal function(s) to place in an NFV environment. This latter fact can be justified due to CPE large scale deployment, which in a long-term can bring a high return on investment both in terms of Capital Expenditure (CAPEX) and Operational Expenditure (OPEX).

Finally, it is important to highlight the need for the coexistence of virtualized and non-virtualized NFs as mandatory and something that must be ensured.

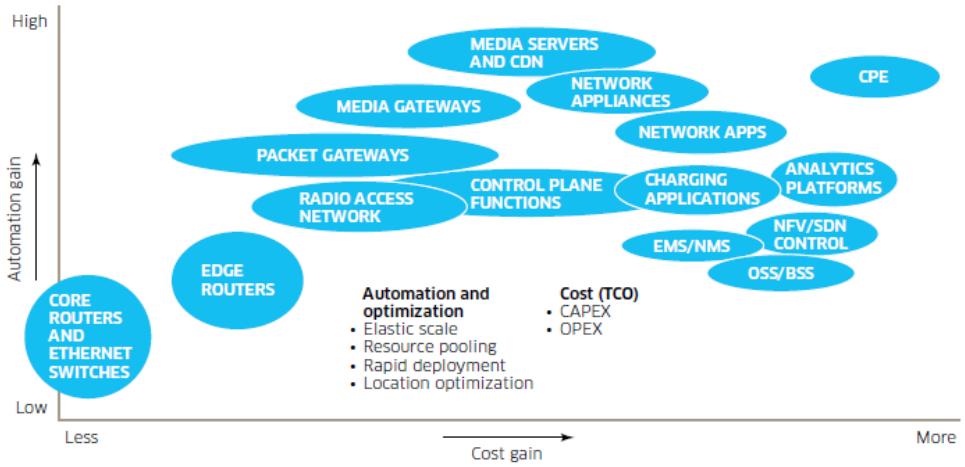


Figure 2-7: Automation Gains vs. Cost Gains in NFV [124]

2.4.1.2. How to virtualize

The answer to this question is focused on the functionalities necessary for the virtualization and the consequent operation of an operator's network, identifying the main functional blocks and the main reference points between those blocks. This is the primarly goal of ETSI NFV, and the following section gives detail on the outcome that ETSI has already provided with respect to this point.

2.4.2. Reference Architecture

Leading the leverage of NFV is performed by ETSI with the NFV Industry Specification Group (ISG) Group. This group has produced what is considered today the reference architecture of an NFV framework, depicted in Figure 2-8. It focuses on the functionalities necessary for the virtualization and the consequent operation of an operator's network, identifying the main functional blocks and the main reference points between those blocks.

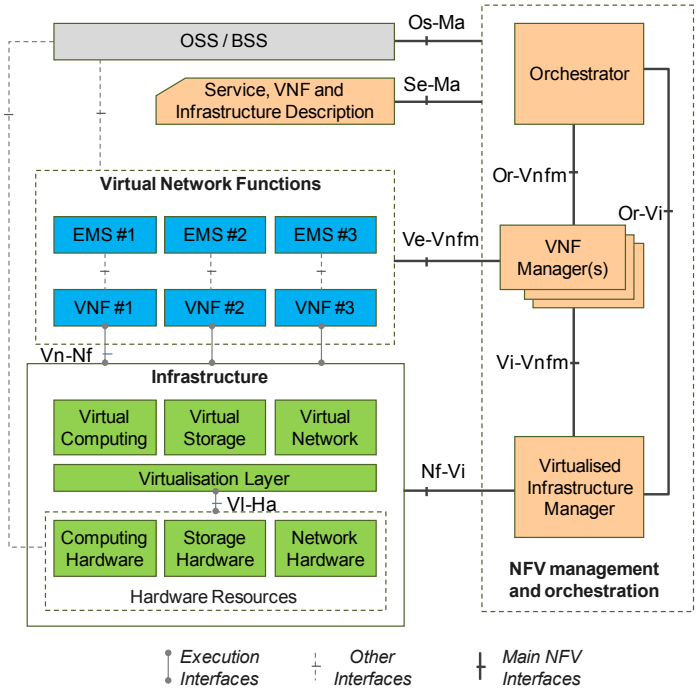


Figure 2-8: NFV Reference architectural framework [125]

Analysing Figure 2-8, the bottom left of the picture represents the NFV Infrastructure (NFVI), which comprises all hardware and software components that support the environment in which Virtual Network Functions (VNFs) are deployed, managed and executed. This infrastructure provides the necessary virtualized resources to the VNFs and can physically span several locations. Looking at Figure 2-9, it is possible to see a NFV infrastructure that comprises: a centralized data centre; Points of Presence (POPs)[5]; and also the costumer site when it has embedded on-site infrastructure to support NFV. The network providing connectivity between these locations is regarded to be part of the NFV Infrastructure.

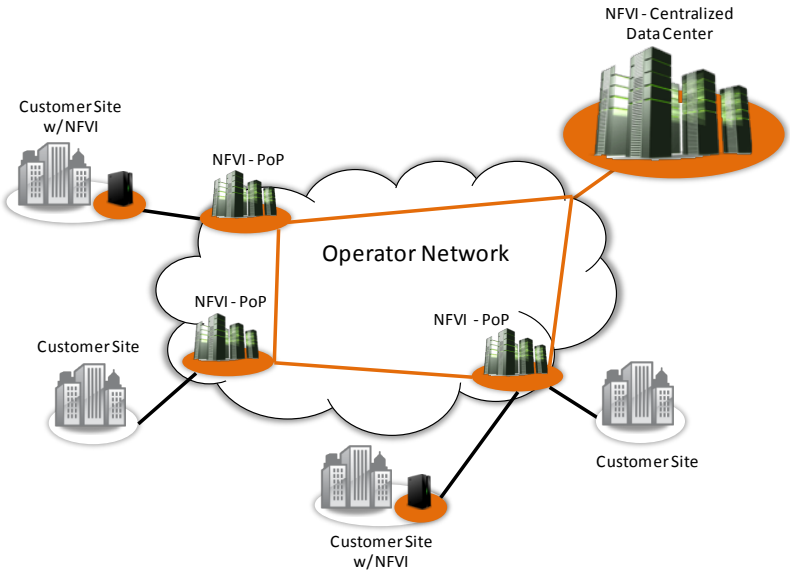


Figure 2-9: NFV Infrastructure

The middle left side of Figure 2-8 contains the VNFs, which use the resources provided by the NFV infrastructure. On the right side there are the management and orchestration elements. The VI Manager (VIM) is responsible for controlling/managing the NFVI resources (e.g. OpenStack). Note that multiple VIM instances may be deployed. A VNF Manager is responsible for the lifecycle management of VNF instances (instantiation, configuration, update, scale up/down, termination, etc). Finally, the Orchestrator is responsible for the orchestration and management of NFV infrastructure and software resources, and realizes network services on the NFV infrastructure. The specification of this functional element still lacks details to be fully understood.

On the top left corner there are the Operational Support System (OSS)/Business Support System (BSS) of an Operator as well as the Service, VNF and Infrastructure Description. The latter comprises a data-set with information regarding the VNF deployment template, VNF Forwarding Graph, service-related information, and NFV infrastructure information models.

More information about this reference architecture can be found in the ETSI public documents - [123] [125] [126] [127] [128]. Nevertheless, note that the ETSI NFV specification work is ongoing and further details and refinements will surely come up in the next few months.

2.4.3. Platforms

The race for NFV management and orchestration platforms seems to be lead by proprietary solutions. Ericsson and Alcatel-Lucent seem to be leading the way, both with their own solutions: Ericsson Cloud System [129] and Alcatel CloudBand [49]. Interesting, but not surprising, is the fact that both solutions are based on OpenStack, a platform that is trying to prove to be ready for carrier-grade deployment scenarios. Alcatel CloudBand, apart from OpenStack, also lays part of its foundation in another open source initiative

[5] In the NFV scope, a PoP is a strategic location in the Operator's Network, which can resemble a small scale DC.

named Cloudify [130] (a cloud orchestration software platform). Focused on NFV, companies following this approach have made partnerships with VNF vendors and other organizations to foster the development and test of these platforms.

2.4.4. Research directions

ETSI hosts what is refered to as the biggest NFV initiative at the moment in its ISG NFV group. It has been driving the establishment of architectural and framework guidelines for the management and orchestration of NFV (among other aspects), and preparing the launch of a second phase focused on the promotion towards implementation. Section 2.8.3 provides further details with respect to this initiative.

CloudNFV [131] is another initiative in the area, keeping a close relation with the ETSI initiative. In fact, CloudNFV is responsible for a Proof of Concept (PoC) already approved within the ETSI group. Although there is this relation between ETSI and CloudNFV, they are not formally associated with each other. CloudNFV main focus is on building a prototype and demonstration model of NFV; however, there are not many public details about it.

ETSI and CloudNFV are the two broadest activities in the NFV domain. But there are other research activities that have specific focus, among which are projects funded by the EU such as the Mobile Cloud Networking (MCN) [28] and T-NOVA [30] projects. Details about these projects are provided in section 2.7. Other initiatives worth mentioning are: the ClearWater project [132], an implementation of IMS designed to be deployed in an NFV environment; OpenEPC [121] and OpenIMS [122], two prototypes under development by Fraunhofer FOKUS.

Naturally, the NFV interest is not confined to wide initiatives. In [133], the authors discuss requirements for NFV-enabled network nodes from a NO's perspective. It also proposed deployment patterns for an NFV VM with OpenFlow switches. Furthermore, it presents a virtual Broadband Remote Access Server (BRAS) prototype developed under an Open vSwitch (OVS) network using Intel Data Plane Development Kit (DPDK) [134].

The work in [135] presents a novel open testbed aimed at offering an experimental facility for SDN and NFV research and experimentation. Part of its focus is on mobility scenarios and the integration with commodity WiFi devices. On the other hand, [136] focus is towards making the data plane more programmable by introducing a tiny, Xen-based VM that can run a wide range of middleboxes. The VM, entitled ClickOS, is small (5 MegaByte (MB) when running), can be instantiated in very small times (order of milliseconds), presents very good networking performance and is able to run on low-cost commodity servers. Similar to ClickOS, [137] presents NetVM, a solution built on top of KVM and Intel DPDK library. On a study perspective, [138] identifies performance bottlenecks in packet processing and forwarding on computers. The authors identify bottlenecks and propose different strategies to overcome them.

2.4.5. Remarks

Due to the novelty of the concept, there are many aspects that are currently target of research and others are still to be addressed. From an NFV management and orchestration perspective, we noticed that currently available solutions are far from completed. Important features are still missing, such as an effective WAN integration and the support for SF composition (e.g. how to map SFs to virtual resource, how to perform SFC). These aspects are mandatory in an NFV “world” and chapter 5 addresses them.

2.5. Scenarios / Business Models and Roles

This subsection discusses the possible roles that a NO can have in the cloud.

Considering the inevitable relation between the NO and the CP, there are four possible situations as Figure 2-10 illustrates: (1) the NO has no notion of cloud services; (2) the NO provides a secure and dedicated connection between user and CP; (3) the NO partners with CPs; (4) the NO is also the CP.

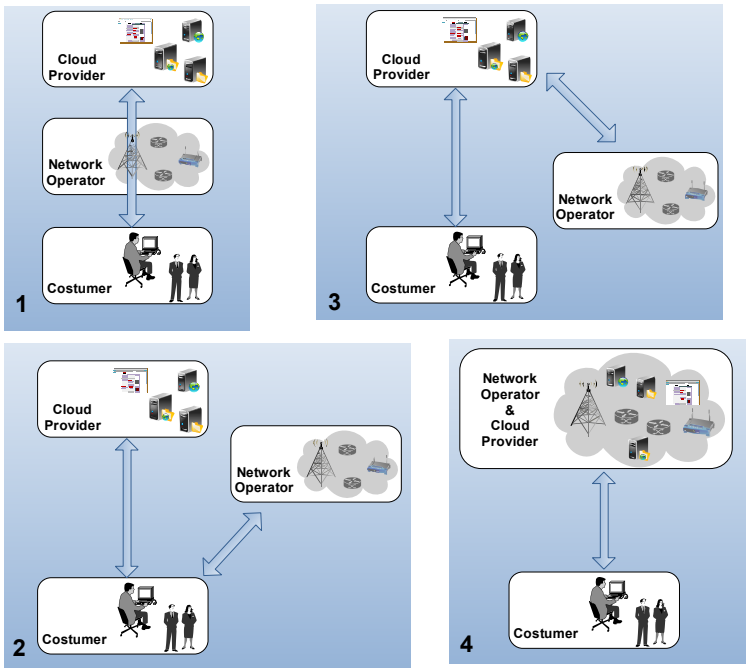


Figure 2-10: Possible Development Models

1. ***NO has no notion of cloud services*** - In this scenario, the end-user consumes services from the CP's through a regular Internet connection granted by its NO. This is the least capable/trustable way of providing and delivering a service to a customer, as the customer cannot be granted with any sort of QoS. Thus, the end-user will receive a service (either free or paid) without any guarantees. In this scenario, which is the most common one today, the NO is a mere bit transporter.
2. ***NO provides dedicated connectivity services*** - In this case the client, usually an enterprise, contracts a private network service (NaaS) from the NO, such as a VPN, and uses it to access the cloud services. Thus, according to its private network service's terms, the client is granted certain guarantees to access the cloud.
3. ***NO partners with the CP*** - Another possible approach is for the NO and the CP to have an SLA entailing service guarantees. In this scenario, according to the agreement made between both parties, the customer is granted a certain service level and QoS. The NO in this scenario charges the CP for the network services, i.e. NaaS. The difference from the previous case lies on the fact that the user in this case has a single point of contact and single SLA, the CP.
4. ***NO as a CP*** - In this last approach the NO is also a CP, providing both networking and cloud services. Thus, such a scenario can easily offer the customer network guarantees in the access to the NO's cloud services.

All four models are not mutually exclusive, since all of them can coexist. Ideally both cloud and network services should be provisioned "as one", and it is in this sense that we develop our work. Nevertheless, as the reader will notice, the architecture presented in chapter 3 is flexible enough to allow all the above mentioned approaches. Part of the proposed architecture relies on the concept of cloud broker, a concept that is further detailed in the next section.

2.5.1. Cloud Service Broker

The concept of a cloud broker is to provide services as an intermediary between CPs and end-users, assisting end-users in selecting the most appropriate cloud platforms/services and in deployment and integration of applications across multiple clouds, as well as providing a choice of multiple competing services. Although the concept is not entirely new, there seems to be a growing opportunity for trusted

intermediaries between end-users and CPs that can help customers to select the right services and provide a "one-stop-shopping" alternative to customers. In a recent Gartner report [139], three categories of cloud brokers were outlined:

- **Intermediation broker** - providing value added services on top of existing cloud platforms, such as identity or access management capabilities.
- **Aggregation broker** - providing the "glue" to bring together multiple services and ensure the interoperability and security of data between systems, usually as fixed services.
- **Arbitration broker** - the main difference between cloud aggregation and arbitration is that services being arbitrated are not fixed. A cloud service arbitrage provides flexibility and "opportunistic choices" by offering multiple similar services to select from, e.g. by providing multiple e-mail services through one service provider.

A cloud broker would normally provide added value, or intermediation, to the aggregated services and "one-stop-shopping" alternative as depicted in Figure 2-11. In the case of a telecom operator, these could include e.g. identity and access management, security policies and billing services. An operator acting as a broker would also encourage users to join the cloud as users trust operators. With a plethora of cloud providers, each with their own API, set of services, pricing models and more, it will be cumbersome to end-users to programmatically access and maintain each service. Thus, a cloud broker creates a layer of abstraction between the users and CPs, so that the end-users can see one cohesive view of the services.

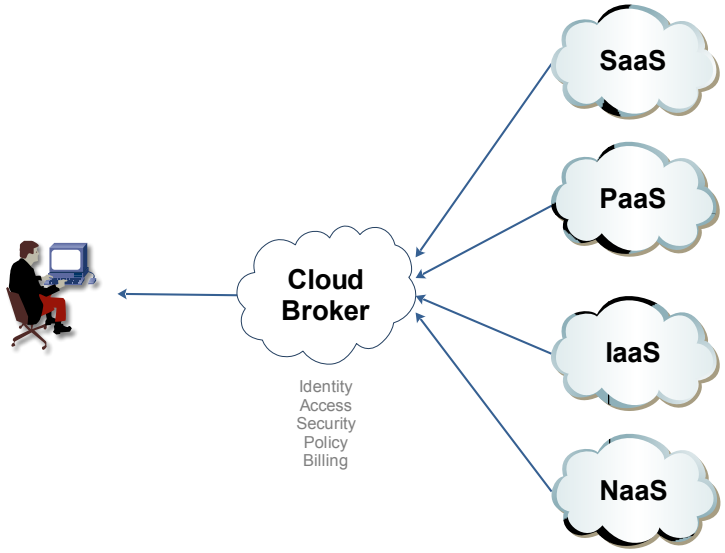


Figure 2-11: A cloud service broker – aggregation and added value

2.6. Virtual Infrastructure Embedding Problem

The increasing interest on a closer integration of cloud/IT and network both from the research community and providers requires several challenges to be addressed properly. One of those challenges is the one of optimally embedding[6] VIs. The VI embedding problem can be described as a graph assignment problem, which involves the embedding of VI requests onto a physical infrastructure. The process requires the simultaneous optimization of virtual nodes and virtual links placement, which is a complex process both in terms of formulation and computation. One can realize that this problem has similarities with the VN embedding problem, which justifies the fact that part of the reviewed state-of-the-art is related with the VN embedding problem. The VI embedding problem can be formulated as an un-splittable flow problem [140], known to be Non-deterministic Polynomial-time (NP)-hard [140] [141] [142] [143].

[6] The term embedding, mapping and assignment are used interchangeable throughout this Thesis

In order for the reader to have a better understanding on the full extent of the VI embedding domain problem, we first present the domain characteristics (section 2.6.1). Then, we present the related work on VI/VN embedding approaches, organized according to the following criteria: resource management in single-domain scenarios (section 2.6.2); resource management in multi-domain scenarios (section 2.6.3). Finally, remarks regarding the presented state-of-the-art are made and other research directions not endorsed in this Thesis are pointed out (section 2.6.4).

2.6.1. Characteristics

2.6.1.1. Online Problem

The online version of the embedding problem refers to VI requests that arrive along time without the embedding system knowing it *a priori* ([144], [142]). The offline version of the problem assumes that all VI requests are known in advanced ([140]). The former case is the one experienced in cloud environments. These cases can result in sub-optimal performance with respect to the physical infrastructure load balancing, energy consumption and VI request acceptance ratio compared to the offline case.

2.6.1.2. Physical and Virtual Resources

There is a set of important considerations that have to be taken into account with respect to the infrastructure resources. First of all, physical resources are limited. Although some approaches neglect this fact [140]), finite resources present a more realistic approach. This latter case can naturally lead to VI requests rejected or eventually postponed in order to avoid resource overbooking and possible violation of guarantees in the existing VIs ([144], [142]).

Furthermore, both physical and virtual resources can be heterogeneous, with distinct characteristics. This heterogeneity can be both in terms of functional (e.g. node type, hardware, location, etc) or non-functional parameters (e.g. available capacity, QoS requirements).

Finally, VI requests may have a diverse set of topologies. This can also happen with the physical infrastructure. When drawing a general solution for the problem, one should also have in mind this fact that both virtual and physical infrastructures can have different topologies.

2.6.1.3. Static vs. Dynamic Mapping

When the initial embedding of a VI has to remain the same along time, the approach is known as static (e.g. [144], [142]). The drawback of such an approach, especially in online scenarios, is that it may lead to inefficient resource utilization. On the other hand, if the VI embedding can change along time, it is possible to: improve resource utilization; increase acceptance ratio; increase energy efficiency; enhance network resilience in case of resource failures. A dynamic scenario is in fact the most desired one; however, it brings costs as the migration of virtual resources may lead to service disruption.

2.6.1.4. Multi-Objective Problem (MOP)

From an embedding objective perspective, it can be a complex problem to map a high-level objective (e.g. map as much VIs as possible) down into a resource management objective. In other words, resource management objectives could be, for example, to map a VI in such a way that: 1) it occupies as less bandwidth as possible; or 2) it balances the occupation of the physical nodes. Moreover, the objectives over the different resources can and should be combined, i.e., to be both considered in the objective. However, in the context of the VI embedding problem these are not standalone objectives because the achievement of one objective is closely related to the others, e.g., the choice of an ideal physical network node may not allow the choice of the ideal physical link. We are therefore faced with what is called a Multi-objective Optimization Problem (MOP) [145], where two or more conflicting objectives subject to constraints need to be simultaneously optimized. In MOP it is not usual to have a solution that is optimal for all the objectives, and our problem is no exception. The general used concept for optimality in a MOP is *pareto optimality*. A *pareto optimal* solution is one that makes it impossible to improve the performance of a criterion without

decreasing the quality of at least another criterion. While typical single objective problems have unique optimal solutions, MOP may have a set of solutions known as *pareto optimal* set [145].

2.6.1.5. Finding the Embedding Solution

The search for an embedding solution can rely on three different categories:

- Heuristics – provide reasonable enough embedding solution according to an objective in a short period of time (e.g. few tens of milliseconds [146]).
- Meta-Heuristics – provide good enough embedding solutions according to an objective in a medium period of time (e.g. few hundreds of milliseconds ([142]).
- Exact Solution – provide optimal solutions according to an objective in a longer period of time (e.g. ranging from milliseconds ([147] to many seconds [148]).

In order to choose the best approach, two aspects should be taken into account: the quality of the VI solution; the time period for the solution to be found.

2.6.1.6. Single-Domain vs. Multi-Domain Scenarios

Single-domain scenarios refer to the case when the embedding system has access to the full state information of physical resources. When that is not the case, the scenario is considered to be a multi-domain one. Approaches for both scenarios have been proposed in the literature as shown in the following two sections.

2.6.2. Single Domain approaches

2.6.2.1. Resource Allocation

The authors of [149] propose a greedy algorithm for node mapping, but the interesting part is the algorithm for virtual link embedding that allows the substrate to split a virtual link over multiple substrate paths, and also employs path migration to periodically re-optimize the utilization of the substrate network. The work in [149] is one of the first to propose path split and path migration. Although it is an interesting approach, this can lead to a level of fragmentation that is unfeasible to manage on large scale networks.

In [142] different algorithms are presented to provide better coordination between the node and link mapping phases. The algorithm takes into account the bandwidth of links and the CPU of nodes. In [150] the authors extended the work in [142] with a generalized window-based VN embedding to evaluate the effect of look-ahead on the mapping of VNs. Both approaches rely on Mixed Integer Linear Programming (ILP) (MILP). Although the approaches provide a better coordination of node and link mapping, they do not support heterogeneity of nodes. [151] formulates the VN embedding problem similar to [142], and proposes a progressively greedy VN embedding algorithm based on linear programming relaxation.

To improve the utilization of physical resources, [152] focuses on having multiple physical nodes to host a virtual node. [153] proposes an approach in which the vertices of the VN are mapped closely as possible in the substrate network, and then virtual edges are assigned to the shortest paths which satisfy the demands. The algorithm tries to optimize the substrate's bandwidth, which naturally allows the accommodation of more VNs as the solution is more optimal.

A VN mapping algorithm that maps nodes and links during the same stage (based on sub-graph isomorphism detection) is presented in [154]. The algorithm proves to be faster than the two stage approach, especially for large VNs with high resource consumption.

In [155], the authors propose an ILP formulation to solve the on-line VN embedding problem. Further, they also propose an enhancement to an existing heuristic [146] and compare both heuristic and ILP, proving that heuristics are far from the optimal values. The authors have recently published [156], where they continue to explore the subject in an ILP perspective. Although it is shown an improvement in performance, the formulation is restricted to VNs and only considers CPU load, bandwidth and delay.

Energy consumption is today a main concern, and entities such as content providers that own huge infrastructures (similar to IaaS CPs) have great interest in reducing their energy consumption. [157] and [158] address the subject, as the former evaluates Power Consumption-Based (PCB) and Transmission Rate-

based (TRB) algorithms for server selection, and the latter proposes an Extended Power Consumption-based (EPCB) algorithm that proves to be able to reduce the power consumption more when compared to TRB and PCB algorithms.

The work in [159] presents an ILP solution for the embedding of VNs in a way that it minimizes the energy consumption of the physical infrastructure. [160] proposes an efficient energy-aware algorithm using a consolidation technique to reduce energy consumption.

Botero et al [161] have extended the VN embedding problem to energy awareness, and proposed a MILP to solve the optimal energy efficiency embedding problem. In the problem formulation they set the objective of minimizing the number of active network nodes and links. The evaluation shows that, depending on the substrate load, the energy consumption can be substantially reduced without drastically affecting the VN acceptance ratio. In this work VNs are characterized by the processing power of the virtual nodes and the bandwidth of the virtual links. However, the work does not consider online scenarios. Nevertheless, the formulations in this work provide part of the base ground for the strategies proposed in our work.

So far, we have focused on VN related work due to the extension of work available. However, there is already some work done with the cloud scenarios in mind. [162] presents an empirical study of bin-packing heuristics for resource allocation for Distributed Real-time Embedded (DRE). A solution for managing infrastructure resources in a PaaS is presented in [163]. The authors consider the initial allocation of requests and a continuous process of optimization/adaptation of the infrastructure. However, the work is focused on CPU and memory resources, while in [164], a fair joint multiple resource (computational and network) allocation method is presented. It is important to note that, from a network perspective, only bandwidth resources are considered.

Recent work has also been developed in the virtual server allocation's domain. [165] has proposed an optimal allocation approach for virtual servers among different DCs, i.e., having multiple DCs to store the user's requested virtual server, the algorithm defines the best one to place the servers. A dynamic, decentralized and self-organizing approach to the allocation of VMs to physical servers in public and private clouds is proposed in [166]. The approach is based on a Cross-Entropy Ant System (CEAS), where ants (i.e. intelligent agents) are used to discover physical servers and make allocation decisions. The system is able to dynamically react to changes in the load of physical servers, as well as to failures in the physical infrastructure (e.g. failures of servers, network links, etc). The mapping of VMs into physical servers is done using near-optimal heuristics.

Other works have looked more carefully to the cloud and network integration field, such as in [167], [168] and [169]. The work in [167] studies the delay minimization in the cloud network through a MILP formulation. In [168] the authors address the reconfiguration of the cloud network in order to maximize the energy savings, and propose two heuristic approaches benchmarked by MILP approaches. The trade-off between energy savings and delay minimization was later studied in [169] and a heuristic was proposed. However, these approaches do not look to complex requests of joint cloud and network resources, and focus mostly on the access of end-users to cloud DCs.

The work in [170] proposes a software-based and hardware-based methods to enhance the networking usage in DC environments. The software solution provides an algorithm to better locate VMs on physical machines, while the hardware solution improves the bandwidth usage across different physical machines. The authors in [171] consider both communication and computation constraints in a cloud environment. It proposes a greedy algorithm to design a cloud network with network characteristics in terms of communication and computation costs. However, the node modelling is based only on self-loop flows, which we believe not to be enough for a consistent solution.

The work in [172] proposes a preventive reliable VN embedding algorithm (PR-VNE) within the cloud's backbone network. The algorithm follows a metaheuristic approach. However, it lacks to provide an end-to-end embedding solution, considering both network and cloud resources.

2.6.2.2. Resource Adaptation and Optimization

The cloud online and dynamic environment requires VIs already deployed to be reconfigured or re-deployed in order, not only to satisfy VI requirements, but also to optimize the use of the physical infrastructure.

In [173] the authors propose a heuristic algorithm for re-deploying the existing VN effectively as the network evolves. The proposal focuses on satisfying node resource constrains and path delay constrains with the minimum upgrading cost of the VN.

The authors of [174] propose a slightly different mechanism that gradually performs reconfigurations based on estimation errors. The particularity of this proposal is that the estimation errors are calibrated and reduced as the reconfigurations on the network are performed.

Adaptation approaches mainly differ on their architecture (i.e. whether they are centralized or distributed), the techniques used to trigger the re-allocation process and the goal they aim at (i.e. whether it is minimizing cost, or avoiding or mitigating SLA violations). [175] presents a centralized system that automates the process of migrating VMs as a response to SLA violations. It uses a combination of black-box and grey-box monitoring techniques to generate distribution and time-series profiles to capture the variations in resource usage and their temporal correlations, respectively. Since the task of deciding which and where VMs to migrate is NP-hard, heuristics are used for this (i.e., the VM with the highest load and the lowest size is chosen to reduce migration overhead, and the target physical server is chosen as the one with the lowest load). Naturally this policy may not be the best approach when taking into account, for example, green aspects. On the other hand, [176] presents a green approach. VM migration decisions are taken in order to minimize the cost of running a data centre. A time-series forecast technique is used to predict the future usage of physical servers based on historical usage data. Based on the forecasting results, a centralized algorithm determines a new mapping of VMs to physical servers that aims at placing the given work load in a reduced number of physical servers with, at the same time, low probability of violating SLAs.

The work in [177] proposes a feedback control system for adjusting the allocation of physical resources to VMs. The system measures the performance of VMs, including the actual resource usage of an application along with its performance. Thus, the actual performance is compared with the desired one, and if necessary, the resource usage is adjusted accordingly.

The work in [178] presents a strategy that considers the ability to reconfigure currently mapped VNs when trying to map a new one. The acceptance ratio of the VNs and the impact of reconfigurations are evaluated. The authors formulate the mapping problem as an ILP problem, and also propose a heuristic to speed up the solving time of the ILP problem. In this work, VNs are characterized only by node CPU and link bandwidth. It is also important to mention that the formulations in this work provide part of the base ground for the strategies proposed in our work.

Also in the SDN and NFV scope, the VI embedding problem is being subject of research. In [179] the authors provide a brief overview of the state-of-the-art of SDN and NFV technologies over optical networks, and present a formal model for the VNF complex scheduling problem, using the complex job formultion. However, the problem formulation does not consider full VIs, where VNFs are directly hosted in VMs. [180] describes an architecture based on an orchestrator that ensures the placement of the virtual nodes and the allocation of network services on them, supported by a monitoring system that collects and reports on the behaviour of the resources.

Looking to a more specific scenario, [181] considers mobile network services to be deployed over a distributed network of cloud computing DCs. More specifically, it looks to avoid and minimize mobility gateway function reallocations. Furthermore, the authors show how this reallocation reduction can be reflected in an effective NF placement algorithm.

2.6.3. Multi-Domain approaches

A framework for VN embedding in multiple infrastructure providers is presented in [182]. The framework also encompasses a discovery of resources and the work in a hierarchical way, relying on Cluster Index Servers (CISs) and local management nodes. The management node of each infrastructure provider is responsible for maintaining and classifying the local virtual resources into conceptual clusters named Micro Cluster, while the CIS aggregates and organizes the conceptual clusters of multiple infrastructure providers with the same root attribute to form the Macro Cluster. The CIS then aggregates different resource clusters from multiple providers to create resource federation on a macro level to further implement the resource sharing effectively and efficiently.

The work in [183] also addresses the inter-domain issue and proposes a policy-based inter-domain VN embedding framework that embeds end-to-end VNs in a decentralized manner. A distributed protocol

coordinates the VN embedding process across infrastructure providers. Moreover it is presented a hierarchical mechanism for location aware VN request forwarding and a location awareness protocol.

The work in [184] studies the embedding of VNs in a multiple infrastructure provider's scenario. The embedding is formulated and solved as a MILP with the focus of decreasing the embedding cost for providers while increasing the VN acceptance ratio. This is an interesting work in the multi-domain area, but it is limited to VNs and not fully compliant with the scenario envisioned in this work, where there is one network domain and multiple DC domains.

Volley, a system for optimizing placement of application data across cloud DC, is presented in [185]. Migration decisions are done once per month based on request logs and collected statistics from all DCs, taking into account WAN bandwidth, data centre utilization and end-user latency. However, the system works in a static and centralized way and only performs optimizations at a specific moment, thus rapid changes in the system cannot be addressed. Decisions on which DC to deploy virtual resources only take into account the DCs location, and scalability may be a problem.

In [186] the authors extend a previous work [67], by also addressing the request split and allocation problem across several domains based on CSPs offerings.

2.6.4. Remarks

The joint manipulation of cloud and network virtual resources has not yet been widely explored. Nevertheless, there is a considerable amount of work in the VN embedding field, a problem that closely relates to the VI one, and that is a good starting point for inspiration to solve the VI embedding problem.

The algorithms found in the literature do not effectively take into account cloud resources such as computing, memory and storage, and selection is not "negotiated" with the network (e.g. bandwidth, latency). Nevertheless, the works presented in the literature were a very good starting point to our work. Furthermore, an ideal solution must also be enabled with green policies without neglecting business goals (i.e. profit).

In summary, despite the fact that some state-of-the-art proposals address some particular aspects of the VI embedding problem, these solutions are completely loose, i.e., they need to be grouped into an "almost" single solution. Consequently, new concepts need to be proposed, which naturally come out from our work. Our work does not ignore or reinvent the state-of-the-art solutions, as it builds upon some of the current solutions. For example, the work presented in [178] and [161] provides part of the base ground for our work.

It is also important not to forget that there are many other research aspects in the scope of the VI embedding problem that are not the focus of this Thesis, such as: how to deal with elasticity; take into account resource pricing; take into account security aspects; consider wireless scenarios. Nevertheless, the solutions proposed in this Thesis are able to accommodate these aspects in future works.

2.7. Future Internet Research Projects

Several research initiates have investigated new architectures and mechanisms for the integration of the cloud and network domains at different levels. This section describes some of the most recent European projects in this area, and that are of special interest to this Thesis scope.

2.7.1. 4WARD

The 4WARD (Architecture and Design for the Future Internet) project [187] aimed to create and design a Future Internet architecture, using a clean slate approach. One of the basic tenets of 4WARD is that the Future Internet shall allow multiple networking solutions to coexist, not only in the link and application layer, but also in the network and transport layer. In this sense, by being able to decouple the infrastructure from the services and further allow network service providers to share a common infrastructure, network virtualization was the chosen solution. Thus, 4WARD developed a systematic and general approach to network virtualization, named VNet, which addressed three main areas:

- Virtualisation of Network Resources – defined a generalised approach that allows the use of a broad variety of resources as part of a unified virtualisation framework, taking into account the performance of shared resources and the secure separation of VNs sharing a resource. A standardized interface for management and control of virtual resources was also developed.
- Provisioning of VNs – developed a systematic approach to instantiate complete VNs using virtual resources, allowing the on-demand deployment of new VNs on a potentially large scale (including discovery of available physical and virtual resources, as well as the scalable provisioning, control, and aggregation of resources to form complete networks).
- Virtualisation Management – management mechanisms supporting the deployment, control, and dynamic re-allocation of resources on-demand during the lifetime of a VN were defined.

2.7.2. SAIL

The SAIL (Scalable & Adaptive Internet soLutions) project's [27] main goal was to reduce costs for setting up, running, and combining networks, applications and services, and increase the efficiency of deployed resources.

SAIL looked at the integration of networking with cloud computing to produce cloud networking, which was refered in SAIL as CloNe. The on-demand concept of cloud computing was extended to the network, and both network and computing resources are managed according to variable demand. A novel cloud networking architecture supporting flash network slices (network resources) was developed. Figure 2-12 illustrates a use case of CloNe, where multiple cloud sites, implementing virtual processing and storage infrastructure, are connected by an operator's network. The interconnection of cloud sites and end users is made through flash network slices implemented by the operator. Furthermore, VI can be deployed on demand throughout DCs and network. In some way, SAIL extends the general concept of the 4WARD VNet to cloud computing and is one of the research projects that closely relates to the work of this Thesis. The developed framework in SAIL was evaluated through the deployment of a large scale prototype distributed across different sites in Europe.

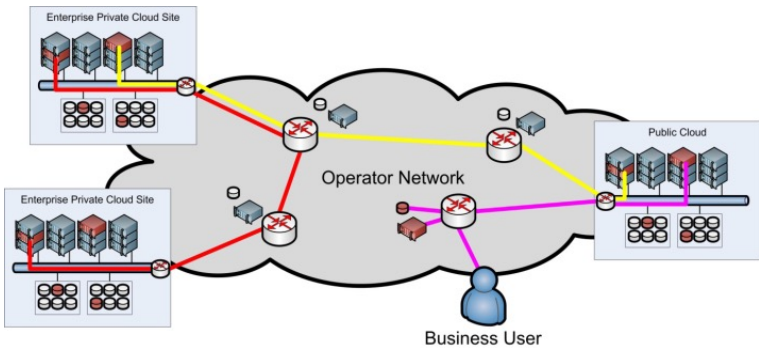


Figure 2-12: Enterprise virtual infrastructure of the Dynamic Enterprise scenario [189]

The SAIL's CloNe solution includes part of the work developed in the scope of this Thesis, namely part of the work presented in chapter 3 and chapter 4. However, it was not in the scope of the project to explore how the NO could take advantage of this integration environment to host its own SFs.

2.7.3. GEYSERS

The GEYSERS (GEneralised architecture for dYnamic infrastructure SERviceS) project [79] goal was to "qualify optical infrastructure providers and NOs with a new architecture, to enhance their traditional business operations". It is expected that this new architecture is capable of: seamless and coordinated provisioning of optical & IT resources; end-to-end service delivery to overcome limitations of network domain segmentation; a novel business framework for infrastructure providers and NOs; a novel mechanism to partition infrastructure resources and compose logical infrastructures; a cost and energy-efficient proof-of-concept implementation.

From GEYSERS perspective, optical network infrastructure providers will compose logical infrastructures and rent them out to NOs, while NOs will run cost-efficient, dynamic and mission-specific networks by means of integrated control and management techniques.

More detailed, GEYSERS proposed to:

- "Specify and develop mechanisms that allow infrastructure providers to partition their resources (optical network and/or IT), compose specific logical infrastructures and offer them as a service to network operators" [79]. Its purpose is to overcome the current limitations of networks/domain segmentation, supporting dynamic and on-demand changes in the logical infrastructures.
- "Specify and develop a Network Control Plane for the optical infrastructure, by extending standard solutions – Automatically Switched Optical Network (ASON)/ Generalized Multi-Protocol Label Switching (GMPLS) and PCE -, able to couple optical network connectivity and IT services automatically and efficiently, and provide them in 1 step, dynamically and on-demand, including infrastructure re-planning mechanisms" [79].

At a high level view, our work and the GEYSERS project aim at a common end: integrate network and IT and support dynamic and on-demand changes in the logical infrastructures. However, GEYSERS has a strong focus on the optical network infrastructure challenges, while we focus more on guaranteeing the independency between domains and the independency from physical network technologies. Furthermore, just like in the SAIL case it was not in the scope of the project to explore how the NO could take advantage of this integration environment to host its own SFs.

2.7.4. Mobile Cloud Networking (MCN)

MCN [28] is mainly exploiting cloud computing as the infrastructure for future mobile network deployment and operation. Furthermore, it is also looking to how a future MCN provider that takes advantage of cloud computing can exploit innovative value-added services like: EPCaaS, IMSaaS, and Radio Access Network (RAN) as-a-Service (RANaaS).

The project will develop a fully cloud-based mobile communication and application platform, assuming the existence of RANs, Micro and Macro DCs. Macro DCs are large-scale computing farms like those that exist today, located in strategic locations. Micro DCs are medium to small-scale server clusters across a certain geographical area (e.g. covering a city of a rural area).

MCN will define and implement an architecture that meets the real-time requirements of mobile NFs and support the efficient and elastic use of the resources. On the mobile NFs side, this will also require adaptation to allow higher decentralization and support elastic behavior. The end-to-end control and management of the entire environment is another key aspect of the MCN architecture. Finally, MCN aims also to integrate solutions like end-to-end SLA, monitoring, AAA, Rating, Charging and Billing.

2.7.5. UNIFY

UNIFY (Unifying Cloud and Carrier Networks) [80] aims to research, develop and evaluate means to orchestrate end-to-end service delivery from home and enterprise networks through aggregation and core networks to DCs. To do so, it aims to create a more flexible network able to open up new business possibilities while also reducing costs.

An automated, dynamic service creation platform, leveraging a fine-granular service chaining architecture will be developed. UNIFY aims to do this by creating a service abstraction model and proper service creation language to enable the dynamic placement of network, compute and storage components across the infrastructure. In this line of reasoning, UNIFY intends to improve virtualized network operations by creating an architecture that optimizes data traffic flows, taking advantages of SDN and networking virtualization technologies. Furthermore, it also looks to design a universal hardware node architecture with advanced programmability.

2.7.6. T-NOVA

T-NOVA's (Network Functions as-a-Service over Virtualized Infrastructures) [30] vision is to introduce and create a framework for Telcos to deploy VNFs. This framework's purpose is not only for operators to deploy NFs for their own needs, but also to be able to offer them to their customers as value-added services. NFs like gateways, proxies or firewalls, can be provided on-demand as-a-Service with the T-NOVA framework, eliminating the need to acquire, install and maintain specialized hardware at customers' premises.

From a more technical perspective, the T-NOVA framework will provide orchestration and management features for the automated provision, configuration, monitoring and optimization of NFaaS over virtualized Network/IT infrastructures. T-NOVA will leverage and enhance current cloud management architectures and SDN to support the deployment of NFs.

Last but not least, T-NOVA will work on a "NFV Marketplace" concept, where on one side NF providers can publish their NFs, and on the other customers can browse and select the services that best match their needs.

T-NOVA is in its early stage and part of the work developed in the scope of this Thesis (part of chapter 5) is planned to be included in the T-NOVA framework technical solutions.

2.7.7. Remarks

There are a considerable number of research projects looking to the topics cover by this Thesis. Some of the projects were used as "inspiration", such as 4WARD, while others were a way to promote our work, which is the case of SAIL, MCN and T-NOVA.

2.8. Standardization and Research Groups

With the prominence interest of the industry in making network services more agile, whether at the connectivity level or function level, and in bringing network and cloud together, several standardization entities have activities to explore these aspects, trying to provide guidelines and identifying potential needs for standardization. Herein are highlighted some of the most relevant standardization activities to this Thesis scope.

2.8.1. Open Networking Foundation (ONF)

The ONF [188] is the leading organization in the SDN area. In their own words, the ONF "is a user-driven organization dedicated to the promotion and adoption of SDN through open standards development." The OpenFlow protocol [98], which was the first SDN standard, is their most important outcome since their foundation in 2011. Most of the relevant players in the networking domain, including NOs and hardware and software vendors, are members of ONF, although the majority of board members come from service providers, R&D and user organizations.

The ONF is dedicated to promote the adoption of SDN through open standards development. It aims to speed innovation through simple software changes in telecommunications networks, wireless networks, DCs and other networking areas. Currently, it has dedicated working groups to tackle what are considered the most important issues related to SDN, which are: architecture and framework, configuration and management, extensibility, forwarding abstraction, market education, migration, northbound interfaces, optical transport, testing and interoperability, and wireless and mobile.

2.8.2. Metro Ethernet Forum (MEF)

The MEF has a set of initiatives within the scope of cloud and SDN. One of the most refered initiatives is the Carrier Ethernet for Cloud (CE4Cloud) [190]. The purpose of this initiative is to introduce the concept of delivering private cloud services via Carrier Ethernet WANs and services [191]. Furthermore, it aims to

foster the dialog between cloud industry stakeholders (enterprise users, CSPs, standards development organizations) as the MEF develops new Ethernet services more aligned with the nature of cloud services, i.e. with dynamic and on-demand features.

2.8.3. The European Telecommunications Standards Institute (ETSI)

Seven of the world's leading telecom NOs initiated in November 2012 the creation of the ETSI ISG for NFV. One can say that this was an important point in the acceleration of progress towards network virtualization. With approximately two years of work, the initiative counts with 2 whitepapers and a specification that counts with 5 out of 19 scheduled public deliverables ([123], [125], [126], [127], [128]). The purpose of the group is to define requirements and architecture for the virtualization of NFs, and also to address a set of technical challenges listed in [192]. Moreover, the initiative has developed a NFV PoC Framework to coordinate and promote multi-vendor PoCs, illustrating key aspects of the group work.

In the group's first phase, which is due in the end of December 2014, ETSI has defined requirements and an architecture for the virtualization of NFs – see section 2.4.2. With the first phase coming to an end, ETSI has already started drawing the guidelines for a second phase. This phase is refered to be much more implementation oriented, with one idea under consideration being the creation of an Open NFV Platform (ONP) which is refered to be hosted in a near future by the Linux Foundation. Nevertheless, what it official is that ETSI is considering during this second phase to produce normative specifications to enable end-to-end interworking of equipment and services.

ETSI ISG NFV is refered to as the biggest NFV initiative at the moment.

2.8.4. BroadBand Forum

The BroadBand Forum (BBF) [193] carries out technical work on different areas, among which are those related to NFV, SFC, and SDN. Regarding NFV, the BBF is looking on how NFV can be accommodated in Multi-Service Broadband Networks (MSBNs) [194]. Moreover, it is looking to the specific case of virtualization of the business [197] and residential gateways [198]. On the latter case, an analysis on what functionalities can and should be moved to the operator's network is being carried out. The aim is to facilitate the deployment, maintenance and evolution of existing and new capabilities without adding complexity to the gateway or the home network.

With respect to the SFC topic, at the moment the BBF is focused on studying the concept to later on provide guidance to the BBF's Technical and Marketing Committees to define broad network element requirements for implementation [195]. In the SDN domain, the BBF is focused on how telecommunication networks are able to evolve towards SDN (e.g. software upgrade only on network equipments) [196]. It is looking on how SDN concepts can eventually be introduced incrementally within the networks without having to rebuild the entire network.

Unfortunately, there is not much public information available with respect to these works.

2.8.5. Internet Engineering Task Force (IETF)

The IETF recently created the "Service Function Chaining" working group [199]. The group's purpose is to document a new approach to service delivery and operation. It aims to accomplish this by producing an architecture for SFC that includes the necessary protocols (or protocol extensions) to transport the necessary information to nodes that are part of the implementation of SFs and SFCs. Traffic Steering mechanisms will also be subject of study in the group. The SFC group will identify the information needed from network and SFs to support SFC, and how the information can be provided to the SFC architectural components.

Specifically, the SFC WG is chartered to deliver the following documents: Problem Statement, Architecture, Generic SFC Encapsulation, Control Plane Mechanisms, and Manageability. There are many draft contributions available; however, there is not yet a consistent contribution from this group.

2.8.6. Remarks

Among the refered initiatives within the standardization groups involved in the topics covered by this Thesis, ETSI's work is definitely the one that has higher impact on our work. The reader will come to realize this in chapter 5.

2.9. Summary (~1 pages)

This chapter provided a comprehensive overview of the State-of-the-Art on different research topics that are comprehended in the work developed in this Thesis, such as cloud computing, virtualization, SDN, and NFV. An overview of the refered concepts was provided, along with some of the most prominent activities that are currently taking place in those domains.

We highlight the fact that there are still many open issues and challenges to be addressed in each domain. If we look from an interdependency perspective between the domains, there is the need for: proper integration; guaranteed network QoS within and in the access to cloud services; adaptation of cloud offers to be able to accommodate NFV; need for networks to evolve to more dynamic architectures to support cloud and NFV requirements, among others.

More concretely, we want to highlight the lack of flexibility in the integration of cloud and NO domains. Most of the approaches assume the scenario where a single entity controls all domains. Moreover, to effectively integrate multiple domains, there is the need for technology agnostic solutions that are able to be framed in different technology environments. Most of the existing research work presented has focused on the technology dependent aspects and not so much on the agnostic ones. Chapter 3 presents a set of flexible solutions for these aspects.

From a networking perspective, on the NO side there is the need for more dynamic, complete and modular approaches targeted at WAN connectivity services, while on the cloud side more flexibility is required. Chapter 3 looks at the former case, while chapter 5 looks to the latter one.

From an NFV management and orchestration perspective, we noticed that currently available solutions are far from completed. Apart from the already mentioned need for an effective WAN integration, there is still a lack of support with respect to the composition of SFs (e.g. how to map SFs to virtual resource, how to perform SFC). These aspects are mandatory in an NFV "world" and are endorsed chapter 5.

Finally, the integration of cloud and network leads to the need for an appropriate management of resources. The joint manipulation of cloud and network virtual resources has not yet been widely explored. The algorithms found in the literature do not effectively take into account cloud resources such as computing, memory and storage, and selection is not "negotiated" with the network (e.g. bandwidth, latency). Nevertheless, there is a considerable amount of work that brought inspiration to address the VI embedding problem. Chapter 4 focuses on this topic, the resource management one, by looking to the VI embedding problem.

3. Cloud Computing and Network Operator Integration

After reviewing the State-of-the-Art related with the main research topics addressed in this Thesis, this chapter introduces the proposed foundations for the integration of the cloud computing paradigm and the Network Operator (NO). These foundations comprise a multi-domain architecture for the integration and joint management of cloud computing and NO services. The main building blocks of the proposed architecture are defined and presented along with an integrated resource management architecture. Furthermore, a mechanism for the on-demand establishment of connectivity services across cloud and NO domains is presented as a fundamental piece of the architecture. With the NO domain in focus, an architecture for the management of NO connectivity services in a cloud fashion way (i.e. self-service and on-demand) is presented, showing that it is flexible enough to be realized both with legacy network technologies as well as more long-term technologies like Software Defined Networking (SDN).

We start by specifying our multi-domain architecture, entitled CloudNet, based on the work published in Annex Paper A and Annex Paper B. Still in the scope of what was published in Annex Paper A, we detail on the integrated management of resources. Further, based on Annex Paper B we elaborate on how the coordination across domains is accomplished and present a mechanism for the on-demand connectivity negotiation.

Sustained in the work presented in Annex Paper B and Annex Paper G, we elaborate on the dynamic establishment of NO connectivity services as well as on the maintenance of their correct and optimal functioning. Both legacy network and SDN approaches are presented in detail.

Finally, we present a proof of concept that supports the architecture and its features and functionalities. In this latter part, an evaluation of the developed components is provided.

3.1. Architecture

The architecture herein presented lays on the aggregation broker principle, presented in section 2.5.1. From a technical perspective this allows the architecture to fit in any of the business scenarios presented in section 2.5, since the communication between all service domains (e.g. Infrastructure-as-a-Service (IaaS), Network as a Service (NaaS)) is done via service Application Programming Interfaces (APIs).

Figure 3-1 presents a high level view of the architecture for cloud and NO integration. The architecture was defined in the scope of this Thesis, although some similarities to the SAIL CLoNe architecture can be found as some of our building blocks were included in the SAIL CloNe architecture. Cloud and NO domains are considered to be managed by dedicated management systems, i.e. Cloud Management System (CMS)

and NO Management System (NOMS). Moreover, there is the Cloud Network Management System (CNMS), an aggregation brokering entity, which has an integrated view and control over cloud and NO domains.

The communication between the CNMS and the CMS(s) and NOMS(s) is done via service APIs which do not expose the underlying details of each domain. In other words, just like today in the cloud, these APIs only expose service abstractions (e.g. Virtual Machines (VMs) available). In other words, one can say that the system is prepared to work in a "black-box" view. By following such an approach, the architecture makes possible the interaction with external Cloud Providers (CPs) or NOs. Nevertheless, this does not exclude, e.g. in the case where a single entity owns all domains, the existence of other interaction mechanisms to allow a more detailed view of the domains at the CNMS level.

The CMS exposes a service interface just like most cloud services do today [62] [200]. However, NO domains do not have such interfaces for their connectivity services: there was neither an apparent need for them, nor NOs were in the past willing to give certain control to external entities. For the sake of integration of cloud and network, we argue for the need of such interface. In this line of reasoning we consider a north-bound interface on the NOMS that can be used by other entities (e.g. in this case, the CNMS) for requesting connectivity services. Just like traditional cloud infrastructure interfaces, it supports Create, Read, Update and Delete (CRUD) operations on abstract logical resources. Section 3.4 elaborates on the service abstraction provided by the NOMS service interface.

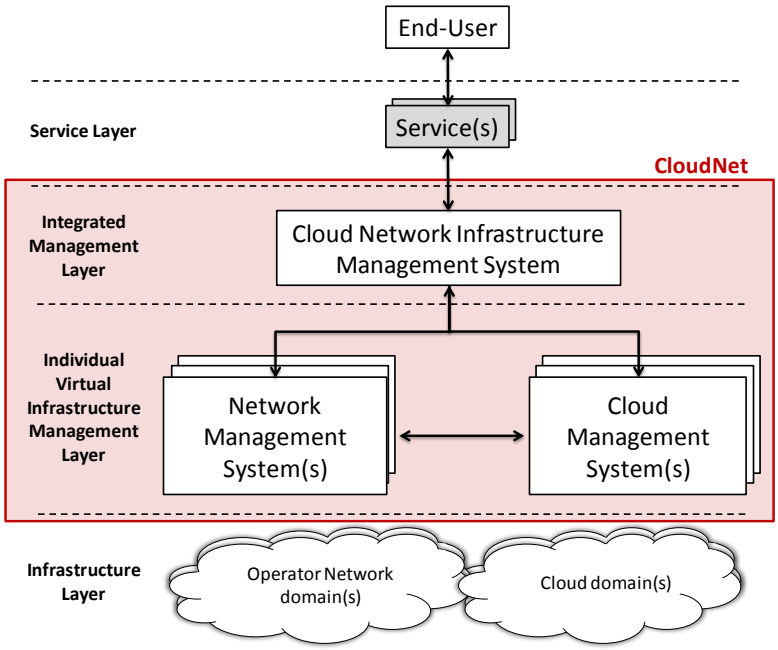


Figure 3-1: CloudNet platform - high-level architecture

To instantiate a connectivity service that spans multiple domains, a multitude of configurations need to be performed on all affected nodes in the network. This typically requires detailed knowledge about the network equipment in use and the design of the involved networks, for example, the topology. NOs typically consider these as sensitive and business critical information which should not be exposed to the tenant requesting the Virtual Infrastructure (VI) or other domains. To support this separation of concerns, a special interface and associated protocol have been defined allowing negotiation of configuration details. This is the east-bound interface between the CMS and the NOMS. Details about this interface are presented in section 3.3.

3.2. Integrated Resource Management

The discovery, allocation, adaptation and re-optimization of resources, addressing simultaneously both network and cloud resources, are the main inherent challenges of resource management in an integrated environment. The management of these resources lays upon the concepts of virtual resource mapping in

the physical infrastructure with self-organized reconfiguration of resources, devices and associated network, according to the services and user requirements, policies (with respect to e.g. location) and changes in the infrastructure.

In Figure 3-2 we present the management block diagram of the CNMS composed by three main blocks: the *Resource Management* block; the *Fault Management* block; and an underlying block entitled *Integrated view of resources*. The former is composed by three sub-blocks: the *Resource Discovery* block; *Resource Allocation* block; and the *Resource Adaptation & Optimization* block. The *Integrated view of resources* has the purpose of providing the upper blocks with the domain agnostic ability to view and interact with resources, whether they are cloud or network resources. Regarding the *Fault Management*, it is illustrated in the picture to facilitate the interpretation of the management system, mainly regarding its interaction with the *Resource Adaptation & Optimization* sub-block.

Although the management block diagram is presented as part of the CNMS, one should not forget that the CMS should also have a similar management system. In this case, the network resources are those within the Data Centers (DCs) providing the connectivity between virtual resources (e.g. VMs).

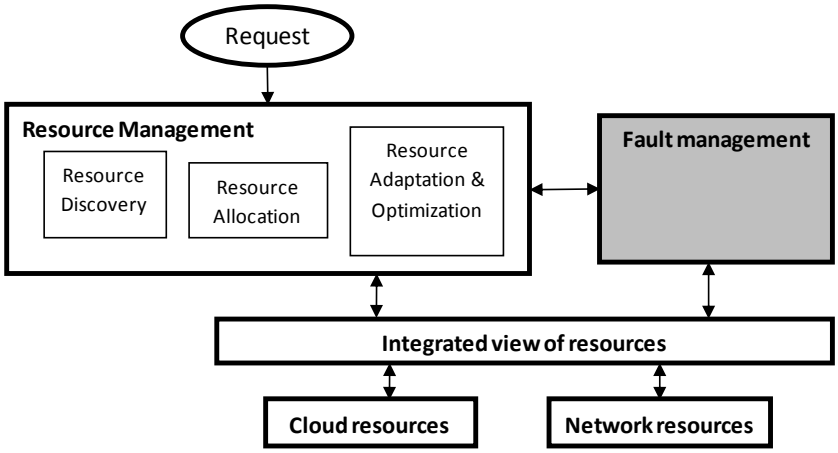


Figure 3-2: Cloud Networking Management diagram.

3.2.1. Resource Discovery (and Monitoring)

A fundamental requirement in virtualized environments is the integrated view of the existing physical and virtual topologies, the resources' characteristics, and the status of all network elements and links. This knowledge can be provided either by a centralized or by a distributed approach [201].

Today the cloud, i.e. DCs and NOs, are two distinct domains which we aim to integrate. However, there are boundaries that cannot be crossed as these domains will not be willing to share full information about them. In this approach, we assume to have access to network information such as topology and physical resources, as well as the ability to retrieve information on the virtual resources that the physical resources may host. On the DC side we do not expect to have such detailed information, we rather expect to see a DC as a single node in the network with a capacity and a set of associated information elements (similar to today's cloud services, e.g., instance types, available OSs, pricing, location).

3.2.2. Resource Allocation

Virtual resources shall be provisioned and placed in an optimal location according to the available resources at the time of the request, based on a number of possible criteria from both cloud and network, e.g.: type of VMs and possible restriction on location of these VMs; latency, bandwidth topology, geographical places where users will access the service, and other possible restrictions.

In order to map resources, it is needed a combined mechanism that performs balanced decisions taking into account the abovementioned requirements of both network and cloud resources. This mechanism must be able to determine a possible solution, i.e., physical hosts are able to allocate the cloud resources which, at the same time, can have an associated network service able to fulfil the requirements in the access to the cloud. Chapter 4 is focused on this subject and presents a set of solutions to perform resource

allocation, both within a single domain (e.g. one DC) and across multiple domains (e.g. across several DCs connected by a NO).

3.2.3. Resource Adaptation and Optimization

With the dynamism of the cloud, reconfigurations and re-optimizations become common operations. These operations are needed in several situations: new virtual resources being instantiated, existing virtual resources being resized, released or migrated, business policies, or triggered by unexpected events (e.g. node or link failure). These unexpected events are triggered by the FM which is responsible for monitoring the resources, detecting faults and collecting performance metrics.

Depending on the specific environment, actions can be taken at different levels: in the cloud, in the network, or in both. Thus, mechanisms are required for extending or moving cloud resources within one DC or across DCs, creating new network paths and reconfiguring existing ones (e.g. need for more bandwidth, less latency, failure, load balancing network resources). Those algorithms must decide on (1) when to reconfigure and (2) how to reconfigure. These decisions must be done based on information provided by the FM, or by an explicit request from the user. Chapter 4 also addresses adaptation and optimization topic by focusing on reconfiguration processes triggered by the arrival of new virtual resources.

3.3. Coordination across domains

3.3.1. Scope

From an administrative standpoint the notion of domain refers to a collection of resources involving hardware and software managed by a single entity, e.g. CMS, or NOMS. However there are challenges on how to negotiate on-demand connectivity between domains. In this section we present a protocol for the dynamic negotiation of connectivity services between domains. We describe how the protocol fits the needs and how it can be integrated in current technical solutions.

Several use-cases can be addressed, whether to connect users directly to the cloud (e.g. virtual desktop, online gaming) or to create distributed clouds in which cloud resources are spread across domains. The use-case depicted in Figure 3-3 focuses on the latter case.

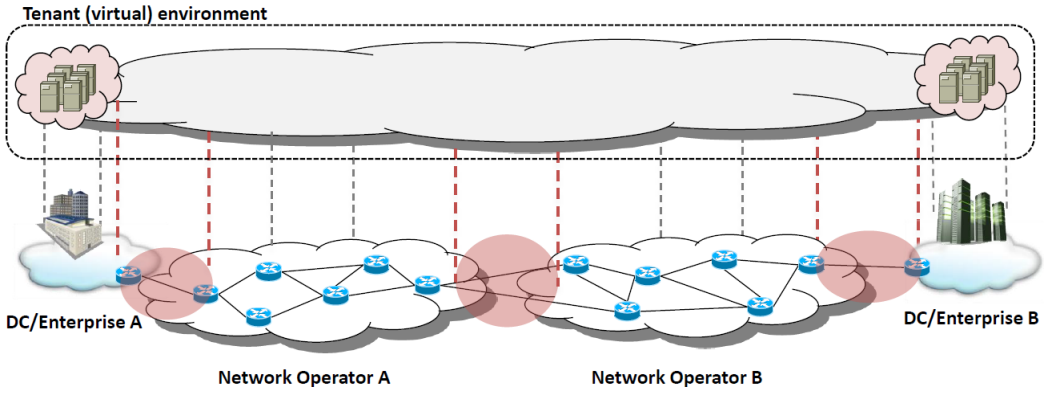


Figure 3-3: Use-Case

Figure 3-3 depicts a single tenant (virtual) environment composed by two portions of resources hosted in different DCs connected over a Wide Area Network (WAN) connectivity service. Note that this single environment involves four domains. The remaining of this section is focused on how these domains coordinate among themselves for the establishment of end-to-end connectivity.

There are several logical and physical entities that need to work together in order to establish the connectivity that spans multiple administrative domains. These entities are the various interfaces, protocols, controllers and input-output processing functions that enable two or more infrastructure service providers to interact and exchange domain specific information needed for the establishment of the

connection. Together we refer to them as Interdomain Coordination Framework (ICF) for distributed infrastructures.

3.3.2. Inter-Domain Connectivity Protocol

A Composite Request (CR) represents a high level description of a VI requested by the user of the CNMS. A VI specified in a CR can span multiple administrative domains. In such cases, the CNMS will decompose the CR into a number of sub-requests for each sub-domain. The individual parts of this VI within each administrative domain need to be connected to its other parts distributed in one or more other domains. Naturally, the link is the virtual resource that will cross domains, and therefore will make these connections.

For this purpose we have defined the *Link Negotiation Protocol* (LNP), which is implemented by the ICF. This protocol is responsible for creating one or more virtual links belonging to the same VI but may be spanning multiple (usually two) domains.

The design of this protocol has the following high-level objectives:

- Simple and low level protocol agnostic
- Support of various transport network solutions (L2, L3)
- Agnostic to any particular networking implementation

First of all it is important to highlight and define three terms:

- **Virtual Link**: A link in the VI.
- **Service Provider Logical Link (SLL)**: A logical data transmission link from one domain to another. This is usually installed and configured by the service provider. Examples of SLL can include physical layer separation schemes like Provider Backbone Bridging or Media Access Control Address (MAC)-in-MAC or tunneling schemes like Internet Protocol (IP) Security (IPsec) or Generic Routing Encapsulation (GRE). Each SLL has a reference to a physical or logical link, which in the latter case means that there might be aggregation happening below this layer at the physical links as well. It is important to note that this work is not concerned with the creation or configuration of SLL, and it assumes that they are already in place. It merely uses them via references.
- **Tenant Logical Link (TLL)**: A logical link part spanning between two domains. This corresponds to a virtual link in the VI of the tenant and has a one to one mapping. The main difference is that a virtual link must be unique within one VI, whereas a TLL must be unique across the two domains in all VIs. For example, Virtual Local Area Networks (VLANs) may be used for creating TLLs and ensuring isolation, in which case the VLAN number can act as a unique TLL. The TLL can have additional configuration parameters, such as interface addresses of the two endpoints and a routing protocol like in the case of a L3 link. The TLL comprises virtual slices of an SLL.

The LNP offers three main functions: *Create*, *Update* and *Delete*. There is another function, *Route Export*, for the cases where there is the need to explicitly export a route to a remote domain. The operations are detailed below.

3.3.2.1. Create Function

The process of creating a TLL between two domains (refered in our description as Domain A and Domain B) is illustrated in Figure 3-4 via a message sequence diagram. Under the natural assumption that both domains have received two "pieces" of the same VI, the element spanning the two domains is a virtual link (one or several). For that reason we refer to "Link Request" in the diagram as the request for a virtual link that will cross and connect across these two domains.

From a high-level perspective and depending on the type of link being established (i.e. whether L2 or L3), the *link negotiation* process can have either one or two phases. For the establishment of a L2 TLL, the process comprises only the L2 Negotiation phase. After receiving the request, one of the domains will trigger the negotiation process. The decision about which domain takes the initiative is outside the scope of the protocol (in practice it is usually the NO).

Table 3-1: Link Negotiation: Overall Message Parameters

Parameters	Description
infra id	identifier of the virtual infrastructure
transaction id	identifier of this transaction
msg type	type of message
sender	message sender
service type	type of service
virtual link	Details of the virtual link: virtual link id, in and out bandwidth, TLL information

In our example we assume Domain A to be the "initiator". After the trigger, the "initiator" starts the process by listing for each TLL, the various SLLs able to accommodate it. This information is then sent to the "receiver", Domain B, using the *Link_Offer* message. The remaining parameters of the *Link_Offer* message can be seen in Table 3-1. The parameters in this table are common to all messages in the protocol. *transaction_id* is used to identify this transaction among the set of ongoing transactions at any moment . The *infra_id* identifies the VI, and the *virtual_link* contains the information about the virtual link crossing two domains. This information is known in advance by both domains (via the north bound interfaces). The remaining elements, *service_type*, *in_bw* and *out_bw* (in and out bandwidth), are used in the perspective of guaranteeing consistency, since this information is expected to be known in advance by both domains.

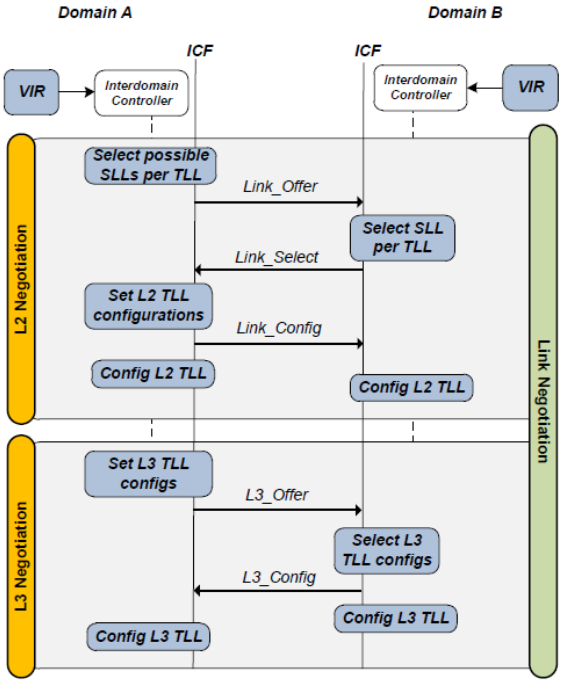


Figure 3-4: Creation Function - sequence diagram

Upon receiving the list of SLLs for each TLL, Domain B selects one SLL per TLL, and sends that information in the *Link_Select* message along with the type of encapsulation scheme used for the establishment of each TLL. Domain A is then responsible for setting the encapsulation scheme configuration attributes of each TLL, sending that information to Domain B using the *Link_Config* message. Table 3-2 shows the message parameters. At this point both domains have the necessary information to establish a L2 link.

Table 3-2: Link Negotiation: Parameters for L2 Negotiation

Parameters	Description
TLL_id	Identifier of TLL
SLL_offer	list of SLLs offered to carry the TLL
SLL_id	SLL id selected to carry the TLL
encap_scheme	identifies the encapsulation scheme (type and attributes) for the TLL

If the virtual link is a L3 link, the process continues and goes to the *L3 Negotiation* phase. Note that the message sequence does not imply that a L3 link can be configured only after a L2 link is configured; this design choice is taken to reduce the total number of messages. This avoids a new sequence with some duplicate information (like identifying the SLL) in both L2 and L3 configurations. The "initiator" sends the *L3_Offer* message with the L3 configuration parameters, i.e. the IPs, to be configured in the endpoints of each TLL, and a list of the supported routing protocols by each TLL. The "receiver" selects the protocol and informs the "initiator" using the *L3_Config*. All parameters for establishing a L3 TLL(s) are now known and the configuration can be applied. The parameters for the L3 Negotiation phase are presented in Table 3-3.

Table 3-3: Link Negotiation: Additional Parameters for L3 Negotiation

Parameters	Description
L3_config	L3 configuration parameters: IPs for the link endpoints (source and destination) and a list of the supported routing protocols.

3.3.2.2. Update Function

Updating or reconfiguring parameters are natural actions in the lifetime of a VI. Such expected actions can be triggered by a user specific update of the VI, or by the domain's management policies for reconfiguration. In either case, if the change in the VI is associated to a link, then the corresponding TLLs may need to be reconfigured. When the update to a TLL is due to an update on the VI, the domain who initiated the TLL creation process is also responsible for starting the update process. On the other hand, when the update is triggered by an internal domain policy, either the "initiator" or the "receiver" must be able to start the process.

There are two possible scenarios (see Figure 3-5): an update is requested by the "receiver" (case A) and an update is triggered by the "initiator" (case B). In the former, case A, the process is initiated using the *Link_Reconf* message, which is a request for reconfiguring one or more TLLs in a certain VI. The "initiator" will, identical to the creation process, prepare a list of SLLs for each TLL and send it to the "receiver" using the *Link_Update* (which in terms of parameters is equal to the *Link_Offer*). From this point on, the process is similar to the creation process.

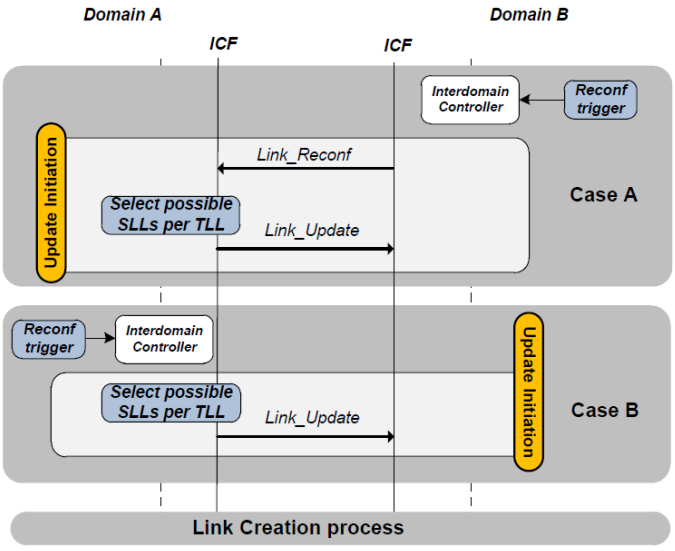


Figure 3-5: Update Function - sequence diagram

3.3.2.3. Delete Function

The delete process for a TLL, shown in Figure 3-6, is always triggered by the "initiator". Since both domains are expected to have received the delete request, the "initiator" domain usually takes the lead on this process. The *Link_Delete* message identifies the VI and the TLL(s) to be deleted.

3.3.2.4. Route Export Function

In the case of L3 TLL(s), for some domains, the defined routing protocol may be static. For such cases, the route export function was defined to cover the need of exporting a route to another domain. To do that the domain wanting to export routes related to a certain VI uses the *Route_Export* message to send one or more routes to a remote domain. Figure 3-7 illustrates the process for the case in which Domain B is exporting a route to Domain A.

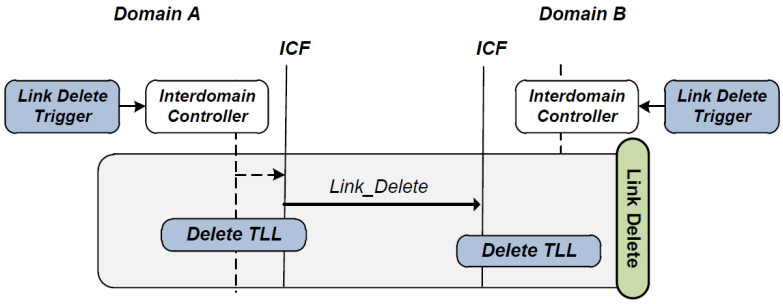


Figure 3-6: Delete Function - sequence diagram

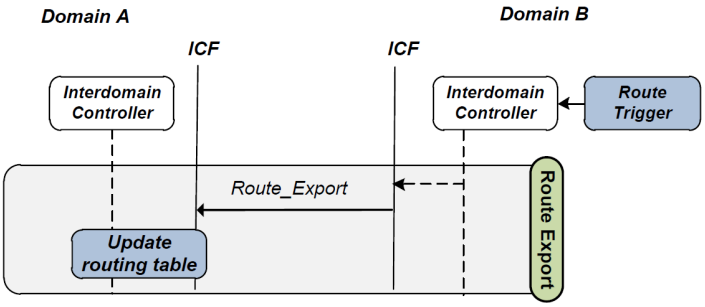


Figure 3-7: Route Export Function - sequence diagram

3.4. Dynamic WAN Connectivity Services

In this section, we first describe the service abstraction considered for WAN connectivity services. We then detail how to apply this abstraction to a legacy network approach and to an SDN network approach.

3.4.1. Service Abstraction

As refered in section 3.1, the NOMS is considered to provide a service interface for connectivity services, similar to today's cloud service interfaces. From a service abstraction perspective, we consider the notion of logical switch (for layer 2 connectivity) or a logical router (for layer 3 connectivity). These abstractions map well into the traditional Virtual Private Network (VPN) service model, providing a simple and intuitive abstraction of the connectivity service. Furthermore, it is considered that this interface relies, just like the cloud interfaces, in Representational State Transfer (REST) principles. Bellow it is provided an overview of the interface abstraction model.

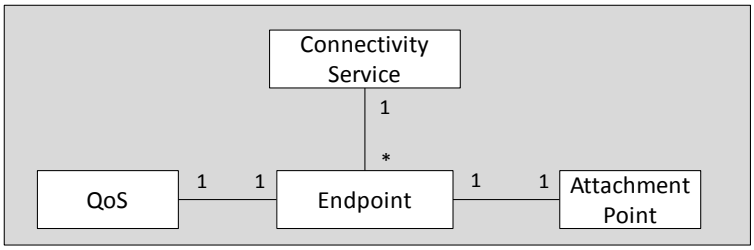


Figure 3-8: WAN Service Abstraction Model

Figure 3-8 depicts the service abstraction model. A C*onnectivity Service* has associated a set of E*ndpoints* that have direct mapping to *Attachment Points* (e.g. DC virtual environment, enterprise customer site). Furthermore, each *Endpoint* has an associated Quality of Service (QoS), *QoS*, which defines bandwidth and latency requirements.

Table 3-6 and Table 3-7 present the attributes of each element. The *id*, *name*, and *description* attributes are shared by all entities. The *id* refers to a unique identifier of the resource, while *name* and *description* are human readable information to better help characterize the resource. Furthermore, the resources that can be manipulated by the tenant have also an associated attribute, the *tenant_id*. The *endpoints* attribute corresponds to a list of endpoint identifiers, while the *type* attribute refers to the type of connectivity service, i.e. L2 or L3 type of service. The attribute *attachment_id* and *qos_id* refer to unique identifiers for *Attachment Point* and *QoS* resources respectively. Finally, the *bandwidth* and *latency* attributes refer to the bandwidth capacity and latency values associated to the *QoS* resource.

Table 3-4: WAN Service Abstraction Model: Connectivity Attributes

Attribute	Type	Default Value	Required	CRUD	Notes
id	uuid	none	Y	R	
name	string	none	N	CRU	
description	string	none	N	CRU	
tenant_id	uuid	none	Y	CR	
endpoints	list	none	Y	CRU	
type	string	none	Y	CR	L2, L3

Table 3-5: WAN Service Abstraction Model: Endpoint Attributes

Attribute	Type	Default Value	Required	CRUD	Notes
id	uuid	none	Y	R	
name	string	none	N	CRU	
description	string	none	N	CRU	
tenant_id	uuid	none	Y	CR	
attachment_id	uuid	none	Y	CRU	
qos_id	uuid	none	Y	CRU	

Table 3-6: WAN Service Abstraction Model: Attachment Point Attributes

Attribute	Type	Default Value	Required	CRUD	Notes
id	uuid	none	Y	R	
name	string	none	N	RU	
description	string	none	N	RU	
tenant_id	uuid	none	Y	R	

Table 3-7: WAN Service Abstraction Model: QoS Attributes

Attribute	Type	Default Value	Required	CRUD	Notes
id	uuid	none	Y	R	
name	string	none	N	R	
description	string	none	N	R	
bandwidth	string	none	Y	R	kbps
latency	string	none	N	R	ms

QoS requirements are very sensitive and we consider the NO not to allow the arbitrary definition of these requirements. Therefore, we consider QoS requirements to be pre-defined by the NO and cannot be edited by the tenant of the NOMS. The tenant can only associate the QoS(s) to endpoint(s). Moreover, it is up to the NOMS to validate the request, and this is especially important when it comes to QoS requirements. For example, a QoS has a certain latency defined; however, latency is intimately associated to the actual attachment points requested for association. A tenant can specify a connectivity service between two endpoints that cannot guarantee the requested QoS. Therefore, it is up to the NOMS to validate the QoS, and in the case of not being able to support the required QoS, it returns back the available options.

3.4.2. Legacy Network Approach

As emphasised in the previous section, the considered service abstraction maps well into the traditional VPN service model. Furthermore, operator VPNs (e.g. [91]) are among the most used connectivity services, and Business Support System (BSS)/Operational Support System (OSS) systems already host the necessary mechanisms to provision these services (although not in a cloud oriented manner, i.e. on-demand and self-service). Therefore, one can say that the accomplishment of the provisioning of these services relying on operator VPN services is an obvious choice.

Figure 3-9 depicts the architecture of the NOMS when relying on a legacy network. The Core Management System and Resource Controller are considered to be legacy OSS systems, while the Service/Application/Frontend is considered to be an improved BSS system. This BSS allows the on-demand and self-service provisioning of the connectivity services to rely on the OSSs to enforce them. Next to the Core Management System is the ICF module for the establishment of connectivity service across domains. The ICF is considered a separate module that interacts with the Core Management System.

In this approach we consider the OSSs and the actual VPN technologies to support self-healing features. In other words, if there are faults in the NO, these are considered able to react to these faults. This is considered because it already happens in real NOs.

Although being able to accomplish the provisioning of the dynamic WAN services following a legacy approach it is obvious that there are certain limitations, such as the slow response time of legacy OSSs and VPN technologies. This will become evident in the evaluation section (section 3.5.3).

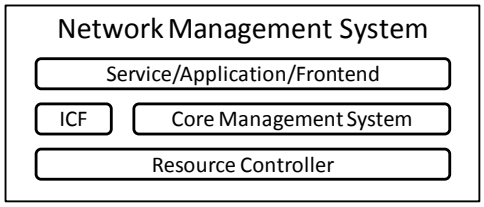


Figure 3-9: NO Management System Architecture – Legacy Network Approach

3.4.3. SDN Network Approach

Just like in the legacy network approach, the overall architecture in this case also comprises a Service/Application/Frontend block, in addition to an SDN control layer. In this case there is little work on how to provision the envisioned connectivity services following an SDN network approach. In this sense, this section presents an SDN framework able to accomplish this. The framework allows the creation and management of network connectivity services over an OpenFlow based network. Moreover, it assures the integrity of these services (fault-management), as well as the optimal usage of the infrastructure (run-time management).

The framework's architecture is presented in Figure 3-10. The control grid of the framework is composed by four modules: *SDN Application*, *Activator*, *Monitoring* and *Resource Mediator*. It is responsible for the connection between the data plane, where the forwarding elements are, and the applications plane, the *Service*. Moreover, it is also responsible for assuring the optimal operation of the network. The individual functionalities of the different modules are described below.

• ***Service/Application/Frontend***: provides a mean for users to make connectivity requests, and it can be seen as a frontend for the framework. It has three functionalities: *Service to SDN Application Translation*, *Service/Application Repository* and *SDN Application Repository*. The *Service to SDN Application Translation* functionality translates connectivity requests into the *SDN Application*, which are then persistently stored using the respective *Service/Application Repository* and *SDN Application Repository* functionalities. This can be roughly considered as an enhanced BSS/OSS system.

• ***SDN Application***: this module comprises the network intelligence by using algorithms to find data paths for individual flows. It has three functionalities*: Flow Allocation*, *Event/Fault Detection*, and *Flow Optimization*. The first refers to the ability to find data paths for flows. The second functionality refers to the evaluation of network changes that can eventually trigger optimization processes. The third functionality consists on reacting to generated events (e.g. a link failure event may lead to flow reallocations). Moreover, the flow allocation algorithm used is the Dijkstra algorithm considering link bandwidth capacity - equation (1). The set of nodes in the physical infrastructure is denoted by N^P; N^{flow} denotes the two endpoints of a flow, bw^{ij} denotes the bandwidth capacity of link $ij \in N^P$, and finally, ${Cost_{bw}}^{mn}$ refers to the cost to allocate flow *mn* in the network.

$$ {Cost_{bw}}^{mn} = \min\left(\frac{1}{\sum_{ij}^{N^P} bw^{ij}}\right) mn \in N^{flow} \quad (1) $$

• ***Monitoring***: this element identifies, monitors, saves and provides information regarding network elements. Two of the functionalities, *Real Time Network Discovery* and *Event Generation*, are subscription based. The first one uses information provided by the controller to retrieve real-time information regarding the network topology and individual links bandwidth. The *Event Generation* functionality relates to the *Network Statistics* functionality, which gathers network statistics to generate events regarding link and switch usage. These event messages are forwarded to subscribers. It should be noted that the module supports multi-tenancy.

• ***Activator***: this module uses two functionalities, *Flow Translation* and *Flow Enforcement* to achieve its goal: store rules in the network elements. The first one consists in the translation of flows sent by the *SDN Application* into rules for individual network elements. The second one is accomplished by communicating to the *Resource Mediator* the rules to be enforced in individual switches.

• ***Resource Mediator***: this element provides the necessary abstraction from individual network elements to upper layers. To fulfill this objective, one functionality is used, *Network Device Mediation*, which establishes the communication between the framework and the SDN network.

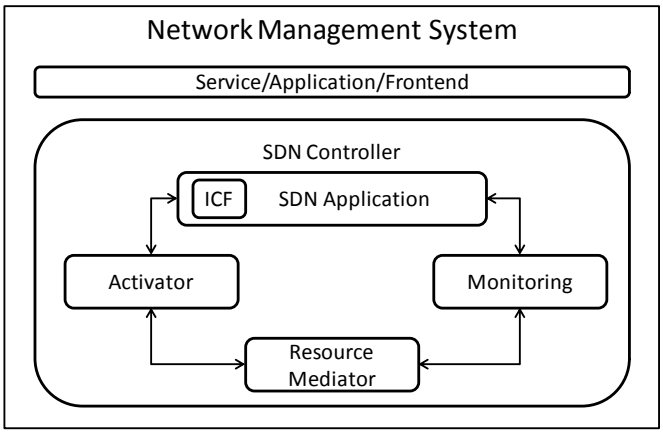


Figure 3-10: NO Management System Architecture - SDN Network Approach

3.5. Proof-of-Concept

3.5.1. Scenario

In order to validate, evaluate and demonstrate the advantages of the architecture and mechanisms proposed in this chapter, a Proof of Concept (PoC) has been developed. The scenario envisioned, depicted in Figure 3-11, consider: two CP's DCs, one NO, and two customer sites. Each DC is managed by a CMS and the NO is managed by a NOMS. Furthermore, we consider all domains to be owned by the NO. The NO also contains a CNMS.

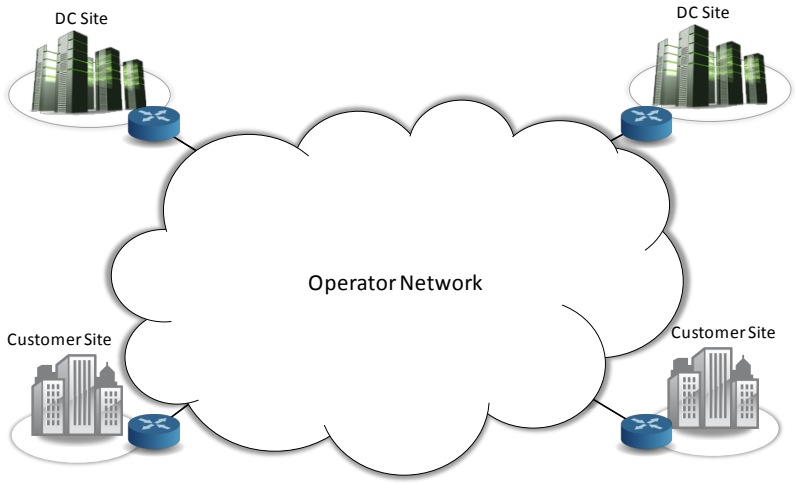


Figure 3-11: Proof-of-Concept Scenario

In this scenario, the CNMS is able to receive requests for the two DC locations and, if necessary, it is able to establish WAN connectivity services, whether to connect virtual resources across the two DCs, or to connect a customer site to virtual resources within the DCs.

From an end-user service perspective, two types of services are provided: IaaS with two possible locations for the virtual resources; and a service which allows the end user to establish connectivity services across his cloud resources and registered sites (section 3.4.1). We will define this later service as VPN as-a-Service (VPNaaS). While from a standalone perspective the IaaS service is identical to the offers today, the VPNaaS is a new concept of service. In summary, the VPNaaS exposes the service abstraction presented in section 3.4.1.

3.5.2. Technologies and Implementation

From a technology viewpoint, the PoC relies on multiple technologies, some are already existent and used currently, others are adapted, and others are built from scratch due to their novelty. In this section we provide some details about the technologies used per domain.

The software developments done rely on the *python* programming language.

3.5.2.1. Cloud Network Infrastructure Management System

The CNMS has been developed for the purpose of this work. The communication to the NO domain is done using an extension of the Open Cloud Computing Interface (OCCI) interface to support the WAN service abstraction model presented in section 3.4.1 [61]. Currently, the communication to the cloud domains can be done using OCCI, OpenNebula API, or the OpenStack Interface. It is relatively easy to add other APIs as the implementation follows a plugin-approach. Currently, the north-bound interface has been developed for this work.

This component also comprises two MySQL databases:

- Client Database - stores the client information (services and profile)
- Infrastructure (cloud, WAN, sites) Database – stores access credentials to the various VI Managers (VIMs), infrastructure topology, and resource information (i.e. available and used).

More details about the internal implementation of this component can be found in [32].

3.5.2.2. Cloud Management System

The CMSs currently in place to manage the DC domains rely on OpenStack and OpenNebula. For both cases, an ICF component that supports the LNP has been integrated. This latter component has been developed for this PoC.

Each DC domain is comprised of two computation nodes. We highlight that for the sake of the operation of the PoC, the size of the DC is not relevant.

3.5.2.3. Network Operator Management System - Legacy Network Approach

In this case, the legacy network relies on a Cisco IP/ Multi Protocol Label Switching (MPLS) network composed by four Provider (P) routers connected in a full mesh and four Provider Edge (PE) routers – see Figure 3-12. Each PE router is connected to one P router. The PE routers then connect to the Customers Premises Equipments (CPEs), whether they are from a cloud domain, or a customer site domain. Figure 3-9 illustrates the testbed with the mentioned setup.

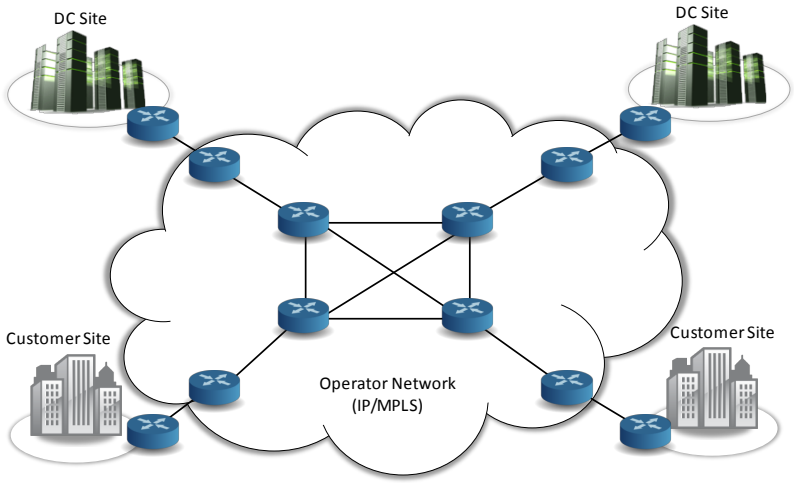


Figure 3-12: Testbed Scenario with Legacy Network

The entire NOMS core has been built from the start for this demonstrator, except for its resource controller module. For this latter component, a Portugal Telecom (PT) Inovação e Sistemas (PTInS) proprietary OSS, called Network Activator (NA) [202], has been used. For the northbound interface, as refered earlier, an extension of the OCCI interface that provides the WAN connectivity service abstraction has been used. Just like in the CMS, the ICF component developed has also been integrated with the NOMS.

3.5.2.4. Network Operator Management System - SDN Network Approach

In the SDN network approach, the PoC relies on an OpenFlow network composed by four computation nodes running the Open vSwitch software [203] – see Figure 3-13. With respect to the SDN controller layer, two distinct implementations were performed: the one presented in [18], which partially relies on the Floodlight SDN controller; and another that relies on the implementation presented in [35] that relies on the OpenDaylight SDN Controller. The reason for the two implementations relies on the fact that, by the time the former was used, the OpenDaylight controller had not yet been released, and Floodlight was one of the most promising SDN controllers. In the meanwhile, OpenDaylight was released and its prominence has been increasing. With this in mind, we pursued the second approach. It is the OpenDaylight approach that is currently in place in the demonstrator.

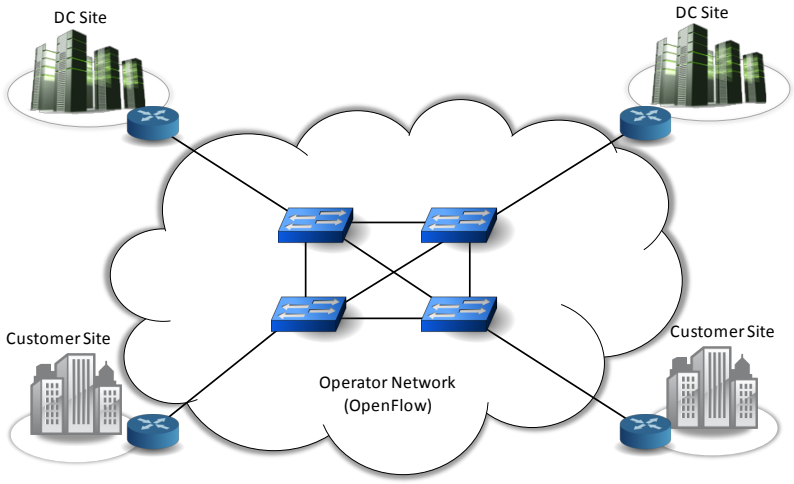


Figure 3-13: Testbed Scenario with SDN Network

3.5.3. Evaluation

In this section, we present a set of the evaluations performed which we consider to be the most relevant ones for the sake of this Thesis. Further evaluations can be found in [32], [33], [34], and [35]. All values presented comprise a mean of 10 measurements and a confidence interval of 90%.

3.5.3.1. Cloud Domain

Table 3-8 presents the average time for the deployment of one VM with standard specs (1vCPU, 2GB of Random-Access Memory (RAM), and 50GB of storage) within the cloud domain. It takes around 6 minutes from the point the request reaches the CMS until the VM is up and running (i.e. it has already booted and its services are all running).

Although this time is very much dependent on the physical computation nodes, it is considered within the average time experienced in public cloud services (e.g. Amazon EC2).

Table 3-8: DC Overall Service Establishment

Phase	Time (s)	Confidence Interval (90%)
DC Management System	378.4	1.734

3.5.3.2. Network Operator Domain - Legacy Network Approach

For the NO domain following a legacy approach, we first present in Table 3-9 the processing time for three phases: request process – time since the NOMS receives the request until it triggers the LNP; LNP – time for the connectivity negotiation; request enforcement – time since the connectivity negotiation ended until the request has been enforced in all network elements. As it is possible to see, the time for enforcing the request is the longest one. This is due to the inherent nature of the OSS used which follows a series of pre-defined procedures until it in fact enforces the actual configurations on the network devices. Although this could be eventually optimized, this was considered out of the scope of our work.

Table 3-9: Legacy NO Management System Analysis – Processing time

Phase	Time (s)	Confidence Interval (90%)
Request Process	29.316×10^{-3}	0.068×10^{-3}
Link Negotiation Protocol	15.767×10^{-3}	0.036×10^{-3}
Request Enforcement	18.713	1.334×10^{-3}
Total	18.779	0.068×10^{-3}

Furthermore, two other factors have been evaluated:

- **Delay Time:** the time it takes the NOMS to start processing the request after it has been sent. Ideally this should be the same as the local time, but due to a lock mechanism, the request may be put on hold until the lock is released.
- **Total Time:** the time it takes the NOMS to complete the request after it has been sent. This value is affected by the delay time and the lock mechanism. In this case, after sending the first LNP message to the DC, it will receive a response message, but if the NMS is busy processing another request, the message will not be consumed. Note that this time comprises the time of the request enforcement.

Figure 3-14 presents the values for two and five simultaneous requests. The values highlight the fact that requests are processed sequentially. Although this could be overcome by optimizing the implementation (e.g. using thread processes), it is considered a safe measure to perform the actions sequentially to avoid concurrent actions on the networking equipment.

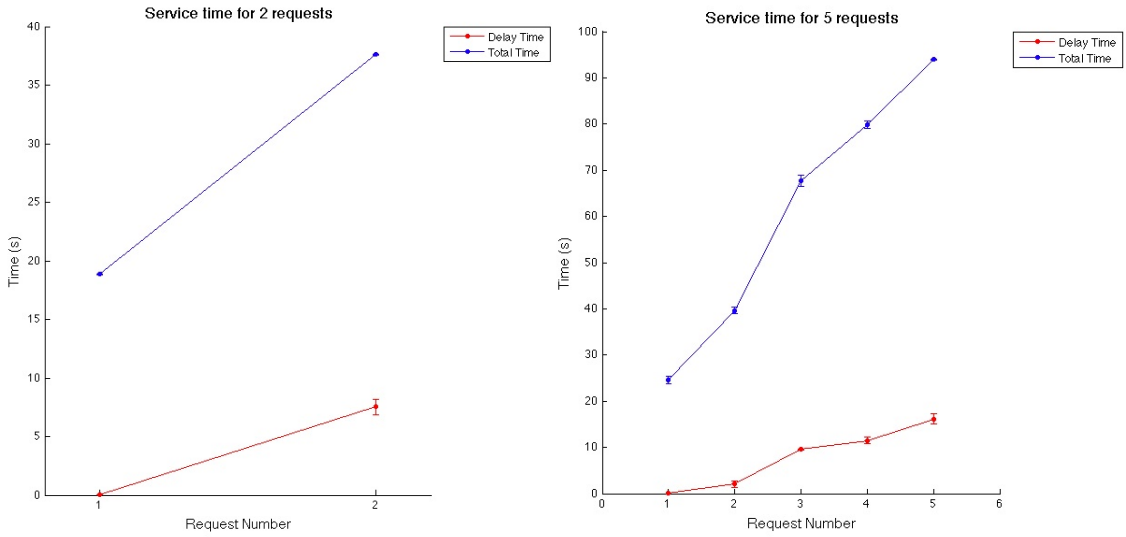


Figure 3-14: Delay and Total service time for 2 and 5 simultaneous requests

Finally, in Table 3-10 one can find the overall time for establishing a service across two DCs. For this evaluation, a request of one VM for each DC (approximately 378.4 seconds) plus a connectivity service across the NO was performed (approximately 81 seconds). The time the NOMS takes to process the request is the sum of the values presented in Table 3-9 (approximately 18.779 seconds). The connectivity time comprises the time after the configurations were enforced in the network devices until the connectivity is

actually up (approximately 62.23 seconds, which comprises internal propagation time of VPN technologies, such as for route exchange).

Table 3-10: Legacy NO Management System Analysis - Service Establishment

Phase	Time (s)	Confidence Interval (90%)
NOMS	18.779	$0.068x10^{-3}$
Connectivity	62.23	0.378
Overall Connectivity - Total	81.009	-
DC Management System	378.4	1.734

In summary, the values obtained are in the order of seconds and minutes. These are considered good values as they are in line with the time today experienced in public clouds, and are acceptable in most cases.

3.5.3.3. Network Operator Domain - SDN Network Approach

In the NO domain following a SDN network approach, we first present in Table 3-11 the time values for service creation and removal. The times are low and inline with the ones presented in the legacy network approach for the request process. The values of the LNP process are the same since the software module itself is the same. The biggest difference in the values here presented is due to the configuration enforcement time in the network devices. This process in this case is much faster as it relies on the OpenDaylight internal mechanisms and not on the OSS component used in the legacy approach.

Table 3-11: SDN Network Management System Analysis – Service Creation and Removal

Phase	Time (s)	Confidence Interval (95%)
Create	0.0281	0.0023
Remove	0.0276	0.0013
Link Negotiation Protocol	$15.767x10^{-3}$	$0.036x10^{-3}$

Furthermore, in Table 3-12 it is presented the average time for a connection drop response. In this case, a connectivity service is in place and a link over which this service is provisioned is turned down. Then, the SDN application detects the change in the topology and triggers a reconfiguration process. The average time for detection of the connection drop is low, as well as for the time to calculate an alternative route and enforce it in the network devices.

Table 3-12: SDN Network Management System Analysis – Connection Drop Response

Phase	Time (s)	Confidence Interval (95%)
Topology Change Detection	0.0809	0.0540
Reroute Total	0.1544	0.0765

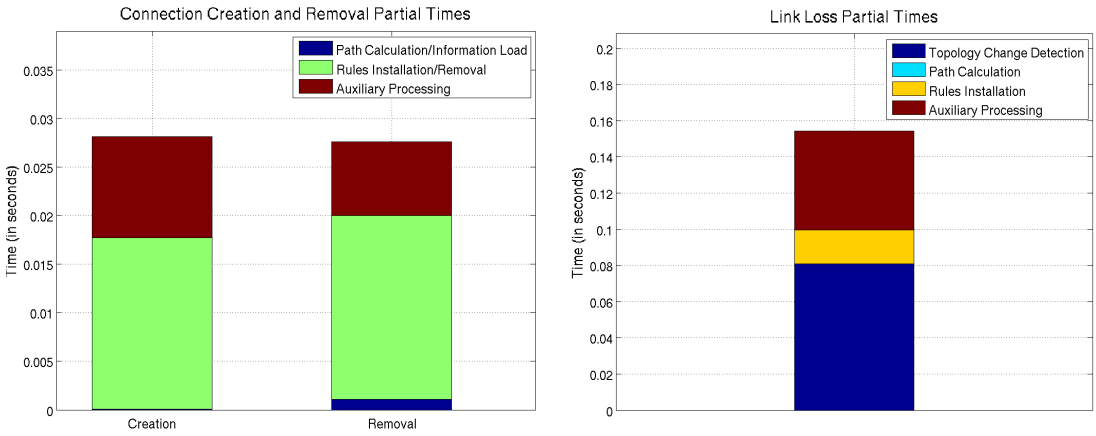


Figure 3-15: Connectivity Service Partial times for Creation and Removal Processes

Figure 3-15 shows the total and partial times for the process of creation, removal and link loss. The partial times for each case consider:

- Total time to create a connectivity service: partial time to calculate a service path route; partial time to enforce rules on network devices (switches); partial time in auxiliary processes – API delay, internal communication and information parsing
- Total time to revoke a connectivity service: partial time to load the connection stored information; partial time to remove rules on network devices (switches); partial time in auxiliary processes - API delay, internal communication and information parsing
- Total time to react to a link failure: partial time to detect a topology change (difference between the total time and the topology change time); partial time to calculate a new path route; partial time spent on updating rules on network devices; partial time in auxiliary processes - internal communication and information parsing

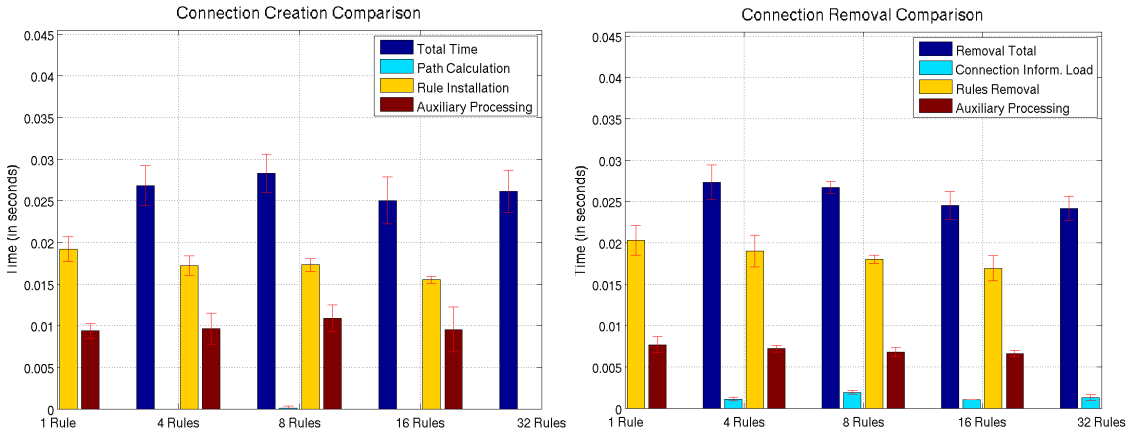


Figure 3-16: Network Service Enforcement Test Stress – Creation and Removal Processes

Finally, Figure 3-16 shows the stress analysis done in the enforcement of connectivity services. The test aims at analysing the impact of increasing the number of the network nodes affected by a connectivity service. As one can see, the number of network elements involved (up to 32) does not seem to affect the time for processing the requests.

3.5.3.4. Conclusion

The evaluation results are encouraging as they are within the expected time frame for the establishment of cloud services (order of minutes). Furthermore, we noticed that some of the legacy OSSs can benefit from adaptation to improve the time performance in accomplishing some processes. Moreover, legacy network technologies like VPNs bring inherent additional time in the connectivity establishment, something that we were able to avoid in the SDN approach.

3.6. Summary

This chapter presented a new architecture for the integration of cloud computing and NO domains. Special focus was given to the integrated resource management, the coordination across domains, and the dynamic establishment of WAN connectivity services. From a resource management perspective, an overall management system was presented. With respect to the coordination across domains, a protocol for the on-demand negotiation of connectivity services across domains was presented. In the scope of WAN connectivity services a service abstraction was presented, and details were provided on how it can be accomplished through legacy network approaches or SDN ones. The proof of concept showed that SDN approaches can significantly improve the establishment of the latter services.

Chapter 4 will focus on resource management aspects, specifically on resource allocation and resource adaptation and re-optimization modules.

4. Virtual Infrastructure Resource Management

In this chapter we focus on the research performed in the scope of resource management and specifically address the allocation and adaptation of virtual resources, i.e. Virtual Infrastructures (VIs). With the ultimate goal of optimally embedding VIs, we present a set of on-line VI Embedding (VIE) mechanisms. The proposed mechanisms have the objective of maximizing the number of accepted requests through minimization of the embedding costs by: minimizing the number of resources used to fulfil requests; and minimizing the physical infrastructure energy consumption. A sub-set of the proposed mechanisms allow the adaptation of already embedded VIs. In these latter cases we study how we can benefit from adaptation processes while at the same time minimizing the associated costs. The VIE problem is addressed in two distinct perspectives: single domain and multi-domain (section 2.6.1.6).

First, we address the single domain case, where we start by describing the problem and highlighting the challenges. Then, different approaches and strategies to solve the VIE problem are presented. Based on the work published in Annex Paper C and Annex Paper D, it is presented a heuristic approach and an optimal approach that try to balance the load in the physical infrastructure. Moreover, sustained in the work presented in Annex Paper D, strategies to consider the physical infrastructure energy consumption are introduced. Based on the work of Annex Paper E, a set of strategies that consider adaptation/re-configuration of already established VIs are presented. Finally, in the scope of what was published in Annex Paper F, we describe the multi-domain scenario case and provide an optimal solution for the VI embedding problem in this scenario.

4.1. Single Domain Approach

4.1.1. Problem Description

This scenario considers a single administrative domain (e.g., a Data Center (DC) of a Cloud Provider (CP)), where two types of physical nodes are considered, network nodes and server nodes, each with its specific set of associated characteristics. Network nodes are characterized by the number of CPU, denoted by *Cs*, the clock CPU frequency, *F*, and by the amount of memory, *M*. Two types of network nodes are considered: those which have virtualization capabilities and can therefore host Virtual Network (VN) entities; and those that do not have virtualization capabilities, considered as network transport elements only. Server nodes are characterized by the same parameters as network nodes (*Cs*, *F*, and *M*) plus storage capabilities, denoted by *STG*. With respect to links, these are characterized by bandwidth capacity (*B*) and

assumed to be bidirectional and with a maximum delay (*D*). Moreover, associated to the number of CPUs of a node (*Cs*), we consider another parameter that reflects the CPU load (or capacity) denoted by *C*.

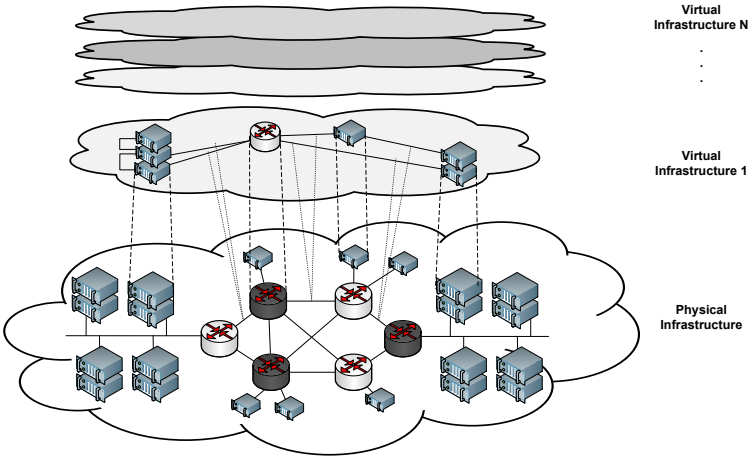


Figure 4-1: Single Domain Physical Infrastructure view

Table 4-1: VI Assignement Problem Notation – Single Domain

Symbol	Description
G^P	Physical Infrastructure
N^P	Set of Physical Nodes
S^P	Set of Server Nodes
R^P	Set of virtual-enabled Network Nodes
Rt^P	Set of transport Network Nodes
L^P	Set of Physical Links
i, j	Physical Nodes
ij	Physical Link
$Cs^P{}_i$	Number of CPUs of Physical Node *i*
$C^P{}_i$	Total CPU of Physical Node *i*
$M^P{}_i$	Total Memory of Physical Node *i*
$STG^P{}_i$	Total Storage of Physical Node *i*
$F^P{}_i$	CPU Frequency of Physical Node *i*
$B^P{}_{ij}$	Total Bandwidth of Physical Link *ij*
$D^P{}_{ij}$	Delay of Physical Link *ij*
$S^{max}{}_{load}$	Load of the physical server node with maximum load
$R^{max}{}_{load}$	Load of the physical network node with maximum load
$L^{max}{}_{load}$	Load of the physical link with maximum load
B_{cons}	Bandwidth being consumed in the physical links
$C^{Ptotal}{}_i$	Total CPU of Physical Node *i*
$M^{Ptotal}{}_i$	Total Memory of Physical Node *i*
$STG^{Ptotal}{}_i$	Total Storage of Physical Node *i*
$C^{Pfree}{}_i$	Free CPU of Physical Node *i*
$M^{Pfree}{}_i$	Free Memory of Physical Node *i*
$STG^{Pfree}{}_i$	Free Storage of Physical Node *i*
$G^V{}_v$	VI request *v*
$N^V{}_v$	Set of Virtual Nodes of VI request *v*
$S^V{}_v$	Set of virtual Server Nodes of VI request *v*
$R^V{}_v$	Set of VN Nodes of VI request *v*
$L^V{}_v$	Set of Virtual Links of VI request *v*
m, n	Virtual Nodes
mn	Virtual Links
Cs	Number of CPUs of Virtual Node *m* of VI request *v*
$C^V{}_{v,m}$	CPU of Virtual Node *m* of VI request *v*
$M^V{}_{v,m}$	Memory of Virtual Node *m* of VI request *v*
$STG^V{}_{v,m}$	Storage of Virtual Node *m* of VI request *v*
$F^V{}_{v,m}$	CPU Frequency of Virtual Node *m* of VI request *v*
$B^V{}_{v,mn}$	Bandwidth of Virtual Link *mn* of VI request *v*
$D^V{}_{v,mn}$	Maximum Delay of Virtual Link *mn* of VI request *v*

The physical infrastructure can host multiple VIs, as it is illustrated in Figure 4-1. These VIs are described in the same way as physical infrastructures. Note that we consider that VI nodes of a certain type (server or network) can only be accommodated in the physical infrastructure by nodes of the same type.

Table 4-1 summarizes the notation used. The reference to physical resources uses letter *P*, e.g., N^P, and virtual resources use letter *V*, e.g., $N^V{}_v$, where *v* refers to a specific VI. With respect to the connectivity of an infrastructure, an adjacent matrix is used: $A^P = N^P \rightarrow N^P$ when referring to a physical infrastructure. The convention used for the index notation is the following: *i*, *j* for nodes and links in the physical network, and *m*, *n* for nodes and links in the VI.

We aim to define mechanisms for the best VIE possible (i.e. maximizing acceptance and minimizing costs) according to the available physical resources and VI requirements. In order to map these resources, a combined algorithm, able to perform balanced decisions taking into account the abovementioned requirements of both network and cloud resources, is needed. This algorithm must be able to determine a possible and best solution, i.e., physical hosts able to allocate the virtual resources according to the specified requirements. Special attention is given to the energy consumption of the physical infrastructure as a high source cost in the overall embedding process. In this sense we intend to study the minimization of the energy consumption without jeopardizing the acceptance ratio. Furthermore, we aim to explore the ability to reconfigure and adapt already embedded VIs to improve the refered embedding factors, i.e. maximizing acceptance ratio, and minimizing energy consumption. In this sense we have defined a heuristic algorithm (section 4.1.2) and a set of optimal strategies (section 4.1.3) to accomplish the VIE process.

4.1.2. Heuristic Approach

In this section we present a heuristic approach (section 2.6.1.5) to solve the VIE problem based on the work published in Annex Paper D. A high level description of the algorithm is provided. Further, two approaches to the virtual link assignment process are considered. A comparative analysis of the two approaches is performed in section 4.1.4.

4.1.2.1. The Algorithm

Herein we detail the different parts of the algorithm. The algorithm is inspired by the concepts of node and link stress, i.e. links and nodes with less stress are more prone to accepting new virtual resources. Its management objective is to minimize the stress of the resources and to balance the stress among the resources. The algorithm uses these concepts to include the joint mapping of network and cloud resources.

Candidates' Selection – The algorithm starts to compare the hardware features of all virtual nodes $m \in N^V$ with all physical nodes $i \in N^P$. In order for a physical node to be a candidate, the characteristics of node *i* have to be higher or equal to the characteristics requested by virtual node *m* ($C_i^P \geq C^V_m$, $F_i^P \geq F^V_m$, $M_i^P \geq M^V_m$, $STG_i^P \geq STG^V_m$). Additionally, $\forall i \in Candidates(\forall m)$ need to have a physical connection $ij \in A^P$ with at least one candidate of the virtual node neighbour $mn \in A^V$. In order to strengthen the quality of the candidates, we introduce an individual selection of $\forall i \in Candidates(m)$ from all the virtual nodes $m \in N^V$ to find the candidates that will certainly lead to a mapping failure.

Resource Stress Calculation – This part includes the stress calculation of the infrastructure elements that will be used in the selection process. Stress is an indicator of how likely a resource should host a virtual resource in comparison with other physical resources (resources with less stress are more prone to accepting new virtual resources). Stress indicators are used on the mapping algorithm to calculate the potential of a certain candidate node to host a virtual resource. The link stress, $S_{LS_{(ij)}}$ (equation (2)), is calculated for $\forall ij \in A^P$. Special attention is taken to the amount of physical links used to map the VI, since one might save bandwidth by using shorter paths. In this sense, a physical link stress is calculated as a sum of the bandwidth occupied by all virtual links passing by that physical link.

Since the substrate has two types of nodes, each stress depends on different characteristics according to the type of the physical nodes. If node $i \in S^P$, the STG^P will be considered in S_{Ni} (equation (3)). If node $i \notin S^P$, STG^P will not be considered in S_{Ni}. Moreover, M_{medReq} represents the average memory of the virtual nodes,

C_{medReq} represents the average CPU load increase for each virtual node embedded, and STG_{medReq} represents the average storage of the virtual nodes; *k* represents a constant value.
After the stress calculation, we introduce a routine to assure that S_{Ni} is never zero. In the cases $S_{Ni}=0$, the algorithm assigns the *MinStressValue*($S_{Ni}!=0$) to $\forall i \in N^p$ that already hosts $\forall m$, in order to not compromise the node potential calculation in equation (4) ($\pi(i)$ - the potential of a candidate is not calculated only using server stress, but also considering the stress of the physical links which might be used to host virtual links). If at this moment there are virtual nodes with only one candidate, the algorithm will simulate an update to the substrate occupied resources of that candidate, in order to simulate its future occupation, and check if that candidate is still a valid candidate for the rest of the virtual nodes that have this candidate as an option.

$$S_{LS_{(ij)}} = \sum_{v \in V, m,n \in N^V(m), m<n} \left(S_{LV}(mn) \times y_{ij}^{v,mn} \right) \quad (2)$$

$$S_{N_i} = \left[N.ActiveVMs.(1 + \frac{k.C^V{}_{med\,\mathrm{Req}}}{C_i^{P_{free}} + \delta}).(1 + \frac{k.M^V{}_{med\,\mathrm{Req}}}{M_i^{P_{free}} + \delta}) \quad (1 + \frac{k.STG^V{}_{med\,\mathrm{Req}}}{STG_i^{P_{free}} + \delta}) \right] \quad (3)$$

Selection Process – The selection of $i \in Candidates(\forall m)$ is performed in this part. At the beginning of the selection process, a verification process of the candidates due to the future occupation status is introduced. After this analysis, it starts the selection process (which is sequential), i.e. a node $m \in N^V$ is selected in order to calculate $\pi(i)$ (equation (4)) for $\forall i \in Candidates(m)$. S_{N_i} is multiplied by the path stress average of $\forall j \in Candidates(n)$, where $mn \in A^V$. When calculating the potential of the candidate nodes, link cost will be the most important parameter until the considered nodes achieve a critical occupation state (that can be adjusted through the constant *k*). The formula for the path stress (*PathStress*) is described in section 4.1.2.2. The candidate $i \in Candidates(m)$ that presents the lowest value of π is chosen to host *m*.

$$\pi(i) = S_{N_i} \frac{\sum_j PathStress(i,j)}{totalCand} \quad (4)$$

Substrate Links mapping – The last part of the algorithm is executed if every virtual node $m \in N^V$ has a candidate. This process comprises the mapping of all virtual links (further details are provided in section 4.1.2.2).

4.1.2.2. Path Find and Path Stress approaches

In the decision process, the candidate that presents the highest potential is chosen to host the virtual node. The candidate potential is inversely proportional to the multiplication of the candidate stress with the average of the path stress necessary to link the candidate with all candidates of the neighbour's virtual nodes – equation (4). Note that the path stress calculation does not influence the path finding process, but it is dependent on the latter. The process of calculating the path stress is preceded by the finding of a proper path. For the latter case, two approaches are studied:

- *Constrained Shortest Path First (CSPF) Dijkstra algorithm* [204] - This approach tries to reduce the physical link's load, however neglecting to a certain extent the amount of bandwidth consumed by a virtual link.
- *Breadth-First Search (BFS) algorithm* [205] - The BFS looks for the path between source and destination that requires less hops. Originally, the algorithm stops as soon as it reaches the first solution (which guarantees the minimum number of hops), or when it does not find a solution. In our case, we let the algorithm find all solutions that guarantee the minimum number of

hops, and the tiebreak is done using the Dijkstra algorithm. In this sense, it guarantees the best path among those with minimum hops.

After finding a suitable path, the path stress is calculated. In the CSPF approach (that uses only Dijkstra), the path stress is retrieved directly from the path find algorithm. However, having in mind that the bandwidth of the physical links was already identified in the past as a main bottleneck to the embedding of VIs [25], the approach now imposes weights to the hops in addition with the substrate occupation by a factor *K* of the virtual link. The weight added by hop follows an exponential behaviour. Equation (5) reflects the stress calculation of mapping a virtual link in a certain physical path. *PathStress* represents the stress, *H* represents the number of hops, $S_{LS}(ij)$ represents the link stress (reserved substrate bandwidth on the substrate link) between the node *i* and the node *j*, *P* represents the sorted list of nodes traversed by the link, *K* represents a constant value, and B^{mn} is the virtual link bandwidth.

$$PathStress = \sum_{h=1}^{H} S_{LS}(P_h, P_{h+1}) + \sum_{h=1}^{H} \left[K.(h-1).B^{\mathrm{mn}} + B^{\mathrm{mn}} \right] \quad (5)$$

In summary, this approach does not allow to choose candidates that will consume a high number of links, even if the physical substrate in that zone has a lower stress value. In this sense fewer link resources from the infrastructure will be consumed. This approach is also used in the final link mapping.

4.1.3. Optimal Approach

In this section we present a set of optimal approaches (section 2.6.1.5) to solve the VIE problem based on the works published in Annex Paper D and Annex Paper E. The presented approaches study the maximization of the embedded VIs as well as the minimization of the physical infrastructure energy consumption. To improve the performance of the approaches, adaptation of already embedded VIs is also considered. A comparative analysis of the different approaches is performed in section 4.1.4.

4.1.3.1. Mathematical Formulations

In this section we present the mathematical formulation to solve the VI embedding problem based on Integer Linear Programming (ILP) formulation. First, we present the assignment variables as well as the constraints common to all approaches. Then, we present four specific formulations in separate sub-sections to: calculate the maximum resource loads which provides the load consumption of each type of resource (server node, network node and link) that is more loaded among all other resources; calculate the bandwidth consumption that VIs consume; set node and link state which keeps track of the state of all resources, i.e. if they are active or not; allow reconfiguration processes and to calculate the amount of reconfigurations that occur.

4.1.3.1.1 Assignment Variables

The formulation considers two binary assignment variables, *x* and *y*, one for virtual nodes and another for virtual links, as equations (6) and (7) show.

$$x_i^{v,m} = \begin{cases} 1, \text{virtual node } m \text{ of VI } v \text{ is allocated at physical node } i \\ 0, \text{otherwise} \end{cases} \quad (6)$$

$$y_{ij}^{v,mn} = \begin{cases} 1, \text{virtual link } mn \text{ of VI } v \text{ is uses physical link } ij \\ 0, \text{otherwise} \end{cases} \quad (7)$$

4.1.3.1.2 Generic Constraints

As mentioned earlier in the VI description, virtual servers are assigned to physical server nodes. Equation (8) reflects this constraint, which is coupled with the fact that a virtual node is assigned just to one

physical node. Moreover, equation (9) assures that VN nodes are assigned to network nodes with virtual capabilities.

$$\forall v,m \in S : \sum_{i \in S} x_i^{v,m} = 1 \quad (8)$$

$$\forall v,m \in R : \sum_{i \in R} x_i^{v,m} = 1 \quad (9)$$

Equations (10), (11) and (12) guarantee that the capacity values of physical nodes, i.e. CPU load, memory, and storage, are kept within range. Equations (13) and (14) make sure that the number of cores and CPU frequency requirements are respected, i.e., a physical node selected to host a virtual node has at least the same number of cores and frequency as the virtual node.

$$\forall i : \sum_{m,v} x_i^{v,m} \times C_{v,m}^{V} \leq C_i^{P_{total}} \quad (10)$$

$$\forall i : \sum_{m,v} x_i^{v,m} \times M_{v,m}^{V} \leq M_i^{P_{total}} \quad (11)$$

$$\forall i : \sum_{m,v} x_i^{v,m} \times STG_{v,m}^{V} \leq STG_i^{P_{total}} \quad (12)$$

$$\forall i,v,m : x_i^{v,m} \times Cs_{v,m}^{V} \leq Cs_i^{P} \quad (13)$$

$$\forall i,v,m : x_i^{v,m} \times F_{v,m}^{V} \leq F_i^{P_{total}} \quad (14)$$

Equation (15) applies the multi-commodity flow constraint [206] with a node-link formulation [207], in order to simultaneously optimize the mapping of virtual links and virtual nodes. Moreover, the notion of direct flows on the virtual links is also used.

$$\forall v,\forall m,n \in N_v^V(m), m < n, \forall i : \sum_{j \in N^P(i)} \left(y_{ij}^{v,mn} - y_{ji}^{v,mn} \right) = x_i^{v,m} - x_i^{v,n} \quad (15)$$

Equation (16) guarantees that each physical link selected has enough available bandwidth to host a virtual link: the sum of the amount of bandwidth of the virtual links that go through *i* must be equal or lower than the amount of free bandwidth in a physical link. Finally, equation (17) guarantees that the virtual link delay constraint is met, i.e. the maximum delay of the physical links assigned to a virtual link does not exceed the virtual link's maximum delay.

$$\forall i,j \in N^P(i), i < j : \sum_{v,m,n \in N^V(m), m < n} B_{v,mn}^{V} \times \left(y_{ij}^{v,mn} + y_{ji}^{v,mn} \right) \leq B_{ij}^{P_{free}} \quad (16)$$

$$\forall v,\forall m,n \in N_v^V(m), m < n : \sum_{\forall i,j \in N^P(i), i < j} D_{ij}^{P} \times \left(y_{ij}^{v,mn} + y_{ji}^{v,mn} \right) \leq D_{v,mn}^{V} \quad (17)$$

4.1.3.1.3 Maximum Node and Link Load Formulation

The maximum load consumption of server nodes, i.e. the load consumption of the server node that is more loaded among all server nodes ($S_{load}^{\max}$), is presented in equation (18). This value is given by the sum of CPU, memory and storage load fractions. Moreover, a fraction of the CPU frequency is taken into account in tiebreak situations (when there are two server nodes with the same load), in a normalized function. In this way, physical nodes with lower frequency are the first to be used, preserving the remaining for eventual future nodes with higher frequency demands. δ_1, δ_2, and δ_3 represent the resource component weights, allowing us to define which resource (CPU, memory or storage) is more relevant to the node overall load. ε is a very small value, so that the CPU frequency factor is only relevant in tiebreak situations. The sum of resource component weights is 1.

The maximum load consumption of network nodes ($R_{load}^{\max}$) is presented in equation (19) - similar to the maximum load consumption of server nodes. It only differs from the previous one by not taking into account storage (*STG*). μ_1 and μ_2 represent the resource components weights (CPU load and memory load), and ε is again a very small value so that the CPU frequency is only relevant in tiebreak situations. Again, the sum of resource component weights is 1.

$$\forall i \in S: \left[\begin{array}{l} \delta_1 \dfrac{\sum_{v,m} \left(x_i^{v,m} \times C_{m,i}^V \right)}{C_i^{P_{total}}} + \delta_2 \dfrac{\sum_{v,m} \left(x_i^{v,m} \times M_{m,i}^V \right)}{M_i^{P_{total}}} + \\ \delta_3 \dfrac{\sum_{v,m} \left(x_i^{v,m} \times STG_{m,i}^V \right)}{STG_i^{P_{total}}} + \varepsilon \dfrac{F_i^P}{F_{\max}^P} \end{array} \right] \leq S_{load}^{\max} \quad (18)$$

$$\forall i \in R: \left[\begin{array}{l} \mu_1 \dfrac{\sum_{v,m} \left(x_i^{v,m} \times C_{m,i}^V \right)}{C_i^{P_{total}}} + \\ \mu_2 \dfrac{\sum_{v,m} \left(x_i^{v,m} \times M_{m,i}^V \right)}{M_i^{P_{total}}} + \varepsilon \dfrac{F_i^P}{F_{\max}^P} \end{array} \right] \leq R_{load}^{\max} \quad (19)$$

Equation (20) defines the maximum load consumption of a physical link, among all links ($L_{load}^{\max}$). $B_{v,mn}^V$ refers to the bandwidth of virtual link *mn* (i.e. the virtual link connecting virtual node *m* and *n*) of a certain VI (*V*). $B_{ij}^{P_{Total}}$ refers to the total bandwidth capacity of physical link *ij* (i.e. the physical link between physical node *i* and *j*). The variable $y_{ij}^{v,mn}$ assumes the value 1 if virtual link *mn* is crossing physical link *ij*, and 0 otherwise. In this way, the formula sums the bandwidth of all virtual links ($B_{v,mn}^V$) that cross a certain physical link (*ij*) and divides that value by the total capacity of the physical link ($B_{ij}^{P_{Total}}$). In the end, $L_{load}^{\max}$ assumes the load value of the most loaded physical link.

$$\forall i, j \in N^P(i), i < j: \left[\frac{\sum_{v,m,n \in N^V(m), m<n} B_{v,mn}^V \times \left(y_{ij}^{v,mn} + y_{ji}^{v,mn} \right)}{B_{ij}^{P_{Total}}} \right] \leq L_{load}^{\max} \quad (20)$$

4.1.3.1.4 Bandwidth Consumption Formulation

Equation (21) defines the percentage of the total bandwidth consumed by a VI in the physical infrastructure. The amount of bandwidth consumed by each virtual link ($B_{v,mn}^V$) in all physical links ($y_{ij}^{v,mn}$) is summed and divided by the total bandwidth capacity of the entire physical infrastructure.

$$\frac{\sum_{v \in V, m, n \in N^V(m), m<n} y_{ij}^{v,mn} \times B_{v,mn}^V}{\sum_{i,j \in N^P(i), i<j} B_{ij}^{P_{total}}} = B_{cons} \quad (21)$$

4.1.3.1.5 Node and Link State Formulation

We present a formulation for the node and link state with two extra (binary) variables, x_i^{active} and y_{ij}^{active}. x_i^{active} denotes if physical node *i* is active ($x_i^{active} = 1$) or not ($x_i^{active} = 0$). A node is considered to be active if it hosts at least one virtual node or if a virtual link traverses this node. Similarly, y_{ij}^{active} denotes if physical link *ij* is active ($y_i^{active} = 1$) or not ($y_i^{active} = 0$). Equation (22) guarantees that, if a physical link is occupied by at least one virtual link, variable y_{ij}^{active} will be set to 1 (0, otherwise). Moreover, equations (23) and (24) guarantee

that, if a physical node has at least one active link variable, x_i^{active} takes value 1 and 0 otherwise. (Note: variable *K* is a high value constant used to assure that variable x_i^{active} does not assume a value higher than 1)

$$\forall i,j \in N^P(i), i<j: \sum_{v,m,n \in N^V(m), m<n} B_{v,mn}^V \times \left(y_{ij}^{v,mn} + y_{ji}^{v,mn}\right) \leq B_{ij}^P \times y_{ij}^{Active} \quad (22)$$

$$\forall i \in N^P: \sum_{i,j \in N^P} \left(y_{ij}^{Active} + y_{ji}^{Active}\right) \geq x_i^{Active} \quad (23)$$

$$\forall i \in N^P: \sum_{i,j \in N^P} \left(y_{ij}^{Active} + y_{ji}^{Active}\right) \leq x_i^{Active} \times K \quad (24)$$

4.1.3.1.6 Reconfiguration

The ability to completely reconfigure (re-arrange) currently mapped VIs when trying to map a new VI increases the likelihood of a successful mapping. In this sense, we keep record of the currently mapped VIs in the physical infrastructure in variables *X* and *Y*. $X_i^{v,m}$ denotes if virtual node *m* of VI *v* is mapped at the physical node *i* ($X_i^{v,m}=1$) or not ($X_i^{v,m}=0$). $Y_{ij}^{v,mn}$ denotes if virtual link *mn* of the VI *v* uses the physical link *ij* ($Y_{ij}^{v,mn}=1$) or not ($Y_{ij}^{v,mn}=0$).

Completely reconfiguring VIs can give an upper bound for the VI acceptance ratio/revenue; however, this is done at a high cost. The disruption caused is extremely high, especially when virtual nodes are migrated. When considered, equation (25) guarantees that VI nodes already embedded have to remain in the same physical hosts. Equation (26) guarantees that virtual links already embedded remain using the same physical path. In this sense, when equations (25) and (26) are not considered, they allow the ILP to reconfigure nodes and/or links respectively.

$$\forall v \in V^{old}, \forall m \in N^V, \forall i \in N^P: x_i^{v,m} = X_i^{v,m} \quad (25)$$

$$\forall v \in V^{old}, \forall m,n \in N^V, \forall i \in N^P, m<n: y_{ij}^{v,mn} + y_{ji}^{v,mn} = Y_{ij}^{v,mn} \quad (26)$$

Moreover, re-optimizations of VIs are registered. We keep track of node and link changes within each VI. $X_{v,m}^{change}$ registers if virtual node *m* in VI *v* is moved ($X_{v,m}^{change}=1$) or not ($X_{v,m}^{change}=0$). This constraint is reflected in equation (27). $Y_{v,mn}^{change}$ registers if virtual link *mn* in VI *v* was reconfigured ($Y_{v,mn}^{change}=1$) or not ($Y_{v,mn}^{change}$=0), which is guaranteed by equations (28), (29) and (30). Note that we assume that the number of changes in the physical links used by a virtual link is counted only once. Variables $Y_{v,mn}^{changeAUX}$ and *K* are auxiliary variables that help to keep track of virtual link changes. $Y_{v,mn}^{changeAUX}$, in equation (28), provides the number of physical links affected by changing (either by supporting or no longer supporting) the embedding of virtual link *mn*. Equation (29) guarantees that, if $Y_{v,mn}^{changeAUX}=0$, then $Y_{v,mn}^{change}=0$. In equation (30) variable *K* is a sufficient large number to ensure that $Y_{v,mn}^{change}=1$ if $Y_{v,mn}^{changeAUX} \geq 1$.

$$\forall v \in V^{old}, \forall m \in N_v^V: \sum_{i \in N^P} \left(X_i^{v,m} - x_i^{v,m}\right) \times X_i^{v,m} = X_{v,m}^{change} \quad (27)$$

$$\forall v \in V^{old}, \forall m,n \in N_v^V, m<n: \sum_{i,j \in N^P} \left(Y_{ij}^{v,mn} - (y_{ij}^{v,mn} + y_{ji}^{v,mn})\right) \times Y_{ij}^{v,mn} = Y_{v,mn}^{changeAUX} \quad (28)$$

$$\forall v \in V^{old}, \forall m,n \in N_v^V, m<n: Y_{v,mn}^{change} \leq Y_{v,mn}^{changeAUX} \quad (29)$$

$$\forall v \in V^{old}, \forall m,n \in N_v^V, m<n: \frac{Y_{v,mn}^{changeAUX}}{K} \leq Y_{v,mn}^{change} \quad (30)$$

Furthermore, we also define V_v^{Change} to count the number of affected VIs. Similarly to what was previously explained, equations (31), (32) and (33) reflect the associated constrains. V_v^{Change} assumes value 1 if VI *v* was subject of a re-optimization process, and 0 otherwise. Variables $V_v^{ChangeAUX}$ and *K* are auxiliary variables that help to keep track of VI changes. $V_v^{ChangeAUX}$, in equation (31), provides the number of virtual nodes and links in VI *v* affected by changing (either by no longer being embedded in a certain physical resource, or by embedding in a new one) the embedding of these resources. Equation (32) guarantees that if $V_v^{ChangeAUX} = 0$, then $V_v^{Change} = 0$. In equation (33) variable *K* is a sufficient large number to ensure that $V_v^{Change} = 1$ if $V_v^{ChangeAUX} \geq 1$.

$$\forall v \in V^{old} : \left(\sum_{m \in N_v^V} X_{v,m}^{change} \right) + \left(\sum_{m,n \in N_v^V : m<n} Y_{v,mn}^{change} \right) = V_v^{ChangeAUX} \quad (31)$$

$$\forall v \in V^{old} : V_v^{Change} \leq V_v^{ChangeAUX} \quad (32)$$

$$\forall v \in V^{old} : \frac{V_v^{ChangeAUX}}{K} \leq V_v^{Change} \quad (33)$$

Since moving nodes and links has a significantly different impact, we define two other counters for affected VIs, one based only on virtual node changes – equations (34), (35) and (36)– and another based only on virtual link changes – equations (37), (38) and (39). $V_v^{NodeChange}$ assumes value 1 if VI *v* was affected by a virtual node change, or 0 otherwise. The same is true for $V_v^{LinkChange}$ with respect to virtual link changes. Variables $V_v^{NodeChangeAUX}$, $V_v^{LinkChangeAUX}$, and *K* are used for the same purpose as in the cases before, i.e. to help keeping the values of $V_v^{NodeChange}$ and $V_v^{LinkChange}$ within their boundaries (0 or 1).

$$\forall v \in V^{old} : \sum_{m \in N_v^V} X_{v,m}^{change} = V_v^{NodeChangeAUX} \quad (34)$$

$$\forall v \in V^{old} : V_v^{NodeChange} \leq V_v^{NodeChangeAUX} \quad (35)$$

$$\forall v \in V^{old} : \frac{V_v^{NodeChangeAUX}}{K} \leq V_v^{NodeChange} \quad (36)$$

$$\forall v \in V^{old} : \sum_{m,n \in N_v^V : m<n} Y_{v,mn}^{change} = V_v^{LinkChangeAUX} \quad (37)$$

$$\forall v \in V^{old} : V_v^{LinkChange} \leq V_v^{LinkChangeAUX} \quad (38)$$

$$\forall v \in V^{old} : \frac{V_v^{LinkChangeAUX}}{K} \leq V_v^{LinkChange} \quad (39)$$

4.1.3.2. Objective Functions

In this section, based on the formulations presented, different embedding optimal strategies are presented.

4.1.3.2.1 Load Balancing (Node and Link) strategy – VIE-Opt-LB

Based on a strategy presented in [156], this strategy aims to minimize the maximum load of node and link resources. Equation (40) represents the strategy. Weights w_1 and w_2 rule the importance of the several loads, and ε is a very small value so that the total bandwidth consumption (B_{cons}) is only considered in tiebreak situations. Note that the maximum load of server ($S_{load}^{\max}$) and network ($R_{load}^{\max}$) nodes is under the effect of the same weight, w_1, and the maximum link load ($L_{load}^{\max}$) is under the effect of w_2. This evens the importance given to the maximum load of nodes and links.

$$min\ \omega_1\left(S_{load}^{\max} + R_{load}^{\max}\right) + \omega_2 L_{load}^{\max} + \varepsilon B_{cons} \quad (40)$$

This strategy will be used as the reference one, since in [156] it is shown that it outperforms several state-of-the-art embedding algorithms - Deterministic Node Mapping with Slipttable Link Mapping using Multi-Commodity Flow Constraint (D-Vine) [150], Randomized Node Mapping with Splittable Link Mapping using Multi-Commodity Flow Constraint (R-Vine) [150], etc).

In contrast to these approaches, in our approach we consider two types of nodes, more parameters in the load of a node, and we do not restrict a physical node to host only one virtual node per VI. Moreover, we will be able to improve the VI embedding performance when using load balancing formulation. This improvement is achieved with the next strategy.

4.1.3.2.2 VIE-Opt-LB with Shortest Path – VIE-Opt-LB+SP

This strategy, represented by equation (41), differs from the previous one, as it does not take into account the total bandwidth consumption of a VI merely as a tiebreaker. In this case, B_{cons} is under the effect of weight w_2. We argue that the bandwidth consumption shall be considered as an active aspect to be taken into account while mapping VIs.

$$min\ \omega_1\left(S_{load}^{\max} + R_{load}^{\max}\right) + \omega_2\left(L_{load}^{\max} + B_{cons}\right) \quad (41)$$

4.1.3.2.3 Green strategy - VIE-Opt –Green

This strategy minimizes the energy consumption of the physical infrastructure. The energy consumption of a device is deeply related to its load. However, the relation between these two parameters is intimately dependent on the specific device. Moreover, in most devices the energy consumption in idle mode (i.e. powered on but with minimal load) is considerably high compared to the maximum energy consumption at full power (i.e. with maximum load). In this sense, we consider that it is better to have one node at full power than two nodes in idle mode. Moreover, we assume that reducing the number of active links in an active node will reduce the node's overall energy consumption.

With these assumptions in mind, we draw the objective to map a VI in such a way that it has a minimum number of active physical resources. In equation (42) the main objective is to minimize the number of active nodes, since they are the main source of energy consumption. The second objective is to minimize the number of active links in order to reduce the overall energy consumption of the active nodes. In this case w_1 is higher than w_2, and ε is a very small value, so that another objective can be considered as a tiebreak objective, presented in the equation as *Strat*.

The objective considered in *Strat* in the evaluation section is the load balancing with shortest path (*VIE-Opt-LB+SP*). The load balancing of the active elements of the infrastructure will ease the future embedding of VIs without having to increase the number of active elements.

$$min\ \omega_1\left(\sum_{i \in N^p} x_i^{Active}\right) + \omega_2\left(\sum_{i,j \in N^p, i<j} y_{ij}^{Active}\right) + \varepsilon(Strat) \quad (42)$$

4.1.3.2.4 Dynamic Green + Load Balancing - VIE-Opt-D-Green-LB

Here we study the combination of different objectives in a dynamic way. In the previous cases, different objectives were considered in a strategy. However, there was always a fixed hierarchy of objectives. In this case, we do not keep that hierarchy fixed, and change it depending on the infrastructure state.

This strategy uses the green strategy (*VIE-Opt –Green*) and the load balancing with shortest path strategy (*VIE-Opt-LB+SP*), as presented in equation (43). Here, weights (z_1 and z_2) are not fixed along time, but change depending on the current state of the infrastructure at the time of each VI arrival. More precisely, we consider that, if the infrastructure is loaded over a certain limit, the load balancing strategy is applied. If the load is below or equal to the limit, the green strategy is applied.

$$min\ z_1(Strat\_1) + z_2(Strat\_2) \tag{43}$$

4.1.3.2.5 Re-Optimization – VIE-Re-Opt

The strategy defined in this category allows the reconfiguration of already deployed VIs. We present two sub-types of strategies: those that allow only link re-optimization, and those that allow both link and node re-optimizations.

Node reconfigurations are considered to be the most expensive ones, while link reconfigurations can be easily achieved and, to a certain extent, they can be neglected when considering the use of technologies like Software Defined Networking (SDN). With this assumption in mind, we do not penalize link reconfigurations in the refered strategies.

The strategies that allow only the re-optimization of links keep the original objective functions. However, those that allow also node re-optimization suffer a change, presented in equation (44). Since node reconfigurations are considered very expensive, the objective functions first try to minimize the number of virtual node reconfigurations. ε is a very small value so that *Strat* is considered as the last objective.

$$min\left(\sum_{v\in V}\sum_{m\in N_v^V} X_{v,m}^{change}\right) + \varepsilon(Strat) \tag{44}$$

Taking advantage of the strategies presented in the previous subsection, the following strategies are defined:

- Load Balancing (Node and Link) + Shortest Path strategy with Link Re-optimization - VIE-ReOpt-LB+SP
- Load Balancing (Node and Link) + Shortest Path strategy with Node and Link Re-optimization - VIE-LinkReOpt-LB+SP
- Green strategy with Link Re-optimization - VIE-LinkReOpt-LB-R
- Green strategy with Node and Link Re-optimization- VIE-ReOpt –Green
- Green + Load Balancing with Link Re-optimization - VIE-LinkReOpt-D-Green-LB
- Green + Load Balancing with Node and Link Re-optimization - VIE-ReOpt-D-Green-LB

Depending on the strategy used, *Strat* will assume different forms. For example, in the strategy VIE-ReOpt-LB+SP, *Strat* will assume the form of equation (41), which leads to the objective function presented in equation (45). In this case, since both virtual node and link reconfigurations are allowed, equations (25) and (26) are not considered. In the case of strategy VIE-LinkReOpt-LB+SP, equations (25) is considered to prevent virtual node reconfigurations.

$$min\left(\sum_{v\in V}\sum_{m\in N_v^V} X_{v,m}^{change}\right) + \varepsilon\left(\omega_1\left(S_{load}^{\max} + R_{load}^{\max}\right) + \omega_2\left(L_{load}^{\max} + B_{cons}\right)\right) \tag{45}$$

4.1.4. Evaluation Results

This section evaluates the proposed approaches. Here, we provide a thorough evaluation of the different optimal embedding strategies presented in section 4.1.3, along with an evaluation of the proposed heuristic presented in section 4.1.2. Note that, in this section, the term *Heuristic* refers to the approach in which the CSPF Dijkstra algorithm is used in the path finding process, while the term *E-Heuristic* refers to the BFS case. Furthermore, the results of the heuristic approaches are only compared with the optimal strategies that do not consider re-optimization processes. This is done to avoid the graphs that consider optimal strategies allowing re-optimization processes and those not allowing becoming illegible, and because presenting the heuristic results there will not bring added value to the analysis. The overall evaluation is presented in Annex Paper C, Annex Paper D, and Annex Paper E.

4.1.4.1. Evaluation scenario description

To evaluate the performance of the proposed approaches, a Matlab [208] simulator is used. For each run, the program designs a random physical infrastructure of 20 nodes based on the *Waxman* network topology generator [209], and it simulates a set of requests of VIs (with a number of nodes between 4 and 14) with Markov-modulated inter-arrival and inter-departure times. Both physical infrastructure and the generated VIs have 70% of the nodes as servers (rounded to the higher integer), and the remaining 30% as network nodes. Details on the nodes and link parameters can be seen in Table 4-2. Moreover, the VI request rates (λ) are between 2 and 5 VIs per time unit (Poisson arrivals), and the average duration ($1/\mu$, where μ is the average service rate) is 20 time units (exponentially distributed).

Table 4-2: Physical and virtual infrastructure parameters

		Physical Infrastructure	Virtual Infrastructures
Network Nodes	**Cs**	{2; 4; 6; 8}	{1; 2; 3; 4 }
	F(Hz)	{2.0-3.2 / 0.2 steps}	{2.0-3.2 / 0.1 steps}
	Memory	{2; 4; 6}(GB)	{64; 128; 256; 512}(MB)
Server Nodes	**Cs**	{8; 16; 32; 64}	{1; 2; 4; 8; 16; 32; 64}
	F(Hz)	{2.8-3.2 / 0.2 steps}	{2.8-3.2 / 0.1 steps}
	STG (GB)	{6400; 12800; 25600}	{100; 200; 400; 800; 1600}
	M (GB)	{256; 512; 1024}	{2; 4; 8; 16; 32; 64}
Links	**B (Mbps)**	{500-2000 / 500 steps}	{10-100 / 10 steps}
	D (ms)	{5-10 / 1 steps }	{0-40 / 5 steps}

In order to solve the ILP, we have used CPLEX [210] version 12.3, integrating a plug-in for our Matlab simulator and setting a time limit of 600 seconds (i.e. 10 minutes) for each VI mapping. The weight values used for the optimal strategies are the following:

- w_1 and w_2 are 0.5 (50%);
- ε is 0.001;
- z_1 and z_2 have dynamic values which change depending on the state of the physical infrastructure. If the maximum load of any resource (server node, network node or link) is below 75%, *Opt-Green* is applied; otherwise, *Opt-LB+SP* is applied. The 75% value was chosen empirically due to its good results in the running experiments.

We analyze the VI acceptance ratio as one of the main indicators of the algorithms' performance. Moreover, the energy consumption of the physical infrastructure is also analyzed, based on the analysis of the active infrastructure resources along time. All values in the following graphics present a mean of 10 runs (with different substrate) with a confidence interval of 95%.

4.1.4.2. Virtual Infrastructure Acceptance Ratio

Figure 4-2 presents the VI acceptance ratio of the strategies that do not allow re-optimization. It is important to note that, given the amount of constraints involved in the embedding problem (delay, bandwidth, processing, memory, storage), many VIs are rejected due to the impossibility of the

infrastructure to simultaneously meet all requirements. Naturally, the values in all strategies present a decreasing linear behaviour as the number of VIs per time unit increases. As the VI arrival rate increases, the substrate reaches a saturation point much faster, which leads to a decreasing behaviour in the acceptance.

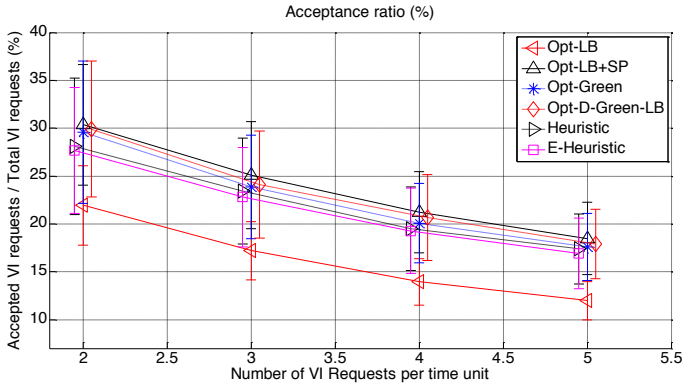


Figure 4-2: VI acceptance ratio

The difference between the VIE-Opt-LB (values between 12% and 22%) and the remaining strategies is clear, approximately above 10%. This shows that actively considering the minimization of the bandwidth consumed by VIs improves the acceptance ratio. The remaining strategies present very close values. The VIE-Opt-LB+SP is the strategy that overall presents better results – between 31% with 2 VIs per time unit, and 18% with 5 VIs per time unit. The VIE-Opt-Green presents lower values. This is an expected behaviour: minimizing the number of active resources will lead to a faster saturation of these resources, making them inadequate for future VIs. Nevertheless, the decrease is low, approximately 3%. In-between these latter two strategies is the Opt-D-Green-LB. This is also an expected result, since the interplay between the green strategy and the load balancing naturally results in a position between these two. With respect to the heuristic approaches, they present results very close, but it is noticeable the slight better performance of the Heuristic over the E-Heuristic. This happens because the E-Heuristic has less freedom to find disperse solutions, leading to higher probability of reaching saturation of certain physical infrastructure points and consequent less embedding solutions.

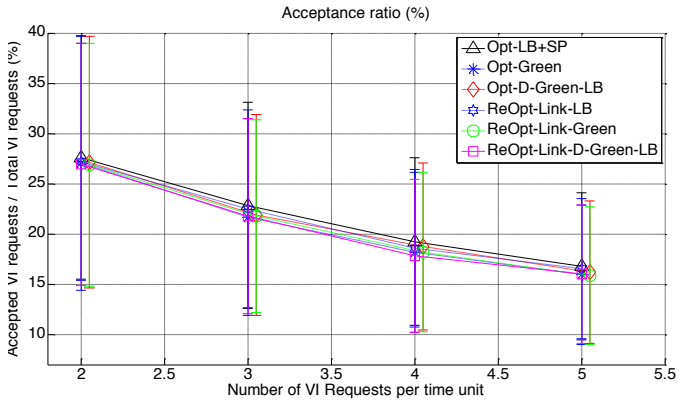


Figure 4-3: VI acceptance ratio – with link re-optimization

Figure 4-3 compares the results of the optimal strategies with and without link re-optimization (VIE-Opt-LB is not considered due to its poor performance compared to the other strategies). It is clear that, in this case, the strategies allowing only link reconfiguration do not present significant differences as compared to the ones without reconfiguration. This happens because the strategies allowing only link re-optimization are limited in terms of virtual link mapping alternative solutions that make a reconfiguration viable. This lack of alternatives is mostly due to the fact that virtual nodes cannot be moved.

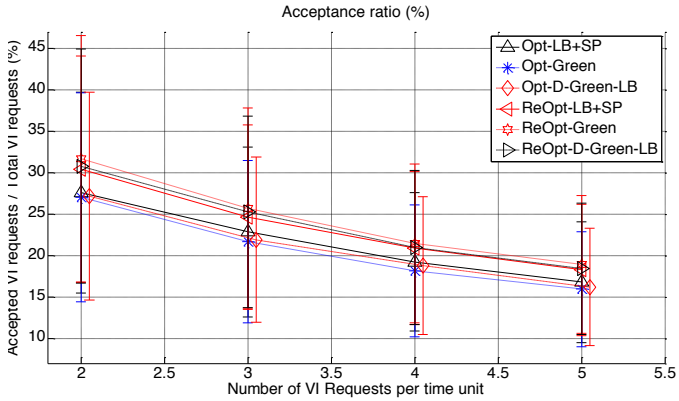


Figure 4-4: VI acceptance ratio – with node and link re-optimization

On the other hand, the strategies allowing both node and link re-optimization outperform all others – see Figure 4-4. The values are, in average, up to 5% higher. The ReOpt-Green strategy is the one presenting the best performance (with values between 32% for a VI arrival rate of 2 and 19% for an arrival rate of 5), followed by the ReOpt-D-Green-LB and the ReOpt-Lb+SP. The difference between the three strategies is low (a gap between strategies of approximately 1%). We notice that, in theory, all three strategies should present the exact same values in terms of acceptance ratio as they all allow full reconfiguration of resources. This does not happen due to the time constraint set to solve the ILP problem (600 seconds). In other words, each strategy takes a different time to find a VI embedding solution, and in some cases no solution is found within the defined timeframe.

4.1.4.3. Delay and Physical Infrastructure size

In this sub-section we briefly analyze the impact of adding a new constraint to the problem formulation. More specifically, it is analyzed the delay constraint impact in the VI acceptance results. Furthermore, the impact of the size of the physical infrastructure is analyzed.

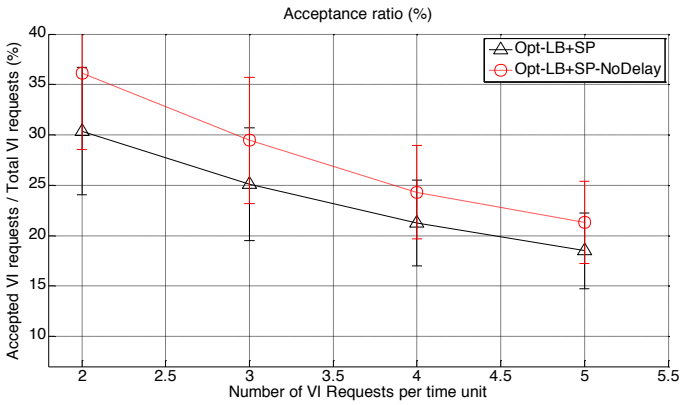


Figure 4-5: Delay constraint impact

Figure 4-5 shows the impact that the delay constraint brings to the acceptance of VIs. The delay factor decreases the acceptance ratio from 3% to 6% depending on the VI arrival rate. In this analysis only one strategy (*VIE-Opt-LB+SP*) is considered, since the comparison between different strategies was already provided in the previous sub-section.

The impact of the physical infrastructure size can be observed in Figure 4-6. Naturally, by increasing the physical infrastructure size, the acceptance rate increases. The overall ranking between strategies remains the same as in Figure 4-2.

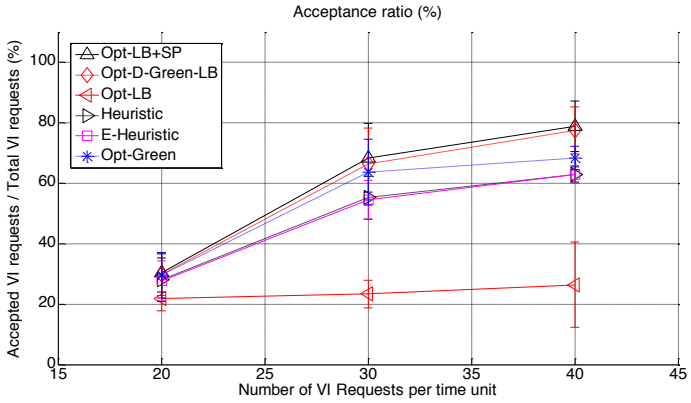


Figure 4-6: Physical Infrastructure size impact

4.1.4.4. Physical Infrastructure Energy Consumption

This sub-section presents the performance of the different strategies in terms of energy consumption of the physical infrastructure. While in terms of acceptance ratio some strategies presented values relatively close, the differences with respect to energy consumption are more pronounced.

The analysis is performed taking into account the cumulative time of active resources (node and link), and the results for strategies not considering re-optimization are depicted in Figure 4-7 and Figure 4-8. The values are presented in percentage, where 100% means that all resources were active during the entire simulation time. First, we analyze the values related to nodes, Figure 4-7, as they are the lead indicators of the energy consumed.

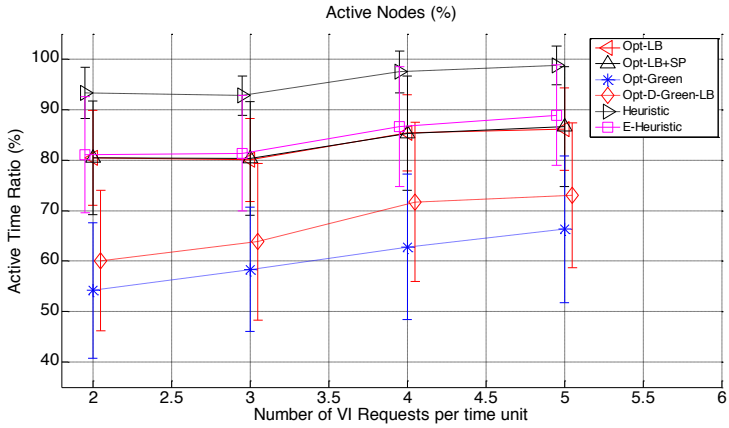


Figure 4-7: Cumulative time of active nodes – strategies without re-optimization

All strategies that consider the green policy perform better than the remaining ones in terms of energy consumption. The VIE-Opt-Green strategy clearly outperforms the remaining ones, with values between 53% for arrivals of 2 VIs per time unit, and 66% for 5 VIs per time unit. The next performing strategy is the VIE-Opt-D-Green-LB strategy, with values between 60% and 72%. It is important to highlight that, although VIE-Opt-D-Green-LB requires more active nodes, it presents a higher rate in terms of acceptance ratio.

The remaining two optimal strategies, those that do not apply the green strategy, present higher and very close values, between 80% and 87%. It is of relevance to highlight here the fact that the VIE-Opt-LB, used as reference point, presents values 10% lower in terms of acceptance ratio in relation to the remaining strategies, and it also presents equal values in terms of active nodes.

Finally, with respect to the heuristic approaches, the E-Heuristic (82-89%) clearly outperforms the Heuristic (93-99%), with improvements above 10%. Also note that, in terms of acceptance ratio, they present very close values.

In terms of active links, the ranking of strategies has a lower variation (Figure 4-8). The optimal strategies with green approaches perform better, with values of active time ratio between 17% and 28%.

The Heuristic is the one presenting higher values of active time ratio, between 42% and 49%. The E-Heuristic and the optimal strategies with load-balancing policies have performance values in between the previous ones.

Note that, just like the acceptance ratio indicator, the values here present a linear behaviour as the VI arrival rate increases. However, as opposed to the acceptance ratio, here they increase as the VI arrival rate increases. This is due to the fact that, although the acceptance ratio decreases with the increase of the VI arrival rate, the overall number of VIs embedded in the physical infrastructure increases, which leads to a higher percentage of active resources in the substrate.

In summary, considering the VI bandwidth consumption an active factor in the embedding strategy (VIE-Opt-LB+SP) clearly improves the acceptance without affecting the cumulative time of active resources (i.e. there is no significant variation of the energy consumption). Moreover, adopting a green approach (VIE-Opt-Green) presents nearly as good results as the best strategy. Finally, its positive impact in terms of energy consumption is considerably high.

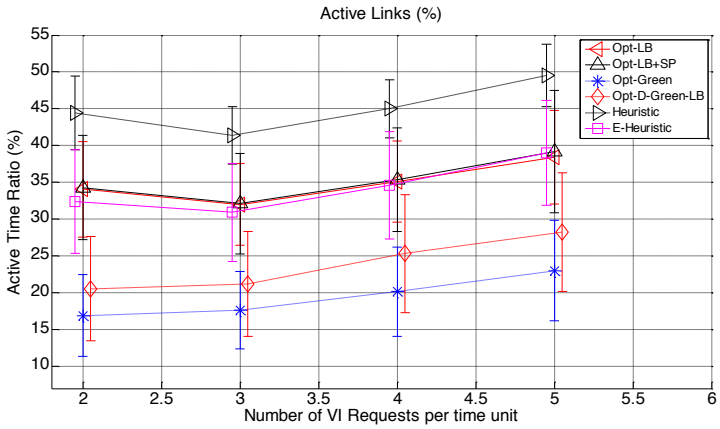


Figure 4-8: Cumulative time of active links – strategies without re-optimization

Now we analyze the values considering the strategies that allow re-optimization - Figure 4-9 and Figure 4-10. In terms of nodes (Figure 4-9), not surprisingly, all strategies that consider the green policy outperform the remaining ones in terms of energy consumption, i.e. they present lower values. The VIE-ReOpt-Link-Green strategy clearly outperforms the remaining ones, with values between 47% for arrivals of 2 VIs per time unit, and 57% for 5 VIs per time unit. The next strategy is the VIE-Opt-Green strategy, with values between 50% and 60%. It is important to highlight that these two strategies perform very similarly in terms of acceptance ratio, but by allowing link reconfiguration, the VIE-ReOpt-Link-Green presents gains up to 3% with respect to active nodes.

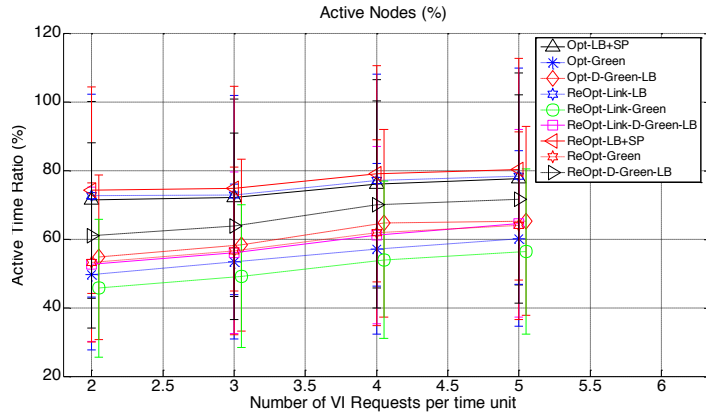


Figure 4-9: Cumulative time of active nodes – strategies with re-optimization

These strategies are then followed by VIE-ReOpt-Green and the VIE-ReOpt-Link-D-Green-LB, followed by the VIE-Opt-D-Green-LB, with the first two with values between 53% and 63%, and the latter one between 55% and 64%. Although having close values, it is important not to forget that the VIE-ReOpt-Green strategy

presents gains up to 5% in terms of acceptance and still manages to present lower values in terms of active nodes. The VIE-ReOpt-D-Green-LB is the following strategy presenting better performance, with values between 61% and 71%. The remaining optimal strategies, those that do not apply the green strategy, present higher and very close values, between 71% and 81%.

We highlight here the fact that, among the strategies that consider energy consumption, those that allow only link re-optimization present similar acceptance ratio values to those not allowing any re-optimization. However, the former ones do improve the node energy consumption indicators. If the impact of link reconfiguration is considered to be low to the point that it can be neglected, these strategies should indeed be considered as better approaches than those performing no reconfiguration.

In terms of active links, Figure 4-10, the strategies have similar performance. The optimal strategies with green approaches perform better, with values of active time between 12% and 27%. The optimal strategies with load-balancing policies present values between 29% and 37%.

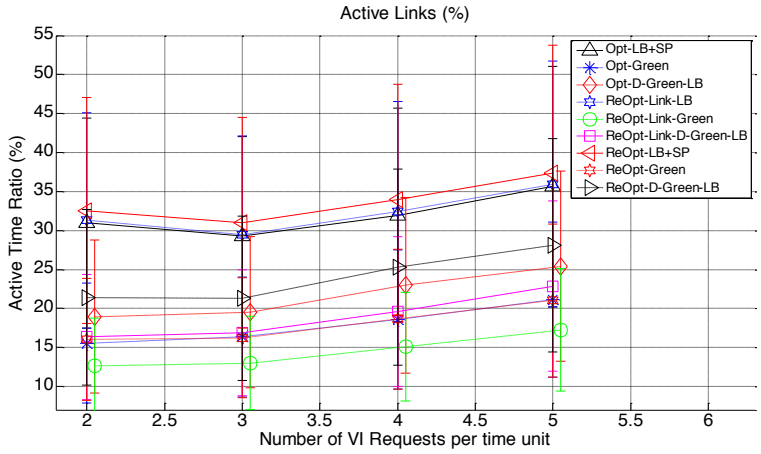


Figure 4-10: Cumulative time of active links – strategies with re-optimization

Note that, just like the acceptance ratio indicator, the values here present a linear behaviour as the VI arrival rate increases. However, as opposite to the acceptance ratio, here they increase as the VI arrival rate increases. This is due to the fact that, although the acceptance ratio decreases with the increase of the VI arrival rate, the overall number of VIs embedded in the physical infrastructure increases, which leads to a higher percentage of active resources in the substrate.

Furthermore, it is clear that, allowing resource optimization in the strategies that consider energy consumption helps reducing the energy consumption of the physical infrastructure without compromising the acceptance ratio. In strategies where node and link reconfiguration are allowed, the acceptance ratio is increased.

4.1.4.5. Re-Optimization Cost

In this subsection, it is performed an analysis to the different strategies that consider re-optimization of resources in terms of the impact of changes in VIs that they naturally lead to. The analysis is done in five different perspectives: i) VIs affected by reconfiguration, whether by link or node; ii) VIs affected by link reconfiguration; iii) VIs affected by node reconfiguration; iv) overall number of link reconfigurations; v) overall number of migrated nodes.

Figure 4-11 shows the number of VIs affected by reconfigurations (due to node and/or link). The VIE-ReOpt-D-Green-LB presents the highest value with a mean of 1.8 VIs affected per re-optimization process, followed by the VIE-ReOpt-LB+SP with a mean of 1.6. Then, it follows the strategies VIE-ReOpt-Green, VIE-ReOpt-Link-LB and VIE-ReOpt-Link-D-Green-LB with mean values of affected VIs of 1.3, 1.2 and 1, respectively. The strategy that shows lower impact is the VIE-ReOpt-Link-Green, with a mean of 0.6 affected VIs.

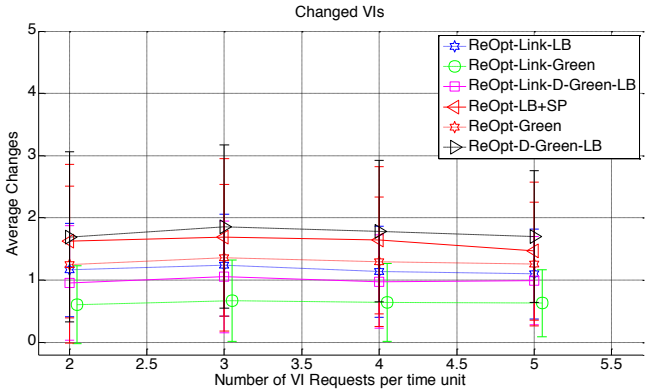


Figure 4-11: VIs affected by changes

The gap between the VIE-ReOpt-Link-Green and the other two strategies that allow only link reconfiguration (i.e. VIE-ReOpt-Link-LB and VIE-ReOpt-Link-D-Green-LB) is due to the fact that the former is a pure green strategy that tends to load active resources (nodes and links), which leads resources to a point of saturation much faster than the load balancing strategy. Therefore, with a higher probability of saturating nodes, trying to reconfigure virtual links to map a new VI will be in some cases inglorious. On the other hand, in the load balancing strategy, the probability of nodes reaching a saturation point is much lower than in the former case, which will allow more fruitful reconfigurations of virtual links.

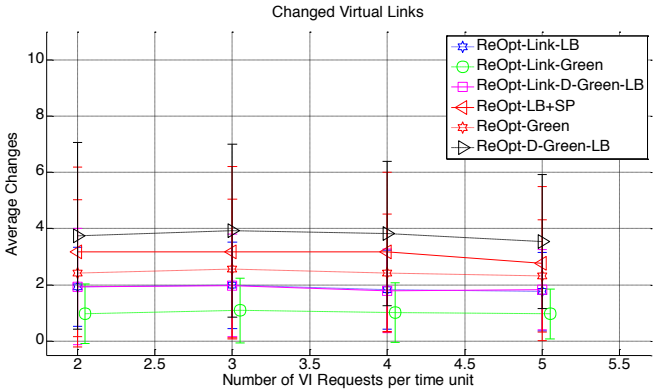


Figure 4-12: Virtual link changes

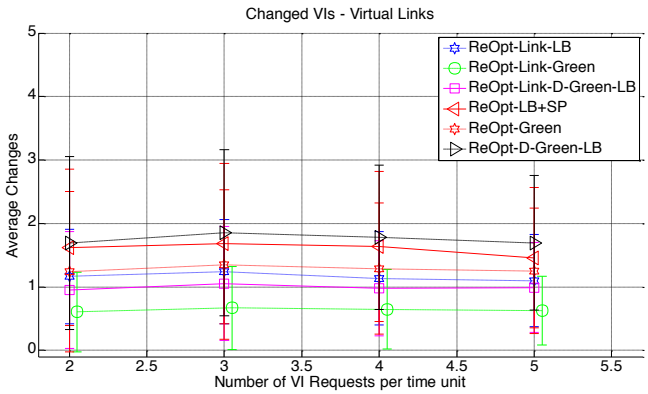


Figure 4-13: VIs affected by virtual link changes

Figure 4-12 shows the values related to the overall number of links reconfigured. The ranking here is the same as before. VIE-ReOpt-D-Green-LB (average of 3.9) is the strategy that triggers more link reconfigurations, followed by the VIE-ReOpt-LB-SP (average of 3.1), VIE-ReOpt-Green (average of 2.5), VIE-ReOpt-Link-LB (average of 1.9), VIE-ReOpt-Link-D-Green-LB (average of 1.9), and finally VIE-ReOpt-Link-Green (average of 1).

The number of VIs affected by the reconfiguration of links is depicted in Figure 4-13. Here the ranking remains the same as before.

With respect to the number of nodes reconfigured, the values between the three strategies that allow node reconfiguration are close – see Figure 4-14. The mean values are between 0.5 and 0.6 of virtual nodes that change. However, the number of VIs affected by the migration of nodes is only nearly 0.15 as shown in Figure 4-15.

These results show that the node re-optimization impact can be low. However, it is considered as future work the characterization of VI nodes in terms of their ability/cost to me migrated. In this sense, the embedding strategy can better assess which nodes it should first try to migrate.

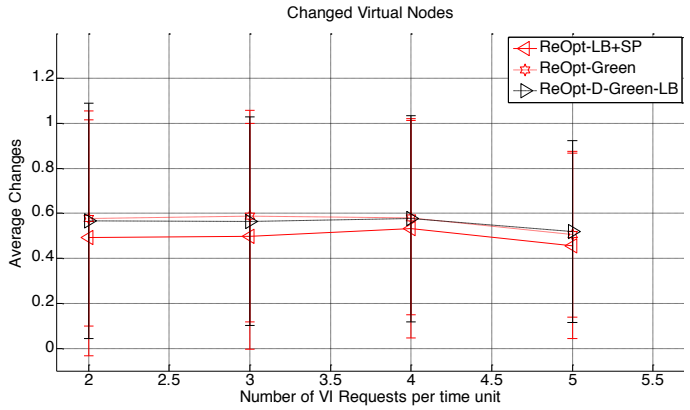


Figure 4-14: Virtual node changes

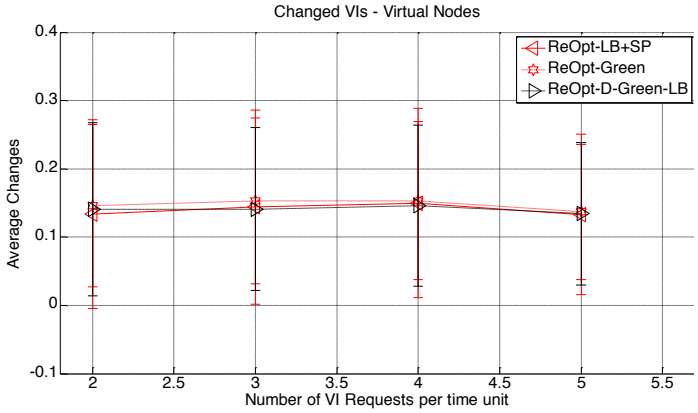


Figure 4-15: VIs affected by virtual node changes

4.1.4.6. Discussion

Strategies allowing full re-optimization of resources clearly outperform the remaining ones in terms of acceptance ratio. However, the adoption of these strategies needs to take into consideration the costs of reconfiguring resources. The values in terms of impact of reconfiguration presented in this work are optimistic. Furthermore, since each VI and resource can have specific requirements, an evolvement of these strategies could be to classify resources with respect to their ability/cost to be migrated. This aspect can be especially important in virtual nodes, since they are the ones more difficult to move. We believe that the strategies presented in this work provide a solid base foundation that can be extended for more specific and detailed scenarios as the one just mentioned.

Moreover, the improvements achieved are not strict to the acceptance ratio. Energy consumption can also be improved in strategies with green objectives, not only by allowing re-optimization, but also by adjusting the embedding strategy depending on the current load of the physical infrastructure.

In the cases of strategies allowing only link re-optimization, the values of acceptance ratio remain nearly the same; however, the energy consumption can be improved with green objectives. These are definitely

strategies worth taking into account, since the cost of reconfiguring links is much lower than those of reconfiguring nodes.

With respect to the heuristic approaches, it was not a surprise that they underperformed most of the optimal approaches. Nevertheless, we have provided a heuristic solution that is not very far from the equivalent optimal approaches in terms of acceptance ratio, and we were also able to improve it to perform better in terms of physical energy consumption.

4.2. Multi-Domain Approach

4.2.1. Problem Description

In this approach we look at a multi-domain perspective, where cloud and network resources are hosted in specific domains (e.g. DCs or Points-of-Presence (PoPs)) that are interconnected by a network domain (e.g. a Network Operator (NO)). This section describes the perspective as well as the notation used. Moreover, it details the embedding allocation problem.

The multi-domain perspective considers an infrastructure where multiple DCs of varied size and capacity are geographically dispersed and interconnected by a network, which can be a NO or an overlay network within the Operator's Network. In both cases, this network is refered to as Inter-DC Network. Figure 4-16 illustrates this perspective, which resembles the European Telecommunications Standards Institute's (ETSI) view of a Network Functions (NFs) Virtualization (NFV) Infrastructure [123].

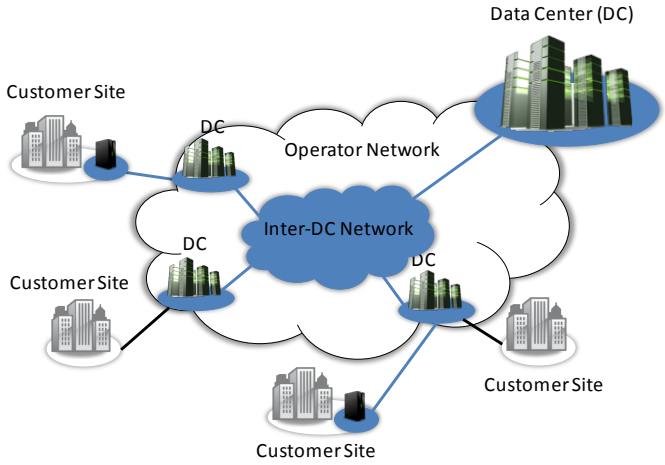


Figure 4-16: Multi-Domain Physical Infrastructure view.

At the VI level two types of nodes are considered, network nodes and server nodes, each with its specific set of associated characteristics. Network nodes are characterized by the number of CPU, denoted by *Cs*, and by the amount of memory, denoted by *M*. Server nodes are characterized by the same parameters as network nodes (*Cs*, and *M*) plus storage capabilities, denoted by *STG*. With respect to links, these are characterized by bandwidth capacity (*B*) and assumed to be bidirectional and with a maximum delay (*D*). Moreover, associated to the number of CPUs of a node (*Cs*), we consider another parameter that reflects the CPU load (or capacity), denoted by *C*.

At the physical infrastructure, two types of nodes are considered: DC nodes (denoted by *DC*) and transport network nodes (part of the Inter-DC Network, denoted by *Rt*). In this case, server nodes and virtual-enabled network nodes are considered to be part of DC nodes; however, the resource allocation within DC nodes is transparent in this perspective, since it is considered to be the responsibility of the DC domain management system (using single domain approaches like in section 4.1 presented).

DC nodes are characterized by their total amount of CPU, memory and storage. This amount does not need to reflect the actual total amount of resources of a DC, and can be just a subset of those resources. A good example is the case of a DC from a third party, where the total amount of available resources is a reflection of a business agreement between the owner of the DC and the VI provider. Moreover, in this case

the geographical factor is very important; therefore, a DC is also characterized by its location, denoted by *Loc*.

From a VI perspective, network or server nodes can only be accommodated in DC nodes. The transport network is used for hosting virtual links that connect virtual nodes in different DCs. Table 4-3 summarizes the notation used.

Table 4-3: VI Assignement Problem Notation – Multi-Domain

Symbol	Description
G^P	Physical Infrastructure
N^P	Set of Physical Nodes
DC^P	Set of DC Nodes
Rt^P	Set of transport Network Nodes (Inter-DC network)
L^P	Set of Physical Links (Inter-DC network)
i, j	Physical Nodes
ij	Physical Link
$C^P{}_i$	Total CPU of DC Node *i*
$M^P{}_i$	Total Memory of DC Node *i*
$STG^P{}_i$	Total Storage of DC Node *i*
$B^P{}_{ij}$	Total Bandwidth of Physical Link *ij*
$D^P{}_{ij}$	Delay of Physical Link *ij*
$Loc^P{}_i$	Location of physical Node *i*
$C^{Ptotal}{}_i$	Total CPU of DC Node *i*
$M^{Ptotal}{}_i$	Total Memory of DC Node *i*
$STG^{Ptotal}{}_i$	Total Storage of DC Node *i*
$B^{Ptotal}{}_{ij}$	Total Bandwidth of Physical Link *ij*
$C^{Pfree}{}_i$	Free CPU of DC Node *i*
$M^{Pfree}{}_i$	Free Memory of DC Node *i*
$STG^{Pfree}{}_i$	Free Storage of DC Node *i*
$B^{Pfree}{}_{ij}$	Free Bandwidth of Physical Link *ij*
$DC^{max}{}_{load}$	Load of the DC with maximum load
$L^{max}{}_{load}$	Load of the physical link with maximum load
B_{cons}	Bandwidth being consumed in the physical links
$Loc^V{}_m$	Location of Virtual Node *m*
$G^V{}_v$	VI request *v*
$N^V{}_v$	Set of Virtual Nodes of VI request *v*
$S^V{}_v$	Set of virtual Server Nodes of VI request *v*
$R^V{}_v$	Set of VN Nodes of VI request *v*
$L^V{}_v$	Set of Virtual Links of VI request *v*
m, n	Virtual Nodes
mn	Virtual Links
Cs	Number of CPUs of Virtual Node *m* of VI request *v*
$C^V{}_{v,m}$	CPU of Virtual Node *m* of VI request *v*
$M^V{}_{v,m}$	Memory of Virtual Node *m* of VI request *v*
$STG^V{}_{v,m}$	Storage of Virtual Node *m* of VI request *v*
$F^V{}_{v,m}$	CPU Frequency of Virtual Node *m* of VI request *v*
$B^V{}_{v,mn}$	Bandwidth of Virtual Link *mn* of VI request *v*
$D^V{}_{v,mn}$	Maximum Delay of Virtual Link *mn* of VI request *v*

Just like in the single domain case, we aim to define mechanisms for the best VIE possible (i.e. maximizing acceptance and minimizing costs) according to the available resources and VI requirements. In order to map these resources, a combined algorithm, able to perform balanced decisions taking into account the abovementioned requirements of both NO and DC resources, is needed. This algorithm must be able to determine a possible solution, i.e., DCs able to allocate the virtual resources according to the specified requirements and guarantee network requirements between DCs if needed. In this sense we have defined optimal strategy (section 4.2.2) to accomplish the VIE process in this scenario.

4.2.2. Optimal Approach

This section presents an optimal approach (section 2.6.1.5) to solve the VIE problem based on the work published in Annex Paper F. The study of the presented approach focuses on the impact of the location factor in the VIE process. The results of this study are presented in section 4.2.3.

4.2.2.1. Mathematical Formulations

This section presents the mathematical formulation to solve the VI embedding problem for multi-domain scenarios. First, we present the assignment variables as well as the generic constraints. Then, we present specific formulations to calculate the maximum load of DC nodes and links, which is the load consumption of each type of resource that is more loaded among all other resources, and to calculate the bandwidth consumption that VIs consume in the Operator's Network.

4.2.2.1.1 Assignment Variables

The formulation considers two binary assignment variables, *x* and *y*, one for the virtual nodes and another for virtual links as equations (46) and (47) show.

$$x_i^{v,m} = \begin{cases} 1, \text{ virtual node } m \text{ of VI } v \text{ is allocated at Data Center } i \\ 0, \text{ else} \end{cases} \quad (46)$$

$$y_{ij}^{v,mn} = \begin{cases} 1, \text{ virtual link } mn \text{ of VI } v \text{ is uses physical link } ij \\ 0, \text{ else} \end{cases} \quad (47)$$

4.2.2.1.2 Generic Constraints

Equation (48) reflects that virtual server and network nodes are assigned to DC nodes plus the fact that a virtual node is assigned just to one DC node.

$$\forall v,m \in DC : \sum_{i \in S} x_i^{v,m} = 1 \quad (48)$$

Further, equations (49), (50) and (51) guarantee that the capacity values of DC nodes, i.e. CPU load, memory, and storage, are kept within range. Equation (49) sums the CPU load of all virtual nodes hosted in each DC, and guarantees that it is kept below or equal to the total amount of CPU available in the corresponding DC. Equations (50) and (51) do the same for memory and storage resources.

$$\forall i : \sum_{m,v} x_i^{v,m} \times C_{v,m}^V \leq C_i^{P_{total}} \quad (49)$$

$$\forall i : \sum_{m,v} x_i^{v,m} \times M_{v,m}^V \leq M_i^{P_{total}} \quad (50)$$

$$\forall i : \sum_{m,v} x_i^{v,m} \times STG_{v,m}^V \leq STG_i^{P_{total}} \quad (51)$$

Equation (52) applies the multi-commodity flow constraint [206] with a node-link formulation [207], in order to simultaneously optimize the mapping of virtual links and virtual nodes. Moreover, the notion of direct flows on the virtual links is also used.

$$\forall v, \forall m,n \in N_v^V(m), m < n, \forall i : \sum_{j \in N^P(i)} \left(y_{ij}^{v,mn} - y_{ji}^{v,mn} \right) = x_i^{v,m} - x_i^{v,n} \quad (52)$$

Equation (53) guarantees that each selected physical link (of the inter-DC network) has enough bandwidth available to host a virtual link. In other words, the sum of the amount of bandwidth of the virtual links that go through *i* must be equal or lower than the amount of the free bandwidth in a physical link. Finally, equation (54) guarantees that the virtual link delay constraint is met, i.e. the maximum delay of the physical links assigned to a virtual link does not exceed the maximum delay of the virtual link. Virtual links hosted within a single DC are not subject to these constraints, since it is considered that the DC is able to internally fulfill the requirements.

$$\forall i,j \in N^P(i), i < j : \sum_{v,m,n \in N^V(m), m < n} B_{v,mn}^V \times \left(y_{ij}^{v,mn} + y_{ji}^{v,mn} \right) \leq B_{ij}^{P_{free}} \quad (53)$$

$$\forall v, \forall m,n \in N_v^V(m), m<n: \sum_{\forall i,j \in N^P(i), i<j} D_{ij}^P \times \left(y_{ij}^{v,mn} + y_{ji}^{v,mn}\right) \leq D_{v,mn}^V \quad (54)$$

Finally, the virtual node location factor is reflected in equation (55). It ensures that a virtual node can only be located in a DC that respects its location constraint.

$$\forall v, m \in DC: \sum_{i \in S} \left\lfloor x_i^{v,m} \times Loc_i^V \right\rfloor = Loc_m^V \quad (55)$$

4.2.2.1.3 Formulations

This section presents the proposed formulations. Three formulations are considered: maximum DC node, maximum link load, and bandwidth consumption.

Equation (56) shows the formulation for the DC maximum load, which looks for the DC node more loaded among all DCs. This value is given by the sum of CPU, memory and storage load fractions. Equal weights are considered for each resource type - δ_1, δ_2 and δ_3 - as they are equally important. Nevertheless, depending on the real scenario, adjustments can be made to these weights, e.g. if VI requests are more processing intensive, a higher weight can be given to δ_1.

$$\forall i \in DC: \left[\begin{array}{l} \delta_1 \dfrac{\sum_{v,m} x_i^{v,m} \times C_{m,i}^V}{C_i^{P_{total}}} + \delta_2 \dfrac{\sum_{v,m} x_i^{v,m} \times M_{m,i}^V}{M_i^{P_{total}}} + \\ \delta_3 \dfrac{\sum_{v,m} x_i^{v,m} \times STG_{m,i}^V}{STG_i^{P_{total}}} \end{array} \right] \leq DC_{load}^{\max} \quad (56)$$

The formulations for maximum link load and bandwidth consumption concern the resources in the inter-DC network domain. Equation (57) defines the maximum load consumption of a physical link among all links ($L_{load}^{\max}$). $B_{v,mn}^V$ refers to the bandwidth of the virtual link *mn* (i.e. the virtual link connecting the virtual node *m* and *n*) of a certain VI (*V*). $B_{ij}^{P_{Total}}$ refers to the total bandwidth capacity of the physical link *ij* (i.e. the physical link between physical node *i* and *j*). The variable $y_{ij}^{v,mn}$ assumes the value 1 if virtual link *mn* is crossing physical link *ij*, and 0 otherwise. In this way, the formula sums the bandwidth of all virtual links ($B_{v,mn}^V$) that cross a certain physical link (*ij*), and divides that value by the total capacity of the physical link ($B_{ij}^{P_{Total}}$). In the end, $L_{load}^{\max}$ assumes the load value of the most loaded physical link.

$$\forall i,j \in N^P(i), i<j: \left[\frac{\sum_{v,m,n \in N^V(m), m<n} B_{v,mn}^V \times \left(y_{ij}^{v,mn} + y_{ji}^{v,mn}\right)}{B_{ij}^{P_{Total}}} \right] \leq L_{load}^{\max} \quad (57)$$

Equation (58) defines the percentage of the total bandwidth consumed by a VI in the physical infrastructure. The amount of bandwidth consumed by each virtual link ($B_{v,mn}^V$) in all physical links ($y_{ij}^{v,mn}$) is summed and divided by the total bandwidth capacity of the entire physical infrastructure.

$$\frac{\sum_{v \in V, m,n \in N^V(m), m<n} y_{ij}^{v,mn} \times B_{v,mn}^V}{\sum_{i,j \in N^P(i), i<j} B_{ij}^{P_{total}}} = B_{cons} \quad (58)$$

4.2.2.2. Objective Functions

Taking into account our findings in the definition of embedded strategies for single domain scenarios, a load balancing strategy is considered the most adequate for multi-domain scenarios. The objective of this

strategy is to balance the load among the different DCs, and at the same time reduce the impact in the inter-DC network, taking into account the location of each domain and the VI location restrictions.

Equation (59) presents the strategy's objective function. The weights v_1 and v_2 are considered equal and its sum is 1. Note that the maximum link load ($L_{load}^{\max}$) and bandwidth consumption (B_{cons}) are under the effect of the same weight, v_2, and the maximum DC load ($DC_{load}^{\max}$) is under the effect of v_1. This evens the importance given to the DC load and inter-DC network occupation.

The approach that considers the network load as a joint analysis of the total bandwidth consumption and the maximum link consumption has proved to be the best approach so far – section 4.1.

$$min\ v_1\left(DC_{load}^{\max}\right)+ v_2\left(L_{load}^{\max} + B_{cons}\right) \quad (59)$$

In contrast with the single domain case, embedding strategies that take into account the energy consumption are not considered in the multi-domain approach. These strategies are considered to be adopted inside each DC domain and are transparent to the multi-domain perspective. On the inter-DC network side, since it is a transport network, we assume that the network nodes are always active.

4.2.3. Evaluation Results

This section provides a thorough evaluation of the optimal embedding solution presented in this work in a multi-domain scenario.

4.2.3.1.1 Scenario description

To evaluate the performance of the proposed approaches, a Matlab [208] simulator is used. For each run, the program designs a random physical infrastructure of 20 nodes based on the Waxman network topology generator [209], and it simulates a set of requests of VIs (with a number of nodes between 4 and 14) with Markov-modulated inter-arrival and inter-departure times. The program builds a random physical infrastructure of 20 nodes, where 40% of the nodes are DCs and 60% are transport network nodes of the inter-DC network. The choice of these values is due to the fact that: the envisioned scenario considers a wide geographical area with several DCs (in this case with 8 DCs); the transport network does not (have to) match an entire NO, but only a subset of it, exclusively dedicated for interconnecting the DCs (in this case with 12 nodes). The reason for these values also lays on the fact that these dimensions allow for the physical infrastructure to reach a saturation point, which will allow a better analysis of the solution.

Details on the nodes and link parameters can be seen in Table 4-4. Moreover, the VI request rates (λ) vary between 2 and 5 VIs per time unit (Poisson arrivals), and the average duration of the VIs ($1/\mu$, where μ is the average service rate) is 20 time units (exponentially distributed duration).

In order to solve the ILP, we have used CPLEX [210] version 12.3, integrating a plug-in for our Matlab simulator and setting a time limit of 600 seconds (i.e. 10 minutes) for each VI mapping.

Table 4-4: Physical and virtual infrastructure parameters

		Physical Infrastructure	Virtual Infrastructures
Net Nodes	**Cs**	-	{1; 2; 3; 4 }
	Memory	-	{64; 128; 256; 512}(MB)
DC Nodes	**Cs**	{32; 64}	{1; 2; 4; 8; 16; 32; 64}
	STG (GB)	{6400; 12800; 25600}	{100; 200; 400; 800; 1600}
	M (GB)	{256; 512; 1024}	{2; 4; 8; 16; 32; 64}
Links	**B (Mbps)**	{500-2000 / 500 steps}	{10-100 / 10 steps}
	D (ms)	{5-10 / 1 steps }	{0-40 / 5 steps}

The weight values used for the optimal strategy, v_1 and v_2, are 0.5 (50%); as it was highlighted in the strategy definition, these values are used to even the maximum load on DCs and inter-DC network.

The analysis is focused on the acceptance ratio, since the energy and re-optimization strategies are considered to be applied within each DC. In this case we analyze the impact of location as a constraint in the allocation of virtual resources. We consider that each virtual node resource has a location restriction

with an associated range: the higher the range, more DC locations are likely to satisfy the virtual resource location restriction. In this work we present the values for two different location ranges: 25% and 50%. The range percentage is calculated based on the geographical size of the physical infrastructure, for example: for a physical infrastructure spanning 600 square kilometres, a range of 25% corresponds to 150 square kilometres, and a range of 50% corresponds to 300 square kilometres. Moreover, this variation is also analyzed with respect to its impact on the physical infrastructure.

All values present a mean of 10 runs (with different substrate) with a confidence interval of 95%.

4.2.3.1.2 Virtual Infrastructure Acceptance Ratio analysis

Figure 4-17 presents the VI acceptance ratio of the multi-domain embedding strategy for the two different location ranges (25% and 50%). It is clear that the impact of location is significantly high. For a VI arrival rate of 2, the acceptance ratio for the range of 50% drops approximately to half (18%) of the acceptance for the range of 25% (35%).

This result is due to the fact that, as the location range is reduced, the number of DCs able to embed virtual nodes is also reduced. This automatically reduces the probability of finding an embedding solution for the VIs.

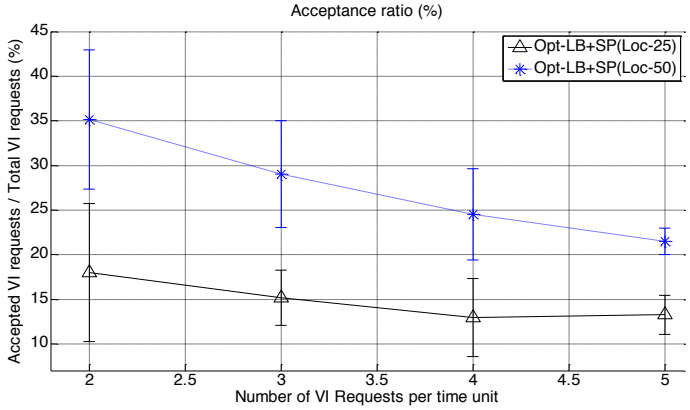


Figure 4-17: VI acceptance ratio vs arrival rate and location restriction

4.2.3.1.3 Physical Infrastructure Impact analysis

The impact of the number of VI requests and the location restriction on the physical infrastructure are analyzed in terms of inter-DC network and DC occupation. Figure 4-18 shows the average number of embedded virtual nodes. The embedding strategy with a location restriction of 50% has, in average, the double of virtual nodes embedded when compared with the strategy with a location restriction of 25%. This happens because the range of available virtual node embedding solutions in the former case is significantly higher than in the latter one. Moreover, the number of embedded virtual nodes in both cases increases linearly as the arrival rate increases. Although the acceptance ratio decreases while increasing the number of VI requests, the number of virtual nodes increases because the total number of VIs embedded still increases (even with a decrease in the acceptance ratio).

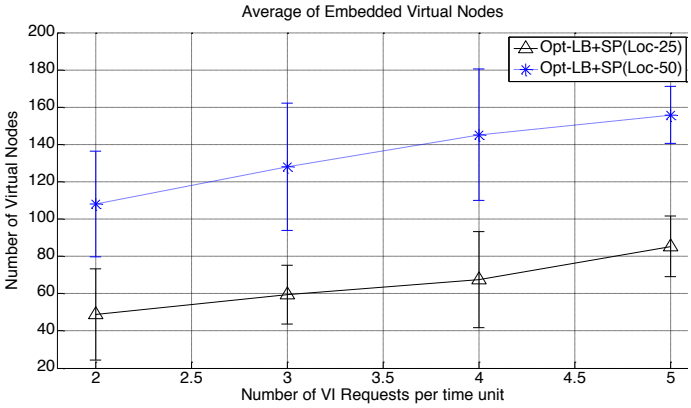


Figure 4-18: Average number of Embedded Virtual Nodes vs arrival rate and location restriction

Figure 4-19 shows the average bandwidth occupation of the Inter-DC network. This value is obtained by dividing the total amount of occupied bandwidth for the total amount of virtual bandwidth effectively requested. In this case, compared to the virtual node one, the ranking is inverted: the strategy with a location restriction range of 25% presents higher occupation than the one with a restriction range of 50%. This happens because, with a lower location range, the number of embedding solutions decreases, which prevents the finding of a better collocation of virtual nodes of a same VI in order to reduce the impact on the inter-DC network. In summary, the results presented in both cases are a reflection of the applied strategies.

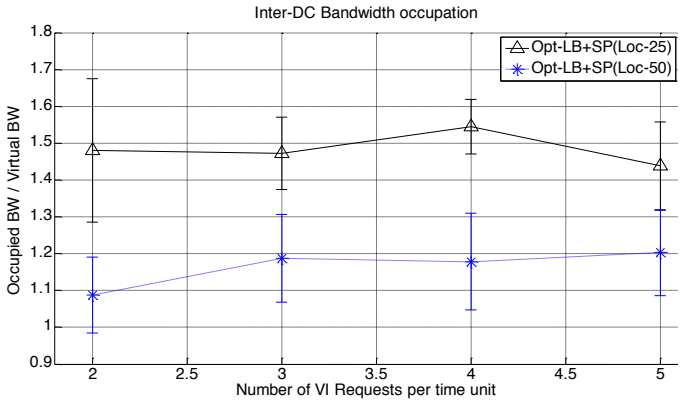


Figure 4-19: Average Inter-DC network bandwidth occupation vs arrival rate and location restriction

4.2.4. Discussion

This section has showed that, in a distributed cloud environment, the embedding process of VI resources with location as a key requirement has high impact on the acceptance ratio. We have showed that relaxing the location constraint significantly improves the acceptance ratio. The impact on the NO shows an opposite behavior, i.e. as the location constraint relaxes, the load on the NO decreases due to the decrease of the number of embedding solutions.

4.3. Summary

This chapter proposed a set of approaches to solve the VI embedding problem in single and multi-domain scenarios. From a chapter 3 perspective, it covered the resource allocation, resource adaptation and re-optimization internal blocks of the resource management block.

For the single domain case, different strategies were defined based on heuristic and optimal approaches. The heuristic approach, although underperforming most of the proposed optimal strategies,

shows an encouraging performance: its difference towards the best optimal approaches is between 2% and 18%, depending on the size of the physical infrastructure (the bigger the infrastructure, higher the gap). The optimal strategies take into account the load balancing of physical resources, the energy consumption of the physical infrastructure, and the impact of the VI re-optimization process. A comparative performance analysis between the different strategies was performed taking into account the acceptance ratio, the energy consumption of the physical infrastructure and the impact of re-optimizations. The results show that there is a clear tradeoff between improving acceptance ratio and reducing the physical infrastructure usage without allowing re-optimizations. The use of different strategies depending on the load of the physical infrastructure resources can be a way to deal with this tradeoff. Moreover, allowing virtual resources to be reconfigured can considerably increase the acceptance ratio, as well as reduce the energy consumption of the physical infrastructure. This is also clear in the case where only virtual links are allowed to be reconfigured.

For the multi-domain case, an optimal approach that takes into account the load balancing of DC and inter-DC network domains was proposed. The results show that location requirements are an important factor that brings a high impact on the VI acceptance ratio and on the occupation of the physical infrastructure. By reducing the location range, the VI acceptance ratio reduces, since less DCs become eligible for accommodating virtual nodes. The same happens with the DC load: with less VIs accepted, the DC's load is lower. However, in terms of inter-DC network, by reducing the location range, the inter-DC network load increases. This happens because the ability to co-locate virtual nodes in the same DC is lower, leading to more disperse nodes across DCs and a larger virtual link accommodation in the inter-DC network.

One should note that although we have studied heuristic approached, most of our work was focused in the study of optimal approaches. We consider this the right approach as the extensive study of optimal approaches is fundamental to further elaborate on other approaches, such as the heuristic ones, provide the upper bound and a way to assess the performance of heuristic approaches. Nevertheless, we are also aware that research on the heuristic side should be further elaborated to consider all the factors considered in the optimal one, and also to bring the heurists performance closer to the optimal ones.

Furthermore, the VI embedding problem is a very complex problem and there are other aspects of it that should be target of further research. Among these aspects is the ability to handle VI elasticity and fault management (e.g. react and prevent failures) in the best way possible.

Chapter 5 will take advantage of the solutions presented in this chapter and in chapter 3 to leverage the NFV concept.

5. Network Service Functions Virtualization

After laying important foundations in the integration of cloud computing and Network Operator (NO) infrastructure services in terms of architecture in chapter 3 and focusing on resource management aspects in chapter 4, this chapter now builds upon those foundations to explore the concept of Network Functions (NFs) Virtualization (NFV). It looks to the cloud and NO integration on the perspective of fulfilling NFV requirements. The chapter presents a platform for the management of Service Functions (SFs) that lays part of its foundation upon the principles and mechanisms presented in chapter 3. Further, it elaborates on the SF virtualization process and presents a set of data models for the definition and chaining of SFs.

First, a platform for orchestrating and managing SFs is introduced, entitled Cloud4NFV, which is based on the work presented in Annex Paper H. The platform architecture follows the most relevant standard NFV guidelines (i.e. European Telecommunications Standards Institute (ETSI) NFV).

Later on, based on the work published in Annex Paper H we present and elaborate on the concept of Carrier cloud. Still in the scope of this work we elaborate on SF virtualization, with special focus on the modelling of SFs towards Virtual Infrastructure (VI) resources. Then, supported on the work published in Annex Paper H we further explore the SF composition and address the concept of SF Chaining (SFC) by laying important foundations and presenting two possible approaches.

Finally, we present a proof of concept that supports all the architecture, features and functionalities presented in this chapter, exploring also the service concept of SF as-a-Service (SFaaS).

5.1. Carrier Cloud

Traditional cloud infrastructures are far from being suitable for all types of business, especially when referring to network SFs. Network SFs have carrier grade requirements, from guaranteed Quality of Service (QoS) in terms of Information Technology (IT) resources and network connectivity, to high-availability and fast fault recovery through redundancy. Telcos, with their already established distributed network infrastructure and hosting centres, are ideally positioned to take the lead in this area, as they can create a compelling end-to-end cloud proposition that integrates their network management capabilities, adapted to a more agile and cloud service-oriented operation model (on-demand, self-service, elastic). Furthermore, they have the experience in providing carrier grade service levels.

We envision a near-future Telco cloud infrastructure that comprises not only the traditional Data Center (DC) domains, but also the Wide Area Network (WAN) domain (inline with what was presented in chapter 3). In such scenario, the Telco takes advantage of its already established distributed facilities (sometimes refered as Points-of-Presence (PoPs)) to host small cloud environments. It is also possible for this

distributed cloud infrastructure to extend itself into the costumer site. A similar scenario is depicted in Figure 2-9.

In this scenario, it is important not to forget the high importance resource management, to which chapter 4 has already provided a very solid contribution.

5.2. Platform for Virtual Network Service Functions

In this section the Cloud4NFV platform, which is an enhancement of the CloudNet platform, is presented. First, its key functionalities are presented, and then the platform's architecture is detailed.

5.2.1. Functionalities

The most relevant functionalities of the Cloud4NFV platform are:

- Automated deployment, configuration and lifecycle management (e.g. instantiation, configuration, update, scale up/down, termination, etc) of SFs.
- Exposure of functionalities such as: service deployment and provisioning; service monitoring and reconfiguration; and service teardown.
- Federated management and optimization of WAN and cloud resources for accommodating SFs.
- Support of SFC composition through SFC.

5.2.2. Architecture

Figure 5-1 provides an overview of the system, which is organized in four major layers: Infrastructure Layer, VI Management Layer, Integrated Management and Orchestration Layer, and Service Layer. The Service Layer handles the services that are built on top of Cloud4NFV, and the Infrastructure Plane comprises all physical resources, whether in the NO or cloud domains. Special attention will be given to the VI Management Plane and the CMS, since we consider it a major lever for enabling SFC. It is important to note that this architecture is aligned with the ETSI NFV architectural guidelines [125]. This fact is highlighted along the description of the platform.

The SF Orchestrator is responsible for exposing and orchestrating SFs. Furthermore, the SF Manager(s) are responsible for managing the life-cycle of SFs. Note that for each SF or composed SF there is a SF Manager. When the Orchestrator is requested a SF, it asks the SF Manager (who knows the SF details) to process the request and start the provisioning process. The Cloud Network Management System (CNMS) gives the Orchestrator an integrated view of NO domains and how they connect to DCs, PoPs and costumer site domains.

Looking to the ETSI's architecture, the SF Orchestrator corresponds to the ETSI's Orchestrator, the SF Manager(s) to the ETSI's Virtual Network (VN) Function (VNF) Manager(s) while the CNMS can be seen as being also part of the ETSI's Orchestrator. Moreover, the CMS and the NMS can be compared to ETSI's VI Manager(s) (VIM). At this point we highlight that the integration of NMS and therefore of NO connectivity services is an add-on towards current ETSI definitions, where this integration is indeed foreseen, but has not taken place yet.

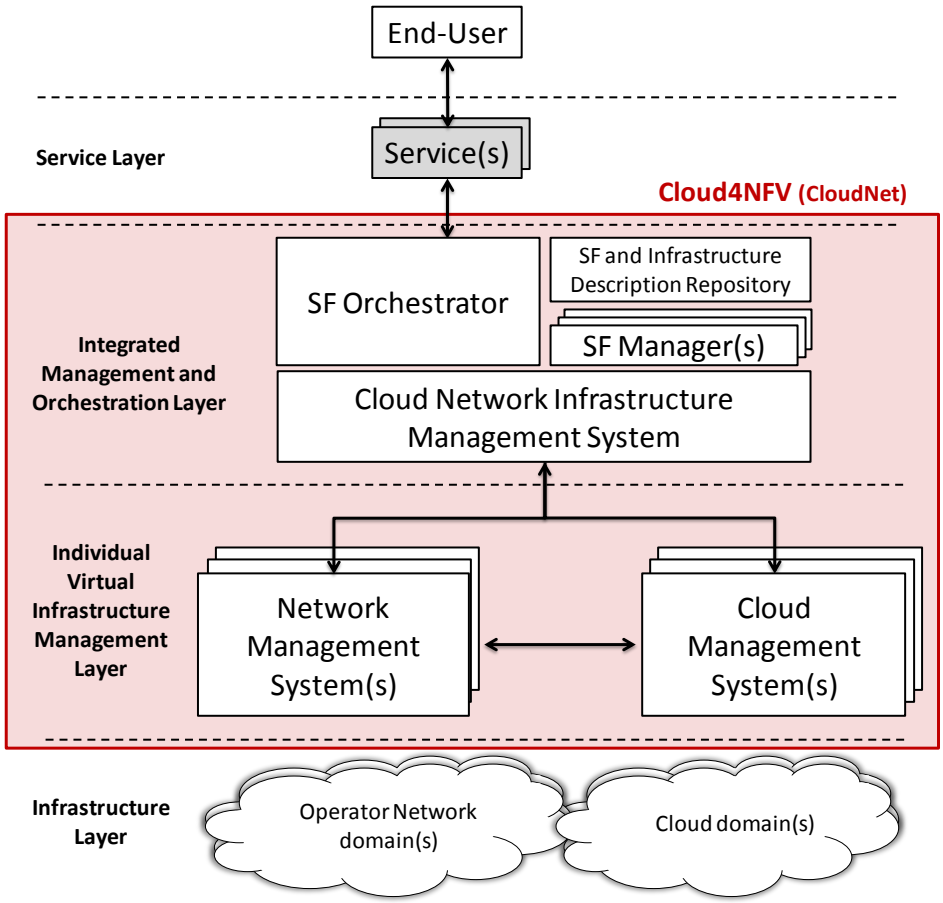


Figure 5-1: Cloud4NFV platform – high-level architecture

5.2.2.1. Service Function and Infrastructure description repository

The information about network topology, available IT and network resources, and information about SF are stored in this component. Currently two categories of SFs are considered: first tier SFs, and second tier SFs. First tier SFs have a direct relation with infrastructure resources (as presented in section 5.3.1), while second tier SFs do not. The latter ones are dependent on first tier SFs as they extend and rely on first tier SFs. Both first and second tier SFs may have software dependencies, which may require the installation and configuration of software components within the compute instances. These dependencies are part of the SF description and are refered in this work as recipes.

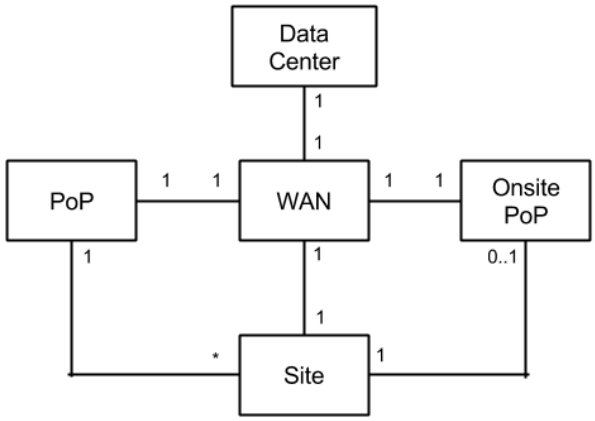


Figure 5-2: Infrastructure Connectivity Data model

Figure 5-2 and Figure 5-3 present the infrastructure modelling. The former shows the connectivity modeling while the latter shows the resource modeling. In Figure 5-2 it is possible to see that the WAN is modeled by a single entity. DCs, PoPs on-site PoP, and costumer sites are connected to the WAN. A PoP can have multiple costumer sites associated (however, one site is considered to have only one PoP link associated – redundancy aspects are not discussed in this work), and a site can have an on-site PoP.

Figure 5-3 shows that DCs and PoPs have multiple *Block Storages* (one *Block Storage* has a specific size associated), *Compute Flavors*, *Link Flavors*, and *Local Networks* (intra-DC networks). The number of *Block Storage*, and *Compute Flavor* represents the total amount of IT resources available to the platform in each premises. This information allows the platform to better allocate resources across DCs and PoPs.

Similarly, the WAN domain has a set of predefined available *Links*, each with a set of available *Link Flavors*. Each *Link* has associted a pair of endpoints (DC, PoP, on-site PoP, site). This aspect is necessary because in the WAN the dispersion of endpoint is much higher and connectivity atributes like bandwidth, delay and jitter are directly related to the location of the endpoints that are being connected.

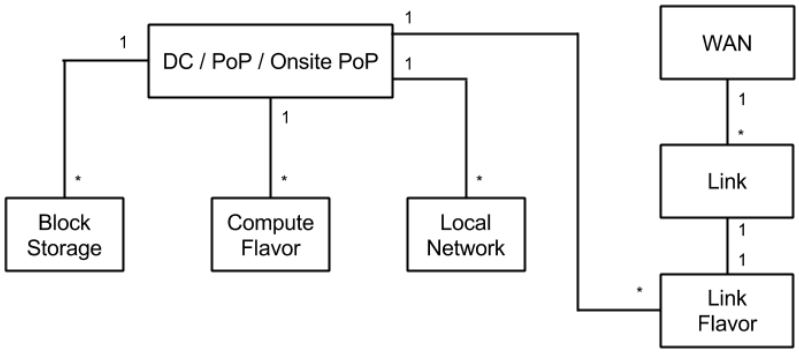


Figure 5-3: Infrastructure Resources Data model

5.2.2.2. Orchestrator

The *Orchestrator* contains a northbound interface that exposes the following functionalities: service advertisement; service deployment and service provisioning; service monitoring and reconfiguration; and service teardown. On the east side it has an interface to the different *SF Managers* and on the south side to the CNMS. Furthermore, it has access to the *SF and Infrastructure description repository* module that provides the *Orchestrator* high-level information about services and infrastructure. The actual service configuration and implementation are transparent to the *Orchestrator*, establishing an abstraction layer towards the *SF Managers* and CNMS.

5.2.2.3. Service Function Manager(s)

The *SF Manager(s)* are responsible for the entire lifecycle of a SF. It can request infrastructure resources (via the *Orchestrator*) and interact directly with them, e.g. to install a software component or configure a SF. This module can retrieve the required information about the SFs (e.g. SF infrastructure description and configuration recipes) by accessing the *SF and Infrastructure description repository*. Moreover, as part of the lifecycle management of a SF this module is responsible for retrieving monitoring information from infrastructure resources and SF themselves in order to, for example, perform scaling operations of virtual resources when needed. It has the following internal modules:

- Resource Mapping – responsible for mapping IT and network connectivity resources.
- Monitoring and Optimization – monitors IT and network resources along with specific SFs parameters, and if necessary, it triggers actions to optimize them (e.g. scaling or migration of resources).

5.3. Service Function Virtualization

In this section we elaborate on the SF modeling towards VI resources and look to the compositions of SFs as a motivation for SFC, addressed in section 5.4.

5.3.1. Data Model towards Virtual Resources

Figure 5-4 shows how SFs can be modelled towards virtual infrastructure resources. Herein each class is detailed. From a SF perspective, two classes have been defined:

- ***Service Function* (SF):** represents an instance of a functional block responsible for a specific treatment of received packets that has well-defined external interfaces – SF *Endpoints* (SFEs).
- ***Service Function Endpoint* (SFE):** represents an external interface of one SF instance that is always associated to a SF. Each SFE can have associated information regarding layer 1 (e.g. physical/virtual interface), layer 2 (e.g. MAC address) and/or layer 3 (e.g. Internet Protocol (IP) address), or even regarding higher layers, e.g. Hypertext Transfer Protocol (HTTP).

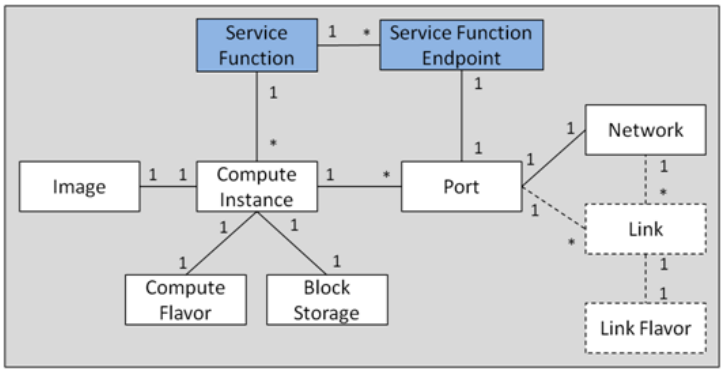


Figure 5-4: Service Function data model towards a cloud infrastructure

From an infrastructure perspective, the resources considered to realize a SF are: *Compute Instance* (i.e. Virtual or Physical Machines), *Image* (disk image), *Compute Flavor* (hardware specification of a compute instance, i.e. CPU, memory and root disk), *Block Storage* (additional disks), *Port* (i.e. network interface), *Network* (a network segment), and *Link* (a connection between two *Ports* from different *Compute Instances*) which has an associated *Link Flavor* (dedicated QoS in terms of bandwidth and delay). A SF can be associated to multiple *Compute Instances*, while the latter has a single *Image*, a single *Flavor* and can have multiple *Ports* and many *Block Storages*. A *Port* can only be associated to a single *Network*; however, it can be associated to multiple *Links*. A SFE is directly associated to a *port*, but not all *ports* need to map to SFEs.

The network QoS, represented in this model by *Link* and *Link Flavor*, is not considered in today's cloud infrastructure systems. However, for a carrier grade cloud this is a must, and platforms like OpenStack already have plans to support it[7].

5.3.2. Service Function Composition

With virtual SFs there will be a greater need to compose and organize virtual SFs dynamically. Figure 5-5 presents an example of how several SFs can eventually be composed and organized. The relation between SFs presented in this figure resembles what the ETSI NFV refers to as Forwarding Graph (FG). According to ETSI, a FG is a "*graph of logical links connecting network function nodes for the purpose of describing traffic flow between these network functions*" [126]. The FG concept is intimately related to the one of SFC. One could think of a FG modelled by multiple SFCs. Figure 5-5 shows how the FG can be materialized relying on three SFCs. In the following section we elaborate on the SFC concept.

[7] OpenStack Neutron QoS support https://wiki.openstack.org/wiki/Neutron/QoS

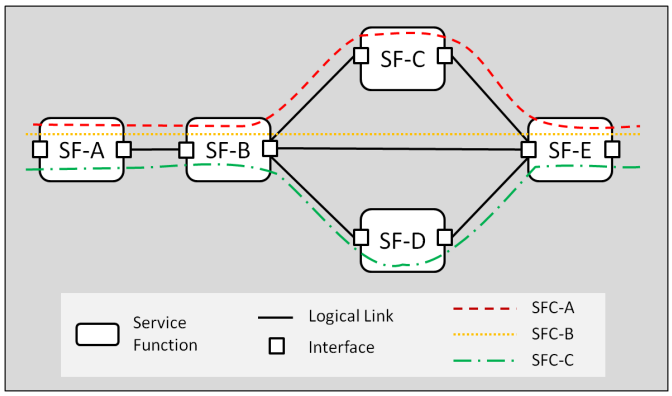


Figure 5-5: Service Function composition - example

5.4. Service Function Chaining

SFC is loosely defined as "*an ordered set of service functions that must be applied to packets and/or frames selected as a result of classification*" [211]. Although there are very good and important contributions (e.g. in Internet Engineering Task Force (IETF) and Open Networking Foundation (ONF)), work is still required, namely when it comes to the details on how to model and actually realize SFCs.

But first we highlight the advantages of SFC in the scope of one of the most prominent NFV use-cases [123]. Then, we elaborate on the most important aspects behind SFC. Finally, we present a data model towards virtual resources to actually realize SFCs.

5.4.1. Customer Premises Equipment Use-Case

CPE SFs are often pointed out as one of the most suitable candidates for virtualization [123] [124]. If we look to the particular case of the enterprise grade CPEs, SFC will surely play a particular relevant role.

The CPE is a collection of SFs, which can be standard routing nodes with Network Address Translation (NAT) and Firewall (FW) capabilities, but also Voice over IP (VoIP) servers, Virtual Private Network (VPN) servers (possibly extending to the operators network), WAN Optimization Controllers (WOC), Deep Packet Inspection (DPI) or Intrusion Prevention System (IPS). Although these services are combined in a single network infrastructure, they are deployed for different scenarios and not all traffic needs to traverse them, leaving room for optimization through SFC, and its capability of relocating functions across network elements. In Figure 5-6 we present a summary of SFC examples for CPE.

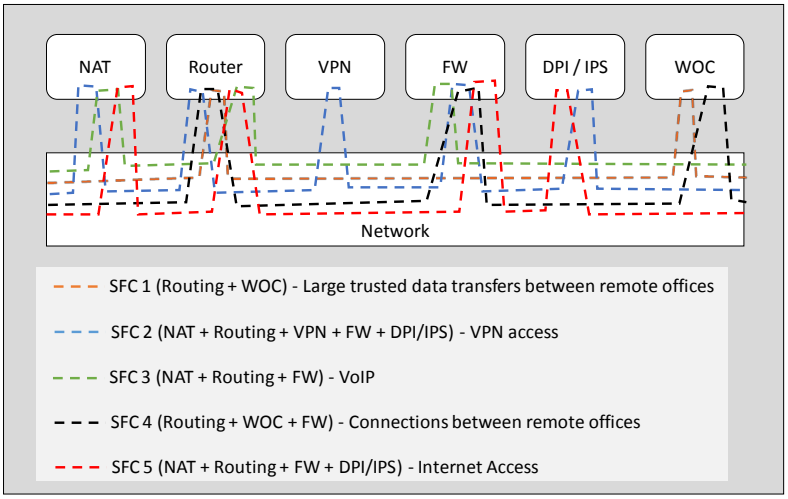


Figure 5-6: CPE Use-Case

It should be noted that some of the chains can even be temporary (e.g. SFC-1), which states the need for a model that enables the dynamic definition of chains.

5.4.2. Fundamentals

In SFC two aspects are vital, classification and traffic steering. Furthermore, in order to actually realize a SFC two approaches can be followed: tagged or non-tagged approach. Finally, depending on the type of processing packets are subject to, two categories of SFs can be derived, active and passive. In this section we elaborate on all of these aspects.

5.4.2.1. Classification

Classification is a policy for matching packets (e.g. *HTTP* traffic) used for the identification of appropriate actions (e.g. forwarding). It can be for example an explicit forwarding entry in a network device that forwards packets from one address (e.g. IP, MAC) into the SFC. (Re)Classification can also occur at each SF of the SFC independent from the previous SFs. In other words, there can be multiple classification points within one SFC. In this sense, multiple classification policy entries should be allowed in an SFC system.

5.4.2.2. Traffic Steering

Traffic Steering refers to the ability to manipulate the route of traffic, i.e. delivering packets from one point to another, at the granularity of subscriber and traffic types [212]. The actual network topology or overlay transports should not be modified to accomplish this.

5.4.2.3. Tagged vs. Non-Tagged packet approaches

The actual combination of classification and traffic steering can be done in two ways, tagged and non-tagged approach. In a tagged packet approach classification can occur only at the initial redirection points to a SFC, if upon this classification packets are tagged. After that, packets are steered to the SFC and routed along it according to the embedded tags. In a non-tagged packet approach classification occurs not only at the redirection points but also at each hop of the SFC. In this case, packets are not tagged and are subject of classification and steering at each SFC hop.

While the first might look a simpler approach, we believe it to be in fact the most complex one. We believe the non-tagged approach to be the smoothest approach to follow due to its lower impact on SFs and virtual infrastructure management systems. The advantage of the tagged approach is that the traffic only needs to be classified and tagged (e.g. with a VLAN tag, or another tag) once along the entire SFC. The drawback is the fact that the SFs need to know how to handle the tags (in the simplest case, they should at least ignore them). Although, we can add in the platform the support for a tagged approach (e.g. classify only at one point, tag, and steer traffic according to tag), it will only make sense if there is also support at the SF level. In this sense, in this work we adopt the non-tagged packet approach.

Further aspects should be taken into account when elaborating a SFC solution, such as: i) no assumption should be done on how functions are deployed, i.e. whether they are deployed on physical hardware, as one or more Virtual Machines (VMs), or any combination thereof; ii) a SF can be part of multiple SFCs; iii) a SF can be network transport independent; iv) a SFC allows chaining of SFs that are in the same layer 3 subnet and of those that are not; v) traffic must be forwarded without relying on the destination address of packets; vi) classification and steering policies should not need to be done by SFs themselves [213].

5.4.2.4. Active vs. Passive Service Functions

Having in consideration the packet treatment that SFs perform, two categories can be defined, active and passive SFs. Active SFs are those that are in fact part of the main course of a packet, in which case two sub-types are considered: a) functions that may drop packets or forward them, such as a Firewall; b) functions that can actually change packets, e.g. an IPSec VPN server. Passive SFs are considered to be out of the main course of the chain. These functions mainly inspect packets, e.g. a monitoring system or a DPI. In practice one can think of a SF in a physical device connected to a hub through a single network interface

configured in promiscuous mode. Traffic is considered to be duplicated when having to reach a passive function.

These two categories are important because they impose constrains on how classification and steering can be implemented. In short, passive functions can rely on packet characteristics as packets are not modified, while active functions must be integrated at a service level because ingress and egress packets can be completely unrelated (e.g., VPN). If a SFC has active functions that change packets, the classification may differ when passing one of these functions. Figure 5-7 illustrates a generic example of a SFC following a non-tagged approach composed of three active SFs and one passive SF, where classification occurs at each hop of the chain.

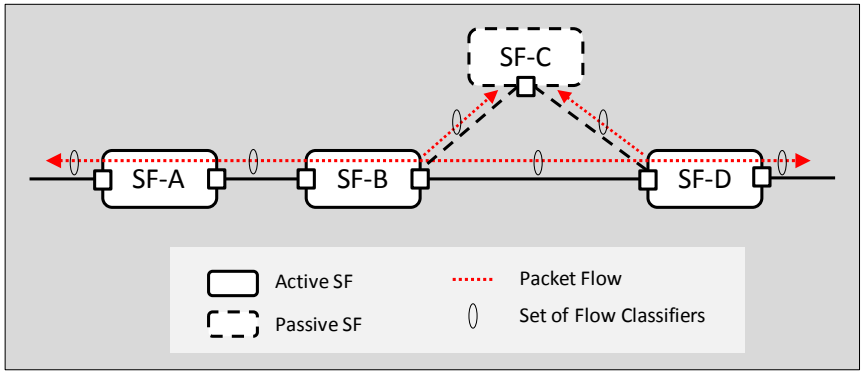


Figure 5-7: SFC example

5.4.3. Data Model towards Virtual Resources

Having in mind the considerations made so far, a base data model for SFC (that supports both tagged and non tagged approaches) is now presented. Naturally, other SFC service abstraction proposals may appear in the future, but we consider that this model lays a strong foundation over which other service abstractions can easily be created by extending the model. Figure 5-8 depicts the model. Five main classes are considered: *SFC*; *SF; SFE; Packet Flow* and *Classifier*.

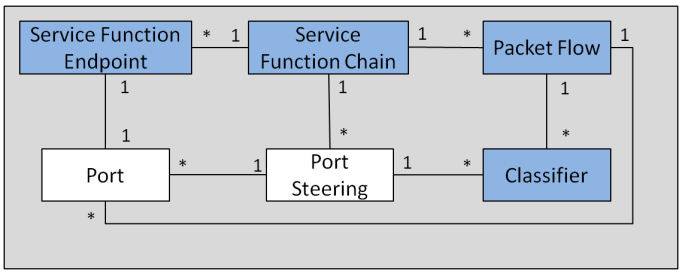


Figure 5-8: Service Function Chain data model towards a cloud infrastructure

All classes have the following attributes: *id*, *name*, and *description*. The *id* refers to a unique identifier able to identify the class instance within the SFC system. The remaining two, *name* and *description*, are attributes that allow a human-readable characterization of the class instance. Below it is provided further detail about each class.

- ***Service Function Chain* (SFC):** a SFC has a set of SFs associated and an attribute that defines the ordered sequence of functions (*path*). Since a function can have more than one SFE, the *path* attribute is specified by an ordered list of SFEs organized by hops. For example:
  "*path= { hop={SF-A_E2, SF-B_E1};*
  hop={SF-B_E2, SF-D_E1}, passive={SF-C_E1} }"
  where the chain crosses SF-A, SF-B, and SF-D and has SF-C as a passive function between SF-B and SF-D.

- ***Classifier*:** a *classifier* represents a classification criteria applied to a packet, which determines if the packet matches that specific criteria or not. In this sense, a *classifier* has an attribute *filter* that contains the classification criteria, e.g.:
  "*filter={protocol='6'; port='80-90';*
  source_IP='192.168.10.20/32'; destination_IP='192.168.10.40/32'}" - matches all TCP traffic using ports between 80 and 90 with source IP address 192.168.10.20 and destination IP address 192.168.10.40.
- ***Packet Flow*:** One *classifier* only identifies packets with a certain criteria, while a *packet flow* represents an aggregator of *classifiers*. In this sense, a *packet flow* can have multiple *classifiers*, and a *classifier* can be associated to multiple *packet flows*. Moreover, a *packet flow* has a *source* and *destination*. The former identifies where the initial classification and redirection of the packet flow to the SFC takes place, while the latter identifies where packets are to be delivered after passing through the SFC. The attributes considered so far would be enough if the system realizing the SFC followed a tagged packet approach. For a non-tagged approach, an additional attribute is considered - *sfc_classifiers*. Due to the possibility of (active) SFs to modify packets, the classification initially done may not be the same along all hops of the SFC, and therefore, the *sfc_classifiers* attribute matches the classification criteria (*classifiers*) at each hop of the SFC.
  Furthermore, the attribute *direction* is also considered to identify the direction of the SFC which the *packet flow* must traverse – this attribute can assume one of two values: *forward*, *reverse*. We consider that multiple *packet flows* can be associated to a single SFC instance.
- ***Port Steering*:** this entity refers to the functionality of steering traffic between *ports*.

In terms of operations, all classes are considered to allow Create, Read, Update and Delete (CRUD) operations.

5.5. Proof-of-Concept

A Proof of Concept (PoC) environment has been deployed to showcase how a Telco can leverage the features proposed in this chapter.

5.5.1. Scenario

In this scenario, a NO exposes a SF Store service via a Web Portal, similar to what happens today in the mobile application world. This concept is refered as SFaaS, and for example purposes and sake of simplicity, only CPE related functions are refered in this paper. The CPE represents a collection of network SFs, such as routing, firewall, NAT, IPsec VPN server, and DPI. Through the SF Store, enterprise customers can acquire CPE functions and associate them to their premises. The costumer account and information (e.g. sites, connection of sites with PoPs) is considered to be set prior to the first access to the store.

Currently, the ability to perform SFC is not exposed to the end-user. The user requests CPE SFs that already have a pre-determined relation with other SFs, and associates them to one of his sites. The instantiation and configuration of the SFs is done in a few minutes and the user is able to control them through a dedicated SF management portal.

The testbed in place is the same depicted in Figure 3-11, focusing on the PoP setup that is further ahead detailed. In this case, the core network connects to two DC premises (managed by OpenStack IceHouse release with traffic steering functionalities), where one represents a centralized DC and the other a PoP. Finally, there is a customer premises represented by switching equipment, which is logically connected to the PoP over an access network. This latter network is a simple switch based network.

Note that the use of the SFaaS service requires the establishment of a basic business relation between the costumer and the Telco (costumer sites registered and with connectivity services). In other words, the user is a client of the Telco who provides connectivity services (e.g. fiber, copper) from the client's sites (e.g. house, enterprise premises) to the Telco network. Each site connection is considered to be directly

associated (physically or logically) to a Telco PoP. On the site side a L2 device (or a L3 device in bridge mode) is considered to be in place.

Currently, from a demonstration story viewpoint it is considered that the user, after having the physical connection in place, must first buy a base CPE function with routing, DHCP and NAT functionalities (the PoC relies on the OpenStack L3 native device). From now on, the user can acquire other CPE functions and services, e.g.: Internet connection, Firewall (PoC relies on *iptables*), VPN Server (PoC relies on *OpenVPN*), NAS (PoC relies on *Samba*) and others. When the base routing function is in place, the user can access the CPE overall management system (in our case a web portal). Afterwards, other CPE functions can be acquired in the SF Store. These will be then made available in the CPE management system. At any time the costumer can delete the previously acquired functions. Finally, SF scheduling is also supported, making it possible to have a start and end date of a specific function.

5.5.2. Technologies and Implementation

From a technology viewpoint, the PoC relies on multiple technologies, some already existent and used as they are, others adapted, and others built from scratch due to their novelty. In this section we provide some details about the technologies used per domain.

All the software developments done rely on the *python* programming language. Furthermore, in the orchestrator module, the Bottle library [214] is used to implement the RESTful Application Programming Interface (API). Finally, CouchDB [215] and MySQLdb [216] libraries are used to connect to the database servers.

5.5.2.1. Service, VNF and Infrastructure description

This module is composed of four MySQL databases:

- Client Database - stores the client information (services and profile)
- Infrastructure (cloud, WAN, sites) Database – stores access credentials to the various CMSs and NO Management Systems (NOMSs), infrastructure topology, and resource information (i.e. available and used).
- SF Frontend Database – stores the list of available SFs in the platform (along with some high-level description) and Sf service registry (i.e. SFs deployed).
- SF Backend Database – stores specific SF information: SF instances to infrastructure resource mapping; SF mapping to correspondent recipes.

Recipes are hosted in a CouchDB database. These comprise the requirements in terms of software and SF configuration. Figure 5-9 shows the example of a simple recipe, where the *Requirements* part refers to the software required to be installed, and the *Files* part refers to the files required to be modified.

```
"Requirements":
{"App":
    [{
        "Name":"application name",
        "Apk":"repository application name"
    }]}
"Files":
{"Copy":
    [{
        "file":"remote file location",
        "location":"destination location"
    }}
```

Figure 5-9: CPE Recipe example

5.5.2.2. Orchestrator

The Cloud4NFV orchestrator prototype was developed using the *python* language, which interacts with both the NOMS and CMS. With this setup we are able to automatically provision SFs (that are in the Cloud4NFV orchestrator repository) across multiple DC locations with guaranteed WAN connectivity.

The Orchestrator exposes through its RESTful API all SFs in the platform and allows their deployment. Although the second tier SFs require a first tier SF, they are treated individually by the orchestrator. This means that they are mapped in a unique URL in the REST API.

Table 5-1: First tier SF REST API attributes

Attribute	Type	Default Value	Required	CRUD
id	uuid	generated	Y	R
name	string	none	N	CRU
description	string	none	N	CRU
tenant_id	uuid	from Auth token	Y	CR
location	uuid	none	Y	CRU

Table 5-1 shows the main attributes (type, default values, if they are mandatory attributes and the operations allowed over each attribute) of the RESTful API associated to a first tier SF, while Table 5-2 shows the same for second tier SFs. The *id* attribute is a unique identifier generated by the orchestrator upon creation of the SFs. *Name* and *description* are optional attributes that help characterize the SF in a human-readable way. The *tenant_id* identifies the client within the system. In first tier SFs, the *location* attribute is used to specify the premises where the SF will be deployed (i.e. which PoP or on-site PoP). Second tier SFs do not have this latter attribute since they are deployed over an existing first tier SF. However they have the *tier1_SF* attribute that identifies the first tier SF over which they will be deployed.

Table 5-2: Second tier SF REST API attributes

Attribute	Type	Default Value	Required	CRUD
id	uuid	generated	Y	R
name	string	none	N	CRU
description	string	none	N	CRU
tenant_id	uuid	from Auth token	Y	CR
tier1_SF	uuid	none	Y	CR

Regarding the connection between the Orchestrator and SF Managers, its implementation is plugin based. When the Orchestrator receives a request to deploy a service, it calls the SF Manager and includes the necessary arguments. The Orchestrator accesses the Client DB, Infrastructure Database and SF Store Database.

The communication to the OpenStack platforms and WAN is done via the CNMS.

5.5.2.3. SF Manager for Costumer Premises Equipment

In this case, there is a single SF Manager to administer the lifecycle of all virtual CPE functions. Currently, for PoC purposes only the instantiation and termination stages of the lifecycle are supported. All CPE functions detailed in this PoC rely on the Alpine Linux, which is a security-oriented, lightweight operating system “designed for x86-routers, Firewalls, VPNs, VoIPs and servers” [217]. Another interesting feature is the embedded EMS exposed in a web-portal to configure the operating system and its applications.

To deploy the first tier SFs, the SF Manager relies on the Orchestrator’s Resource Mapping module that then interacts with the cloud VIM and WAN VIM. The second tier SFs are deployed on top of the first tier

SFs. The SF Manager uses SSH to send the commands and configuration files necessary for their instalment within Alpine Linux.

Finally, note that the SF Manager has access to the Infrastructure Database and SF Backend Database.

5.5.2.4. Cloud Management System

Although the cloud model may require, to a large extent, the redefinition of SFs and the way they are managed, SFs also require adaptation from today's cloud solutions to cope with their requirements, especially in terms of networking features. A clear evidence of this fact is the OpenStack project, a reference open-source cloud management platform, which has been witnessing a tremendous evolution of its networking features - in its networking project mostly known by the codename *Neutron*. It is also important to note that *Neutron* provides the network service logics, and relies on different backends called "drivers" to interact with different networking technologies. Among these drivers is a recent one for the OpenDaylight Software Defined Networking (SDN) controller. OpenDaylight is today seen as an initiative equivalent to OpenStack in the SDN domain. With this in mind, the CMSs in place are based on OpenStack and OpenDaylight.

5.5.2.4.1 OpenStack

From a networking perspective, OpenStack today allows to create and manage VNs (L2 network segments) and ports (attachment points for devices connecting to networks, e.g. VN Interface Cards (vNICs) in VMs). The OpenStack community has been doing a considerable effort on keeping up with users' demand on introducing new *Neutron* network service types - L3 routing, firewall as a service (FWaaS), Load Balancer as a service (LBaaS) and VPN as-a-Service (VPNaaS); however, it is unfeasible in the long run to keep up with demands at this pace in a timely manner. Therefore, we argue that OpenStack should also offer the basic tools for network services to be orchestrated at a higher level and be deployed as VMs.

With the orchestration and composition of SFs in mind, it is easy to identify the need to fill a gap in OpenStack, the one of being able to steer traffic between OpenStack elements (e.g. VMs, routers). We envision a new OpenStack service abstraction that allows steering traffic between *Neutron* "ports" according to classification criteria. The current definition of the abstraction introduces new entities into the OpenStack *Neutron* data model: *Port Steering*, and *Classifier*. Both entities have a set of common OpenStack data model attributes, i.e. *id*, *name*, *description*, and *tenant_id*. The *Port Steering* adds to this common set a list of ports (*ports* attribute), and a list of classifiers (*classifiers* attribute). The former lists the sets of ports that must be targeted of classification and then steered. The *Classifier* entity adds the following attributes: *type*, *protocol*, *port_min*, *port_max*, *src_ip* and *dst_ip*. The purpose of this latter class is further explained ahead. As one will see, this functionality is very useful as it provides the means to realize, among other things, SFC. How the SFC is actually performed is further explained below. More details about the traffic steering functionality can be found in the OpenStack proposal [218], for which we developed a prototype implementation.

5.5.2.4.2 OpenDaylight

OpenDaylight has a module that integrates with OpenStack *Neutron* for the actual enforcement of services in the infrastructure. This module was extended in order to support and enforce the previously refered OpenStack traffic steering feature. It is important to highlight that this implementation relies on the OpenFlow and the Open vSwitch (OVS) Database Management Protocol (OVSDB) for the management of network resources.

5.5.2.5. Point of Presence Setup

Figure 5-10 provides an overview of the PoC prototype setup with the four functions abovementioned as example. It is given special attention to the setup at the PoP level.

Each customer has a dedicated virtual private environment in the PoP that is serving his site. This environment allows the creation of VNs and VMs (in OpenStack this is known as *tenant* or *project*). There is a point to point logical connection between the customer's premises (L2) device (currently we are using Virtual Local Area Network (VLAN) encapsulation to establish this connection, but others can be used). On

the PoP side this logical connection is extended to a VN in the tenant virtual environment – in the figure "Site Network", which has the private IP range of 192.168.1.0/24 (in OpenStack we use the *provider network* concept to achieve this). Moreover, there is a VN that is shared among all tenants, which in the figure is the "Internet Network" (in OpenStack this is achieved using the *external network* concept). This latter network is then connected to the core network which provides the Internet access. Also depicted in the figure is the "Inter-DC Network" that provides access between the PoP and the DC over a VPN service in core network (also here on the PoP side we rely on the OpenStack *provider network* concept). What is been explained so far is considered to be in place as soon as the customer establishes the basic business relation with the Telco (costumer sites registered and with connectivity services).

All functions when deployed upon request are connected to the "Site Network". When an Internet connection is requested, the base CPE is connected to the "Internet Network" and configured to perform NAT. In the figure is also highlighted a SFC that comprises the base CPE, VPN server and firewall.

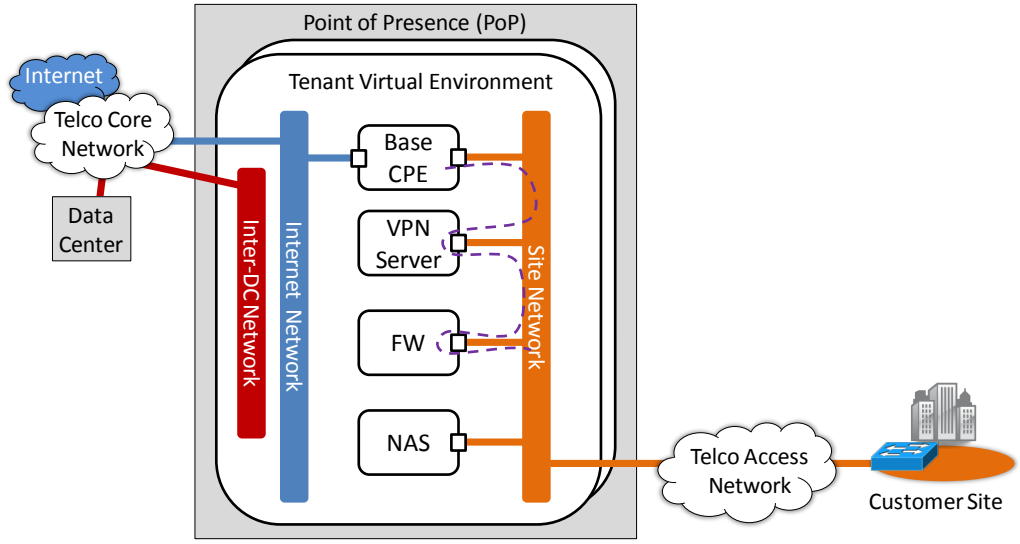


Figure 5-10: PoC prototype setup

5.5.2.6. Network Operator Management System

The NOMS used is the one presented in chapter 3.

5.6. Summary

This chapter explored the NFV concept, focusing on the orchestration and management of SFs. In this sense, leveraging the platform initially presented in chapter 3, the Cloud4NFV platform was presented. This platform is able to orchestrate and manage the lifecycle of virtual SFs. Moreover, special attention was given to the modeling of SFs towards virtual resources and to the combination of SFs, i.e. SFC.

Finally, a PoC that showcases how the platform and principles presented can be leveraged in a Telco environment was described.

6. Conclusion and Future Work

This chapter resumes the research achievements resulting from the work developed in the scope of this Thesis. We first present how the proposed research objectives were accomplished. Then, due to the enterprise related nature of this Thesis, we also highlight the most significant achievements for Portugal Telecom (PT) Inovação e Sistemas (PTInS). Following, the main conclusions about the impact of the performed work are presented. In the end, we point out possible directions for future research, addressing topics that were out of the scope of this Thesis, but given recent communication trends, they are an added value if integrated in our work.

6.1. Achievements on the Research Objectives

The main question addressed in this Thesis is how to integrate the cloud computing paradigm with the Network Operator's (NO) infrastructure. This main problem was decoupled into 4 research objectives in chapter 1, which were defined from the outlined issues in the current integration model. In this section, conclusions of those research objectives are presented:

Research Objective 1: How should cloud computing and NO domains be integrated?

- In a scenario where cloud computing and NOs are treated as distinct administrative domains, we have specified a multi-domain architecture that allows the integration of both domains, and the provision of end-to-end services with the respective Quality of Service (QoS). The architecture is flexible enough to allow a more centralized cloud approach (e.g. one or two cloud locations - Data Centers (DCs)), as well as a distributed approach (i.e. with several cloud locations throughout the network). We elaborate how the coordination across domains should be accomplished and present a mechanism for the on-demand connectivity negotiation. This latter mechanism plays a fundamental role in the architecture, as it guarantees the technology independency of the different domains. Furthermore, we studied and detailed on the most relevant aspects of this integration, with special focus on the management aspects. The outcome from this work fills in one of the biggest gaps in cloud computing today by unlocking the effective integration of cloud computing and NO domains and allowing the establishment of real end-to-end services. The features provided by this solution allow for the creation of added value services on top of the network that could not be done before.

Research Objective 2: How should the NO evolve to accommodate cloud properties?

- The integration of cloud computing and NO includes requirements on how the NO should adapt to such environment. With the identified need for the NO to provide the establishment of

network connectivity services in the Wide Area Network (WAN) in a cloud fashion way, we defined a generic architecture for the management of these services. In this scope, two detailed solutions were provided: one on a short-term perspective, relying on legacy technologies; and another on a more long-term perspective, relying on Software Defined Networking (SDN) technologies. The outcome demonstrates that the dynamic establishment of network connectivity services in the WAN can be accomplished whether by adapting legacy solutions or by taking advantage of more recent ones (i.e. SDN). Naturally, there are pros and cons on the solution chosen. We have demonstrated that the adaptation of legacy technologies brings their inherent limitations, which is their low time performance. The positive side is the fact that they are already well established solutions, which make them a more suitable approach in a short-term perspective. SDN technologies on the other hand present better time performance and provide a more clean approach as they can be easily programmed and adapted. The drawback of SDN solutions today is its lack of deployment in the WAN. The use of SDN technologies is becoming more and more common within DC environments; however, in the WAN side this has not happened yet, mostly due to its extremely high impact and cost for NOs.

Research Objective 3: How should cloud and network resources be embedded in the best way?

- When embedding virtual resources in cloud environments, guaranteeing the requested Service Level Agreements (SLA) and minimizing the embedding costs (e.g. minimize resources used to fulfil the request, minimize energy consumption of the physical infrastructure) are two major concerns. In this sense, Virtual Infrastructure (VI) embedding mechanisms that take into account the refered aspects should be available. We developed a set of resource management mechanisms to allow the embedding of complete VIs (i.e. with compute, storage, and network resources) taking into account the abovementioned aspects. The proposed mechanisms rely on heuristic and optimal approaches. Heuristics are more time-efficient, while optimal ones provide better solutions and a way to assess heuristic approaches. We explored single domain and multi-domain scenarios. For the single domain case, we presented heuristic and optimal strategies for balancing the load of the physical infrastructure. Moreover, we elaborated further on the optimal approaches and presented strategies for minimizing the physical infrastructure energy consumption. Finally, we explored the ability to perform adaptation/re-optimization of VIs in order to optimize the overall embedding. For the multi-domain case we provided an optimal solution for the VI embedding problem, and explored how the location factor in a distributed cloud can impact the VI embedding. The outcome of this work demonstrates that the embedding of VIs can be done in a way that the acceptance ratio is improved while at the same time the embedding costs associated to the physical infrastructure energy consumption can be reduced. This is definitely a win-win situation in the VI embedding process. These benefits can be further improved by applying adaptation and reconfiguration processes over already embedded VIs. Furthermore, we have demonstred that in multi-domain cases the location factor has a high impact on the VI Embedding (VIE) process, especially in terms of acceptance performance.

Research Objective 4: How should the cloud computing and NO integration evolve in order to fulfil Network Functions (NFs) Virtualization (NFV) requirements?

- Layng upon the results of the previous research objectives, we defined a platform to support, manage and compose virtual Service Functions (SFs). With the purpose of enabling the management of SFs in cloud environment, we defined a data model for SFs towards VI resources. We then further elaborated on the SF composition topic and provided a solution to perform SF Chaining (SFC). As emphasized earlier, it is not only the Telco domain that has to evolve to accommodate cloud requirements, but also the cloud has to evolve in order to accommodate NFV/Telco requirements. In this sense, we have defined a new cloud service abstraction, one that enables to realize traffic steering in a virtual cloud environment (a key

functionality for building other services, like SFC). It is also important to note that this functionality is scheduled to be supported in the OpenStack platform in its next release (codename *Kilo*). The outcome from this work demonstrates how NOs can take adavantge of cloud and SDN technologies to host their own network SFs and also provide them to third parties. It also points out solutions for some of the key aspects that need to be improved in the cloud domain in order for it to be able to support network SF requirements, i.e. ability to perform SFC. The outcome of this work is also an example of how the results of research objective 1 and research objective 2 can be used to create added value services.

6.2. Achievements for PTInS

In this section we highlight how the research outcomes of this Thesis impacted PTInS. PTInS, as the research and innovation branch of Group PT, plays a decisive role in the ongoing evolution of PT as a NO as well as Cloud Provider (CP). In this sense, the most obvious and important achievement was the know-how that was continuously absorbed by PTInS during the lifetime of the Thesis.

Moreover, the two new service concepts that were studied under the scope of this Thesis, Virtual Private Networks (VPNs) as a Service (VPNaaS) and SF as-a-Service (SFaaS), led to important outcomes within PTInS. The Proof of Concepts (PoCs) developed were advertised (with live demonstrations) in several events both within as well as outside the PT Group, such as the Future Network & Mobile Summit (FuNeMS) 2013 and the "O Melhor do Portugal Tecnológico". Figure 6-1 shows a screen shot of the PoC's client integrated cloud portal, where the services Infrastructure-as-a-Service (IaaS), VPNaaS and Virtual Network (VN) Function (VNF) as a Service (VNFaaS) are available.

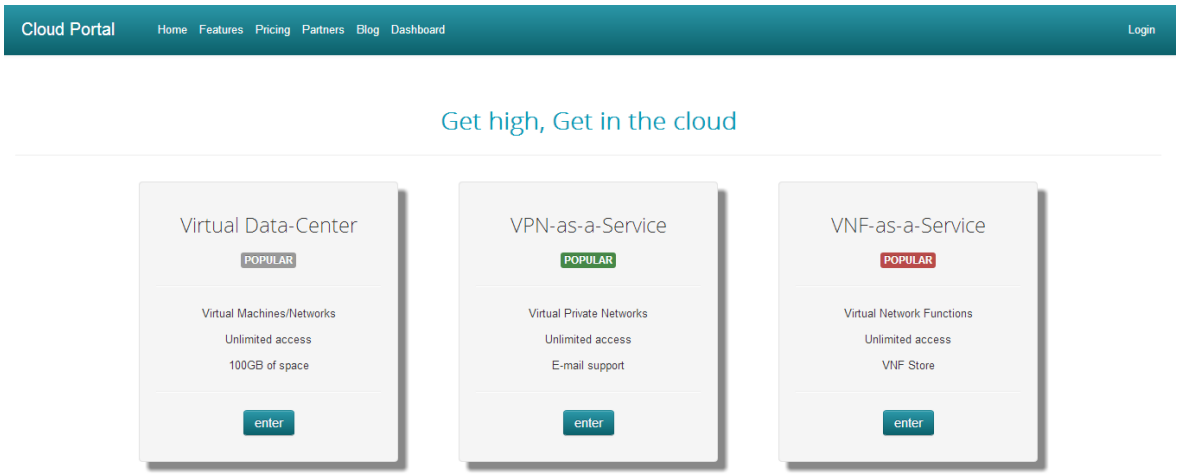


Figure 6-1: Integrated cloud service portal

Figure 6-2 and Figure 6-3 show two screen shots of the client web portal for the VPNaaS. In Figure 6-2, it is possible to observe the registered client sites: two are client premises locations, and one is a Virtual DC (VDC) – a virtual location that corresponds to the location where the cloud resources are hosted. Figure 6-3 shows a VPN which has already been provisioned (to connect the three refered locations), highlighting the QoS details in each endpoint. It is important to highlight that this service concept led to the creation of a PoC at the PT Group level, which gathered some of the most important vendors in the Telco industry. Furthermore, part of the concept has already been adopted. This consists in giving costumers, that have VPN services connecting cloud services, the ability to configure in a cloud fashion way (i.e. self-service and on-demand) the network QoS parameters in the VPN endpoint associated to the cloud resources (i.e. the VDC in Figure 6-2).

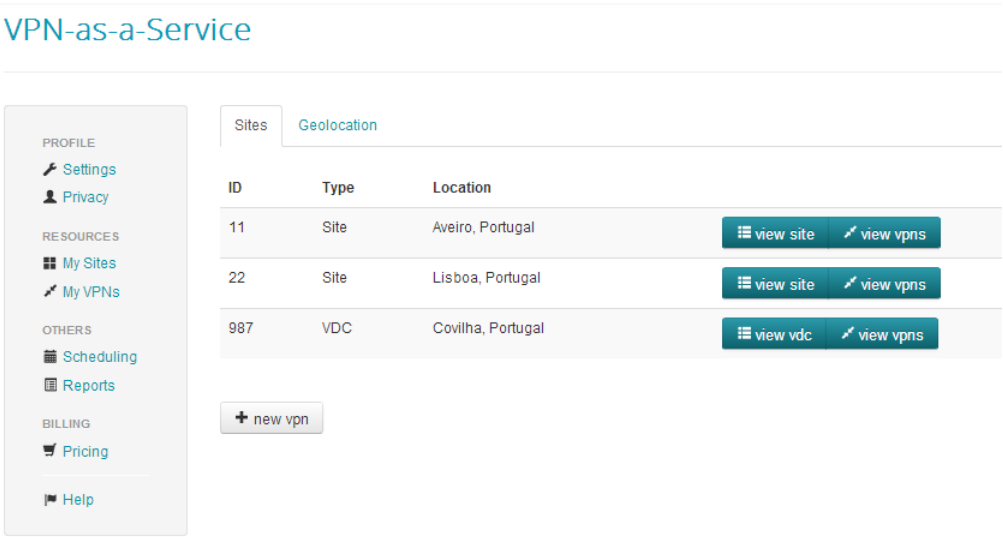


Figure 6-2: Integrated cloud service portal – VPNaaS 1

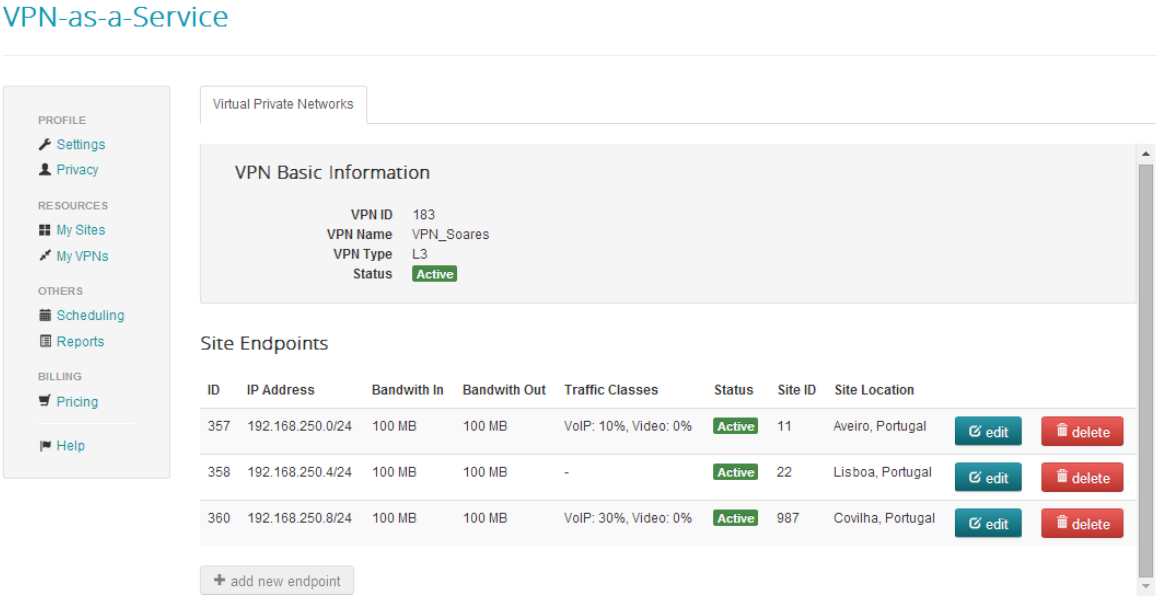


Figure 6-3: Integrated cloud service portal – VPNaaS 2

With respect to the VNFaaS, Figure 6-4 and Figure 6-5 show screen shots of the service's web portal. The service portal is divided in two main parts: one which is the VNF Store, Figure 6-4, where the client can see the list of available SFs and buy them; another which is the client's SF "park", where he can see and manage the list of SFs that he owns. Figure 6-5 shows the process of acquiring a base CPE function (with routing, DHCP, firewall and NAT). When acquiring a SF, the client can schedule for an immediate deployment or specify data and time. Also note that, when buying CPE functions, these have to be associated to a site (in the example, the site of Aveiro).

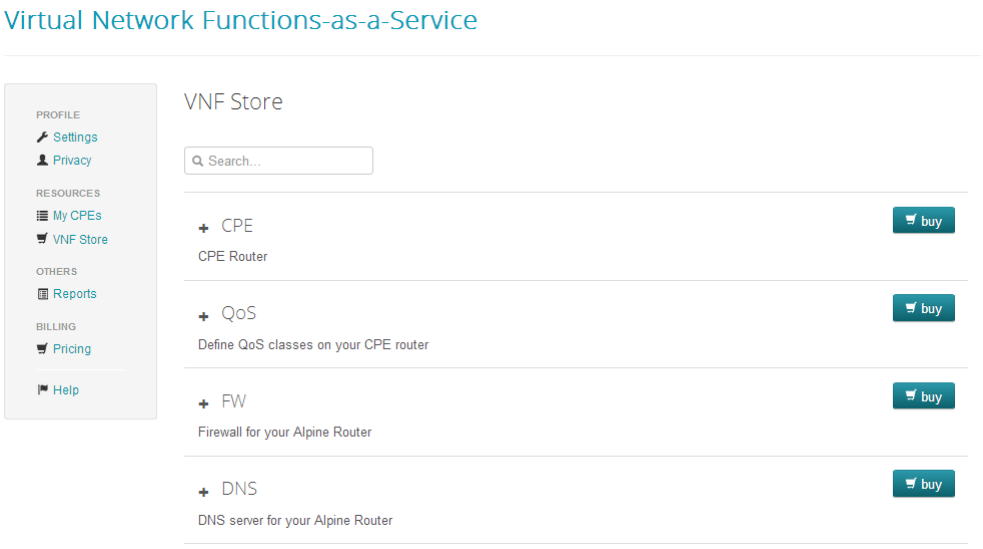


Figure 6-4: Integrated cloud service portal – VNFaaS 1

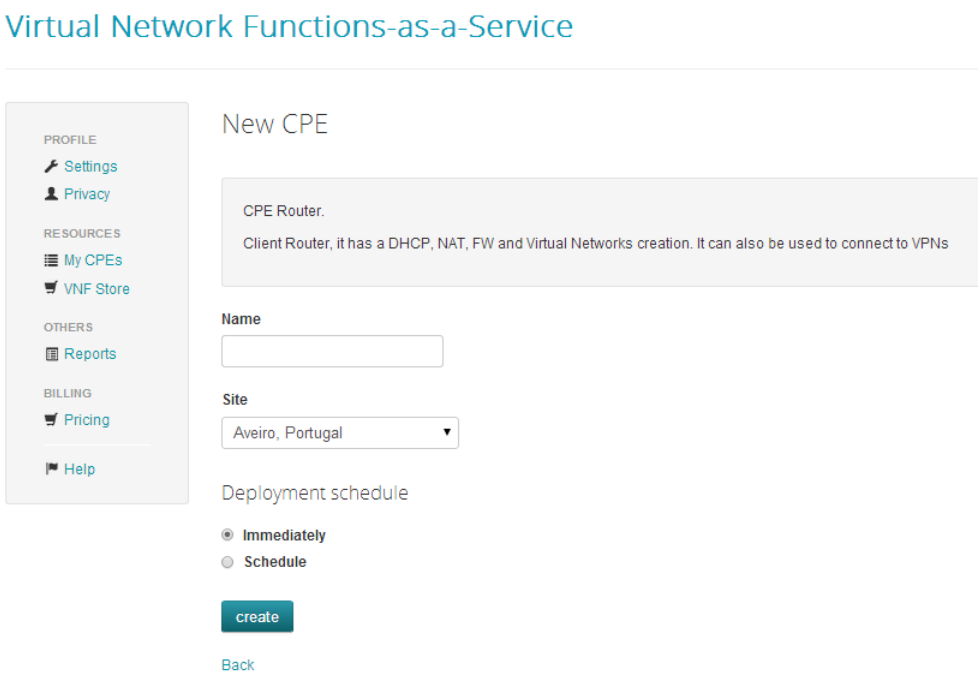


Figure 6-5: Integrated cloud service portal – VNFaaS 2

6.3. Final Remarks

In the scope of this Thesis we defined, studied and evaluated (through different means) a multi-domain architecture along with a set of mechanisms for the integration of the cloud computing paradigm with the Operator's Network. The architecture guarantees independency from the actual business model as the interaction between domains is purely service interface based. Another key aspect of this architecture is the inter-domain connectivity protocol, which guarantees the technology independency of domains. Moreover, we studied and defined a high-level architecture for handling dynamic WAN connectivity services. In more detail, we provided two solutions for this architecture, one laying on legacy network technologies, and another layng on SDN technologies.

The resource management of virtual resources, more specifically the embedding process, was another major concern of this work. The ability to handle requests for virtual resources in an optimal way was our ultimate goal, and in this sense we studied, defined and evaluated a set of strategies for the embedding of virtual resources. Part of the proposed solutions addressed single domain scenarios, while another tackled multi-domain scenarios. Heuristic and optimal approaches were proposed, although a more extensive study was done on the latter cases. The proposed strategies have into account several aspects, such as physical infrastructure load balancing, energy consumption, and the impact of reconfiguration processes.

Enabling the architecture to handle network SFs was another concern of our work. In this sense, we studied and defined a modeling solution for SFs towards physical infrastructure resources. Further, we extended our work to the ability to compose SFs, i.e. to perform SFC. In the scope of SFC, we defined a modeling solution along with an enforcement solution, by defining a traffic steering service abstraction for cloud management platforms.

Finally, and considering the validation provided through real PoCs, a proper integration of cloud computing and the Operator's Network will for sure take place. The concepts defined in this Thesis can help in this process, as we have proved them to be easily deployable in real environments without major modifications on current systems.

In a glance, this Thesis was able to demonstrate how cloud computing and the Operator's Network can be integrated in a flexible way, and how the NO can play a decisive role in the cloud computing world.

6.4. Future Research Direction

Considering the work developed in this Thesis and its broad research scope, there are several topics that may benefit from further development in order to cover all aspects in the cloud and Telco integration. These topics are presented next:

Wireless Environment

The proposed architecture is mostly targeted to the fixed network segment of the Operator's network. However, the integration cannot be confined to fixed networks and must be therefore extended to mobile network environments. This is an extremely important research topic that should be seriously addressed. A possible integration path could be by evolving the proposed architecture to integrate Internet Protocol (IP) Multimedia Subsystem (IMS) systems.

Continue improving the VI Embedding

As it was highlighted in this Thesis, the VI embedding problem is a very complex problem and there are still some aspects of the problem that should be targeted of further research. Among these aspects is the ability to handle VI elasticity and fault management (e.g. react and prevent failures) in the best possible way. We provided a complete and solid foundation in terms of optimal strategies and also did some research on the heuristic side. However, we believe that research on the heuristic side must be further elaborated to consider all the factors addressed in the optimal one, and also to bring the heurists performance closer to the optimal ones. Finally, depending on the application area (e.g. NFV), further research might be done to the formulations proposed to consider specific requirements.

Continue improving the SF Modeling

In this Thesis we provided a modeling solution for SFs towards VI resources. However, this is one side of the challenge. Among the others is the need to model higher-level functionalities of SFs, i.e. how to express what SFs actually do (e.g. express what a SF does to a packet). By doing this, one can, for example, better infer a SFC. In other words, when following a non-tagged approach, if active SFs express how they handle packets, it is easier for a system to understand how to steer traffic among a set of SFs.

SF Intelligent Embedding

This aspect relates to the second point, "Continue improving the VI Embedding". Each SF has specific characteristics and requirements, not only in terms of VI resources, but also in terms of higher-level requirements such as: the location of other specific SFs; the location according to the users it is serving. Therefore, the optimal placement of one SF (e.g. a FW) is not the same for all SFs (e.g. Access Network

Discovery and Selection Function (ANDSF), Packet Data Network Gateway (PGW)) just by looking to the virtual resources it needs. In this sense, we argue that embedding SFs is a research that we will see being addressed in a massive way in a near future.

References

[1] D. Evans, “The internet of everything: How more relevant and valuable connections will change the world", 2013. [Online]. Available: http://www.cisco.com/web/about/ac79/docs/innov/IoE.pdf. [Accessed: September 29, 2014].

[2] P. Mell, T. Grance, “The NIST Definition of Cloud Computing”, National Institute of Standards and Technology (NIST) , Technical report, 800-145 , Gaithersburg, MD, 2011.

[3] E. Bugnion, S. Devine, M. Rosenblum, J. Sugerman, and E. Y. Wang, “Bringing Virtualization to the x86 Architecture with the Original VMware Workstation”, ACM Transactions on Computer Systems 30, 4, Article 12, 51 pages, November 2012.

[4] V. Moreno and K. Reddy, “Network Virtualization”, Cisco Press, 2006.

[5] F. Baroncelli, B. Martini, P. Castoldi, “Network virtualization for cloud computing”, Annales des Télécommunications 65 (11-12): 713-721, Springer, 2010.

[6] Open Networking Foundation: “Software-Defined Networking: The New Norm for Networks”, white paper, April, 2012.

[7] M. L. Badger, T. Grance, R. Patt-Corner, and J. M. Voas, “Cloud Computing Synopsis and Recommendations”, Technical Report, NIST, Gaithersburg, MD, United States, 2012.

[8] J. Cook and A. Hutchinson, “Network and Operations Planning for Telecommunications: Getting the Edge in the Network-Expansion Boom”, Accenture, 2012.

[9] M. Sapien, “Cloud–network integration is coming to global cloud services”, January 2014. [Online]. Available: http://www.cloudcomputing-news.net/news/2014/jan/06/cloudnetwork-integration-is-coming-to-global-cloud-services/. [Accessed: September 29, 2014].

[10] Pwc, “Cloud-enabled telco opportunities”, April 2013.

[11] P. Murray, A. Sefidcon, R. Steinert, V. Fusenig, J. Carapinha, "Cloud networking: An infrastructure service architecture for the wide area," Future Network & Mobile Summit (FuNeMS) 2012 , vol., no., pp.1,8, 4-6 July 2012.

[12] ETSI, “Network Functions Virtualisation (NFV): An Introduction, Benefits, Enablers, Challenges & Call for Action”, White Paper, October 2012.

[13] Sophia Antipolis, “Leading operators create ETSI standards group for network functions virtualization”, 22 January 2013. [Online]. Available: http://www.etsi.org/news-events/news/644-2013-01-isg-nfv-created. [Accessed: September 29, 2014].

[14] Akami, “The state of the internet, 2nd quarter 2013 report", September 2013. [Online]. Available: http://www.akamai.com/dl/documents/akamai_soti_q213.pdf?WT.mc_id=soti_Q213 [Accessed: September 29, 2014]..

[15] J. Soares, J. Carapinha, M. Melo, R. Monteiro, S. Sargento, "Building Virtual Private Clouds with Network-aware Cloud", The Fifth International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP) 2011, Lisbon, Portugal, November 2011.

[16] J. Soares, R. Monteiro, S. Sargento, J. Carapinha, M. Melo, “Resource Allocation in the Network Operator's Cloud: A Virtualization Approach”, Second Workshop on Management of Cloud Systems (MoCS 2012), in the Seventeenth IEEE Symposium on Computers and Communications (ISCC) 2012, Cappadocia, Turkey, July 2012.

[17] H. Puthalath, J. Soares (joint first author), B. Melander, et. al, “Negotiating On-Demand Connectivity between Clouds and Wide Area Networks”, The first IEEE International Conference on Cloud Networking (CloudNet) 2012, Paris, France, November, 2012.

[18] R. Gouveia, J. Aparicio, J. Soares, et. al “SDN Framework for Connectivity Services”, IEEE International Conference on Communications (ICC) 2014, 10-14 June, 2014.

[19] J. Soares, S. Sargento, “Optimizing the Embedding of Cloud and Network Virtual Infrastructures”, The twenty first IEEE International Conference on Telecommunications (ICT) 2014, Lisbon, Portugal, 4-7 Maio, 2014.

[20] J. Soares, B. Parreira, M. Dias, J. Carapinha, S. Sargento, “Cloud4NFV: A Platform for Virtual Network Functions”, The third IEEE International Conference on Cloud Networking (CloudNet), Luxembourg, 8-10 October, 2014.

[21] J. Soares, S. Sargento, "Optimizing the Embedding of Virtualized Cloud Network Infrastructures across Multiple Domains", submitted to IEEE International Conference on Communications (ICC 2014), 9-12 June 2015.
[22] J. Soares, J. Aparício, S. Sargento, "Optimizing the Embedding of Virtualized Cloud Network Infrastructures", submitted to IEEE Transactions on Cloud Computing, June 2014.
[23] J. Soares, C. Gonçalves, et. al, "Towards a Telco Cloud Environment for Service Functions", submitted to IEEE Communications Magazine, special issue on Network and Service Virtualization, June 2014.
[24] J. Soares, S. Sargento, "Re-Optimizing the Embedding of Virtualized Cloud Network Infrastructures", submitted to Transactions on Cloud Computing, October 2014.
[25] J. Soares, R. Monteiro, et. al, "The Cloud inside the operator's network: Resource allocation", Communication Infrastructures for Cloud Computing: Design and Applications, IGI Global, September 2013.
[26] P2051 - Opportunities and Challenges for Operator in the Mobile Cloud, Eurescom, 2011.
[27] SAIL - Scalable & Adaptive Internet Solutions, EU FP7 Project. [Online]. Available: http://www.sail-project.eu/.
[28] Mobile Cloud Networking, EU FP7 Project. [Online]. Available: http://www.mobile-cloud-networking.eu/site/. [Accessed: September 29, 2014].
[29] CloudAnchor, Portuguese Quadro de Referência Estratégica Nacional (QREN) Project.
[30] T-NOVA - Network Functions as-a-Service over Virtualised Infrastructures, EU FP7 Project. [Online]. Available: http://www.t-nova.eu/.
[31] R. Monteiro, "Criação e Reconfiguração de Redes Virtuais na Perspectiva do Operador", Masther's Thesis, Universidade de Aveiro, 2011.
[32] B. Parreira, "Integração da Cloud com Rede na Perspectiva do Operador", Masther's Thesis, Universidade de Aveiro, 2012.
[33] R. Gouveia, "Demonstrador de uma rede com tecnologia OpenFlow", Masther's Thesis, Universidade de Aveiro, 2013.
[34] J. Aparicio, "Integração da Cloud com a rede do Operador", Masther's Thesis, Universidade de Aveiro, 2013.
[35] G. Pardal, "Demonstrador de uma Rede de Operador com Tecnologia OpenFlow e Serviços na Cloud", Masther's Thesis, Universidade de Aveiro, 2014.
[36] L.M. Vaquero, L.Rodero-Merino, J. Cáceres, M. Lindner, "A Break in the Clouds: Towards a Cloud Definition", ACM Computer Communication Reviews, 2009.
[37] J. Carapinha, "Network Virtualisation – Opportunities and Challenges", Eurescom Study Report P1956, December, 2010.
[38] Amazon Web Services. [Online]. Available: http://aws.amazon.com/. [Accessed: September 29, 2014].
[39] OpenStack. [Online]. Available: http://www.openstack.org/. [Accessed: September 29, 2014].
[40] OpenNebula. [Online]. Available: http://www.opennebula.org/
[41] OpenStack, OpenStack Instalation Guide for Ubuntu 12.04/14.04 (LTS). [Online]. Available: http://docs.openstack.org/icehouse/install-guide/install/apt/content/. [Accessed: September 29, 2014].
[42] Libvirt, The virtualization API. [Online]. Available: http://libvirt.org/. [Accessed: September 29, 2014].
[43] Kernel Based Virtual Machine. [Online]. Available: http://www.linux-kvm.org/. [Accessed: September 29, 2014].
[44] The XEN Project. [Online]. Available: http://www.xenproject.org/. [Accessed: September 29, 2014].
[45] VMware. [Online]. Available: http://www.vmware.com/. [Accessed: September 29, 2014].
[46] Microsoft Azure. [Online]. Available: https://azure.microsoft.com. [Accessed: September 29, 2014].
[47] RedHat OpenStack. [Online]. Available: http://www.redhat.com/openstack/. [Accessed: September 29, 2014].
[48] HP Helion Openstack. [Online]. Available: http://www8.hp.com/us/en/cloud/hphelion-openstack-overview.html. [Accessed: September 29, 2014].
[49] Alcatel Lucent CloudBand Platform. [Online]. Available: http://www.alcatel-lucent.com/solutions/cloudband. [Accessed: September 29, 2014].

[50] T. Wood, A. Gerber, K. Ramakrishnan, et al, “The Case for Enterprise- Ready Virtual Private Clouds”, HotCloud 09, 2009.
[51] T. T. Huu, G. Koslovski, F. Anhalt, P. Vicat-Blanc Primet, and J. Montagnat. “Joint elastic cloud and virtual network framework for application performance optimization and cost reduction”. Journal of Grid Computing (JoGC), pp. 27-47, 2010.
[52] F. Anhalt, G. Koslovski, and P. Vicat-Blanc Primet. “Specifying and provisioning virtual infrastructures with HIPerNET”. Int. J. Netw. Manag., pp. 129-148 May 2010.
[53] G. Koslovski, P.V.-B. Primet, A. S. Charão, “VXDL: Virtual Resources and Interconnection Networks Description Language”, in GridNets 2008.
[54] So, et al, “VPN Extensions for Private Clouds”, IETF Internet Draft, February 2011.
[55] Wen-Hwa Liao, Shuo-Chun Su, "A Dynamic VPN Architecture for Private Cloud Computing," Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on , vol., no., pp.409,414, 5-8 Dec. 2011
[56] D. Cai, S. Natarajan, "The Evolution of the Carrier Cloud Networking," Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on , vol., no., pp.286,291, 25-28 March 2013
[57] Hai Anh Tran, A . Mellouk, S. Hoceini, "QoE Content Distribution Network for Cloud Architecture," Network Cloud Computing and Applications (NCCA), 2011 First International Symposium on , vol., no., pp.14,19, 21-23 Nov. 2011
[58] Yu-Hunag Chu, Yao-Ting Chen, Yu-Chieh Chou, Min-Chi Tseng, "A simplified cloud computing network architecture using future internet technologies," Network Operations and Management Symposium (APNOMS), 2011 13th Asia-Pacific , vol., no., pp.1,4, 21-23 Sept. 2011
[59] T. Benson, A. Akella, A. Shaikh, and S. Sahu, “CloudNaaS: a cloud networking platform for enterprise applications”, In Proceedings of the 2nd ACM Symposium on Cloud Computing (SOCC’11). ACM, New York, NY, USA, Article 8, 13 pages.
[60] E. Keller and J. Rexford, “The “Platform as a service” model for networking”. In Proceedings of the 2010 internet network management conference on Research on enterprise networking (INM/WREN’10). USENIX Association, Berkeley, CA, USA, 4-4.
[61] H. Medhioub, B. Msekni, D. Zeghlache, "OCNI - Open Cloud Networking Interface," Computer Communications and Networks (ICCCN), 2013 22nd International Conference on , vol., no., pp.1,8, July 30 2013 - August 2 2013.
[62] Open Grid Forum, Open Cloud Computing Interface (OCCI). [Online]. Available: http://occi-wg.org/. [Accessed: September 29, 2014].
[63] C.R. Senna, M.A. Soares, L.F. Bittencourt, E. Madeira, "An architecture for adaptation of virtual networks on clouds," Network Operations and Management Symposium (LANOMS), 2011 7th Latin American , vol., no., pp.1,8, 10-11 Oct. 2011
[64] Qiang Duan, "Modeling and Performance Analysis on Network Virtualization for Composite Network-Cloud Service Provisioning," Services (SERVICES), 2011 IEEE World Congress on , vol., no., pp.548,555, 4-9 July 2011
[65] Q. Duan; Y. Yan; AV. Vasilakos, "A Survey on Service-Oriented Network Virtualization Toward Convergence of Networking and Cloud Computing," Network and Service Management, IEEE Transactions on , vol.9, no.4, pp.373,392, December 2012
[66] A. Tzanakaki, M.P. Anastasopoulos, S. Peng, et. al, "A converged network architecture for energy efficient mobile cloud computing," Optical Network Design and Modeling, 2014 International Conference on , vol., no., pp.120,125, 19-22 May 2014
[67] M. Hamze, N. Mbarek, O. Togni, "Self-establishing a Service Level Agreement within autonomic cloud networking environment," Network Operations and Management Symposium (NOMS), 2014 IEEE , vol., no., pp.1,4, 5-9 May 2014
[68] Kim-Khoa Nguyen; M. Cheriet, M. Lemay, "Enabling infrastructure as a service (IaaS) on IP networks: from distributed to virtualized control plane," Communications Magazine, IEEE , vol.51, no.1, pp.136,144, January 2013
[69] A. Mahimkar, A. Chiu, R. Doverspike, et al. “Bandwidth on demand for inter-DC communication”, In Proceedings of the 10th ACM Workshop on Hot Topics in Networks (HotNets ’11). ACM, New York, NY, USA, Article 24 , 6 pages, 2011.

[70] N. Bitar, S. Gringeri, T.J. Xia, "Technologies and protocols for DC and cloud networking," Communications Magazine, IEEE , vol.51, no.9, pp.24,31, September 2013
[71] I. Fajjari, N. Aitsaadi, G. Pujolle, "Cloud networking: An overview of virtual network embedding strategies," Global Information Infrastructure Symposium, 2013 , vol., no., pp.1,7, 28-31 Oct. 2013
[72] M. Fiorani, S. Aleksic, P. Monti, J. Chen, M. Casoni, L. Wosinska, "Energy efficiency of an integrated intra-data-center and core network with edge caching," Optical Communications and Networking, IEEE/OSA Journal of , vol.6, no.4, pp.421,432, April 2014
[73] Mon-Yen Luo; Jun-Yi Chen, "Software Defined Networking across Distributed Datacenters over Cloud," Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on , vol.1, no., pp.615,622, 2-5 Dec. 2013
[74] M. Mechtri, I. Houidi, W. Louati, D. Zeghlache, "SDN for Inter Cloud Networking," Future Networks and Services (SDN4FNS), 2013 IEEE SDN for , vol., no., pp.1,7, 11-13 Nov. 2013
[75] G. Carella, T. Magedanz, K. Campowsky, F. Schreiner, "Network-aware Cloud Brokerage for telecommunication services," Cloud Networking (CLOUDNET), 2012 IEEE 1st International Conference on , vol., no., pp.131,136, 28-30 Nov. 2012
[76] G. Breiter, V.K. Naik, "A Framework for Controlling and Managing Hybrid Cloud Service Integration," Cloud Engineering (IC2E), 2013 IEEE International Conference on , vol., no., pp.217,224, 25-27 March 2013
[77] T. Taleb, "Toward carrier cloud: Potential, challenges, and solutions," Wireless Communications, IEEE , vol.21, no.3, pp.80,91, June 2014
[78] Hyunseok Chang; A. Hari, S. Mukherjee, T.V. Lakshman, "Bringing the cloud to the edge," Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on , vol., no., pp.346,351, April 27 2014-May 2 2014.
[79] GEYSERS - GEneralised architecture for dYnamic infrastructure SERviceS, EU FP7 Project. [Online]. Available: http://www.geysers.eu/. [Accessed: September 29, 2014].
[80] UNIFY - Unifying Cloud and Carrier Networks, EU FP7 Project. [Online]. Available: https://www.fp7-unify.eu/. [Accessed: September 29, 2014].
[81] R. P Goldberg, “Survey of virtual machine research”, IEEE Computer, 7(6):34-35, 1974.
[82] IEEE, “802.1q – virtual lans”, August 2011. [Online]. Available: http://ieee802.org/1/pages/802.1Q.html. [Accessed: September 29, 2014].
[83] IEEE, "802.1ad – provider bridges”, May 2006. [Online]. Available: http://ieee802.org/1/pages/802.1ad.html. [Accessed: September 29, 2014].
[84] T. Sridhar, L. Kreeger, D. Dutt, et. al, “Vxlan: A framework for overlaying virtualized layer 2 networks over layer 3 networks”, 2013.
[85] M. Sridharan, M. Pearson, I. Ganga, et al., “Nvgre: Network virtualization using generic routing encapsulation”, 2013.
[86] QEMU (Quick EMUlator). [Online]. Available: http://www.qemu.org/. [Accessed: September 29, 2014].
[87] Virtualbox. [Online]. Available: https://www.virtualbox.org/. [Accessed: September 29, 2014].
[88] LXC (LinuX Container). [Online]. Available: https://linuxcontainers.org/. [Accessed: September 29, 2014].
[89] Ronald J Vetter, “Atm concepts, architectures, and protocols”, Communications of the ACM, 38(2):30–ff, 1995.
[90] E. Rosen, A. Viswanathan, R. Callon, et al., “Multiprotocol label switching architecture”, 2001.
[91] E. Rosen and Y. Rekhter, “BGP/MPLS IP Virtual Private Networks (VPNs)”, Technical report, RFC 4364, February 2006.
[92] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, “Resilient overlay networks”, volume 35. ACM, 2001.
[93] M. Hicks, P. Kakkar, J. T. Moore, C. A. Gunter, and S. Nettles, “Plan: A packet language for active networks”, In ACM SIGPLAN Notices, volume 34, pages 86–93. ACM, 1998.
[94] Open Network Foundation, “Software-Defined Networking: The New Norm for Network”, White paper, April 2012.
[95] M. Mendonca et al., “A Survey of Software-Defined Networking: Past, Present and Future of Programmable Networks”, HAL-Inria archive, May, 2013.

[96] Open Networking Foundation, Northbound Interfaces Working Group. [Online]. Available: https://www.opennetworking.org/working-groups/northbound-interfaces. [Accessed: September 29, 2014].
[97] Linux Foundation, “OpenDaylight: An Open Source Community and Meritocracy for Software-Defined Networking”, April, 2013.
[98] Open Networking Foundation, “OpenFlow Switch Specification version 1.4”, October 2013.
[99] Cisco Open Network Environment. [Online]. Available: http://www.cisco.com/c/dam/en/us/products/collateral/ios-nx-os-software/at_a_glance_c45-707979.pdf. [Accessed: September 29, 2014].
[100] Cisco Systems Inc., “Cisco’s One Platform Kit (onePK),” [Online]. Available: http://www.cisco.com/en/US/prod/iosswrel/onepk.html. [Accessed: September 29, 2014].
[101] VMWare, “VMware NSX Network Virtualization Design Guide,”. [Online]. Available: http://www.vmware.com/files/pdf/products/nsx/vmw-nsx-network-virtualization-design-guide.pdf. [Accessed: September 29, 2014].
[102] Floodlight Project. [Online]. Available: http://www.projectfloodlight.org/floodlight/. [Accessed: September 29, 2014].
[103] NOX Project. [Online]. Available: http://www.noxrepo.org/. [Accessed: September 29, 2014].
[104] R. Sherwood, G. Gibb, et al., “FlowVisor: A Network Virtualization Layer”, October 2009.
[105] U. Krishnaswamy, P. Berde, J. Hart, et al., “ONOS, Operating Network Operating System”, April 2013.
[106] Linux Foundation, OpenDaylight Technical Overview. [Online]. Available: http://www.opendaylight.org/project/technical-overview. [Accessed: September 29, 2014].
[107] Nuage Networks, “Nuage Networks,” [Online]. Available: http://www.nuagenetworks.net/. [Accessed: September 29, 2014].
[108] VMWare, “VMWare NSX, Network Virtualization,” [Online]. Available: http://www.vmware.com/products/nsx/resources.html. [Accessed: September 29, 2014].
[109] N. Feamster, J. Rexford, and E. Zegura, “The road to SDN: an intellectual history of programmable networks”, SIGCOMM Computer Communications Review 44, 2, 87-98, April 2014.
[110] J. Kempf et al., "Scalable fault management for OpenFlow", IEEE International Conference on Communications (ICC), 10-15 June, 2012.
[111] Y. Yu, C. Shanzhi, L. Xin and W. Yan, "A framework of using OpenFlow to handle transient link failure", Int. Conf. Transportation, Mechanical, and Electrical Engineering (TMEE), 16-18 December, 2011.
[112] P. Sharma et al., “Enhancing Network Management Frameworks with SDN-like Control”, IFIP/IEEE Int. Symp. Integrated Network Management (IM 2013), 27-31 May, 2013.
[113] H. Egilmez et al., "Scalable video streaming over OpenFlow networks: An optimization framework for QoS routing", 18th IEEE Int. Conf. on Image Processing (ICIP), 11-14 September, 2011.
[114] H. Egilmez, S. Civanlar and A. Tekalp: "An Optimization Framework for QoS-Enabled Adaptive Video Streaming Over OpenFlow Networks", IEEE Trans. on Multimedia, vol.15 (3), pp.710-715, April, 2013.
[115] B. Sonkoly et al., "OpenFlow Virtualization Framework with Advanced Capabilities", European Work. Software Defined Networking (EWSDN), 25-26 October, 2012.
[116] M. Banikazemi, D. Olshefski, A. Shaikh, J. Tracey, Guohui Wang, "Meridian: an SDN platform for cloud network services," Communications Magazine, IEEE , vol.51, no.2, pp.120,127, February 2013.
[117] IBM SmartCloud Provisioning. [Online]. Available: http://www-03.ibm.com/software/products/en/smartcloud-provisioning. [Accessed: September 29, 2014].
[118] H. Farhadi, P. Du, and A. Nakao, “User-defined actions for SDN”, In Proceedings of The Ninth International Conference on Future Internet Technologies (CFI '14). ACM, New York, NY, USA, , Article 3 , 6 pages, 2014.
[119] Z. A. Qazi, Cheng-Chun Tu, L. Chiang, et al., “SIMPLE-fying middlebox policy enforcement using SDN”, SIGCOMM Computer Communications Review 43, 4, 27-38, August 2013.

[120] NEC, “O3 Project launched for achieving the world’s first wide area SDN,” [Online]. Available: http://www.nec.com/en/press/201309/global_20130917_01.html. [Accessed: September 29, 2014].
[121] OpenEPC Project. [Online]. Available: http://www.openepc.net/. [Accessed: September 29, 2014].
[122] OpenIMS Project. [Online]. Available: http://www.openimscore.org/. [Accessed: September 29, 2014].
[123] ETSI, “Network Functions Virtualisation (NFV): Use Cases”, Technical Report ETSI GS NFV 001 v1.1.1, Oct. 2013.
[124] Alcatel-Lucent, “Network Functions Virtualization – Challenges and Solutions”, Strategic White Paper, 2013.
[125] ETSI, “Network Functions Virtualisation (NFV): Architectural Framework”, Technical Report ETSI GS NFV 002 v1.1.1, Oct. 2013.
[126] ETSI, “Network Functions Virtualisation (NFV): Terminology for main concepts in NFV”, Technical Report ETSI GS NFV 003 v1.1.1, Oct. 2013.
[127] ETSI, “Network Functions Virtualisation (NFV): Virtualization requirements”, Technical Report ETSI GS NFV 004 v1.1.1, Oct. 2013.
[128] ETSI, “Network Functions Virtualisation (NFV): Proofs of Concepts; framework”, Technical Report ETSI GS NFV-PER 002 v1.1.1, Oct. 2013.
[129] Ericsson Cloud System. [Online]. Available: http://www.ericsson.com/ourportfolio/products/cloud-system. [Accessed: September 29, 2014].
[130] Cloudify. [Online]. Available: http://getcloudify.org/. [Accessed: September 29, 2014].
[131] CloudNFV Project. [Online]. Available: http://www.cloudnfv.com/. [Accessed: September 29, 2014].
[132] ClearWater Project. [Online]. Available: http://www.projectclearwater.org/. [Accessed: September 29, 2014].
[133] H. Masutani, Y. Nakajima, T. Kinoshita, T. Hibi, H. Takahashi, K. Obana, K. Shimano, M. Fukui, "Requirements and design of flexible NFV network infrastructure node leveraging SDN/OpenFlow”, Optical Network Design and Modeling, 2014
[134] Intel Data Plane Development Kit, [Online]. Available: http://www.intel.com/content/dam/www/public/us/en/documents/presentation/dpdk-packet-processing-ia-overview-presentation.pdf. [Accessed: September 29, 2014].
[135] R. Riggio, T. Rasheed, F. Granelli, "EmPOWER: A Testbed for Network Function Virtualization Research and Experimentation," Future Networks and Services (SDN4FNS), 2013 IEEE SDN for , vol., no., pp.1,5, 11-13 Nov. 2013
[136] J. Martins, M. Ahmed, C. Raiciu, and F. Huici, “Enabling fast, dynamic network processing with clickOS”, In Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking (HotSDN '13). ACM, New York, NY, USA, 67-72, 2013.
[137] J. Hwang, K. K. Ramakrishnan, and T. Wood, “NetVM: high performance and flexible networking using virtualization on commodity platforms”, In Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation (NSDI'14), USENIX Association, Berkeley, CA, USA, 445-458, 2014.
[138] Hirochika Asai, “Where are the bottlenecks in software packet processing and forwarding?: Towards high-performance network operating systems”, In Proceedings of The Ninth International Conference on Future Internet Technologies (CFI '14). ACM, New York, NY, USA, Article 5 , 6 pages, 2014.
[139] Gartner says Cloud consumers need brokerages to unlock the potential of Cloud services, July 2009, Gartner. Retrieved, November 2010, from: [Online]. Available: http://www.gartner.com/it/page.jsp?id=1064712. [Accessed: September 29, 2014].
[140] Yong Zhu and M. Ammar, “Algorithms for assigning substrate network resources to virtual network components”, In INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings, pages 1–12, 2006.
[141] M. Yu, Y. Yi, J. Rexford, and M. Chiangl, “Rethinking virtual network embedding: substrate support for path splitting and migration”, SIGCOMM Computer Communications Review, 38(2):17–29, March 2008.

[142] N.M.M.K. Chowdhury, M.R. Rahman, and R. Boutaba, “Virtual network embedding with coordinated node and link mapping”, In INFOCOM 2009, IEEE, pages 783 –791, april 2009.

[143] J. Lu and J. Turner, “Efficient mapping of virtual networks onto a shared substrate”, Technical report, Washington University in St. Louis, 2006.

[144] M. Yu, Y. Yi, J. Rexford, and M. Chiang, “Rethinking virtual network embedding: substrate support for path splitting and migration”, SIGCOMM Comput. Commun. Rev., 38(2):17–29, March 2008.

[145] Hoboken, “Metaheuristics: from design to implementation”, John Wiley \& Sons, 2009.

[146] J. Nogueira, M. Melo, J. Carapinha, and S. Sargento, “Virtual network mapping into heterogeneous substrate networks”, In Computers and Communications (ISCC), 2011 IEEE Symposium on, pages 438 –444, 28 June 2011 - July 1 2011.

[147] IBM ILOG Optimization Products. [Online]. Available: http://www-03.ibm.com/software/products/en/category/decision-optimization. [Accessed: September 29, 2014].

[148] Andrew Makhorin, “Glpk (gnu linear programming kit)”. [Online]. Available: http://www.gnu.org/software/glpk/. [Accessed: September 29, 2014].

[149] M. Yu, Y. Yi, J. Rexford, M. Chiang, “Rethinking virtual network embedding: substrate support for path splitting and migration”, SIGCOMM Computer Communications Review, 38 (2), 17-29, 2008.

[150] N. Chowdhury, M. Rahman, R. Boutaba, “ViNEYard: Virtual Network Embedding Algorithms With Coordinated Node and Link Mapping”, IEEE/ACM Transactions on Networking, 20 (1), 206-219, 2012.

[151] Xiujiao Gao, Hongfang Yu, V. Anand, Gang Sun, and Hao Di, "A new algorithm with coordinated node and link mapping for virtual network embedding based on LP relaxation," Communications and Photonics Conference and Exhibition (ACP), 2010 Asia , vol., no., pp.152-153, 8-12 Dec. 2010.

[152] Ye Zhou, Yong Li, Depeng Jin, Li Su, Lieguang Zeng, "A virtual network embedding scheme with two-stage node mapping based on physical resource migration," Communication Systems (ICCS), 2010 IEEE International Conference on , vol., no., pp.761-766, 17-19 Nov. 2010

[153] A. Razzaq, M.S. Rathore, "An Approach towards Resource Efficient Virtual Network Embedding," Second International Conference on Evolving Internet (INTERNET), vol., no., pp.68-73, 20-25 Sept. 2010

[154] Jens Lischka and Holger Karl, “A virtual network mapping algorithm based on subgraph isomorphism detection”, In Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures (VISA '09). ACM, New York, NY, USA, 81-88, 2009.

[155] M. Melo, J. Carapinha, S. Sargento, L. Torres, P.N. Tran, U. Killat, et al., “Virtual Network Mapping - An Optimization Problem”, In K. Pentikousis, R. Aguiar, S. Sargento, R. Aguéro, O. Akan, P. Bellavista, et al. (Edits.), Mobile Networks and Management (Vol. 97, pp. 187-200). Springer Berlin Heidelberg, 2012.

[156] M. Melo; S. Sargento; U. Killat; A. Timm-Giel; J. Carapinha, "Optimal Virtual Network Embedding: Node-Link Formulation," Network and Service Management, IEEE Transactions on , vol.10, no.4, pp.356,368, December 2013.

[157] T. Enokido, A. Aikebaier, M. Takizawa, S.M. Deen, "Power Consumption-Based Server Selection Algorithms for Communication-Based Systems,", 13th International Conference on Network-Based Information Systems (NBiS), vol., no., pp.201-208, 14-16 Sept. 2010

[158] T. Enokido, A. Aikebaier, M. Takizawa, "Energy-Efficient Server Selection Algorithms for Network Applications", International Conference on Broadband, Wireless Computing, Communication and Applications (BWCCA), vol., no., pp.159-166, 4-6 Nov. 2010

[159] E. Rodriguez, G. Alkmim, D.M. Batista, N.L.S. da Fonseca, "Green virtualized networks", IEEE International Conference on Communications (ICC), vol., no., pp.1970,1975, 10-15 June 2012.

[160] Sen Su; Zhongbao Zhang; Xiang Cheng; Yiwen Wang; Yan Luo; Jie Wang, "Energy-aware virtual network embedding through consolidation", IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), vol., no., pp.127,132, 25-30 March 2012

[161] J.F. Botero, X. Hesselbach, M. Duelli, D. Schlosser, A. Fischer, A.; De Meer, H., Energy Efficient Virtual Network Embedding, Communications Letters, IEEE , vol.16, no.5, pp.756,759, May 2012

[162] N. Roy, J.S. Kinnebrew, N. Shankaran, G. Biswas, D.C. Schmidt, "Toward Effective Multi-Capacity Resource Allocation in Distributed Real-Time and Embedded Systems", 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC), vol., no., pp.124-128, 5-7 May 2008.

[163] Ying Zhang, Gang Huang, Xuanzhe Liu, Hong Mei, "Integrating Resource Consumption and Allocation for Infrastructure Resources on-Demand", IEEE 3rd International Conference on Cloud Computing (CLOUD), vol., no., pp.75-82, 5-10 July 2010.

[164] Y. Osana and S. -i. Kuribayashi, "Enhanced Fair Joint Multiple Resource Allocation Method in All-IP Networks", IEEE 24th International Conference on Advanced Information Networking and Applications Workshops (WAINA), vol., no., pp.163-168, 20-23 April 2010

[165] K. Bouyoucef, I. Limam-Bedhiaf, O. Cherkaoui, "Optimal allocation approach of virtual servers in cloud computing,", 6th EURO-NF Conference on Next Generation Internet (NGI), vol., no., pp.1-6, 2-4 June 2010.

[166] M.J. Csorba, H. Meling, and P.E. Heegaard, “Ant system for service deployment in private and public Clouds”, Proceeding of the 2nd workshop on Bio-inspired algorithms for distributed systems, ACM, p. 19–28, 2010.

[167] B. Kantarci, H. Mouftah, “Minimizing the provisioning delay in the cloud network: Benefits, overheads and challenges”, IEEE Computers and Communications (ISCC), 2012 IEEE Symposium on, (pp. 806-811), 1-4 July 2012.

[168] B. Kantarci, H. Mouftah, “Optimal Reconfiguration of the Cloud Network for Maximum Energy Savings”, 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), (pp. 835-840), 13-16 May 2012.

[169] B. Kantarci, H. Mouftah, “Overcoming the energy versus delay trade-off in cloud network reconfiguration”, IEEE Symposium on Computers and Communications (ISCC), vol., no., pp. 53, 58, 1-4 July 2012.

[170] R.C.S. Chen; Chung-Ting Kao, Hui-Kuang Chung, "Efficient Usage of Network Bandwidth in the Cloud Architecture", IEEE Ninth International Conference on e-Business Engineering (ICEBE), vol., no., pp.338,343, 9-11 Sept. 2012

[171] S. Feizi, A. Zhang, M. Medard, "A Network Flow Approach in Cloud Computing", 47th Annual Conference on Information Sciences and Systems (CISS), vol., no., pp.1,6, 20-22 March 2013.

[172] O. Soualah, I. Fajjari, N. Aitsaadi, A. Mellouk, "PR-VNE: Preventive reliable virtual network embedding algorithm in cloud's network", IEEE Global Communications Conference (GLOBECOM), vol., no., pp.1303,1309, 9-13 Dec. 2013

[173] Zhiping Cai, Fang Liu, Nong Xiao, Qiang Liu, Zhiying Wang, "Virtual Network Embedding for Evolving Networks", IEEE Global Telecommunications Conference (GLOBECOM), vol., no., pp.1-5, 6-10 December 2010.

[174] Y. Ohsita, T. Miyamura, S. Arakawa, S. Ata, E. Oki, K. Shiomoto, M. Murata, "Gradually Reconfiguring Virtual Network Topologies Based on Estimated Traffic Matrices", 26th IEEE International Conference on Computer Communications (INFOCOM), vol., no., pp.2511-2515, 6-12 May 2007.

[175] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, “Black-box and gray-box strategies for virtual machine migration,” Proc. Networked Systems Design and Implementation, 2007.

[176] N. Bobroff, A. Kochut, and K. Beaty, “Dynamic placement of virtual machines for managing SLA violations,” Proc. of 10th IFIP/IEEE International Symposium on Integrated Network Management (IM’07), 2007, p. 119–128.

[177] Q. Li, Q. Hao, L. Xiao, and Z. Li, “Adaptive Management of Virtualized Resources in Cloud Computing Using Feedback Control,” Proc. of 1st International Conference on Information Science and Engineering (ICISE’09), IEEE, 2010, p. 99–102.

[178] P.N. Tran, L. Casucci, A. Timm-Giel, “Optimal mapping of virtual networks considering reactive reconfiguration”, IEEE 1st International Conference on Cloud Networking (CLOUDNET), vol., no., pp.35,40, 28-30 November 2012.

[179] J. Ferrer Riera, E. Escalona, J. Batalle, E. Grasa, J.A. Garcia-Espin, "Virtual network function scheduling: Concept and challenges,"International Conference on Smart Communications in Network Technologies (SaCoNeT), vol., no., pp.1,5, 18-20 June 2014

[180] S. Clayman, E. Maini, A. Galis, A. Manzalini, N. Mazzocca, "The dynamic placement of virtual network functions", IEEE Network Operations and Management Symposium (NOMS), vol., no., pp.1,9, 5-9 May 2014.

[181] Tarik Taleb and Adlen Ksentini, "Gateway relocation avoidance-aware network function placement in carrier cloud", In Proceedings of the 16th ACM international conference on Modeling, analysis & simulation of wireless and mobile systems (MSWiM '13). ACM, New York, NY, USA, 341-346, 2013.

[182] Bo Lv, Zhenkai Wang, Tao Huang, Jianya Chen, Yunjie Liu, "Virtual Resource Organization and Virtual Network Embedding across Multiple Domains," International Conference on Multimedia Information Networking and Security (MINES), vol., no., pp.725-728, 4-6 Nov. 2010

[183] M. Chowdhury, F. Samuel, and R. Boutaba, "PolyViNE: policy-based virtual network embedding across multiple domains", In Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures (VISA '10). ACM, New York, NY, USA, 49-56, 2010.

[184] I. Houidi, W. Louati, W.B. Ameur, D. Zeghlache, Virtual network provisioning across multiple substrate networks, Computer Networks, Volume 55, Issue 4, Pages 1011-1023, ISSN 1389-1286, 10 March 2011.

[185] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, and H. Bhogan, "Volley: Automated data placement for geo-distributed Cloud services," Proceedings of the 7th USENIX conference on Networked systems design and implementation, USENIX Association, 2010, p. 2.

[186] M. Hamze, N. Mbarek, O.Togni, "Autonomic Brokerage Service for an End-to-End Cloud Networking Service Level Agreement", IEEE 3rd Symposium on Network Cloud Computing and Applications (NCCA), vol., no., pp.54,61, 5-7 February 2014.

[187] 4WARD - Architecture and Design for the Future Internet, ET FP7 Project. [Online]. Available: http://www.4ward-project.eu/. [Accessed: September 29, 2014].

[188] Open Networking Foundation. [Online]. Available: https://www.opennetworking.org/. [Accessed: September 29, 2014].

[189] SAIL Project, D-A.1 "Description of project wide scenarios and use cases", February 2011.

[190] Metro Ethernet Forum, Cloud Services. [Online]. Available: http://metroethernetforum.org/carrier-ethernet/cloud-services. [Accessed: September 29, 2014].

[191] Metro Ethernet Forum, "MEF Carrier Ethernet for Delivery of Private Cloud Services", February 2012.

[192] ETSI, "Network Functions Virtualization ISG, Our Role & Activities". [Online]. Available: http://www.etsi.org/technologies-clusters/technologies/nfv. [Accessed: September 29, 2014].

[193] Broad Band Forum. [Online]. Available: http://www.broadband-forum.org/. [Accessed: September 29, 2014].

[194] BroadBand Forum, Service Innovation & Market Requirements Working Group, "Stage 1 for introduction of Network Function Virtualization in MSBN", SD-340, Working Text.

[195] BroadBand Forum, Service Innovation & Market Requirements Working Group, "Flexible Service Chaining", SD-326, Work in Progress, End to End Architecture Working Group.

[196] BroadBand Forum, Service Innovation & Market Requirements Working Group, "High Level Requirements and Framework for SDN in Telecommunication Broadband Networks", SD-313, Work in Progress.

[197] BroadBand Forum, End to End Architecture Working Group, "Virtual Business Gateway", WT-328, Working Text.

[198] BroadBand Forum, End to End Architecture Working Group, "Network Enhanced Residential Gateway", WT-317, Working Text.

[199] Internet Engineering Task Force, Service Function Chaining Working Group. [Online]. Available: https://datatracker.ietf.org/wg/sfc/documents/. [Accessed: September 29, 2014].

[200] OpenStack, OpenStack API specifications. [Online]. Available: http://docs.openstack.org/api/api-specs.html. [Accessed: September 29, 2014].

[201] J. Nogueira, M. Melo, J. Carapinha, and S. Sargento, "A distributed approach for virtual network discovery," GLOBECOM Workshops (GC Wkshps), 2010 IEEE , vol., no., pp.277-282, 6-10 December 2010.
[202] Network Activator, Mediation and Service Activation Platform, Portugal Telecom Inovação e Sistemas. [Online]. Available: http://www.ptinovacao.com.br/download.do?image=/pdf/Ne-Activator.pdf. [Accessed: September 29, 2014].
[203] Open vSwitch. [Online]. Available: http://openvswitch.org/. [Accessed: September 29, 2014].
[204] E.W. Dijkstra, "A note on two problems in connexion with graphs", Numerische Mathematik 1: 269–271, 1959.
[205] S. Russel and P. Norvig, "Artificial Intelligence, a modern approach", 2003.
[206] S. Even, A. Itai, A. Shamir, "On the complexity of time table and multi-commodity flow problems", Foundations of Computer Science, 16th Annual Symposium on, (pp. 184-193), 1975.
[207] M. Pioro, D. Medhi, "Routing, Flow, and Capacity Design", in Communication and Computer Networks. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc, 2004.
[208] MATLAB Release 2010, The MathWorks, Inc., Natick, Massachusetts, United States.
[209] MATLAB, Waxman Network Topology Generator. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/2517-waxman-network-topology-generator/content/waxtop.m. [Accessed: September 29, 2014].
[210] IBM. IBM ILOG Optimization Products. [Online]. Available: http://www-01.ibm.com/software/websphere/products/optimization. [Accessed: September 29, 2014].
[211] P. Quinn, Kumar, T. Nadeau, "Service Function Chaining Problem Statement," IETF Internet draft, Informational, December 2013.
[212] Y.Zhang, N. Beheshti, L. Beliveau, G. Lefebvre, et al. "StEERING: A Software-Defined Networking for Inline Service Chaining," Proceedings of. IEEE ICNP 2013, Goettingen, Germany, Oct. 2013.
[213] W. John, K. Pentikousis, G. Agapiou, E. Jacob, et al., "Research Directions in Network Service Chaining", IEEE SDN for Future Networks and Services (SDN4FNS), vol., no., pp.1,7, 11-13 November 2013.
[214] Bottle: Python Web Framework. [Online]. Available: http://bottlepy.org/docs/dev/index.html. [Accessed: September 29, 2014].
[215] CouchDB Python library. [Online]. Available: https://code.google.com/p/couchdb-python/. [Accessed: September 29, 2014].
[216] MySQLdb. [Online]. Available: http://mysql-python.sourceforge.net/. [Accessed: September 29, 2014].
[217] Alpine linux. [Online]. Available: http://alpinelinux.org/. [Accessed: September 29, 2014].
[218] OpenStack Traffic Steering blueprint. [Online]. Available: https://review.openstack.org/#/c/92477/. [Accessed: September 29, 2014].

Annex

Paper A. Building Virtual Private Clouds with Network-aware Cloud

João Soares, Jorge Carapinha, Márcio Melo, Romeu Monteiro, Susana Sargento

in ***International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP), 2011***

The format has been revised

Building Virtual Private Clouds with Network-aware Cloud

João Soares, Jorge Carapinha, Márcio Melo, Romeu Monteiro, Susana Sargento

Abstract

Cloud computing presupposes on-demand network access to pool of computing resources. However, network access through the WAN is usually not compliant with any kind of service guarantees, including reliability, security and performance. In this work, two types of network services able to fulfill cloud requirements are presented. In addition, an extension of the Virtual Private Cloud concept is proposed by integrating these network services. Managing cloud and network resources in an integrated way is a need and an obvious challenge, thus resource management in such environment is a major focus. We identify the main inherent challenges in resource management and how they can be overcome. Further, an experimental platform is presented, along with a preliminary analysis of results.

Keywords - cloud computing; cloud networking; virtual private cloud; network-as-a-service; connectivity-as-a-service.

A.1 Introduction

Today, the proliferation of broadband access gives users the possibility to use services available directly through the Internet, which represents a change of the paradigm for using applications and communicating, thus popularizing the so-called Cloud Computing (CC).

CC brings with it requirements at two different levels: at the cloud level, i.e. DCs; and at the network level, where required levels of performance, reliability and security must be guaranteed.

So far, the cloud and the network have been seen as two separate entities in this picture, with the network playing a relatively minor role, mostly as provider of connectivity between the cloud resources and the user premises. We argue that, to provide assured levels of performance to cloud services, cloud and network resources need to be provisioned, managed, controlled and monitored in an integrated way.

There are several reasons for the integration of network and cloud. First, the establishment of Service Level Agreements (SLA) is essential to encourage customers, particularly enterprises, to adopt cloud services. Today, the lack of reliability and performance guarantees is one of the main obstacles against the widespread use of cloud services. It is clear that these SLAs can only be implemented through network integration. Just like the Wide Area Network (WAN) component of enterprise networks is usually based on reliable managed network services such as Virtual Private Networks (VPNs), rather than the public Internet, there is no reason to believe that future enterprise cloud services will require a lesser degree of reliability and performance guarantees from the network.

Secondly, it is essential that cloud properties such as elasticity and self-provisioning be also extended to network resources. Quite often, expanding or reducing cloud resource capacity, or provisioning new cloud resources, requires a corresponding reconfiguration of network resources, e.g. bandwidth admitted into the network. Today, by contrast, reconfiguration of network services is supposed to be relatively infrequent and usually involves a significant amount of manual effort.

Thirdly, the dynamism of the cloud will often require live migration of resources (e.g. from a local enterprise DC to the cloud, or between two different sites of the cloud service provider) without interrupting the operating system and any noticeable impact on the running application. This requires IP addressing to remain unchanged after migration and all relevant QoS, security and traffic policies applied on network equipment (e.g. routers, switches, firewalls) to be adapted appropriately in real time.

For the reasons stated above, it is clear that next generation of cloud services must handle network and cloud resources in an integrated way. This paper presents the concept of Cloud Networking (CN) to achieve this integration in the context of virtual private environments, and its resource management aspects to develop an integrated view and allocation of both network and cloud resources.

This paper is organized as follows. Section A.2 summarizes relevant work in the area and how this work intends to progress, and section A.3 presents how the concept of CN can be applied in the context of virtual

private environments. Further, section A.4 provides an overview of the resource management challenges that arise in a CN environment. The experimental platform that embraces this approach is described in section A.5. Section A.6 describes how a Virtual Private Cloud (VPC) request is instantiated and also presents the prototyping results achieved so far to demonstrate the concept of CN. Finally, section A.7 provides general conclusions and indicates directions for future work.

A.2 Related Work

Based on recent trends and evolvements, it is clear that the network will play a key role in the provisioning of cloud computing services, by giving the necessary guarantees to access the cloud. The importance of this role will be increasingly evident. In this area, some research works have been presented in the literature, such as [2] where the critical impact of network performance on the applications is shown and an extension of [3] is presented based on a platform for provisioning of virtual infrastructures, to extend the traditional cloud paradigm to network provisioning. [4] presents a software-based network resource management system for VPCs, able to handle heterogeneous network equipment and proposes a virtual network point for multipoint network provisioning.

The European Commission, through the Seventh Framework Programme (FP7), has also been supporting research in this area, FP7 Projects SAIL (Scalable & Adaptive Internet soLutions) [5] and GEYSERS (Generalised Architecture for Dynamic Infrastructure Services) [6] are two relevant examples.

From the industry side, both standardization bodies and enterprise efforts have highlighted the need for cloud and network resources to be handled together. Verizon has been working on the extension of VPNs for Private Clouds and an Internet Engineering Task Force (IETF) Internet-Draft has been released on this matter [7]. Meanwhile, IBM already offers enterprises a cloud data backup supported by Verizon's VPN services.

Although there are works on management addressing VPCs and others addressing virtual networks, there are few addressing the management of both in an integrated way. This paper addresses this subject and proposes an extension of the VPC concept to also provide WAN guarantees. A platform able to provide this VPC model is also presented in the paper.

A.3 Cloud Networking and Virtual Private Clouds

Virtualization has been the key enabler of agility in DCs, which in turn led to the emergence of CC. The fundamental breakthrough offered by virtualization is the separation of operating systems (OSs) and applications from the underlying physical infrastructure.

Applying the same concept to networks has been often advocated – by decoupling networks from infrastructure through virtualization, it should be possible to establish and reconfigure (virtual) networks with great flexibility, nearly on-demand. Network virtualization has been explored by different research initiatives in multiple contexts and application scenarios. The idea of on-demand provisioning of network services has been demonstrated in practice [9].

Providing the network infrastructure with the ability to match the dynamism of the cloud would be required to overcome the problems and limitations identified in the previous section. From this point of view, network virtualization would be the perfect companion for virtualization in the DC, in order to build seamless end-to-end elastic and agile offer of cloud services.

A virtual network (VN) is supposed to fully replicate the behavior of a physical network, from all points of view. While this replication may be useful in some cases (e.g. when the customer is itself a service provider), in most cases the effort of managing a VN is a burden that customers would prefer to avoid.

Thus, just like the CC service model defines three basic services - Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) - we propose a similar approach for networks. From this perspective, we define two types of network services (Figure A-1): Network as a Service (NaaS) and Connectivity as a Service (CaaS). NaaS allows the user to request a network by specifying precisely the network topology, link bandwidths, routers' computing capacities, routing protocols, as well as possibly other features (e.g. physical location, security properties). As for CaaS, the user is provided with the ability to define, just like in VPNs, a set of customer edge equipments (CEs) (e.g. enterprise sites and possibility the

cloud CE, however this latter does not necessarily needs to be defined) and certain characteristics such as bandwidth at ingress/egress points and routing protocols between the CE and the provider edge equipment (PE). Everything that runs inside the service provider network domain is not visible to the customer.

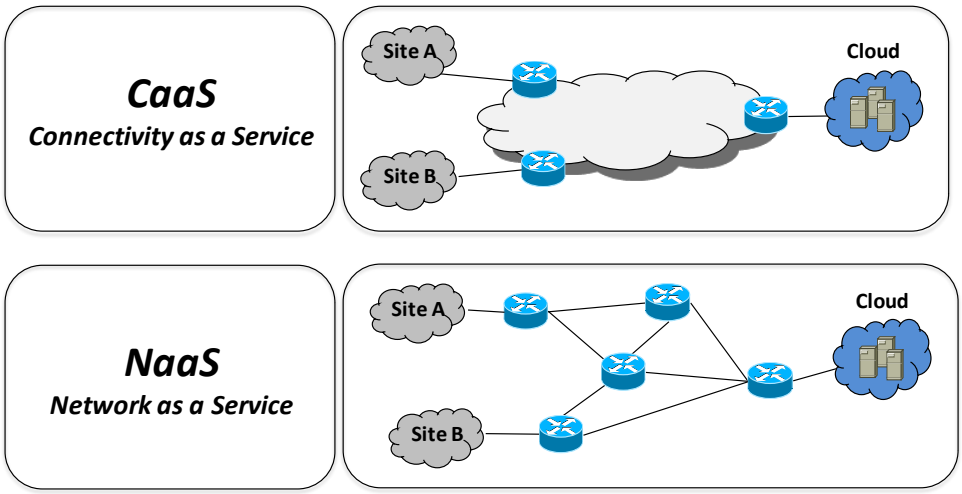


Figure A-1: Network layering model.

Both network service options can be materialized in multiple ways. A fully virtualized network is the "natural" way to materialize NaaS, whereas managed VPNs (e.g. BGP/MPLS VPNs) are a typical example of CaaS.

In this paper, we propose a solution that is able to embrace both types of network service.

At this point it is useful to define the concept of VPC. In [8] a VPC is defined as "*a combination of cloud computing resources with a VPN infrastructure to give users the abstraction of a private set of cloud resources that are transparently and securely connected to their own infrastructure*". In the context of this paper, we propose to generalize this concept, in order to embrace any kind of private network service, either materialized as a VPN, or as more advanced service types, including those based on fully virtualized networks. Moreover, a network service should allow the handling of network resources (e.g. bandwidth, add/remove a costumer site) with a certain level of freedom in order for it to keep up with the cloud.

In order to address the coupling of both cloud and network services, the concept of CN was put forward. Similarly to CC, CN has no standard definition, but we can say that it goes beyond classical networks, encompassing on-demand provisioning i.e. scalability, guaranteed performance, self-healing and extensible management. So far, no real attempt to merge networking and cloud resources in a common framework has actually taken place.

We pursue the concept of CN by envisioning a unified management framework for computing and communication, where the network operator can provide simultaneously the network and cloud resources (IaaS), in an integrated approach, optimizing overall resource allocations by considering network and computing resources as a unified whole. In this work, network services are materialized in VNs, however we do not exclude the possibility of other network approaches (e.g. VPN, OpenFlow).

In the following section we identify what we consider to be the most important challenges to CN that resources management raises.

A.4 Resource Management in Cloud Networking

Bringing together network and CC resources so that users can access services in the cloud with guaranteed performance and reliability raises several challenges. The discovery, allocation, adaptation and re-optimization of resources, addressing simultaneously both network and cloud resources, are the main inherent challenges of resource management in CN. The management of these resources lays upon concepts of virtual resource mapping in the physical infrastructure with self-organized reconfiguration of

resources, devices and associated network, according to the services and user requirements, policies (with respect to e.g. location) and changes in the infrastructure.

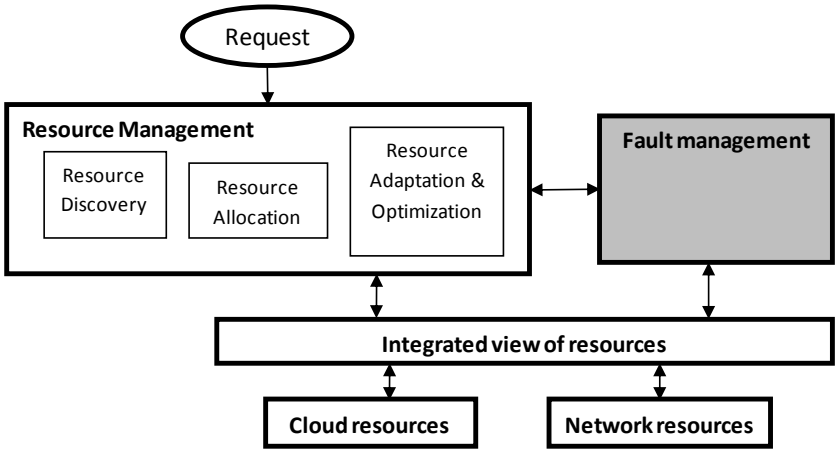


Figure A-2: Cloud Networking Management diagram.

In Figure A-2 we present a management block diagram composed by three main blocks: the *Resource Management* block (RM); the *Fault Management* block (FM); and an underlying block entitled *Integrated view of resources* (IVR). This work is focused on the RM and IVR. The former is composed by three sub-blocks: the *Resource Discovery* block (RD); *Resource Allocation* block (RA); and the *Resource Adaptation & Optimization* block (RAO). These sub-blocks will be detailed ahead. As for the latter, the IVR, it has the purpose of providing the upper blocks with the domain agnostic ability to view and interact with resources, whether they are cloud or network resources. Regarding the FM, it is illustrated in the picture to facilitate the interpretation of the management system, mainly regarding its interaction with the RAO sub-block.

A.1.1 Resource Discovery (and Monitoring)

A fundamental requirement in virtualized environments is the integrated view of the existing physical and virtual topologies, the resources' characteristics, as well as the status of all network elements and links. This knowledge can be provided either by a centralized or by a distributed approach [10].

Today the cloud, i.e. DCs, and the operator's network are two distinct domains which CN aims at integrating. However, there are boundaries that cannot be crossed as these domains will not be willing to share full information about their domain. In this approach, we assume to have access to network information such as topology and physical resources, as well as the ability to retrieve information on the virtual resources that the physical resources may host. On the DC side, we do not expect to have such detailed information, we rather expect to see a DC as a single node in the network with unlimited capacity and a set of associated information elements (similar to today's cloud services, e.g., instance types, available OSs, pricing, plus location).

A.4.1 Resource Allocation

Virtual resources should be provisioned and placed in an optimal location according to the available resources at the time of the request, based on a number of possible criteria from both cloud and network, e.g.: type of VMs and possible restriction on location of these VMs; latency, bandwidth topology, geographical places where users will access the service, and other possible restrictions.

In order to map resources, a combined mechanism, able to perform balanced decisions taking into account the abovementioned requirements of both network and cloud resources, is needed. This mechanism must be able to determine a possible solution, i.e., physical hosts able to allocate the cloud resources which, at the same time, can have an associated network service able to fulfill the requirements in the access to the cloud.

A.4.2 Resource Adaptation and Optimization

With the dynamism of the cloud, reconfigurations and re-optimizations become common operations, whether to cater for possible side effects of new virtual resources being instantiated and existing virtual resources being resized, released or migrated, business policies, or triggered by unexpected events (e.g. node or link failure). These unexpected events are triggered by the FM which is responsible for monitoring the resources, detecting faults and collecting performance metrics.

Depending on the specific environment, actions can be taken at different levels: in the cloud, in the network, or in both. Thus, mechanisms are required for extending or moving cloud resources to other DCs, creating new network paths and reconfiguring existing ones (need for more bandwidth, less latency, failure, load balancing network resources). Those algorithms must decide on (1) when to reconfigure and (2) how to reconfigure. These decisions must be done based on information provided by the FM, or by an explicit request from the user.

Towards this aim, the Network Virtualization System Suite (NVSS) presented in [9] was extended with the control and management of the Suite enabling now cloud (IaaS) and network services (NaaS and CaaS) to be provisioned together to meet the user's requirements, apart from its original feature, the deployment of VNs. The implementation work presented in this paper specifically targets the challenge of resource discovery (and monitoring) of cloud resources and resource allocation of both cloud and network. The next section gives an overview on the evolved NVSS.

A.5 Network-Aware Cloud System Suite

The Network-aware Cloud System Suite (NCSS) is a platform that provides integrated deployment and management of cloud and network resources in a single tool. This platform is an evolution of NVSS [7], an experimental platform that provides VN design, embedding, creation, discovery, monitoring, and management. NCSS extends NVSS in two fundamental ways: it handles cloud resources (rather than just network resources) and CaaS (rather than just NaaS).

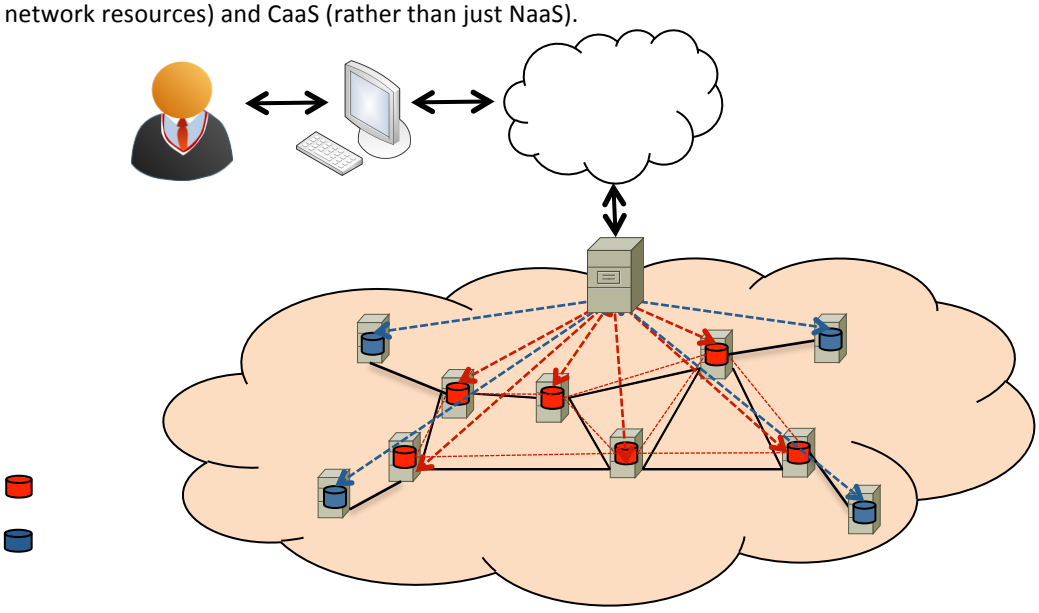


Figure A-3: NCSS Architecture

NCSS is composed of 3 software modules: the Agent module, the Manager module and the Control Centre module. Their hierarchical decomposition can be analyzed on Figure A-3. The Agent module is designed to run on network nodes ('Net Agent'), as well as on those acting as computing nodes ('IT Agent'), in order to act upon them and periodically gather data from them. The two types of Agents, besides interacting with each other, receive and send requests to the Manager, which is a centralized entity in charge of aggregating all Agents' knowledge and sending them commands. Additionally, the Manager also

communicates with the Control Centre, which is the user's front-end, and provides him with graphical and simple to use VN creation, management, and monitoring functionalities.

A.5.1 Functionalities

The NCSS platform provides a set of main functionalities: distributed network and cloud resource discovery, network and cloud mapping and creation, network and computing monitoring, and network and cloud resource management. These functionalities are described below.

Distributed Network and Cloud Discovery.

Network and cloud resource discovery is not only an administrator's utility that provides a fast and easy way of viewing how the cloud resources and the network resources are been used and where they are been consumed, but it is also fundamental when embedding new cloud and network resources, since the embedding process requires an accurate and up-to-date view of the substrate and currently running cloud and network resources.

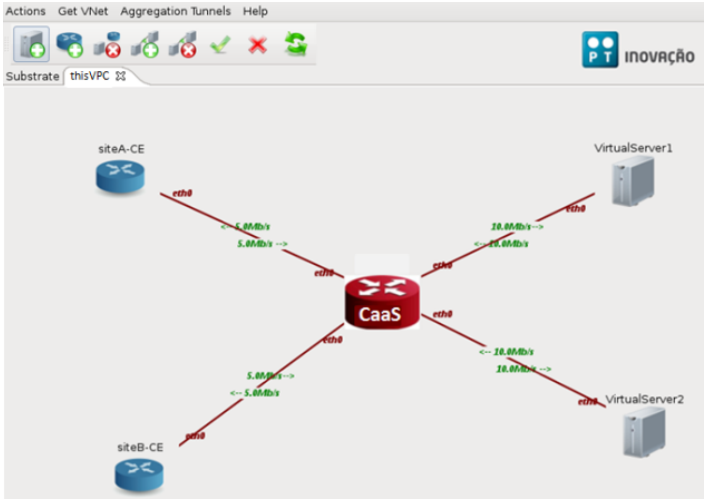


Figure A-4: Control Centre– VPC Requirement Example

Network and Cloud Mapping and Creation

The Control Centre module provides the user with means to create and embed new cloud and network resources in runtime. By selecting and placing either cloud or network resources, i.e. servers or routers, on the platform GUI and by connecting them with links, as depicted in Figure A-4. The user can specify both cloud and network resource capabilities, CPU, RAM amount, location, number of interfaces and also perform network addressing configurations.

The final step in creating a new set of cloud and network elements is to commit it to the Manager, which will then map it in the physical infrastructure.

The embedding problem of cloud and network elements is a complex one, which requires a trade-off between computation time and embedding optimization. In order to lower the computational requirements, a heuristic mapping algorithm was developed, which aims to embed both types of elements taking into consideration both the load of physical links and load of network and computing nodes.

Cloud and Network Monitoring

Dynamic resource monitoring is fundamental to provide an accurate view of the state of both types of resources and to quickly react to failures or configuration problems. The implemented monitoring functions periodically update the information on resources; therefore it is possible to quickly identify diverse situations, such as failures and high resource usage.

Cloud and Network Management.

The management feature provides functionalities like the change of the resource state (i.e., reboot, shutdown, suspend or power up), the change of the assigned RAM memory in runtime and the deletion of either a single resource or a complete set of resources, which greatly simplifies the administrator work.

The following section describes the process of establishing a VPC using the NCSS.

A.6 Virtual Private Cloud Establishment & Evaluation

The establishment of a VPC using the NCSS can be supported by a NaaS or CaaS. The process of establishing a VPC, from the moment a user requests a VPC until the moment it is ready to be enforced in the physical infrastructure, will be detailed in this section. Two VPC requests will be considered, one for NaaS and another for CaaS. In addition, results on the request's processing time are presented. Note that these results do not intend to perform any comparison between a VPC with NaaS and CaaS since they are two different services.

The process of establishing a VPC supported by a NaaS service is divided in 6 main phases, as Figure A-5 shows: r*equest formulation*; *request conversion; resource discovery; node mapping*; *link mapping*; *node* and *link embedding*. Phase 1 encompasses the formulation of a request, in which the user defines all resources: virtual servers (CPU, RAM and HDD); network topology and the characteristics of all nodes and links. The request is then sent to the *Manager* which first performs the conversion of the request from XML to a structured topology (phase 2). Moreover it is performed the resource discovery (phase 3), and the nodes and links are mapped (phase 4 and 5) using a mapping algorithm which considers both the occupation of physical nodes and links. Finally, the VPC is enforced (phase 6).

Based on the tools already available in the NCSS, which already supported NaaS, we developed the necessary mechanisms to support CaaS. The request of a CaaS service was done taking into consideration some aspects of the widespread VPN concept as example - PE, CE, fully mesh topology.

The user is able to define CEs as if he was requesting a VPN, specify ingress and egress bandwidth requirements for each network endpoint (hose model), rather than specifying the requirements between all pairs of endpoints, as in NaaS. The user can drag and drop graphical depictions of routers and servers, which represent the sites and cloud resources of the network, and then connect them to a central graphical element representing an abstraction of the network (Figure A-4). These connections contain information about the bandwidth from each element to and from the network. In the end the user just has to press the commit button and the CaaS information is processed and the request enforced. In the end, the user gets a VN which connects the customer sites and cloud resources, according to the user's configuration parameters.

A VPC supported by CaaS releases the user from the NaaS complexity, but adds an intermediate process step. Nevertheless this does not imply an increase of time from the request to the enforcement moment.

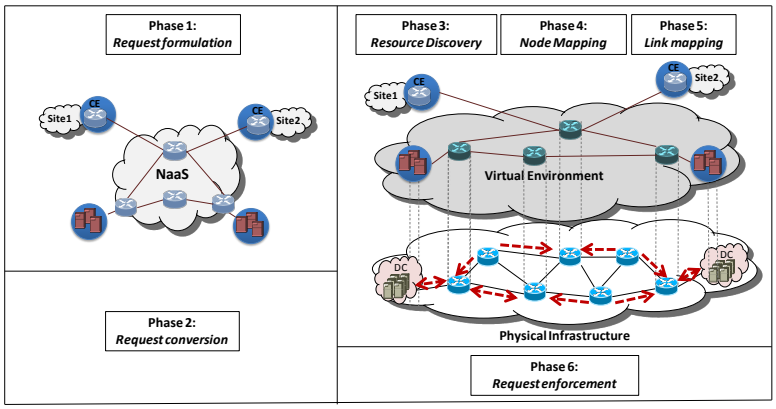


Figure A-5: Virtual Private Cloud mapping process with NaaS.

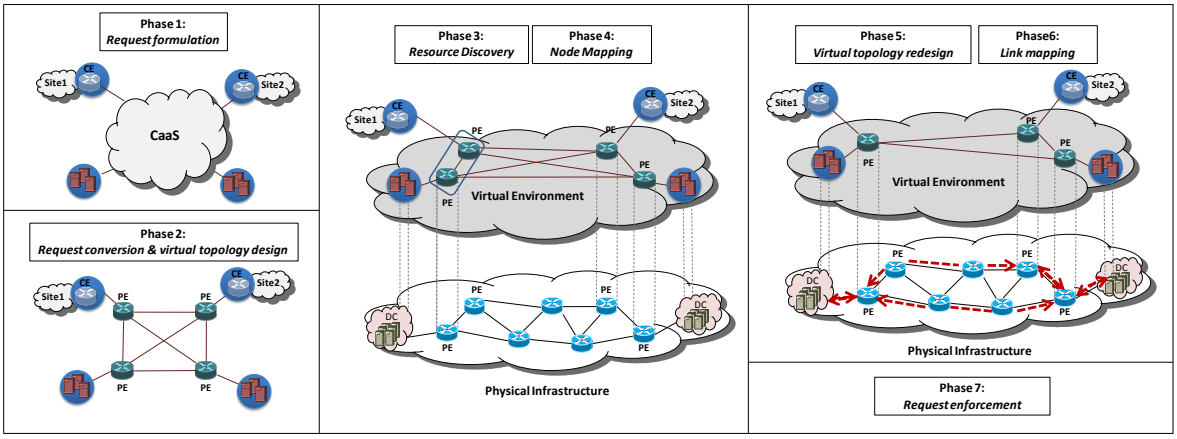
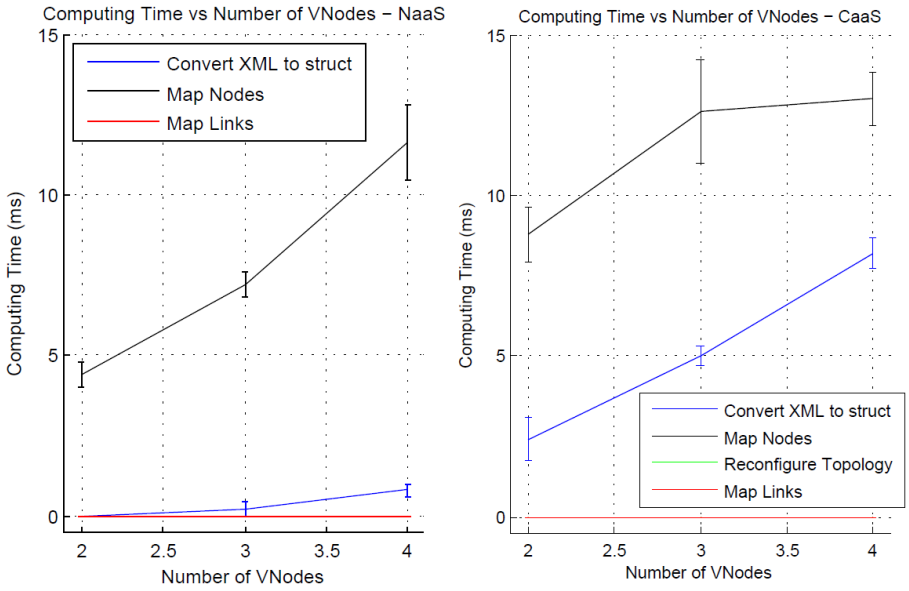


Figure A-6: Virtual Private Cloud mapping process with CaaS.

As depicted in Figure A-6, the process has 7 phases: *request formulation*; *request conversion* (includes the creation of the virtual topology which is not entirely defined); *resource discovery*; *node mapping*; *virtual topology reconfiguration*; *link mapping*; *node* and *link embedding*. First, the user configures the CaaS requirements on the GUI (phase 1). Once the *Manager* has received the request, it converts it to a topology structure, where a virtual PE router is connected to each site and cloud resource, with the PE routers connected in full mesh (phase 2). The bandwidth of the links is set to the minimum necessary to fulfill the worst case hose-model requirements. Then resources are discovered (phase3), and virtual PE routers and cloud resources are mapped to physical PEs and datacenters according to the temporary topology (phase 4). Note that the set of candidate PE routers for a CE encompasses those located near that CE, which eases the mapping process. Phase 5 comprises the virtual topology reconfiguration, so that virtual PEs mapped to the same physical PEs are joined together in one virtual PE. The resulting links are mapped onto the physical substrate in phase 6. In the end, all elements are enforced, phase 7.



(a) Computing times for mapping NaaS&Cloud

(b) Computing times for mapping CaaS&Cloud

Figure A-7: Virtual Private Cloud mapping process with NaaS.

To finalize, Figure A-7 presents the time that the *Manager* takes to process a VPC request. The presented values are an average of 5 requests.

Figure A-7 (a) shows the results for NaaS. The time to convert from the XML message to a structure increases with the number of nodes (0-1ms), but is still a small value when compared to the time to map the nodes (4.5-11.5ms).

As for the mapping with CaaS, Figure A-7 (b), we can see an extra time stage, topology reconfiguration, which is not visible because it is at a 0ms value in both cases, and thus is below the red line. The conversion process takes a little longer with CaaS, 3-14ms, since there is the need to create the full topology. Node mapping values for CaaS are higher, 9-13ms. This might be explained by the fact that the node mapping process in CaaS is made using the temporary topology, which includes 1 PE router per site or server + 1 server element per virtual server. Times for topology reconfiguration in the CaaS are very small, which might be explained by the simple nature of this action, which mainly consist in removing some virtual nodes and links. Link mapping on both cases is always close to 0 (not perceptible since the out time unit while measuring was ms). This might be due to the small number of nodes in both VNs and substrate.

A.7 Conclusion and Future Work

A major limitation of CC is the lack of coordination between CC resource control and network resource control. To overcome this limitation, elasticity and agility of the cloud must be extended to the network infrastructure. In this sense we have associated the concept of VPC with network virtualization, allowing cloud and network resources to be handled as a single set in a dynamic and flexible way.

Two network service models are proposed, CaaS and NaaS. The former roughly corresponds to the traditional managed network-based VPN paradigm. The latter provides a service which is functionally identical to a network. Both models should have a role to play in future networks. Moreover, we present a platform able to handle NaaS and CaaS services along with cloud resources.

One of the tasks that is still open for future work is the integration of BGP/MPLS VPNs in the platform as a CaaS, since VPNs are today in the market and represent the strongest short-time deployment possibility. The integration with the cloud using standardized application programming interfaces (e.g. Open Grid Forum Open Cloud Computing Interface) is also in the evolution roadmap.

Acknoledgement

The authors are thankful to the members of the SAIL project, particularly those involved in Work Package D, for the collaboration and fruitful discussions.

References

[1] P.Mell and T. Grance, “The NIST Definition of Cloud Computing (Draft)”, National Institute of Standards and Technology, January 2011.

[2] T. T. Huu, G. Koslovski, F. Anhalt, P. Vicat-Blanc Primet, and J. Montagnat. “Joint elastic cloud and virtual network framework for application performance optimization and cost reduction”. Journal of Grid Computing (JoGC), pp. 27-47, 2010.

[3] F. Anhalt, G. Koslovski, and P. Vicat-Blanc Primet. “Specifying and provisioning virtual infrastructures with HIPerNET”. Int. J. Netw. Manag., pp. 129-148 May 2010.

[4] T. Miyamoto, M. Hayashi, and K. Nishimura, , "Sustainable Network Resource Management System for Virtual Private Clouds," Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on , vol., no., pp.512-520, Nov. 30 2010-Dec. 3 2010.

[5] FP7 Project SAIL, http://www.sail-project.eu/.

[6] FP7 Project GEYSERS, http://www.geysers.eu/.

[7] So et al, “VPN Extensions for Private Clouds” IETF Internet Draft, February 2011.

[8] T. Wood, A. Gerber, K. Ramakrishnan, P. Shenoy, J. Van der Merwe, “The Case for Enterprise-Ready Virtual Private Clouds”, HotCloud’ 09, 2009.

[9] J. Nogueira, M. Melo, J. Carapinha, and S. Sargento, “Network Virtualization System Suite: Experimental Network Virtualization Platform”, in TridentCom 2011, 7th International ICST Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, April 2011

[10] J. Nogueira, M. Melo, J. Carapinha, and S. Sargento, "A distributed approach for virtual network discovery," GLOBECOM Workshops (GC Wkshps), 2010 IEEE , vol., no., pp.277-282, 6-10 Dec. 2010

[11] J. Nogueira, M. Melo, J. Carapinha, S. Sargento, “Virtual Mapping into Heterogeneous Susbtrate Networks”, Computers and Communications (ISCC), IEEE Symposium, pp.438-444, 2011.

Paper B. Negotiating On-Demand Connectivity between Clouds and Wide Area Networks

Hareesh Puthalath, João Soares (joint first author), Azimeh Sefidcon, Bob Melander, Jorge Carapinha, Márcio Melo

in IEEE International Conference on Cloud Networking (CloudNet), 2012

The format has been revised

Negotiating On-Demand Connectivity between Clouds and Wide Area Networks

Hareesh Puthalath, João Soares (joint first author), Azimeh Sefidcon, Bob Melander, Jorge Carapinha, Márcio Melo

Abstract

Infrastructure as a Service (IaaS) provides the capability to deploy infrastructure on demand, but today there are gaps in the IaaS model for network connectivity, especially when user and/or application require guaranteed connectivity between the deployed infrastructures. This problem is aggravated when virtual infrastructures are geographically distributed, for performance and redundancy reasons. Connectivity is also crucial to acquire and integrate virtual resources in remote cloud datacenters with on-site IT resources. In enterprise environments where connectivity requirements cannot be met by best effort networks like Internet, such distributed infrastructure needs to make use of VPN services for inter-domain connectivity, provided by service providers over their trusted networks. To establish this connectivity, information regarding the interconnection link, such as interfaces and underlying protocols need to be negotiated and agreed upon. Today there is a lack of a technology independent mechanism and interface to perform such negotiation. This is a hindrance to automation of on-demand connectivity. In this paper we present a protocol for dynamic negotiation of connectivity service between domains. We describe how it has been used for negotiations between OpenStack and OpenNebula cloud datacenters and MPLS-based WAN, for establishment of L2/L3 VPN services.

B.1 Introduction

Until very recently the network component of Cloud Computing was the most neglected one, however its importance is highly recognized today due to its fundamental role in guaranteeing performance, reliability and security in the cloud.

Today from an administrative standpoint the notion of domain refers to a collection of recources involving hardware and software managed by a single entity, e.g. a DC, or a communication network.

The network component of the cloud may span one or several domains.

In other words, we can look at cloud networking within a DC (single-domain), or go further and and bring the Wide Area Network (WAN), i.e. network operators, into the picture (multi-domain).

Infrastructure-as-a-Service (IaaS) provides the possibility for deploying infrastructure on-demand but there are limitations when user and/or application relies on guaranteed connectivity between the deployed infrastructure that is spread across geographically distributed sites or domains.

This lack of reliability and performance guarantees over the WAN is today one of the main obstacles against the widespread use of cloud services, particularly in the enterprise market sector.

Enterprises usually rely on managed network services such as Virtual Private Network (VPN), rather than the public Internet (best-effort model), and there is no reason to believe that future cloud services will require a lesser degree of reliability and performance guarantees from the network.

However, the current establishment of these services is done through static processes that require a significant amount of manual effort involving network administrators at different domains. Moreover, reconfigurations of these services are supposed to be relatively infrequent.

In order to cope with the cloud, future network services will certainly require on-demand and self-provisioning properties allowing its deployment across multiple domains. However there are challenges on how to negotiate on-demand connectivity between domains.

We present in this work a protocol for the dynamic negotiation of connectity services between domains.

We describe how the protocol fits the needs and how it can be integrated in current technical solutions.

The paper is organized as follows. Section B.2 highlights the most relevant work and trends in the area. Section B.3 starts with the motivation of our work, a generic use-case description followed by a high-level architectural framing. Section B.4 explores the interdomain connectivity aspects. It points out the need for

on-demand negotiation, presents a technology independent mechanism for on-demand link negotiation. Section B.5 details how the integration/implementation of the protocol can be done. Moreover, it describes a real testbed set-up, using cloud management solutions such as OpenStack and OpenNebula along with MPLS-based WAN. Finally, in section B.6 conclusions are drawn and some future work is pointed out.

B.2 Related Work

The importance of the network role in the cloud service delivery is increasingly evident. Recent trends and evolvements show that academy, industry, standardization bodies and the cloud market itself are alert and active on the subject.

B.2.1 Standardization bodies and industrial efforts

Both standardization bodies and enterprises have been looking into this subject. AT&T and the University of Massachusetts released one of the first studies highlighting the need for more comprehensive control over network resources and security on the cloud service delivery, especially in the enterprise market sector [1]. More recently [2] was published, exploring the on-demand bandwidth reservation for inter-DC communication.

Verizon performed an initial study on the extension of VPNs for Private Clouds and an Internet Engineering Task Force (IETF) Internet-Draft was released [3].

IBM recently published [4], in which they presented CloudNaaS, a cloud networking platform for enterprise applications.

From a standardization perspective, the Metro Ethernet Forum (MEF) [5] is working on a cloud project to introduce the concept of delivering private cloud services via Carrier Ethernet in WANs. MEF uses cloud broker, an entity that manages the use, performance and delivery of cloud services and negotiates the relationships between cloud service providers, cloud customers and Ethernet based carriers. MEF's cloud broker is the peer of NIST [6] cloud broker that performs the same negotiation and management between cloud service providers and cloud consumers. We consider our work to be very much in alignment with the objectives of both these projects.

Open Grid Forum (OGF) is also active in the definition of the Open Cloud Computing Interface (OCCI) [7], and recently a cloud networking extension has been defined by Institut Télécom, python Open Cloud Networking Interface (OCNI) [8]. The Distributed Management Task Force (DMTF), the holder of Cloud Infrastructure Management Interface (CIMI), has more recently also been working on network extensions [9].

B.2.2 Ongoing projects and Academia

Ongoing EU-funded projects such as SAIL [10], GEYSERS [11] and EURO-NF [12] are examples of the strong work carried out by the European research community.

[13] presents the Platform-as-a-Service (PaaS) model for networking, in which the network is abstracted by a single router representation. In the same line of thought [14] proposes two types of network services to cope with the cloud, Network as-a-Service (NaaS) and Connectivity-as-a-Service (CaaS), and an experimental platform able to deal with cloud infrastructure resources and network services are presented.

B.2.3 Related products

On the commercial side, Amazon AWS Virtual Private Cloud (VPC) [15] was one of the first solutions that looked to the connectivity component of the cloud, however without any interaction with network providers. More recently Amazon has made available the Direct Connect service [16], which brings network operators into the picture. Nevertheless it is important to note that this does not provide a fully automated and on-demand process.

Furthermore, AT&T has introduced a VPC solution [17] and IBM offers enterprises a cloud data backup supported by Verizon's VPN services. However the establishment of these services are not done either on-demand or in a self-provisioning manner.

B.3 A Multi-Domain Cloud Networking Architecture

B.3.1 Motivation and Requirements

In the previous section we have drawn attention to various efforts in the cloud networking area.

The research community is highly committed to leverage cloud networking, and the wide number of studies and projects are proof of it. Moreover, we have also shown that the commercial viewpoint is not absent, since there are already some initial offers.

We believe that this work is unique with respect tackling the multi-domain connectivity challenge without depending on any particular technology, where IaaS services can be provisioned on-demand and delivered across multiple domains with assured network connectivity guarantees.

B.3.2 Use-Case

Several use-cases could be pointed out, whether to connect users directly to the cloud (e.g. virtual desktop, online gaming) or to create distributed clouds in which cloud resources are spread across domains. In this work we focus on the latter case without aiming at any specific application/service. We take a generic use-case in which the purpose is to deploy cloud infrastructure resources, i.e. virtual machines, in different sites and connect them as illustrated in Figure B-1.

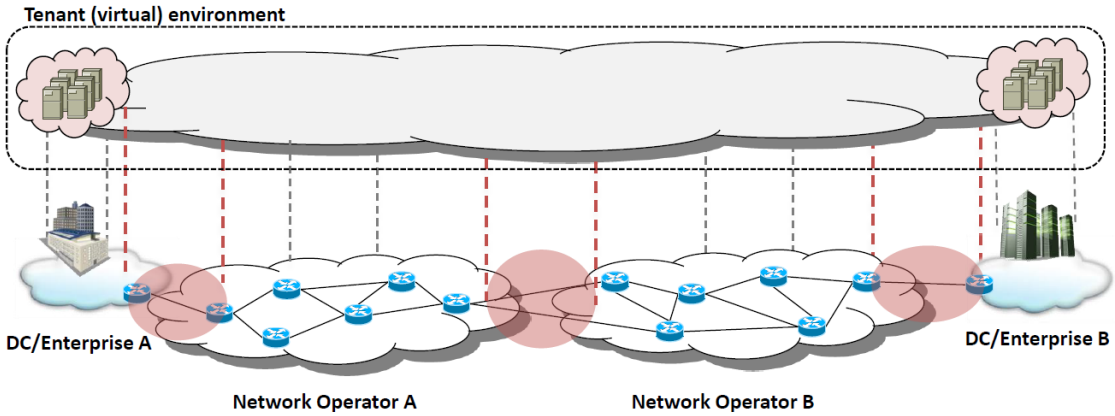


Figure B-1: Use-Case

In Figure B-1 we can see a single tenant (virtual) environment composed by two portions of resources hosted in different DCs connected over a WAN service. Note that this single environment involves four domains. The remaining of this paper is focused on how these domains coordinate among themselves for the establishment of end-to-end connectivity.

B.3.3 Architectural Context

Whether the same provider, or different providers, are in charge of the distributed infrastructure (network and DC), there are several logical and physical entities that need to work together in order to establish the connectivity that spans multiple administrative domains. These entities are the various interfaces, protocols, controllers and input-output processing functions that enable two or more infrastructure service providers to interact and exchange domain specific information needed for the establishment of the connection. Together we refer to them as Interdomain Coordenation Framework (ICF) for distributed infrastructures.

We consider that the DCs and the networks in between belong to different administrative domains. In order to communicate between these domains, the solution includes a north bound interface as well as an east bound interface. The request for connectivity comes via the northbound interface. The actual connectivity between domains needs more detailed information such as interface addresses and agreement

on routing protocols which are not available in the original request. Figure B-2 shows the high level view of these entities.

A Composite Request (CR) represents a high level description of a Virtual Infrastructure (VI) requested by the user of this system, hereafter called tenant. This request is sent to a Cloud Infrastructure Orchestrator (CIO). The CIO will decompose the CR into a number of sub-requests for each sub-domain in a distributed infrastructure scenario. This sub-request is called a VI Request (VIR). Each domain has an infrastructure controller implementing the North-bound interface for receiving the VIR.

The request is parsed by the Message De-serialiser and sent to the Inter-domain Controller, which in essence is a finite state machine acting on the received message based on the actual state. As per defined sequence of actions in case of each message, the next event will be triggered, which might be a new message or an action to the resource management system.

The next message is built and sent out via the Message Serializer to the relevant recipient domains which have the same logical entities (Message (De)Serializers and Inter-domain Controllers).

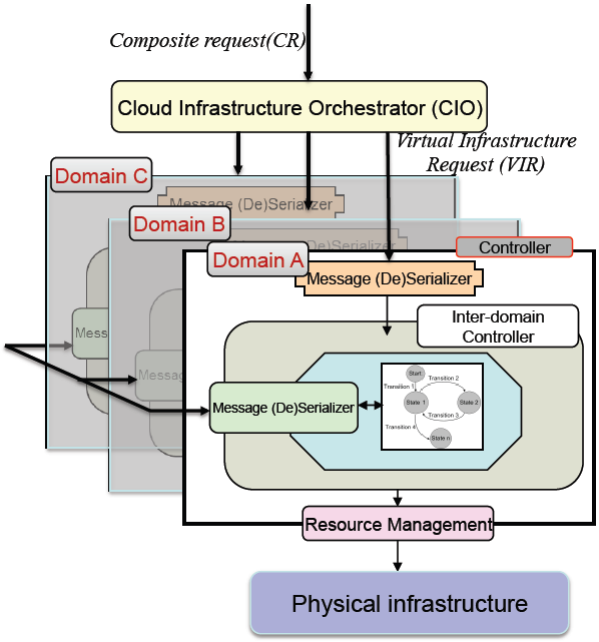


Figure B-2: Architecture

B.4 Interdomain Connectivity

B.4.1 Coordination across domains

For requesting connectivity services there is a north-bound interface that can be used by tenants. The interface supports create, read, update and delete (CRUD) operations on an abstract logical switch (for layer 2 connectivity) or a logical router (for layer 3 connectivity). This maps well onto traditional VPN service model, providing a simple and intuitive abstraction of the connectivity service. The details of the north-bound interface is outside the scope of this paper. It suffices for the reader to understand it is a RESTful interface.

To instantiate a connectivity service, a multitude of configurations need to be performed on all affected nodes in the network. This typically requires detailed knowledge about the network equipment in use and the design of the involved networks, for example, topology. Network operators typically consider these as sensitive and business critical information which should not be exposed to the tenant requesting the VI or other domains.

To support this separation of concerns a special interface and associated protocol have been defined allowing negotiation of configuration details. This is the east-bound interface and it is the main focus of the remaining sections of this paper.

When virtual resources, such as VMs with one or several virtual network interface cards (vNICs), are created in multiple clouds, care must be taken so that property values of those resources that must be unique are not assigned colliding values. Examples of such properties can be the MAC address assigned to vNICs, IP address assigned to vNICs or fully qualified host names.

In a static, non-virtual environment collisions are less likely to occur since there is often global coordination, e.g., IEEE assigns equipment vendor unique MAC value ranges. However, in a cloud environment where values of properties like these are generated on-the-fly, overlap is possible across domains. In lack of a global mechanism for ensuring uniqueness, coordination is necessary across the participating domains. In the design proposed here, parameter arbitration and collision resolution are part of the ICF.

B.4.2 Link Negotiation Protocol

A VI specified in a CR can span multiple administrative domains. In such cases the individual parts of this VI within each administrative domain need to be connected to its other parts distributed in one or more other domains. Naturally, link is the virtual resource that will cross domains, and therefore will make these connections.

For this purpose we have defined the *link negotiation protocol*, which is implemented by the Inter-domain controller. This protocol is responsible for creating one or more virtual links belonging to the same VI but may be spanning multiple (usually two) domains.

The design of this protocol was done with the following high-level objectives:

- Simple and low level protocol agnostic
- Support of various transport network solutions (L2, L3)
- Agnostic to any particular networking implementation

First of all it is important to highlight and define three terms:

- **Virtual Link**: A link in the VI.
- **Service Provider Logical Link (SLL)**: A logical data transmission link from one domain to another. This is usually installed and configured by the service provider. Examples of SLL can include physical layer separation schemes like Provider Backbone Bridging or MAC-in-MAC or tunnelling schemes like IPSec or GRE. Each SLL has a reference to a physical or logical link, which in the latter case means that there might be aggregation happening below this layer at the physical links as well. It is important to note that this work is not concerned with the creation or configuration of SLL and assumes that they are already in place. It merely uses them via references.
- **Tenant Logical Link (TLL)**: A logical link part spanning between two domains. This corresponds to a virtual link in the VI of the tenant and has a one to one mapping. The main difference is that a virtual link must be unique within one VI, whereas a TLL must be unique across the two domains in all VIs. For example, VLANs may be used for creating TLLs and ensuring isolation, in which case the VLAN number can act as a unique TLL. The TLL can have additional configuration parameters, such as interface addresses of the two endpoints and a routing protocol like in the case of a L3 link.The TLL are thus virtual slices of an SLL.

The Link Negotiation Protocol offers three functions: Create, Update and Delete. There is another function, route export, for the cases where there is the need to explicitly export a route to a remote domain. The operations are detailed below.

Create Function

The process of creating a TLL between two domains (refered in our description as Domain A and Domain B) is illustrated in Figure B-3 via a message sequence diagram. Under the natural assumption that both domains have received two "pieces" of the same VI, the element spanning the two domains is a virtual link

(one or several). For that reason we refer to "Link Request" in the diagram as the request for a virtual link that will cross and connect across these two domains.

Table B-1: Link Negotiation: Overall Message Parameters

Parameters	Description
infra id	identifier of the virtual infrastructure
transaction id	identifier of this transaction
msg type	identifier of the type of message
sender	identifier of the message sender
service type	identifier of the type of service
virtual link	Details of the virtual link: virtual link id, in and out bandwidth, TLL information

From a high-level perspective and depending on the type of link being established, (i.e. whether L2 or L3) *link negotiation* process can have either one or two phases.

For the establishment of a L2 TLL, the process comprises of only the L2 Negotiation phase.

After receiving the request, one of the domains will trigger the negotiation process. The decision about which domain takes the initiative is outside the scope of the protocol. (In practice it is usually the network operator).

In our example we assume Domain A to be the "initiator". After the trigger, the "initiator" starts the process by listing for each TLL, the various SLLs able to accommodate it.

This information is then sent to the "receiver", Domain B, using the *Link_Offer* message. The remaining parameters of the *Link_Offer* message can be seen in Table B-1. The parameters in this table are common to all messages in the protocol. *transaction_id* is used to identify this transaction among the set of ongoing transactions at any moment .The *infra_id* identifies the VI in question and the *virtual_link* contains the info about the virtual link crossing two domains. This information is known in advanced by both domains (via the north bound interfaces). The remaining elements, *service_type*, *in_bw* and *out_bw* (in and out bandwidth) are used in the perspective of guaranteeing consistency, since this information is expected to be known in advance by both domains.

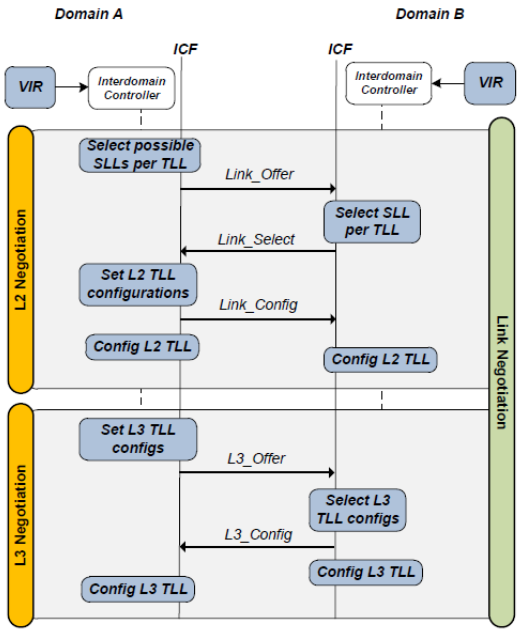


Figure B-3: Creation Function - sequence diagram

Upon receiving the list of SLLs for each TLL, Domain B selects one SLL per TLL, and sends that information in the *Link_Select* message along with the type of encapsulation scheme used for the establishment of each TLL.

Domain A is then responsible for setting the encapsulation scheme configuration attributes of each TLL, sending that information to Domain B using the *Link_Config* message. Table B-2 shows the message parameters. At this point both domains have the necessary information to establish a L2 link.

Table B-2: Link Negotiation: Parameters for L2 Negotiation

Parameters	Description
TLL_id	Identifier of TLL
SLL_offer	list of SLLs offered to carry the TLL
SLL_id	SLL id selected to carry the TLL
encap_scheme	identifies the encapsulation scheme (type and attributes) for the TLL

If the virtual link in question is a L3 link, the process continues and goes to the *L3 Negotiation* phase. Note that the message sequence does not imply that a L3 link can be configured only after a L2 link is configured, but rather this design choice was taken to reduce the total number of messages. This avoids a new sequence with some duplicate information (like identifying the SLL) in both L2 and L3 configuration. The "initiator" sends the *L3_Offer* message with the L3 configuration parameters, i.e. the IPs) to be configured in the endpoints of each TLL, and a list of the supported routing protocols by each TLL. The "receiver" selects the protocol and informs the "initiator" using the *L3_Config*. All parameters for establishing a L3 TLL(s) is now known and the configuration can be applied. The parameters for the L3 Negotiation phase are presented in Table B-3.

Table B-3: Link Negotiation: Additional Parameters for L3 Negotiation

Parameters	Description
L3_config	L3 configuration parameters: IPs for the link endpoints (source and destination) and a list of the supported routing protocols.

Update Function

Updating or reconfiguring the parameters are natural actions in the lifetime of a VI. Such expected actions can be triggered by a user specific update of the VI, or by the domain's management policies for reconfiguration. In either case if the change in VI is associated to a link, then the corresponding TLLs may need to be reconfigured. When the update to a TLL is due to an update on the VI, the domain who initiated the TLL creation process is also responsible for starting the update process. On the other hand, when the update is triggered by an internal domain policy either the "initiator" or the "receiver" must be able to start the process.

There are two possible scenarios, an update is requested by the "receiver" (case A) and an update is triggered by the "initiator" (case B). In the former, case A, the process is initiated using the *Link_Reconf* message, which is a request for reconfiguring one or more TLLs in a certain VI. The "initiator" will, identical to the creation process, prepare a list of SLLs for each TLL and send it to the "receiver" using the *Link_Update* (which in terms of parameters is equal to the *Link_Offer*). From this point on the process is similar to the creation process.

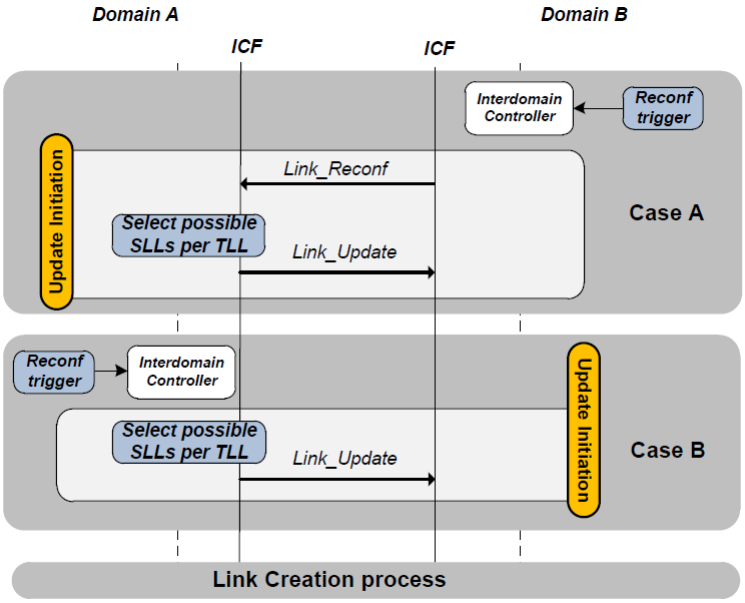


Figure B-4: Update Function - sequence diagram

Delete Function

The delete process for a TLL, shown in Figure B-5, is always triggered by the "initiator". Since both domains are expected to have received the delete request, the "initiator" domain usually takes the lead on this process. The *Link_Delete* message identifies the VI and the TLL(s) to be deleted.

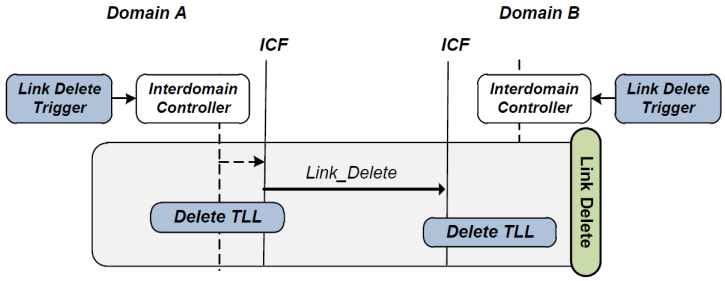


Figure B-5: Delete Function - sequence diagram

Route Export Function

In the case of L3 TLL(s), for some domains, the defined routing protocol maybe static. For such cases, the route export function was defined to cover the need of exporting a route to another domain. To do that the domain wanting to export routes related to a certain VI uses the *Route_Export* message to send one or more routes to a remote domain. Figure B-6 illustrates the process for the case in which Domain B is exporting a route to Domain A.

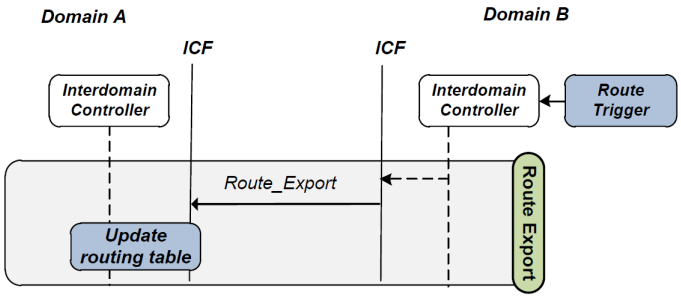


Figure B-6: Route Export Function - sequence diagram

B.5 Implementation & Testbed

B.5.1 DC implementation

Figure B-7 shows the various components involved in the link negotiation and setup process in a DC. It also shows a runtime snapshot with some already configured links and their associated internal configuration as well.

The DC has modules for implementing the north bound and the east bound interfaces, namely the API server and the ICF modules respectively. The DC Network Manager module is responsible for creating and managing tenant's virtual networks. The VM Manager module is responsible for creating and controlling VMs for the tenant and places them in the corresponding tenant network by interacting with the DC Network Manager module. Each has an associated database for keeping their configuration information.

The DC Network Manager module together with the ICF module implements link negotiation (i.e. setup of TLL). In practice, this requires the setup of an intermediate network device for connecting the tenant's virtual network to the WAN.

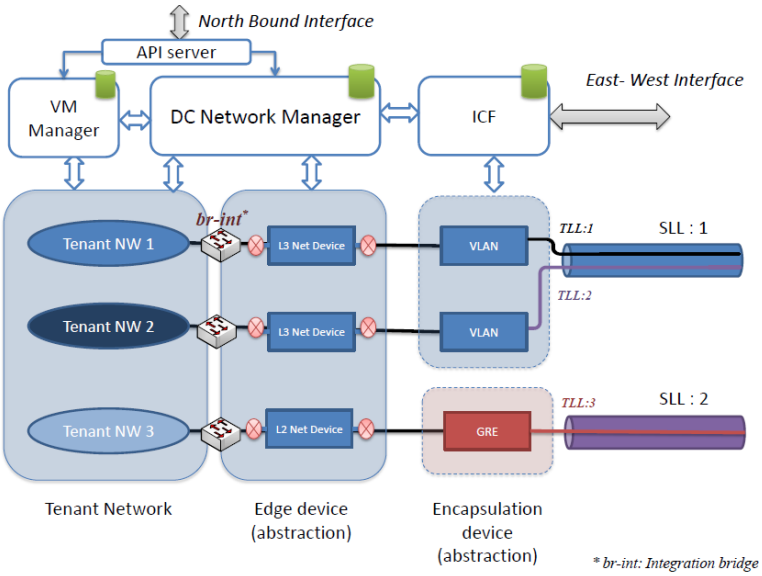


Figure B-7: DC Modules

The DC Network Manager module creates the interconnecting network device. In the case of an L3 VPN this means a router for routing between the two subnets. This functionality may be implemented in software (running inside a VM or a routing service) or by delegating to a hardware based router. In the case

of L2 VPNs this means a layer 2 device, namely a switch or even just a link. Here also the implementation may be software based or delegated to an appropriate hardware device. Note that in both cases the edge network devices need to expose two (virtual) interfaces. One of these virtual interfaces are then plugged into the bridge corresponding to the tenant network.

Finally an encapsulation device also needs to be configured. It separates the different tenant links going into the same SLL. This is done by the ICF module and done as per the parameters agreed in the corresponding TLL. Examples of encapsulation include VLANs and tunnelling schemes like GRE or IPSec etc. Figure B-7 shows an example using VLANs for SLL\#1 and GRE for SLL\#2. Note that the encapsulation type each SLL has already been pre-agreed and known to both domains and only the parameters of that specific flow are set here as agreed in the earlier link negotiation process. Here also, from an implementation perspective, both a pure software only or a combination of hardware and software are equally feasible.

Figure B-8 shows the interaction among the modules based on the messages.

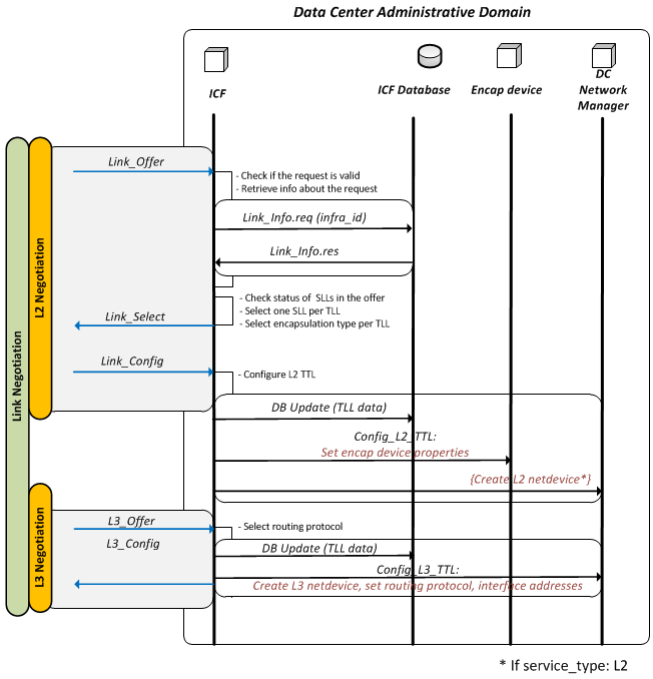


Figure B-8: Link setup - DC network functions interactions

B.5.2 WAN implementation

Similar to the DC, the network management system in the WAN needs to be modified to incorporate the necessary modules and functions needed for link negotiation and setup. Figure B-9 shows these modules. This domain follows the architectural model described in section B.3 and has north and east bound interfaces are similar to the DC.

The request for a connectivity resource (a L3 or L2 VPN) is received via the North bound interface, the controller initiates the negotiation process with the peer domains (DCs in this case) in accordance with the link negotiation protocol (section B.4.2). After the protocol is completed, along with the parameters received via the north bound interface, the controller will have all the necessary information needed to setup the network. Then it contacts a more traditional network management system responsible for provisioning the network with all the necessary information.

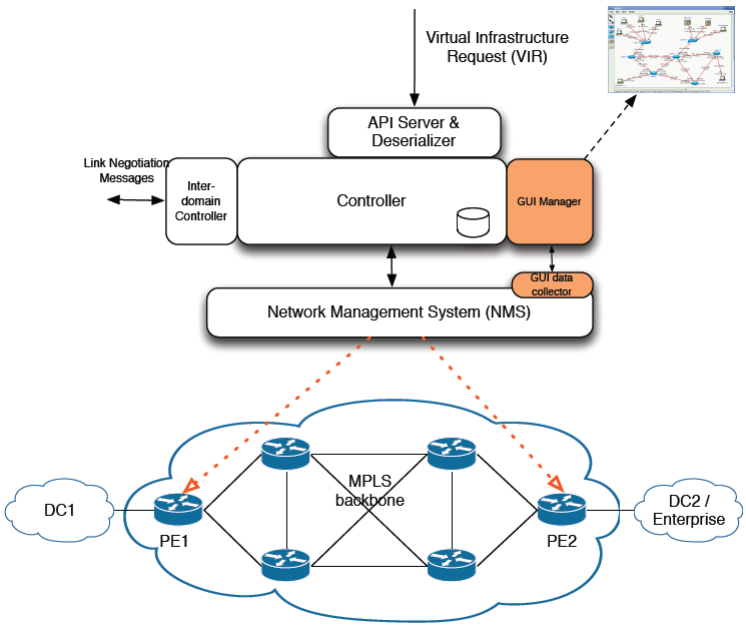


Figure B-9: WAN modules

B.5.3 Testbed

We have built a testbed composed of five domains, namely three DCs and two network operators. Figure B-10 shows the high level overview of the same. OpenStack [18] and OpenNebula [19] platforms provide the base for the VM Manager and DC Network Manager functionalities in the DCs. Both platforms were extended for the ICF as described in section B.5.1. Our OpenStack implementation is based on 'Diablo' release, while the OpenNebula implementation is based on version 3.4. The WAN testbeds use MPLS as the network technology for L2 and L3 VPN services and are managed by their associated Network Management Systems.

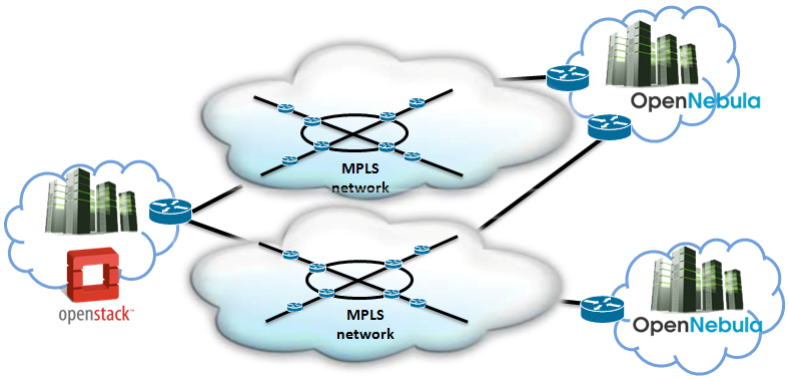


Figure B-10: Testbed

B.6 Conclusions

In this work we presented a framework for negotiating on-demand connectivity between domains. This allows the creation of distributed VIs.

In order to connect the individual parts of a VI lying in different domains we described a protocol for the negotiation of virtual links between domains.

Moreover we provided an overview of the framework implementation on DC and network operator domains.

This general framework was implemented as part of a testbed using popular open source platforms for the management of datacenters (OpenStack and OpenNebula). On the network operator part we implemented a framework with a network management system for MPLS VPNs. Our experiments over this proof of concept show that we can indeed create a VI spanning multiple domains in an on-demand manner.

As future work we intend to present concrete experimental values with respect to the VI creation process, i.e. overall time, link negotiation protocol message exchange time, virtual machine creation time, network service establishment time, and the time for the establishment of connectivity. Also, a more thorough evaluation to verify the possibility of incorporating the proposed protocol within an existing one is necessary.

References

[1] T. Wood, A. Gerber, K. Ramakrishnan,et al, “The Case for Enterprise- Ready Virtual Private Clouds”, HotCloud 09, 2009.

[2] Ajay Mahimkar, Angela Chiu, Robert Doverspike, et al. 2011. Bandwidth on demand for inter-DC communication. In Proceedings of the 10th ACM Workshop on Hot Topics in Networks (HotNets ’11). ACM, New York, NY, USA, , Article 24 , 6 pages.

[3] So et al, “VPN Extensions for Private Clouds”, IETF Internet Draft, February 2011.

[4] Theophilus Benson, Aditya Akella, Anees Shaikh, and Sambit Sahu. CloudNaaS: a cloud networking platform for enterprise applications. In Proceedings of the 2nd ACM Symposium on Cloud Computing (SOCC’11). ACM, New York, NY, USA, , Article 8 , 13 pages.

[5] Metro Ethernet Forum, “MEF Carrier Ethernet for Delivery of Private Cloud Services”, February 2012.

[6] F. Liu, J. Tong, J.Mao, R.B.Bohm, et al, ”NIST Cloud Computing Reference Architecture”, National Institute of Standards and Technology U.S. Department of Commerce, September 2011.

[7] Open Grid Forum, Open Cloud Computing Interface (OCCI).

[8] Institut T´el´ecom, pyOCNI [Online]. Available: http://occi-wg.org/2012/02/20/occi-pyocni/ [June. 15, 2012].

[9] Doug Davis (IBM), DMTF, Cloud Infrastructure Management Interface (CIMI) Model and REST Interface over HTTP An Interface for Managing Cloud Infrastructure.

[10] FP7 Project SAIL, http://www.sail-project.eu/.

[11] FP7 Project GEYSERS, http://www.geysers.eu/.

[12] FP7 Project EURO-NF, http://euronf.enst.fr/

[13] Eric Keller and Jennifer Rexford, “The “Platform as a service” model for networking”. In Proceedings of the 2010 internet network management conference on Research on enterprise networking (INM/WREN’10). USENIX Association, Berkeley, CA, USA, 4-4.

[14] João Soares, Jorge Carapinha, Márcio Melo, Romeu Monteiro, and Susana Sargento, “Building Virtual Private Clouds with Networkaware Cloud”, ADVCOMP 2011, Lisbon, November 2011.

[15] Amazon, AWS Virtual Private Cloud. Available: http://aws.amazon.com/vpc/ [June. 15, 2012].

[16] Amazon, AWS Direct Connect. Available: http://aws.amazon.com/directconnect/ [June. 15, 2012].

[17] AT&T, Introducing AT&T’s Virtual Private Cloud. Available: http://www.att.com/gen/press-room?pid=22362&cdvn=news&newsarticleid=33844 [June. 15, 2012].

[18] Openstack, http://www.openstack.org/

[19] OpenNebula, http://www.opennebula.org/

Paper C. The Cloud inside the operator's network: Resource allocation

João Soares, Romeu Monteiro, Márcio Melo, Susana Sargento, Jorge Carapinha

in IGI Global The Communication Infrastructures for Cloud Computing, 2013

The format has been revised

The Cloud inside the operator's network: Resource allocation

João Soares, Romeu Monteiro, Márcio Melo, Susana Sargento, Jorge Carapinha

Abstract

The access infrastructure to the cloud is usually a major drawback that limits the uptake of cloud services. Attention has turned to rethinking a new architectural deployment of the overall cloud service delivery. In this chapter we argue that it is not sufficient to integrate the cloud domain with the operator's network domain based on the current models. We envision a full integration of cloud and network, where cloud resources are no longer confined to a DC, but are spread throughout the network and owned by the network operator. In such an environment, challenges arise at different levels, such as at the resource management, where both cloud and network resources need to be managed in an integrated approach. We particularly address the resource allocation problem through joint virtualization of network and cloud resources, by studying and comparing both Integer Linear Programming formulation of the problem and a heuristic algorithm.

C.1 Introduction

"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." [1]. This is part of one of the most cited cloud computing definitions, defined by the United States National Institute of Standards and Technology (NIST). It clearly states that network is an inherent component of the cloud, not only as a mean of access to other cloud resources, but also as a resource itself. Although a definition would not be necessary to confirm this, it is interesting to highlight it, since in the cloud early stages the network component of the cloud has been neglected to a large extent. On the other hand, its importance is highly recognized today because of its fundamental role in guaranteeing performance, reliability, and security.

In today's network scenarios, the network component of the cloud has implications at two different levels: Data Center (DC) and Wide Area Network (WAN). Depending on the type of service, different Quality of Service (QoS) guarantees are required both on the DC and in the WAN. Moreover, scalability and elasticity of the cloud may suggest variations on the necessary network resources as the cloud scales up or down. However, from an administrative standpoint, DCs and WANs (which are in practice operator networks) are completely different entities, which do not cooperate on an active basis, and consequently the access to cloud services is typically done over best-effort Internet.

The lack of cooperation between cloud and WAN represents a major drawback that has limited the uptake of cloud services. The current best-effort support for many cloud services is not enough as an increasingly large number of services cannot be handled in this way (e.g., Netflix, OnLive). Furthermore, looking at the enterprise market sector, network reliability is a "must have", not only from a performance perspective but also from a security one. In some cases an independent network service that tries to fulfil the cloud service requirements can be purchased, backed up by a Service Level Agreement (SLA), connecting the user and the cloud hosting the service. This typically happens in the enterprise sector, namely through operator-managed Virtual Private Network (VPN) service models, such as Border Gateway Protocol (BGP)/Multiprotocol Label Switching (MPLS) Internet Protocol (IP) VPN [2] or Virtual Private LAN Service (VPLS) [3] [4]. There is no reason to believe that future cloud services will require a lesser degree of reliability and performance guarantees from the network.

However, the traditional VPN model is not able to handle essential cloud properties such as elasticity and self-provisioning, which means that those properties should be also extended to network resources. Quite often, expanding or reducing cloud resource capacity, or provisioning new cloud resources, requires a corresponding reconfiguration of network resources, e.g., bandwidth assigned between two DCs, whether they are in the same geographical place or not, or between the DC and the end user. Today, the

reconfiguration of network services is supposed to happen on a relatively infrequent basis and usually involves a significant amount of manual effort. In order to cope with the cloud, future network services will certainly require on-demand and self-provisioning properties. This will be the basis for an active participation of the network in the cloud computing service delivery.

Moreover, the dynamism of the cloud will often require live migration of resources (e.g., from a local enterprise DC to the cloud, or between two different sites of the cloud service provider) without interrupting the operating system or making any noticeable impact on the running application. This requires IP addressing to remain unchanged after migration, and all relevant QoS, security and traffic policies applied on network equipment (e.g., routers, switches, firewalls) to be adapted appropriately in real time.

Lately, as a result of the above mentioned aspects, attention has turned to rethinking architectural deployment of the overall cloud service delivery [5] [6], where cloud and network resources need to be provisioned, managed, controlled and monitored in an integrated way to provide a specific service support. Therefore, a joint management of cloud and network resources will be required, along with other requirements, e.g., security, on-demand provisioning, elasticity, reliability. However, how far this integration will go is still to be unveiled.

The current business relation between the end-user and the cloud provider can have two forms: 1) the user has an SLA with the cloud provider and uses the Internet to access the service; or 2) the user has an SLA with the cloud provider, and a separate one with the network provider that ensures a certain network service between the user and the cloud hosting the service. Note that in the latter case there is no end-to-end SLA, but two partial SLAs, one with the cloud provider and another with the network provider. In a ultimate future scenario, the end-user would have a single truly end-to-end SLA.

In order to achieve the ultimate future scenario, Figure C-1 illustrates what we see as the natural evolutionary process from the current scenario to a future one. Today the network can provide static connectivity to cloud resources, to what we call conventional networking. The next evolutionary step is to make the network elastic and adaptable according to the cloud dynamics. This has been refered in literature as cloud networking [7]. However, we believe that it is not sufficient to integrate the cloud domain with the operator's network domain based on the current models, where cloud resources are confined to big DCs. Today, cloud computing relies on the power of big DCs, which has proven to reduce costs [8]. However, this is not the best solution for every problem, in particular when:

- Many individuals or organizations in a certain geographical area need to access the same resource/content/service. In such a case, it seems more appropriate that this object is located, moved or cached near the users, rather than being repeatedly transported across the network (negatively impacting the network performance) [5];
- The access to a shared resource/content/service often requires low latency; therefore, it might be appropriate to locate the data closer to the users. In such cases, instead of relying on a single DC, cloud providers can use content delivery networks, e.g., Akamai, to improve the delivery of their services. This is indeed true, but the level of dispersion of DCs in the network will be always limited to some degree.

Therefore, we envision a second step in cloud networking, illustrated in Figure C-1, as a full integration of cloud and network, where cloud resources are no longer confined to DCs, but may be spread throughout the network. In such an environment, challenges arise at different levels, such as at the resource management, where both cloud and network resources need to be managed in an integrated approach. It is also important to note that on the network side, current deployed technologies are not fully capable of coping with the cloud requirements, namely with its dynamism. In this sense, network virtualization8 and Software Defined Networking (SDN) [9] have been pointed out as basic components of the next network generation.

Although the integration of cloud and network is needed at the various levels of cloud services – Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS) –, we focus on the most basic layer of the stack, IaaS. To provide a flexible and cost-effective infrastructure for this

[8] Network virtualization – in this chapter we consider network virtualization as a way to create several virtual networks identical to physical ones over a physical infrastructure, through the deployment of virtual routers and virtual links.

integration, we consider the provisioning of cloud and network resources through virtualization (virtualized computing, storage, and network), enabling the support of what we call Virtual Infrastructures (VIs). In this virtual integrated environment, issues such as discovery, allocation, adaptation, and re-optimization of both network and cloud resources, are major challenges of joint resource management. In particular, the allocation problem is addressed in this chapter, and two different approaches to allocate the resources in an integrated way will be proposed and tested.

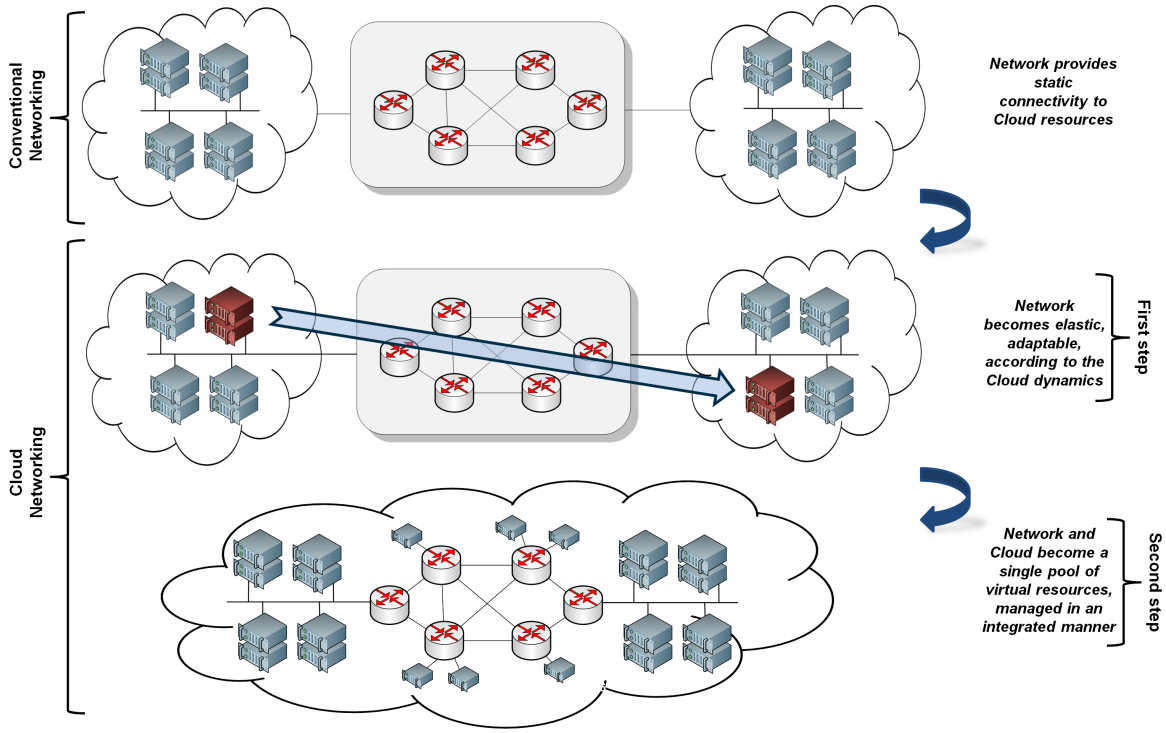


Figure C-1: The evolutionary process of cloud and network

The remainder of this chapter is organized as follows. In the next section we look further to a future cloud networking scenario and sustain our virtualization approach. The allocation problem is then followed by an overview on important state-of-the-art work in the related work section. In order to address the problem, an optimal solution based on an Integer Linear Programming (ILP) formulation is proposed. Further, a heuristic algorithm to solve the problem is presented followed by a comparative analysis between the two approaches via simulation. Moreover, experimental results over a real testbed are presented taking into consideration the heuristic algorithm. Finally, we provide the conclusions and future research directions.

C.2 The Cloud inside the operator's network – A virtualization approach

Agility is a key feature of cloud computing. Resources are elastic, can scale and even be moved (i.e. migrated). Further, this can all be done in an on-demand and self-service way. The key agility enabler is virtualization, by allowing the decoupling of operating systems and applications from the underlying physical infrastructure. It is thus fundamental that this agility is preserved when bringing the network into the picture.

However, on one hand most network services today are not prepared to deal with the cloud agility, whether from a technological or operation perspective. Managed network VPN (e.g., BGP/MPLS), which represents a widely deployed network service for enterprises, is a significant example. This type of services

has been conceived to work in a relatively stable network environment (which is the case with most enterprise networks today), but is not appropriate to cope with the typical dynamics of cloud services. Although some agility can be provided over these services from an operational perspective, i.e. by empowering Telco's operational support systems with appropriate mechanisms, there will always be technological barriers. In the VPNs case, part of the reason lies in the characteristics of the BGP protocol, in charge of handling intra-VPN routing, which suffers from well-known slow convergence issues [10].

In contrast to this model, new forms of network virtualization allow the establishment and reconfiguration of (virtual) networks with great flexibility, nearly on-demand. It all starts with an architecture that, as in server virtualization, decouples the network from the underlying infrastructure and enables the creation of multiple Virtual Networks (VNs) on top of a common physical substrate, using the same operational model of virtual machines found in server virtualization. The main components are node and link virtualization: node virtualization consists on partitioning the physical resources of a substrate node (e.g., CPU, memory, storage capacity, link bandwidth) into slices, where each slice is allocated to a virtual node; link virtualization allows the transport of multiple separate virtual links over a shared physical link. The combination these two enables the creation of isolated VNs, over which any kind of network architecture can in principle be built [11] [12]. These VNs can be easily set up and turned down without changing the physical network [13]. Different research initiatives in multiple contexts and application scenarios have explored the potential of network virtualization. Among these research initiatives, the idea of on-demand provisioning of network services has been demonstrated in practice [14]. However, network virtualization is no longer confined to the research world and has become a reality with companies like [15].

Providing the network infrastructure with the ability to match the dynamism of the cloud is required to overcome the problems and limitations already identified. From this point of view, network virtualization would be the perfect companion for virtualization in the DC, in order to build seamless end-to-end elastic and agile offer of cloud services. An alternative approach would be SDN. However, contrary to SDN, our approach builds upon a scenario in which all resources can be virtualized. We pursue the concept of cloud networking by envisioning a unified management framework for computing and communication, where the network operator can provide simultaneously the network and cloud resources (IaaS), in an integrated approach, optimizing overall resource allocations by considering network and computing resources as a unified whole. In this work, network services are materialized in VNs.

When coupling network and cloud resources in such a way, several resource management challenges arise: discovery, allocation, adaptation, and re-optimization of both types of resources in an integrated way. Virtual resources should be provisioned and placed in an optimal location according to the available physical resources and the service requirements, based on a number of possible criteria from both cloud and network, e.g., type of virtual machines and possible restriction on the location of these resources.

C.3 Virtual Infrastructure Assignment Problem

In this section a description of what we consider to be a virtual and physical infrastructure that combines both cloud and network resources followed by a description of the VI assignment problem is provided.

C.3.1 Physical and Virtual Infrastructure Description

A network operator physical infrastructure is considered to be composed of a given number of nodes, *N*, and with a random topology. The set *N* comprises two types of nodes, routing (or network) nodes and server nodes, each with its specific set of associated characteristics. Routing nodes are characterized by the number of Central Processing Units (CPU), denoted by *Cs*, the clock CPU frequency, *F*, and by the memory amount it contains, *M*. Server nodes are characterized by the same parameters as the routing nodes, *Cs*, *F*, and *M*, and also by its storage capabilities, denoted by *STG*. Note that the subset of server nodes is denoted by *S*. With respect to the links, these are characterized by bandwidth capacity, denoted by *B*, and assumed to be unidirectional. An example is depicted in Figure C-2. Moreover, note that associated to the number of CPUs of a node (*Cs*) is another parameter that reflects the CPU load (or capacity) denoted by letter *C*.

VIs are described in the same way as physical infrastructures. Naturally, VI routing nodes can only be accommodated in the physical infrastructure by routing nodes, and the same applies to server nodes. The letter *P* is used to refer to the physical resources, e.g., N^P, and the letter *V* is used for virtual resources, e.g.,

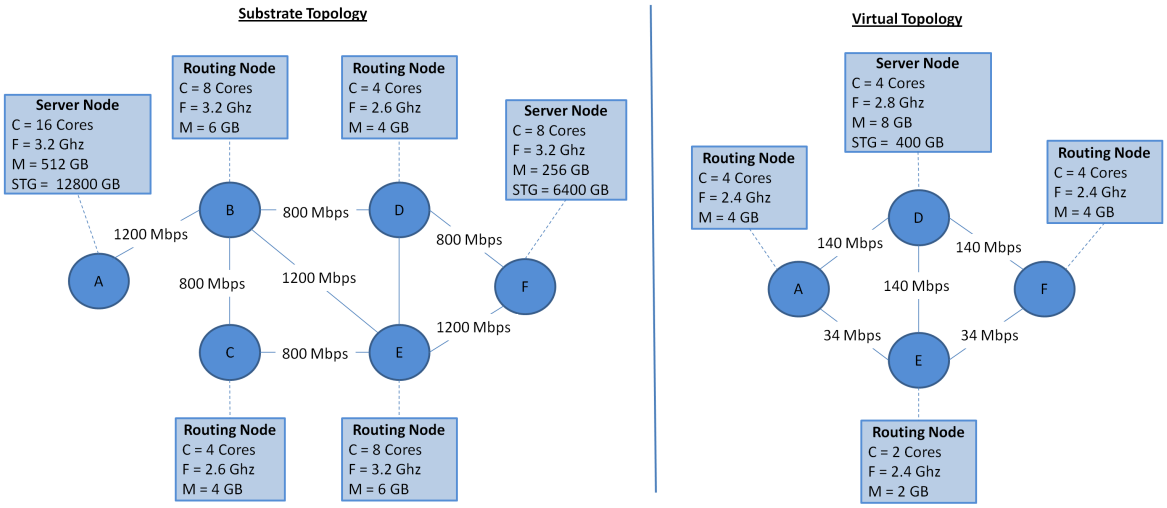
N^V. Moreover, the convention used for the index notation is the following: *i*, *j* for nodes and links in the physical network, and *m*, *n* for nodes and links in the VN.

The CPU number of cores, capacity, frequency, memory size, and the storage capacity of nodes are stored arrays with *N* entries (N^P or N^V depending if it refers to the physical or virtual infrastructure), e.g., $C^P \rightarrow N^P \times 1$. Note that the storage capacity of the routing nodes is considered to be null. Moreover, the total CPU capacity of a physical node is denoted by $C^{P_{total}}$, the free capacity (i.e. non allocated one) by $C^{P_{free}}$, and the allocated capacity $C^{P_{used}}$, where $C^{P_{total}} = C^{P_{used}} + C^{P_{free}}$. The same notation is used for memory and storage.

With respect to the connectivity of an infrastructure, the description is done by an adjacent matrix: $A^P = N^P \rightarrow N^P$ when referring to a physical infrastructure – equation 1; and $A^V = N^V \rightarrow N^V$ when referring to a VI – equation 2. Bandwidth capacity is also described by adjacent matrixes: $B^P = N^P \rightarrow N^P$ and $B^V = N^V \rightarrow N^V$.

$$A_{ij}^P = \begin{cases} 1, \text{the physical node } i \text{ is neigbor of } j \\ 0, \text{else} \end{cases} \quad (1)$$

$$A_{mn}^V = \begin{cases} 1, \text{the physical node } m \text{ is neigbor of } n \\ 0, \text{else} \end{cases} \quad (2)$$



(a) Physical Infrastructure - example *(b) VI – example*

Figure C-2: Example of a physical topology and a virtual topology description

The VI example in Figure C-2 is used to help to better understand the variables description. The number of CPU cores, memory capacity, and storage capacity variables are bellow presented:

$$Cs^V = \begin{bmatrix} 4 \\ 4 \\ 2 \\ 4 \end{bmatrix}; \; F^V = \begin{bmatrix} 2.4 \\ 2.8 \\ 2.4 \\ 2.4 \end{bmatrix}; \; M^V = \begin{bmatrix} 4 \\ 8 \\ 2 \\ 4 \end{bmatrix}; \; STG^V = \begin{bmatrix} 0 \\ 400 \\ 0 \\ 0 \end{bmatrix}$$

As to the connectivity and bandwidth matrixes these are 4 by 4 matrixes with $m, n \in [a, b, c, d]$:

$$A^V = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}; \; B^V = \begin{bmatrix} 0 & 140 & 34 & 0 \\ 140 & 0 & 140 & 140 \\ 34 & 140 & 0 & 34 \\ 0 & 140 & 34 & 0 \end{bmatrix}$$

It is important to note that a VI is not the typical cloud service that the general user uses/wants, but more an enterprise oriented service. For example, an enterprise could buy a VI to deploy a service (e.g. video distribution service) that could be accessed by the general end-user via the Internet.

C.3.2 Problem Description

From a general perspective, one can state that the VI assignment problem is very close to the well-known VN assignment problem. It is indeed true, since in both cases the major challenge lies in the efficient assignment of virtual resources into physical ones: in the VN assignment problem the considered resources are routing nodes and links; in the VI assignment problem, not only routing nodes and links are considered, but also server nodes. In other words, the VI assignment problem adds one extra variable (server nodes) to an already complex problem, known to be *NP-hard*.

In terms of high-level objective, our purpose is to map as much VIs as possible into a physical infrastructure. However, such a high-level objective has to be mapped down into a resource management objective, which in this case cannot be performed directly. In other words, resource management objectives could be, for example, to map a VI in such a way that: 1) it occupies as less bandwidth as possible; or 2) it balances the occupation of the physical nodes. Moreover, the objectives over the different resources can and should be combined, i.e., to be both considered in the objective. However, in the context of our problem these are not standalone objectives because the achievement of one objective is closely related to the others, e.g., the choice of an ideal physical network node may not allow the choice of the ideal physical link. We are therefore faced with what is called a multi-objective optimization problem (MOP), where two or more conflicting objectives subject to constraints need to be simultaneously optimized. In MOP it is not usual to have a solution that is optimal for all the objectives, and our problem is no exception. The general used concept for optimality in a MOP is *pareto optimality*. A *pareto optimal* solution is one that makes it impossible to improve the performance of a criterion without decreasing the quality of at least another criterion. While typical single objective problems have unique optimal solutions, MOP may have a set of solutions known as *pareto optimal* set [16].

Moreover, we consider a reality in which an operator that possesses a physical infrastructure, as the one described earlier, receives VI requests to be mapped along time. These VIs come and go, similarly to what happens today in the cloud environment. In such a scenario physical resources will be gradually occupied over time, and VI requests might be or not accepted depending on the amount of free resources available and on whether the mapping algorithms find a viable mapping solution. The evaluation of the behaviour of a mapping algorithm with the defined objective is mainly performed through the analysis of the amount of accepted VIs during a certain period of time.

C.4 Related Work

This section starts by presenting state-of-the-art works that tackle the VN allocation problem. Furthermore, an overview on state-of-the-art studies which simultaneously embed[9] VNs and cloud computing resources (i.e. virtual servers) is given. The section concludes by presenting the future internet research projects that more closely relate to our work.

[9] The words embedding, mapping, assignment and allocating are used interchangeable throughout this chapter.

C.4.1 Virtual Network Mapping

The embedding of a VN can be considered as a simultaneous optimization of virtual nodes and links assignment, which can be formulated as an unsplittable flow problem [17], known to be *NP-hard* [18]. In order to solve this problem, several approaches have been suggested, mostly considering the *off-line* version of the problem where the VN requests are fully known in advance.

In [19] a backtracking method based on sub-graph isomorphism was proposed; it considers the *on-line* version of the mapping problem, where the VN requests are not known in advance, and proposes a single stage approach where nodes and links are mapped simultaneously, taking constraints into consideration at each step of the mapping. When a bad mapping decision is detected, a back-track to the previous valid mapping decision is made, avoiding a costly re-map.

The work in [20] defines a set of premises about the virtual topology, i.e. the backbone nodes are star-connected and the access-nodes connect to a single backbone node. Based on these premises, an iterative algorithm is run, with different steps for core and access mapping. However, the algorithm can only work for specific topologies.

A distributed algorithm was studied in [21]. It considers that the virtual topologies can be decomposed in hub-and-spoke clusters, and that each cluster can be mapped independently, therefore reducing the complexity of the full VN mapping. This proposal has lower performance and scalability, when compared with centralized approaches.

Zhu *et al.* [17] propose a heuristic, centralized algorithm, to deal with VN mapping. The approach tries to solve an online version of the problem, considering reconfigurations of existing VNs, when VN requests arrive. The goal of the mapping algorithm is to maintain a low and balanced stress of both nodes and links of the substrate network; with that goal in mind, the algorithm starts by determining each node's stress (number of virtual nodes running on the substrate node) and the links' stress (number of virtual links whose substrate path passes through each substrate link). With these weights determined, the Neighbourhood Resource Availability (NR), that takes into account both the node stress and the local links stress, is calculated for each node. The node with the highest NR is selected as the start node to begin the candidate selection. Next, a set of substrate nodes is determined weighted by their distance to the previously selected substrate node, its node potential is calculated, and in the final step the virtual nodes are mapped. Virtual nodes with more interfaces are assigned substrate nodes with higher NR since virtual nodes with more interfaces are also more likely to setup more virtual links and increase the load on both the substrate node and neighbour links. However, the stress of nodes and links does not consider heterogeneity on their characteristics.

Yu *et al.* [22]propose a mapping algorithm which considers finite resources on the physical network, and enables path splitting (i.e. virtual link composed by different paths) and link migration (i.e. to change the underlying mapping) during the embedding process. However, this level of freedom can lead to a level of fragmentation that is unfeasible to manage on large scale networks.

Chowdhury *et al.* [18] propose different algorithms with better coordination between the node and link mapping phases, by using deterministic rounding techniques in one of them and randomized rounding techniques in the other. The evaluation of the algorithms is made in terms of revenue and cost, by using a discrete events simulator to emulate a physical network receiving network requests and then verifying how many resources could be hosted with the different algorithms, and at the same time the VN request acceptance ratio and the ratio of utilization on links and nodes. This approach, despite providing a better coordinated node and link mapping, does not solve the VN assignment problem as a simultaneous optimization problem, and does not support heterogeneity of nodes.

Butt *et al.* [23] proposed a topology aware heuristic for VN mapping, and also suggest algorithms to avoid bottlenecks on the physical infrastructure, where they consider virtual node reallocation and link reassignment for this purpose. Nogueira *et al.* [24] proposed a heuristic that takes into account the heterogeneity of the *VN*s and also of the physical infrastructure. The heuristic is evaluated by means of simulation and also on a small scale testbed, where it achieves mapping times of the order of tens of milliseconds. This heuristic will be further detailed in the section Heuristic Algorithm as it forms the basis from which we build our own approach to the VI mapping problem.

Botero *et al.* [26] proposed an algorithm to solve the *VN* mapping problem, where it also considers the CPU demand of the hidden hops. Melo *et al.* [27] proposed an Integer Linear Programming (ILP) formulation to solve the *on-line* VN embedding problem. It also proposes an enhancement to an existing heuristic [24]

and compares both heuristic and ILP, showing that heuristics can be far from the optimal values. Chowdhury *et al*. [25] extended their preliminary results [18] and included a generalized window-based VN embedding to evaluate the effect of look ahead on the mapping of VNs.

C.4.2 Cloud and Network Resource Management and Mapping

Several works on cloud resource management have been produced in the past years. Roy *et al.* [28] present a study of bin-packing heuristics for resource allocation for Distributed Real-time Embedded (DRE). Li and Tang [29] address the placement decision method taking into account latency and bandwidth constrains in the situation where the user accesses content from several servers.

Energy costs in data centers represent a major consideration nowadays, because of the very high amounts of energy they use. Reducing them is one of the ways of getting competitive advantage and increasing profit margins, besides creating 'green publicity'.

Enokido *et al.* [30] [31] address this subject. The first evaluates Power Consumption-Based (PCB) and Transmission Rate-Based (TRB) algorithms for server selection, while the second one proposes an Extended Power Consumption-Based (EPCB) algorithm that proves to be able to reduce the power consumption more than TRB and PCB algorithms.

As for virtual server placement in Cloud Computing, there has also been significant research. [32] has proposed an optimal allocation approach to choose the best data-center to store the virtual server request by the user, from a pool of multiple data-centers. A dynamic, decentralized, and self-organizing approach to the allocation of Virtual Machines (VMs) to physical servers in public and private Clouds is proposed in [33]. The approach is based on a Cross-Entropy Ant System (CEAS), where intelligent agents are used to discover physical servers and make allocation decisions. The system is able to dynamically react to changes in the load of the physical servers, as well as to failures in the physical infrastructure. The mapping of VMs into physical servers is done using near-optimal heuristics. [34] analyses the interplay between Internet Service Provider (ISP) and content providers. The ISP represents the network part of the problem, while the content providers represent the servers. This paper considers 3 different situations regarding the sharing of information and control between ISP and content providers, concluding that separating server selection and traffic engineering leads to sub-optimal equilibrium, but also that extra visibility might also result in a less efficient outcome.

Moreover, Kantarci and Mouftah have presented several studies within the cloud network field, including the study of the delay minimization in the cloud network [35] through a Mixed Integer Linear Programming (MILP) formulation. Moreover, they address the reconfiguration of the cloud network in order to maximize the energy savings in [36] by proposing two heuristic approaches benchmarked by MILP approaches. In a latter work the authors have studied the trade-off between energy savings and delay minimization [37] and proposed a heuristic.

Moreover, the authors of this chapter, Soares *et al.* [38] have presented a heuristic algorithm that tackles the VI allocation problem. This latter work is an inherent part of this chapter and will be further detailed.

Although the refered studies are of great importance and look towards relevant challenges, whether from a network perspective, from a cloud perspective, and more recently from an integrated perspective, none has looked to an integrated deployment of cloud and network resources within a complete virtualized environment, with exception, or course, of the latter refered work. Apart from this work, most that look from a virtualization perspective usually disregard the network or, when considering it, they take into consideration QoS constraints but do not strive to optimize the use of network resources in order to allow for a better embedding of future requests. Also, it is generally considered that virtual server requests are known before-hand, which does not provide a decision method for requests arriving in real-time. Furthermore, the proposed algorithms do not consider the interplay between particular cloud resources such as CPU capacity, memory, and storage.

C.4.3 Future Internet Research Projects

We believe also to be of the reader's interest to acquaint with ongoing projects that relate in some way with the content of this chapter. Such projects are the SAIL project [7] and GEYSERS project [39].

GEYSERS' goal is to define a new architecture capable of: seamless and coordinated provisioning of optical & IT resources; end-to-end service delivery to overcome limitations of network domain segmentation; a novel business framework for infrastructure providers and network operators; a novel mechanism to partition infrastructure resources and compose logical infrastructures; a cost and energy-efficient proof-of-concept implementation. GEYSERS proposes also to develop mechanisms that allow infrastructure providers to partition their resources and compose specific logical infrastructures to offer as a service [39].

At a high level view our work and the GEYSERS project aim at a common end: integrate network and IT and support dynamic and on-demand changes in the logical infrastructures. However, GEYSERS clearly assumes the network to be an optical infrastructure which makes its' approaches very technologically specific.

On the other hand, the SAIL project is more network technology agnostic and its aims are to integrate networking with cloud computing to produce cloud networking. The on-demand concept of cloud computing will be extended to the network, and both network and computing resources will be managed according to variable demand. Furthermore VIs can be deployed on demand throughout the cloud network. In one of its perspectives, SAIL extends the general concept of the 4WARD project [40] from VN to cloud computing and in this way it is the research project that more closely relates to our work.

In the following two sections we will present two different approaches to the VI allocation problem, one that relies upon an ILP problem formulation and another that is based on a VN allocation algorithm.

C.5 An Optimal Solution / ILP Problem formulation

It was mentioned earlier that converting the high level objective of mapping as much VIs as possible into a concrete resource management objective problem is not straight forward, especially when dealing with a MOP. In this work we have defined that our overall resource management objective lies upon the combination of three different objectives:

1. To minimize the maximum load consumption of the physical network nodes – *min* $R_{load}^{\max}$.
2. To minimize the maximum load consumption of the physical server nodes – *min* $S_{load}^{\max}$.
3. To minimize the physical bandwidth consumption of a VI – *min* B_{cons}.

The first two objectives balance the load consumption of the physical nodes, which can prevent, if possible, physical nodes from getting fully loaded and therefore become ineligible to host future virtual nodes. The third objective minimizes the overall bandwidth consumption of a VI on the physical infrastructure, trying to save bandwidth for future VIs.

Furthermore, we decided to apply a well known method in the generation of *pareto optimal* solutions in MOPs, which is the aggregation (or weighted) method. The method consists in using an aggregation function to transform a MOP into a mono-objective problem (*MOP*$_\lambda$) by combining the different objective functions (f_i) into a single one (f) as in equation 3:

$$f(x) = \sum_{i=1}^{n} \lambda_i f_i(x), \quad x \in S \tag{3}$$

where the weights $\lambda_i \in [0..1]$ and $\sum_{i=1}^{n} \lambda_i = 1$. We combine the three objectives (which are further detailed later in this section) in a single objective function (equation 4) using the aggregation method:

$$min\ \lambda_1 R_{load}^{\max} + \lambda_2 S_{load}^{\max} + \lambda_3 B_{cons} \tag{4}$$

This way, the objective function reflects the three abovementioned objectives. Note that the values of each variable must be normalized so that the mathematical operation can make sense.

Moreover, we use the well known ILP method to solve the problem. The method consists in the optimization of a linear objective function, subject to linear equality and linear inequality constraints. We now go over the ILP problem formulation.

C.5.1 ILP Problem formulation

Assignment variables

We use two binary assignment variables, x and y, for the VI mapping: one for the virtual nodes, shown in equation 5, where $x_i^m \rightarrow N^V \times N^P$; another for the virtual links, represented in equation 6, where $y_{ij}^{mn} \rightarrow \left(N^V\right)^2 \times \left(N^P\right)^2$ is a 4-dimensional matrix.

$$x_i^m = \begin{cases} 1, \text{virtual node } m \text{ is allocated at physical node } i \\ 0, \text{else} \end{cases} \quad (5)$$

$$y_{ij}^{mn} = \begin{cases} 1, \text{virtual link } mn \text{ uses physical link } ij \\ 0, \text{else} \end{cases} \quad (6)$$

Constraints

A set of constraints are associated to the problem, which are now pointed out.

Each virtual routing nodes and virtual servers are assigned to physical routing nodes and physical servers - equations 7 and 8. Note also that a virtual node is assigned just to one physical node.

$$\forall m \notin S : \sum_{i \notin S} x_i^m = 1 \quad (7)$$

$$\forall m \in S : \sum_{i \in S} x_i^m = 1 \quad (8)$$

Each physical node can only accommodate in maximum one virtual node per VI request, although each physical node can accommodate other virtual nodes from different VIs - equation 9.

$$\forall i : \sum_{i \in N} x_i^m \leq 1 \quad (9)$$

The available capacity of the physical nodes, i.e. CPU load, memory, and storage, cannot be exceeded - equations 10, 11, and 12.

$$\forall i : \sum_m x_i^m \times C_m^V \leq C_i^{P_{free}} \quad (10)$$

$$\forall i : \sum_m x_i^m \times M_m^V \leq M_i^{P_{free}} \quad (11)$$

$$\forall i : \sum_m x_i^m \times STG_m^V \leq STG_i^{P_{free}} \quad (12)$$

CPU frequency requirement must be respected - equation 13 – as well as the number of CPU cores – equation 14, i.e., a selected physical node to host a virtual node must have at least the same number of CPU cores and value of CPU frequency as the virtual node.

$$\forall i : \sum_{m} x_i^m \times Cs_m^V \leq Cs_i^P \tag{13}$$

$$\forall i : \sum_{m} x_i^m \times F_m^V \leq F_i^{P_{free}} \tag{14}$$

In order to optimize the mapping of the virtual links and at the same time cope with the optimization of the virtual nodes the multi-commodity flow constraint [41] with a *node-link* formulation [42] is used – equation 15. The notion of direct flows on the virtual links is also used.

$$\forall m,n \in N^V(m), m < n, \forall i : \sum_{j \in N^P(i)} \left(y_{ij}^{mn} - y_{ji}^{mn} \right) = x_i^m - x_i^n \tag{15}$$

Finally, each physical link selected must have enough bandwidth available to host a virtual link - equation 16. In other words, the amount of free bandwidth in a physical link must be equal or greater than the sum of the amount of bandwidth of the virtual links that go through it.

$$\forall i,j \in N^P(i), i < j : \sum_{m,n \in N^V(m), m<n} B_{mn}^V \times \left(y_{ij}^{mn} - y_{ji}^{mn} \right) \leq B_{ij}^{P_{free}} \tag{16}$$

Constraints derived from the optimization function

Apart from the already presented constraints there are still those that are derived from the optimization function (presented in equation 4), which in our case are three. Equation 17 denotes the maximum load consumption of network nodes, i.e. the load consumption of the network node that is more loaded among all network nodes. The value is given by the sum of the CPU and memory load consumptions multiplied by a fraction of the CPU frequency. This latter one is a division of the CPU frequency of the node by the maximum value of CPU frequency that a physical resource can have in order to normalize the function. This constraint enables to first use physical nodes with lower frequency and to preserve the remaining for virtual nodes with higher frequency demands, while keeping the values normalized. β_1 and β_2 represent the weights of each resource component. The weights allow us to define which resource, CPU load or memory load, is more important on the overall load consumption of the node.

Equation 18 is very similar to equation 17 and represents the load consumption of server nodes, being given by the sum of the CPU, memory, and storage load consumptions multiplied by a fraction of the CPU frequency, for the same reason as in equation 17. δ_1, δ_2, and δ_3 represent the resource components weights.

As to the third objective, represented by equation 19 it is denoted by the sum of the substrate bandwidth currently in use with the substrate bandwidth that the VI will consume, divided by the substrate bandwidth capacity.

$$\forall i \notin S : \frac{F_i^P}{F_{\max}^P} \times \left[\beta_1 \frac{C_i^{P_{used}} + \sum_m x_i^m \times C_m^V}{C_i^{P_{total}}} + \beta_2 \frac{M_i^{P_{used}} + \sum_m x_i^m \times M_m^V}{M_i^{P_{total}}} \right] \leq R_{load}^{\max} \tag{17}$$

$$\forall i \notin S : \frac{F_i^P}{F_{\max}^P} \times \left[\delta_1 \frac{C_i^{P_{used}} + \sum_m x_i^m \times C_m^V}{C_i^{P_{total}}} + \delta_2 \frac{M_i^{P_{used}} + \sum_m x_i^m \times M_m^V}{M_i^{P_{total}}} + \delta_3 \frac{STG_i^{P_{used}} + \sum_m x_i^m \times STG_m^V}{STG_i^{P_{total}}} \right] \leq S_{load}^{\max} \tag{18}$$

$$\frac{\sum_{i,j \in N^P(i), i<j} B_{ij}^{P_{used}} + \sum_{m,n \in N^V(m), m<n} y_{ij}^{mn} \times B_{mn}^{V}}{\sum_{i,j \in N^P(i), i<j} B_{ij}^{P_{total}}} \leq B_{cons} \quad (19)$$

The optimization function tries to minimize the sum of these three maximum load consumption values, not each individual value. Therefore, it will look for the solution that provides a better interplay among the three inherent objectives.

C.6 Heuristic Algorithm

In this section another approach to solve the VI assignment problem is used by presenting a heuristic algorithm. The heuristic is based on the algorithm proposed by Nogueira *et al.* [24].

The goal of the mapping algorithm proposed by Nogueira *et al*. [24] is to maintain a low and balanced stress of both nodes and links of the substrate network, where the stress parameter combines one or more values of usage of different features of a resource into a single indicator of how much the resource is being used. With the mapping goal in mind, the algorithm starts by determining each node's stress (which depends on CPU Load, processor frequency, memory, and number of virtual machines running over the physical node) and by ordering the mapping of the virtual nodes by starting with those with smallest number of candidate physical hosts. The physical links' stress (allotted bandwidth of the physical link) is also calculated. The algorithm then uses these parameters to calculate the node and link stresses. Then, it uses both node and link stresses to calculate the potential of a node to be chosen as host of the virtual node, by multiplying the node stress by the link cost, which is a value composed by the link stresses of the physical paths from the candidate to the virtual neighbour candidates. After this, the following virtual node in order is mapped until every virtual node is mapped onto a physical host. Finally, the virtual links are mapped onto the physical links using a Dijkstra algorithm considering the physical links' stress.

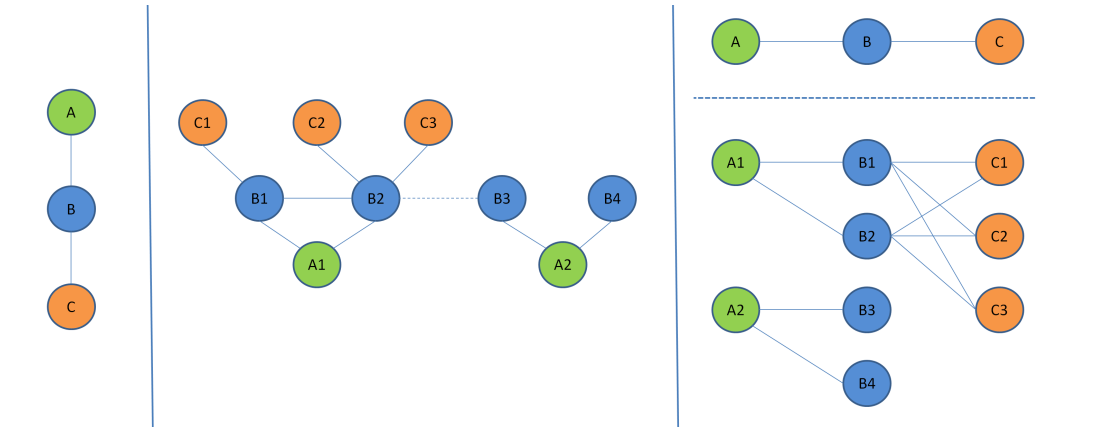
Improvements to the original algorithm were performed by introducing a mechanism which increases the likelihood of finding a mapping solution by taking into account the impact that each mapping option has on the mapping possibilities for other virtual nodes. We name that mechanism interdependency mapping. Moreover, we work on the formulas to calculate the stresses for network nodes and server nodes in order to increase the amount of embeddable virtual resources. Linear and non-linear approaches to node and server stress calculation are studied and compared through simulation.

C.6.1 Interdependency mapping

The algorithm proposed by Nogueira *et al.* [24] starts by producing a list of physical nodes that possess the adequate hardware features to host each virtual node: the virtual node's *candidate list*. However, one should also consider that a host needs not only to have the physical capacity to host the virtual node, but also to adequately connect to other hosts. This means that, for a physical node to be a valid candidate to host a specific virtual node, it needs to be able to establish an adequate physical link with at least one of the candidates to each of its virtual node's neighbors, according to the QoS requirements of each virtual link between the virtual nodes. If the host does not fulfill these conditions, it should not be considered as a candidate, and it should be removed from the virtual node list of candidates. This removal also implies that candidates that would host neighboring virtual nodes that could only connect to this candidate (as host of a specific virtual node) should be removed as well.

Thus, we propose that, for each virtual link, the algorithm verifies if there is at least one physical path available, with the virtual link QoS requirements, between each pair of candidates to the virtual source and destination of the link. These possibilities of connection are registered and the candidates that do not possess at least one possible connection to one candidate of each virtual neighbor node to the virtual node they are applying are removed. Each candidate removal along the mapping algorithm is followed by a check of the remaining possible connections for the candidates that had possible connections to the removed candidate, and a removal of those candidates will take place if appropriate.

Let us take the example of a VI and a physical infrastructure in which we want to map the VI, see Figure C-3.



(a) VI - example *(b) Physical Infrastructure – example* *(c) Possible connections*

Figure C-3: Example of a VI and the Physical Infrastructure in which is being mapped

Let us consider that a VI is made of 3 nodes - A, B, and C - each of them having a set of candidates which are represented with their letters and numbers in Figure C-3(b). Assume that the physical link between B2 and B3 is unfit to host any virtual link (the reason is irrelevant). The mapping of the possible connections between each pair of candidates to each pair of virtual neighbor nodes would result in a representation as depicted in Figure C-3(c).

In this case, every candidate node would be checked up after the mapping of the possible connections. Those that did not have at least one connection to other virtual neighbor nodes candidates would be removed. This is the case of candidates B3 and B4 which do not have any possible connection to any candidate of C, which is a virtual neighbor of B. So, B3 and B4 would be removed. As a consequence, A2 would also not have any connection to any candidate of node B, the neighbor of virtual node A, thus eliminating candidate A2 as well. Since now there is only candidate A1 for virtual node A, A will be mapped to A1. The following virtual node with less candidates is B. It is possible to choose B1 or B2 to host B. When choosing B1, B2 will be removed together with its connections to A1 and to C1, C2 and C3. But these candidates will not be removed since they still have possible connections to the virtual neighbors' candidates. Afterwards, a physical host for node C is chosen, for example, C2. If this procedure had not been used to remove unfit candidates, one might have started by selecting A2 to host the virtual node A. Afterwards the algorithm would choose C1, C2 or C3 to host C. But when it would check B, none of the candidates to B would be able to connect to A2 and to C1, C2 or C3 simultaneously, thus it would not find a mapping solution.

Although this method removes several inadequate candidates, there are some situations where inadequate candidates are not found to be inadequate before they are chosen as hosts. We propose that, before selecting a candidate node as host for a certain virtual node, the candidates list of each virtual node and data of the current possible connections are saved. If the selection of this candidate makes all other candidates for all other virtual nodes inadequate, the data is restored and the (wrongly) chosen candidate is removed from the candidate list and a new candidate is chosen.

C.6.2 Integrating Cloud and VN mapping

In order to map cloud and network resources in an integrated fashion, we expand the algorithm proposed by Nogueira *et al.* [24] and introduce physical servers and virtual servers as cloud elements. The main difference from cloud resources (servers) to the other resources (routing nodes) is that cloud resources also include storage capacity. The final pseudo-code of the VI mapping algorithm considering long interdependency is presented in Algorithm D-1.

***Algorithm D-1*: Pseudo-Code of the VI Algorithm**

```
      input : Substrate (Substrate Network) , V_Request (Requested VI)
      output: V_Map (Mapped VI)

 1    foreach Link i in Substrate.Links do
 2        S_LS(i) =CalcLinkStress(Link(i)) ;
 3    end
 4    foreach Node i in Substrate.Nodes do
 5        if Node(i).Server==true then
 6            S_Ni = CalcServerStress(Node(i)) ;
 7        else
 8            S_Ni = CalcRouterStress(Node(i)) ;
 9        end
10    end
11    foreach Node n in V_Request.Nodes do
12        n.Candidates.(i) = FindCandidates(Substrate.Nodes) ;
13    end
14    foreach Link v in V_Request do
15        PossiblePath(v) = FindPossiblePath();
16    end
17    foreach Node i in V_Request do
18        RemoveCandidatesWithoutAnyPossiblePathToOneVirtualNeighbor(i);
19    end
20    while 9 Node x in V_Request.Nodes | NumberOf(Node(x).Candidates) >1 do
21        n = SelectUnmappedNodeWithLessCandidates(V_Request.Nodes);
22        foreach SourceCandidate v in n.Candidates do
23            π(v) = CalculateNodePotential(v) ;
24        end
25        n.Map = v : π(v) = min(π) ;
26        SaveData(ListsOfCandidates, PossiblePath);
27        RemoveNonSelectedCandidates(n);
28        if NumberOf(n.Candidates)==0, 8 n in V_Request.Nodes then
29            RestoreData(ListsOfCandidates, PossiblePath);
30            RemoveCandidate(n,n.Map);
31        end
32    end
33    foreach Node n in V_Request.Nodes do
34        V_Map.Nodes U n ;
35        foreach Link k connected to n do
36            ConnVNode=GetLinkDestination(k) ;
37            V_Map.Links [ CSFP_Dijkstra(n.Map,ConnVNode.Map) ;
38        end
39    end
```

C.6.3 Node and Server Stress calculation

The same principles used for routing nodes' stress are used to calculate the servers' stress; the link stress formula is kept unchanged. This way, the interplay between server stress and link stress (with the link stresses aggregated in the link cost indicator) allows for a node placement that considers both the servers' characteristics as well as the network node characteristics (and the routers as well).

C.6.4 Non-proportional and Proportional approaches

We suggest a new way of deriving the node and server stress, where the goal is to provide mapping solutions that maximize the embedding of VIs onto the same physical substrate, in spite of the possible load unbalance. Since Nogueira *et al.* [24] pointed out that the bandwidth of the physical links was the main

constraint to the amount of embeddable virtual resources. We make efficient link use the most important aspect (i.e., using short physical paths to host virtual links), at least as long as the physical nodes still have a minimum amount of free resources as tuned through constant k. The proposed non-proportional node and server stress formulas presented in equations 20 and 21:

$$S_N = \text{Number of Active VMs} . \left(1 + \frac{k.C^V_{med\,\text{Req}}}{C^{P_{free}}}\right)\left(1 + \frac{k.M^V_{med\,\text{Req}}}{M^{P_{free}}}\right) \quad (20)$$

$$S_S = \text{Number of Active VMs} . \left(1 + \frac{k.C^V_{med\,\text{Req}}}{C^{P_{free}}}\right)\left(1 + \frac{k.M^V_{med\,\text{Req}}}{M^{P_{free}}}\right)\left(1 + \frac{k.STG^V_{med\,\text{Req}}}{STG^{P_{free}}}\right) \quad (21)$$

In these equations, $M^V_{med\,\text{Req}}$ represents the average memory of the virtual nodes and virtual servers, $C^V_{med\,\text{Req}}$ represents the average CPU load increase for each virtual node / server embedded, $STG^V_{med\,\text{Req}}$ represents the average storage memory of the virtual servers and *k* represents a constant value. This way link cost will be the most important parameter to calculate the potential until the considered nodes achieve a critical level of occupied resources (that can be adjusted through constant *k*). Equation 21, for server stress calculation considers storage memory the same way that memory and CPU load are considered in equation 20.

As for the proportional approach, we consider that the node stress is calculated as proposed by Nogueira *et al.* [24] and that the server stress is calculated according to similar principles, as seen in equations 22 and 23:

$$S_N = \frac{\text{Number of Active VMs}}{\delta + M^{P_{free}} \cdot C^{P_{free}} \cdot F} \quad (22)$$

$$S_S = \frac{\text{Number of Active VMs}}{\delta + M^{P_{free}} \cdot STG^{P_{free}} \cdot C^{P_{free}} \cdot F} \quad (23)$$

It should be noted that, from equations 22 and 23, we chose not to include the *F* (CPU frequency) parameter since in [43] the role of this parameter was evaluated, and it was shown that its removal would have a negligible impact.

C.7 Simulation Results

This section presents the results over the different approaches. First an insight on the simulator and on some considerations are given followed by the simulation results.

C.7.1 Simulator

In order to analyze the behavior of the allocation approaches, a Matlab® [44] simulator was used. For each run, the program designs a random physical infrastructure and it simulates a set of requests of VIs, according to a pool of parameters, with Markov-modulated inter-arrival and inter-departure times. The refered pool of parameters is described in Table C-1.

Moreover, both physical substrate and VIs generated have 20% of the nodes as servers (rounded to the higher integer), and the remaining 80% as routing nodes. The same substrate and VI requests are used for the study of the different approaches. When not used as independent variables, the VI request rate is λ = 2 VIs per time unit (Poisson arrivals), and the average duration is $1/\mu$ = 20 time units (exponentially distributed duration), where μ is the average service rate. The virtual servers' characteristics are based on the Amazon's EC2 instance types [45]. Each scenario runs 10 times, each with 500 time units. The results presented have a 95% confidence interval.

Table C-1: Simulation parameters

		Physical Networks	Virtual Networks
Router Nodes	**N. CPUs**	{2; 4; 6; 8}	{1; 2; 3; 4 }
	CPU Freq(Hz)	{2.0-3.2 / 0.2 steps}	{2.0-3.2 / 0.1 steps}
	Memory	{2; 4; 6}(GB)	{64; 128; 256; 512}(MB)
Server Nodes	**N. CPUs**	{8; 16; 32; 64}	{1; 2; 4; 8; 16; 32; 64}
	Storage (GB)	{6400; 12800; 25600}	{100; 200; 400; 800; 1600}
	Memory (GB)	{256; 512; 1024}	{2; 4; 8; 16; 32; 64}
Links	**Bandwidth (Mbps)**	{800; 1200}	{34.368 139.264}

With respect to mapping algorithms, the simulator is prepared to support several ones and to provide as output a comparative set of results. These results comprise the averaged time values for parameters such as: the acceptance ratio of VI requests, memory in use, storage in use, CPU load in use, virtual nodes per physical node, occupied bandwidth, and mapped virtual bandwidth. Note that, when comparing different mapping algorithms, the same physical and virtual sets are used for the different mapping algorithms. In order to solve the ILP we have used CPLEX [46] version 11, integrating a plug-in for Matlab® and setting a time limit of 600 seconds for each VI mapping, a value which was never overcome during our experiments.

A priori ILP formulation

We followed an a priori approach that consists in having the weights λ_i defined not by the solver, but according to the decision maker preferences, i.e. our own preference [16]. Regarding the optimization function (equation 4), we decided to split in two the overall weight value between node and link constraints. Moreover, taking into consideration the fact that according to Nogueira et.al [24] the links' capability is a main limiting factor on the mapping process, we therefore provide the B_{cons} objective a higher weight. In this sense we apply a weight of 0.5 to B_{cons} (λ_3), and a weight of 0.25 to both λ_1 and λ_2. By doing this we try to keep a balance between node load occupation, from both routing and server nodes, and substrate link occupation.

With respect to the weight variables of the nodes' resource components (equations 17 and 18), we consider equal values for each component, i.e. in the case of routing nodes, which take into account CPU load and memory load, the values of β_1 and β_2 are equal to 0.5; in the server node case, that considers CPU, memory and storage load, the values of δ_1, δ_2, and δ_3 were set to 0.333.

Heuristic node critical level tuning

In the heuristic algorithm, the link cost is considered the most important parameter to calculate the potential until the considered nodes achieve a critical level of occupied resources. This level is adjusted through constant *k*. After a thorough analysis through simulation, we reached the best values for *k* = 3, a value that is used in the presented results.

C.8 Evaluation

In the evaluation we focus on the most relevant results: the VI acceptance ratio, link parameters, and node parameters are presented.

In the following figures, the term *H-P* corresponds to the heuristic algorithm proportional stress approach, where server stress is inversely proportional to the server's free resources. The heuristic algorithm non-proportional stress approach, where server stress is calculated in a non-linear way, is refered to as *H-NP*. As to the ILP approach, it is denoted in the same way, by *ILP*.

VI Acceptance Ratio

From the results obtained in Figure C-4, one may observe that the heuristic algorithm presents a better performance when using the non-proportional node stress approach. When varying the number of

substrate nodes, the non-proportional approach starts to stand out as the number of substrate nodes increases. On the other hand, when maintaining the substrate size and varying the number of VI requests per time unit, the non-proportional approach performance starts to get closer to the proportional one. Nevertheless the performance of the non-proportional approach is always better than the one from the proportional.

As to the ILP approach, it stands out that it presents stable values higher than the heuristic non-proportional approach around an average of 5% in both substrate node variation and number of VI requests per time unit. Note also that, as the number of VI requests per time unit increases, the performance of the heuristic approaches tend to approximate while the ILP, although presenting a slow decrease, maintains a noticeable better performance.

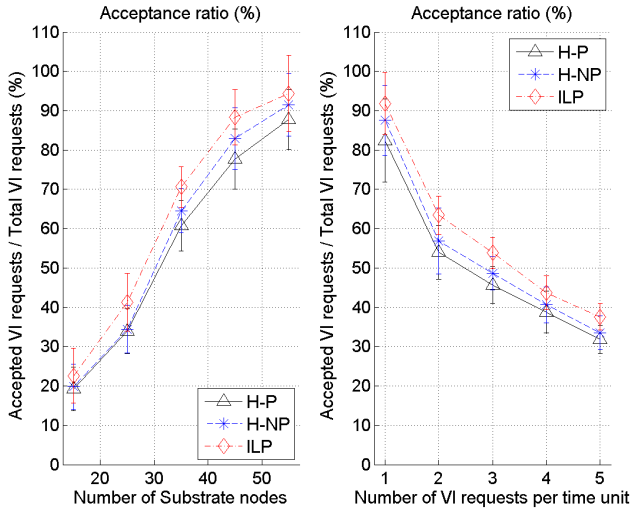


Figure C-4: VI Acceptance Ratio - fraction of successfully mapped VIs

Link parameters

With respect to the link parameters, i.e. bandwidth, by looking to Figure C-5 we can observe, denoted by H-P-VN, H-NP-VN, and ILP-VN, the amount of virtual bandwidth of the embedded VIs divided by the total bandwidth of the substrate network. Denoted by H-P, H-NP, and ILP, we observe the amount of bandwidth of the substrate that is being used by the VIs, divided by the total bandwidth of the physical network. These values are not the same because a virtual link can span through one or more physical links. This provides a great opportunity for optimization, since a better mapping should be able to reduce the amount of substrate bandwidth necessary to host the virtual links. The difference between what VIs require and what they actually end up occupying is considerably high as well as its absolute value. Values can reach up to 80% of occupied bandwidth for 5 VIs per time unit, more than the double of what the VIs actually require.

Moreover, the difference in the acceptance ratio between the approaches is easily understandable observing the value difference in the values presented in Figure C-5, where bandwidth values increase as the number of VI requests per time unit increases. Still in this figure, the behaviour of the average amount of storage in servers is presented. The values never reach a critical level and, with no major surprise, the overall behaviour seems constant, with values increasing as the number of VI requests increases.

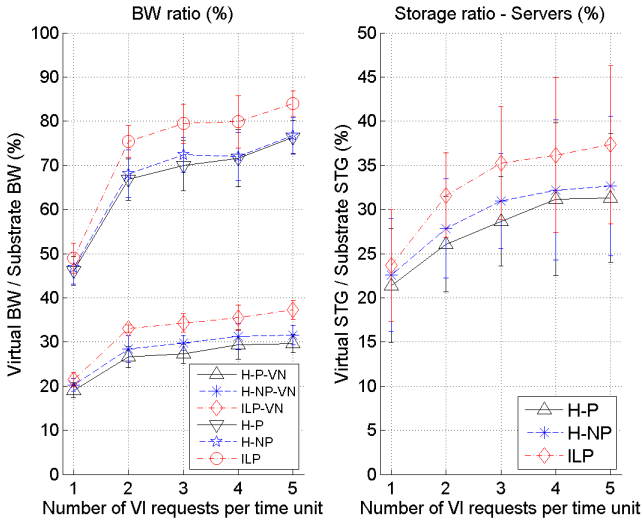


Figure C-5: Bandwidth Ratio - VI bandwidth over substrate bandwidth capacity & Storage ratio

Node parameters

Figure C-6 presents results of the average number of virtual nodes that are hosted per physical node, making a distinction between routing nodes and server nodes. The overall behaviour is identical to the already presented ones, with the ILP approach having higher values because of its higher VI acceptance rate. Nevertheless there is an aspect that is interesting to highlight with respect to the heuristic. While in the ILP case the average number of virtual nodes increases in a convergent way, the heuristic presents a change in this behaviour as it reaches the rate of 5 VI requests per time unit. In this latter case the average number of virtual nodes decreases when increasing the rate from 4 to 5. This can represent a turning point in the heuristic performance if this behaviour tends to maintain itself as the number of VI requests per time unit increases. This aspect will be further analyzed in the future.

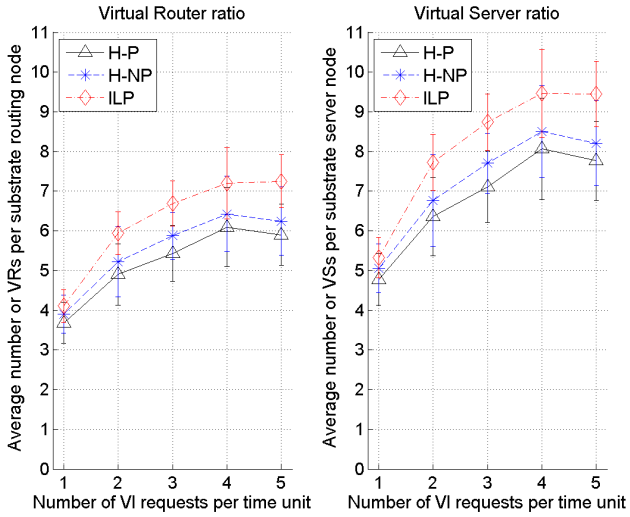
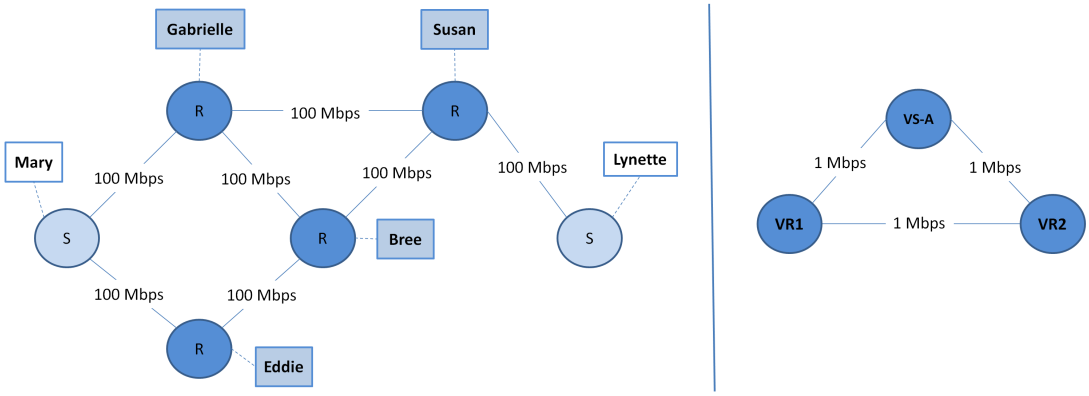


Figure C-6: Average number of virtual routers and virtual servers per number of VI requests

C.9 Testbed & Experimental Results

C.9.1 Testbed Description

The testbed is composed by 6 physical nodes (four network nodes and two server nodes) and is connected according to Figure C-7(a) obtained from the developed virtualization platform and modified to indicate the nodes roles. Table C-2 presents the characteristics of each physical node.



(a) Tested *(b) Mapped VI*

Figure C-7: Tested description & Mapped VI

Table C-2: Characteristics of the Tesbed Machines

Name	**Susan**	**Lynette**	**Eddie**	**Mary**	**Gabrielle**	**Bree**
CPU Freq. (GHz)	3.40	3.40	2.40	2.66	2.13	3.00
CPU Cores	2	2	4	4	2	2
HDD Memory (GB)	89	40	303	277	145	195
RAM Amount (GB)	6	6	6	6	4	6

C.9.2 Experiment Description

In the experiment, there is a standard VI to be mapped, which is presented in Figure C-7(b). Virtual router *VR1* is restricted to be mapped either in *Susan* or in *Eddie*, by using geographical restrictions, while *VR2* is restricted to be mapped on *Gabrielle*. *VS-A* (virtual server A) can be mapped either in *Mary* or in *Lynette*. All pre-existing VIs were erased and mapped 39 of these VIs in sequence, and repeated this process 3 times. The results show the node occupation as a function of the number of pre-existing VIs, both for the server and the routing nodes.

C.9.3 Results

Figure C-8 presents the occupation of server and router machines. In each mapped VI there is a virtual server that can be mapped either in *Mary* or in *Lynette*. It is visible that *Lynette* has significantly more free storage than *Mary*, thus expectantly making it more prone to host the virtual server of each mapped VI. This is confirmed by the experimental results, as virtual servers tend to be hosted in *Mary*, while *Lynette* only gets a smaller portion of them. As *Lynette's* free storage gets smaller, the frequency with which it hosts new virtual servers is also reduced. Meanwhile, *Susan* and *Eddie* dispute one of the VRs. With respect to the route machines, both *Susan* and *Eddie* get a similar number of virtual routers, with a regular difference in occupation, where *Eddie* is slightly more occupied than *Susan*. This is to be expected, as *Eddie* has the double number of cores of *Susan*, thus making *Eddie* more prone to accept more virtual resources. It should be noted that *Susan* has a higher CPU frequency than *Eddie*, but it is not even near double the CPU frequency of *Eddie*.

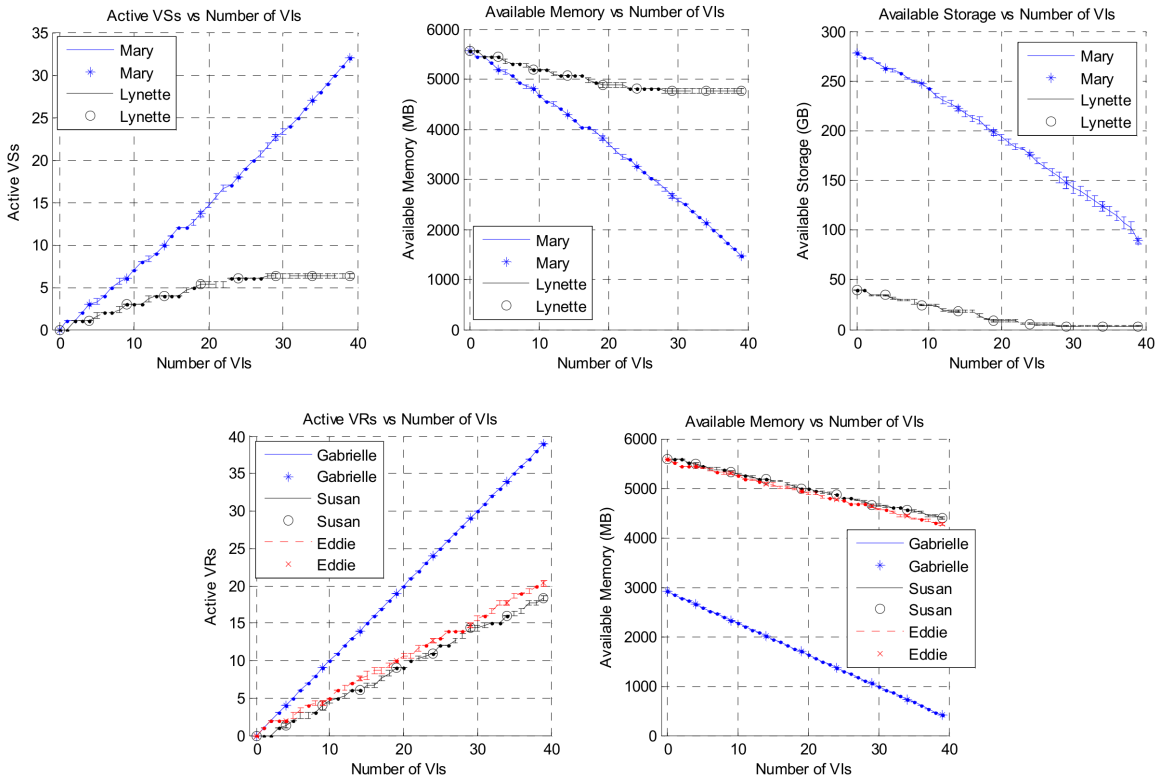


Figure C-8: Use of resources in the server and router nodes as a function of the number of mapped VIs

In this section the mapping decisions for VIs were analyzed and it was shown that machines with more resources or with better connections for other physical nodes are able to host more resources. It was also possible to observe how link and node stress interplayed and influenced mapping decisions, and how physical machines with a critical level of free resources tended not to be used.

C.10 Future Research Directions

Regarding future developments, we aim at extending the research on resource allocation in virtualization environments where cloud and network resources coexist. We will further analyse the ILP formulation in order to understand if there is a formulation capable of improving performance, i.e., improving the successful VI mapping ratio, whether by adjusting weight values or by modifying objectives. Moreover, power consumption considerations will be introduced in both ILP and heuristic approaches.

Resource allocation is one of the resource management challenges identified in a cloud networking environment. In dynamic scenarios such as those that we consider, reconfiguration is a key feature to better exploit the efficiency of a physical infrastructure. Therefore, we intend to integrate a reconfiguration mechanism in our resource management stack.

Finally, as it was highlighted in the beginning of this chapter, network virtualization is not the only possible approach to a future cloud networking scenario, and we clearly identified SDN as another strong possibility. In this sense we plan to study an approach based on SDN.

C.11 Conclusion

This chapter presented a future scenario in which cloud and network resources coexist in a single environment, the operator's network. We looked into a future that may not be that far away. The cloud world is evolving at an astonishing rate and the network is suffering considerable pressure to keep up with it and also to foster its vision. A possible future scenario was presented, where cloud resources are no longer confined to DCs, but may be spread throughout the network. Moreover, as current deployed

technologies are not fully capable of coping with the cloud dynamism, network virtualization was considered as a component of the next network generation, bringing to life a scenario where all resources, cloud and network, can be virtualized. Such an environment requires an integrated management of resources and several challenges were identified, among which the resource allocation problem addressed in this chapter. Our work focused on the most basic layer of the cloud stack, IaaS, considering the provisioning of VIs. The problem was described in detail followed by a review on the most relevant state-of-the-art works in the area.

An ILP formulation to address the VI allocation problem was proposed as well as a heuristic algorithm. Regarding the latter one two approaches were studied. Both approaches aim to take the most out of a physical infrastructure, i.e. to allocate as much VIs as possible. To understand the operation of both approaches a simulation analysis was performed. In this analysis it was clear that the ILP approach performs better, but nevertheless we can say that the heuristic approach is not that far behind.

Finally, experimental results over a real testbed were retrieved in order to confirm and evaluate the performance of the heuristic algorithm under a real testbed.

References

[1] Mell, P., & Grance, T. (2011). The NIST Definition of Cloud Computing. The NIST Definition of Cloud Computing . http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.

[2] Rosen, E., & Rekhter, Y. (February de 2006). BGP/MPLS IP Virtual Private Networks (VPNs). BGP/MPLS IP Virtual Private Networks (VPNs) .

[3] Kompella, K., & Rekhter, Y. (January de 2007). Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling. Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling .

[4] Lasserre, M., & Kompella, V. (January de 2007). Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling. Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling .

[5] Akamai (2011). Can Cloud and High Performance Co-Exist?

[6] Cisco. (2009). The Cisco Powered Network Cloud: An Exciting Managed Services Opportunity.

[7] FP7 Project “Scalable and Adaptive Internet Solutions” (SAIL). (s.d.).

[8] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., et al. (2009). Above the Clouds: A Berkeley View of Cloud Computing. Tech. rep., EECS Department, University of California, Berkeley.

[9] Cisco. (2012). Software-Defined Networking: The New Norm for Networks.

[10] Pei, D., & Van der Merwe, J. (2006). BGP convergence in virtual private networks. Proceedings of the 6th ACM SIGCOMM conference on Internet measurement (pp. 283-288). New York, NY, USA: ACM.

[11] Carapinha, J., & Jiménez, J. (2009). Network virtualization: a view from the bottom. Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures (pp. 73-80), New York, USA.

[12] Chowdhury, N., Boutaba, R., (2009). Network virtualization: state of the art and research challenges. IEEE Communications Magazine, vol.47, no.7, pp.20,26.

[13] Nicira. (2012a). The Seven Properties of Network Virtualization. Retrieved October 2012 from http://nicira.com/sites/default/files/docs/Nicira%20-%20The%20Seven%20Properties%20of%20Virtualization.pdf

[14] Nogueira, J., Melo, M., Carapinha, J., & Sargento, S. (2011). Network Virtualization System Suite: Experimental Network Virtualization Platform. TridentCom 2011, 7th International ICST Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities.

[15] Nicira. (2012b). http://nicira.com/

[16] Hoboken. (2009). Metaheuristics : from design to implementation . John Wiley \& Sons .

[17] Zhu, Y., & Ammar, M. (2006). Algorithms for Assigning Substrate Network Resources to Virtual Network Components. INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings, (pp. 1-12).

[18] Chowdhury, N., Rahman, M., & Boutaba, R. (2009). Virtual Network Embedding with Coordinated Node and Link Mapping. INFOCOM 2009, IEEE, (pp. 783-791).
[19] Lischka, J., & Karl, H. (2009). A virtual network mapping algorithm based on subgraph isomorphism detection. Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures (pp. 81-88). New York, NY, USA: ACM.
[20] Lu, J., & Turner, J. (2006). Efficient Mapping of Virtual Networks onto a Shared Substrate. Tech. rep., Washington University in St. Louis, Department of Computer Science and Engineering.
[21] Houidi, I., Louati, W., & Zeghlache, D. (2008). A Distributed Virtual Network Mapping Algorithm. Communications, 2008. ICC '08. IEEE International Conference on, (pp. 5634-5640).
[22] Yu, M., Yi, Y., Rexford, J., & Chiang, M. (2008). Rethinking virtual network embedding: substrate support for path splitting and migration. SIGCOMM Comput. Commun. Rev. , 38 (2), 17-29.
[23] Farooq Butt, N., Chowdhury, M., & Boutaba, R. (2010). Topology-awareness and reoptimization mechanism for virtual network embedding. Proceedings of the 9th IFIP TC 6 international conference on Networking (pp. 27-39). Berlin, Heidelberg: Springer-Verlag.
[24] Nogueira, J., Melo, M., Carapinha, J., & Sargento, S. (2011). Virtual network mapping into heterogeneous substrate networks. Computers and Communications (ISCC), 2011 IEEE Symposium on, (pp. 438-444).
[25] Chowdhury, N.., Rahman, M., & Boutaba, R. (2012). ViNEYard: Virtual Network Embedding Algorithms With Coordinated Node and Link Mapping. Networking, IEEE/ACM Transactions on , 20 (1), 206-219.
[26] Botero, J., Hesselbach, X., Fischer, A., & De Meer, H. (2011). Optimal mapping of virtual networks with hidden hops. Telecommunication Systems, 1-10.
[27] Melo, M., Carapinha, J., Sargento, S., Torres, L., Tran, P. N., Killat, U., et al. (2012). Virtual Network Mapping - An Optimization Problem. In K. Pentikousis, R. Aguiar, S. Sargento, R. Aguéro, O. Akan, P. Bellavista, et al. (Edits.), Mobile Networks and Management (Vol. 97, pp. 187-200). Springer Berlin Heidelberg.
[28] Roy, N., Kinnebrew, J., Shankaran, N., Biswas, G., & Schmidt, D. (2008). Toward Effective Multi-Capacity Resource Allocation in Distributed Real-Time and Embedded Systems. Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on, (pp. 124-128).
[29] Li, L., & Tang, M. (2010). Novel spectral method for server placement in CDNs. Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on, (pp. V6-197-V6-199).
[30] Enokido, T., Aikebaier, A., Takizawa, M., & Deen, S. (2010). Power Consumption-Based Server Selection Algorithms for Communication-Based Systems. Network-Based Information Systems (NBiS), 2010 13th International Conference on, (pp. 201-208).
[31] Enokido, T., Aikebaier, A., Takizawa, M., & Deen, S. (2010). Energy-Efficient Server Selection Algorithms for Network Applications. Broadband, Wireless Computing, Communication and Applications (BWCCA), 2010 International Conference on, (pp. 159-166).
[32] Bouyoucef, K., Limam-Bedhiaf, I., & Cherkaoui, O. (2010). Optimal allocation approach of virtual servers in cloud computing. Next Generation Internet (NGI), 2010 6th EURO-NF Conference on, (pp. 1-6).
[33] Csorba, M., Meling, H., & Heegaard, P. (2010). Ant system for service deployment in private and public Clouds. Proceeding of the 2nd workshop on Bio-inspired algorithms for distributed systems, ACM, (pp. 19-28).
[34] Jiang, W., shen, R. Z., Rexford, J., & Chiang, M. (s.d.). Cooperative Content Distribution and Traffic Engineering in an ISP Network. Cooperative Content Distribution and Traffic Engineering in an ISP Network.
[35] Kantarci, B., & Mouftah, H. (2012). Minimizing the provisioning delay in the cloud network: Benefits, overheads and challenges. Computers and Communications (ISCC), 2012 IEEE Symposium on, (pp. 806-811).
[36] Kantarci, B., & Mouftah, H. (2012). Optimal Reconfiguration of the Cloud Network for Maximum Energy Savings. Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on, (pp. 835-840).

[37] Kantarci, B., & Mouftah, H. (2012). Overcoming the energy versus delay trade-off in cloud network reconfiguration. Computers and Communications (ISCC), 2012 IEEE Symposium on, (pp. 53-58).
[38] Soares, J., Carapinha, J., Melo, M., Monteiro, R., & Sargento, S. (2012). Resource allocation in the network operator's cloud: A virtualization approach. Computers and Communications (ISCC), 2012 IEEE Symposium on, (pp. 800-805).
[39] FP7 Project “Generalised Architecture for Dynamic Infrastructure Services” (GEYSERS).
[40] FP7 Project "4WARD – Architecture and Design for the Future Internet" (4WARD). (s.d.).
[41] Even, S., Itai, A., & Shamir, A. (1975). On the complexity of time table and multi-commodity flow problems. Foundations of Computer Science, 1975., 16th Annual Symposium on, (pp. 184-193).
[42] Pioro, M., & Medhi, D. (2004). Routing, Flow, and Capacity Design in Communication and Computer Networks. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.
[43] Monteiro, R. (2011). Creation and Reconfiguration of Virtual Networks from an Operator Point of View. Master's Thesis, Universidade de Aveiro.
[44] Mathwords. (2010). Matlab Simulator 2010.
[45] Amazon Web Services. Amazon Elastic Compute Cloud. Retrieved February 2012 from http://aws.amazon.com/ec2/instance-types/
[46] IBM. IBM ILOG Optimization Products. Retrieved February 2012 www-01.ibm.com/software/websphere/products/optimization

Paper D. Optimizing the Embedding of Virtualized Cloud Network Infrastructures

João Soares, João Aparício, Susana Sargento

submitted to IEEE Transactions on Network and Service Management, 2014

The format has been revised

Optimizing the Embedding of Virtualized Cloud Network Infrastructures

João Soares, João Aparício, Susana Sargento

Abstract

So far, cloud computing offers, namely those at the infrastructure level, have focused on providing computing and storage resources, and consider the network mostly as a required communication add-on and not truly as a resource itself. In other words, the ability to define network resources, such as routing/switching elements, bandwidth and delay, is still very limited. However, the need to have more robust solutions is becoming clearer. In this paper we argue that, in a near future, cloud infrastructure services will evolve and allow the definition of complete infrastructures that comprise both computing and network resources, which we refer to as Virtual Infrastructures (VI).

This paper addresses the VI embedding problem through the joint virtualization of computing and network resources. In order to address this problem, an Integer Linear Programming (ILP) formulation is presented and different embedding strategies are proposed: load balancing strategy, that combines physical resources load balancing with minimal bandwidth consumption; green strategy, that reduces the energy consumption of the physical infrastructure by minimizing the number of active resources; dynamic strategy, that combines both load balancing and green strategies depending on the substrate state. In order to enable a real-time optimization, we also propose a heuristic approach. Finally, the performance of the different strategies is carefully evaluated.

Keywords - Cloud networking, heuristic approaches, integer linear programming, virtualization, virtual infrastructure.

D.1 Introduction

The importance of the network on cloud computing is today highly recognized due to its fundamental role in guaranteeing performance, reliability, and security to cloud services both inside DCs as well as outside, i.e. on the Wide Area Network (WAN). An increasing number of services that are being migrated to the cloud require, not only computing, but also network guarantees. The Telco sector is an example, as it seems eager to "cloudify" some of its solutions. The momentum around the Network Functions Virtualization (NFV) working group within the European Telecommunications Standards Institute (ETSI) [1], which has brought together most leading network operators, shows the relevance of cloud in the networking environments. Complete systems such as the Evolved Packet Core (EPC) and IP Multimedia Subsystem (IMS) are already virtualized and prepared to be deployed in the cloud and provided "as-a-service". However, these services have very strict requirements both in terms of computing and network that cannot be met in current cloud offers.

The concept of Network as a Service (NaaS) is not new, but the cloud computing "boom" has re-leveraged it. Connectivity as a Service (CaaS) [2], dynamic VPNs [3] or Bandwidth on-demand [4] are some of the service concepts that have been brought up recently under the NaaS concept. Nevertheless, it is important to notice that the concept of NaaS is much broader and can go up to providing a full network as a service, i.e. network elements (nodes) and connectivity elements (links). Just like the current cloud offers, the key enabler behind the concept lays upon virtualization techniques.

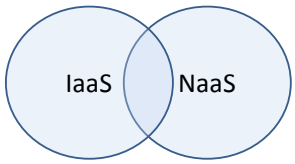


Figure D-1: Mapping IaaS and NaaS [5].

We argue that, in a near future, the NaaS and IaaS concepts will tend to merge – see Figure D-1. In other words, cloud and network resources will become part of a single pool of resources managed in an integrated way. Resource management is one of the challenges that such an environment rises. This integration can be achieved at different levels, i.e. within a single domain (single cloud service provider), across different domains, and it can even include the cloud service consumer. The latter two cases require that Telcos be part of the solution, as they are the ones controlling the WAN.

However, traditional network services were not conceived to handle properties such as elasticity, self-provisioning or on-demand resources. Due to the need for a more dynamic network, the concept of Software Defined Networking (SDN) emerged. Among the advantages of SDN is the easy and agile adaptation of the network, following requirements raised by applications' dynamics and business requirements. The work of this paper is laid out with the SDN concept in mind.

This paper tackles the joint embedding of cloud and network resources, a NP-hard problem, through Integer Linear Programming (ILP). We consider the provisioning of cloud and network resources through virtualization (virtualized computing, storage, and network), enabling the support of Virtual Infrastructures (VIs) that contain both types of resources. We present three main embedding strategies: the first two strategies balance the load of the physical resources; the third one minimizes the energy consumption of the overall physical infrastructure. The different embedding strategies are submitted to an evaluation process, where it is shown that there is a tradeoff between improving acceptance and reducing the energy consumption. Nevertheless, the gains of reducing energy consumption might actually compensate the acceptance loss. In addition, we study the interplay of two strategies depending on the physical infrastructure state, which adapts its objective depending on the status of the infrastructure.

Finally, we propose a heuristic for the optimization of the embedding process. The performance evaluation shows that the heuristic achieves results close to the optimal ones in terms of the VI acceptance, and that it is also able to reduce the energy consumption of the physical infrastructure.

The remainder of this paper is organized as follows. Section D.2 presents the related work. Section D.3 details the concept of a cloud network virtual-enabled infrastructure and the resource allocation problem. Section D.4 presents the mathematical formulations for the embedding, and section D.5 presents the different strategies based on the formulations. Further, section D.5.1 details the heuristic algorithm. Finally, section D.7 depicts the evaluation of the different strategies, and section D.8 shows the final conclusions and future work.

D.2 Related Work

This section presents works that closely relate to the VI allocation problem.

Melo et al. [9] proposed an ILP formulation to solve the on-line VN embedding problem. Further, they also proposed an enhancement to an existing heuristic [10] and compared both heuristic and ILP, proving that heuristics are far from the optimal values. The authors have recently published [11], where they continue to explore the subject in an ILP perspective. Although it is shown an improvement in performance, the formulation is restricted to VNs and only considers CPU load, bandwidth and delay. In [12] it is proposed an algorithm that enables path splitting and link migration during the embedding process. However, this can lead to a level of fragmentation that is unfeasible to manage on large scale networks.

The work in [13] presents a strategy that considers the ability to reconfigure currently mapped VNs when trying to map a new one. The acceptance ratio of the VNs and the impact of reconfigurations are evaluated. The authors formulate the mapping problem as an ILP problem, and also propose a heuristic to speed up the solving time of the ILP problem. It is important to note that, in this work, VNs are characterized only by node CPU and link bandwidth.

In [14] different algorithms are proposed to provide better coordination between the node and link mapping phases. In [15] the authors extended the work in [14] with a generalized window-based VN embedding to evaluate the effect of look-ahead on the mapping of VNs. Although the approaches provide a better coordination of node and link mapping, they do not support heterogeneity of nodes.

Botero et al [16] have extended the VN embedding problem to energy awareness, and proposed a Mixed ILP to solve the optimal energy efficiency embedding problem. In the problem formulation they set the objective of minimizing the number of active network nodes and links. The evaluation shows that,

depending on the substrate load, the energy consumption can be substantially reduced without drastically affecting the VN acceptance ratio. In this work VNs are characterized by the processing power of the virtual nodes and the bandwidth of the virtual links. However, the work does not consider online scenarios. The formulation in [16] provides part of the base ground for the strategies proposed in our work.

Furthermore, our previous works [17] and [18] were the first steps on addressing the VI embedding problem. In [17] it is proposed a first ILP formulation for the problem, and in [18] it is proposed a heuristic. In this paper we extend both previous works to embrace new objectives, new constraints and new features: load balancing, energy consumption and an adaptive objective depending on the status of the infrastructure. Some initial insights of the work presented in this article were presented in [19].

Other works have looked to the cloud network field, such as Kantarci in [20], [21] and [22]. The work in [20] studies the delay minimization in the cloud network through a Mixed ILP (MILP) formulation. In [21] the authors address the reconfiguration of the cloud network in order to maximize the energy savings, and propose two heuristic approaches benchmarked by MILP approaches. The trade-off between energy savings and delay minimization was later studied in [22] and a heuristic was proposed. However, these approaches do not look to complex requests of joint cloud and network resources, and focus mostly on the access of end-users to cloud DCs.

Past and present research projects have looked to similar subjects, such as FP7 projects SAIL [24], GEYSERS [25] and MCN [26]. SAIL and MCN deal with the integration of cloud and network resources, but they consider current models (i.e. DCs and networks). GEYSERS focused on the optical communications. However, these do not consider the interplay between resources: CPU capacity, memory, storage and network related characteristics, such as bandwidth.

A considerable amount of work is available with respect to the mapping of VNs (the problem that closely relates to the VI problem). However, the joint management of cloud and network virtual resources has not yet been widely explored.

D.3 Cloud Network Virtual-Enabled Infrastructure

In this section we describe the scenario and the notation used. Moreover, we detail the embedding allocation problem applied to this scenario.

D.3.1 Cloud Network Infrastructure description

It is well known that virtualization is a key enabler in cloud scenarios, as it provides the ability of resources to be elastic, scale and even be moved (i.e. migrated). We consider the concept of cloud network virtual-enabled infrastructure, where cloud and network resources are part of a single pool of resources, can be virtualized, and are managed in an integrated way. Figure D-2 depicts an example of a cloud network infrastructure that is composed by *N* nodes in a random topology. Two types of nodes are considered, network nodes and server nodes, each with its specific set of associated characteristics. Network nodes are characterized by the number of CPU, denoted by *Cs*, the clock CPU frequency *F*, and the amount of memory *M*. Two types of network nodes are considered: those which have virtualization capabilities and can, therefore, host virtual network entities; and those that do not have virtualization capabilities, considered as network transport elements only. Server nodes are characterized by the same parameters as network nodes (*Cs*, *F*, and *M*) plus storage capabilities, denoted by *STG*. With respect to links, these are characterized by bandwidth capacity *B*, and assumed to be bidirectional, and maximum delay *D*. Moreover, the CPU load (or capacity) denoted by *C* is associated to the number of CPUs of a node (*Cs*).

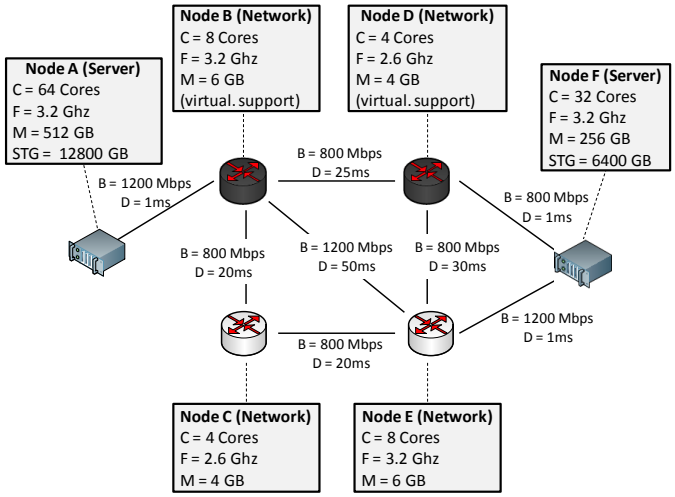


Figure D-2: Cloud Network Virtual-Enabled Infrastructure description example.

Table D-1 summarizes the notation used. The reference to physical resources uses letter *P*, e.g., N^P, and virtual resources use letter *V*, e.g., N^V. With respect to the connectivity of an infrastructure, an adjacent matrix is used: $A^P = N^P \rightarrow N^P$ when referring to a physical infrastructure – equation (1). The convention used for the index notation is the following: *i*, *j* for nodes and links in the physical network, and *m*, *n* for nodes and links in the VI. Bandwidth capacity is also described by adjacent matrixes: $B^P = N^P \rightarrow N^P$.

$$A_{ij}^P = \begin{cases} 1, \text{the physical node } i \text{ is neigbor of } j \\ 0, \text{else} \end{cases} \quad (1)$$

The cloud network infrastructures can host multiple VIs, as illustrated in Figure D-3. We consider that VI nodes of a certain type (server or network) can only be accommodated in the physical infrastructure by nodes of the same type.

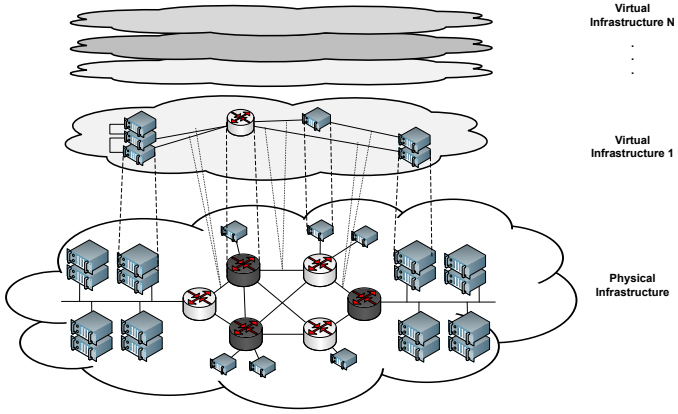


Figure D-3: Multiple Virtual Infrastructures over a Physical Infrastructure.

The number of CPU cores, capacity, frequency, memory size, and storage capacity of nodes are stored arrays with *N* entries (N^P or N^V depending if it refers to the physical or virtual infrastructure), e.g., $C^P \rightarrow N^P \times 1$. Note that the storage capacity of the network nodes is considered to be null. Furthermore, the total CPU capacity of a physical node is denoted by $C^{P_{total}}$, the free capacity by $C^{P_{free}}$, and the allocated capacity $C^{P_{used}}$, where $C^{P_{total}} = C^{P_{used}} + C^{P_{free}}$. The same notation is used for memory and storage. Also note that *v* is used to denote a VI, e.g. $C_{v,m}^V$ denotes the CPU capacity of virtual node *m* in VI *v*.

Table D-1: VI Assignement Problem Notation

Symbol	Description
G^P	Physical Infrastructure
N^P	Set of Physical Nodes
S^P	Set of Server Nodes
R^P	Set of virtual-enabled Network Nodes
Rt^P	Set of transport Network Nodes
L^P	Set of Physical Links
i, j	Physical Nodes
ij	Physical Link
$Cs^P{}_i$	Number of CPUs of Physical Node i
$C^P{}_i$	Total CPU of Physical Node i
$M^P{}_i$	Total Memory of Physical Node i
$STG^P{}_i$	Total Storage of Physical Node i
$F^P{}_i$	CPU Frequency of Physical Node i
$B^P{}_{ij}$	Total Bandwidth of Physical Link ij
$D^P{}_{ij}$	Delay of Physical Link ij
$S_{LS(ij)}$	Stress of Physical Link ij
$S_{N(i)}$	Stress of node i
$C^{Pfree}{}_i$	Free CPU of Physical Node i
$M^{Pfree}{}_i$	Free Memory of Physical Node i
$STG^{Pfree}{}_i$	Free Storage of Physical Node i
$G^V{}_v$	VI request v
$N^V{}_v$	Set of Virtual Nodes of VI request v
$S^V{}_v$	Set of virtual Server Nodes of VI request v
$R^V{}_v$	Set of virtual Network Nodes of VI request v
$L^V{}_v$	Set of Virtual Links of VI request v
m, n	Virtual Nodes
mn	Virtual Links
Cs	Number of CPUs of Virtual Node m of VI request v
$C^V{}_{v,m}$	CPU of Virtual Node m of VI request v
$M^V{}_{v,m}$	Memory of Virtual Node m of VI request v
$STG^V{}_{v,m}$	Storage of Virtual Node m of VI request v
$F^V{}_{v,m}$	CPU Frequency of Virtual Node m of VI request v
$B^V{}_{v,mn}$	Bandwidth of Virtual Link mn of VI request v
$D^V{}_{v,mn}$	Maximum Delay of Virtual Link mn of VI request v
$S_{LV(mn)}$	Stress of Physical Link mn

D.3.2 Virtual Embedding Problem

The process of embedding VIs within a physical infrastructure is known to be NP-hard. A VI is mapped according to one or multiple objectives, e.g.: occupy the lowest bandwidth possible, balance the occupation of the physical nodes, or consider both previous objectives. Combining both objectives is not simple as they are not standalone, i.e. the choice of an ideal physical node may not allow the choice of the ideal physical link. This is considered to be a multiple-objective optimization problem (MOP), where two or more conflicting objectives subject to constraints need to be simultaneously optimized. In MOP it is not usual to have solutions that are optimal for the different objectives individually; instead, there are solutions that make it impossible to improve the performance of a criterion (i.e. one of the objectives) without decreasing the quality of at least another criterion. These solutions are refered to as pareto optimal solutions [27]. Furthermore, finding optimal solutions may require a large computing effort. In this sense, heuristic algorithms are usually designed to reduce this effort and to find solutions close to the optimal ones.

In the following sections we will present the mathematical formulation for the optimal embedding (section D.4) as well as different strategies (section D.5) to solve the problem. Section D.6 presents the heuristic algorithm.

D.4 Optimal Embedding –Mathematical Formulation

In this section we present the mathematical formulation to solve the VI embedding problem based on ILP formulation. First, we present the assignment variables as well as the constraints common to all

approaches. Then, we present four specific formulations in separate sub-sections to calculate: the maximum node and link loads; bandwidth consumption; set of node and link state formulation.

Assignment Variables

The formulation considers two binary assignment variables, *x* and *y*, one for virtual nodes and another for virtual links, as equations (2) and (3) show.

$$x_i^{v,m} = \begin{cases} 1, \text{ virtual node } m \text{ of VI } v \text{ is allocated at physical node } i \\ 0, \text{ otherwise} \end{cases} \quad (2)$$

$$y_{ij}^{v,mn} = \begin{cases} 1, \text{ virtual link } mn \text{ of VI } v \text{ is uses physical link } ij \\ 0, \text{ otherwise} \end{cases} \quad (3)$$

D.4.1 Generic Constraints

As mentioned earlier in the VI description, virtual servers are assigned to physical server nodes. Equation (4) reflects this constraint, which is coupled with the fact that a virtual node is assigned just to one physical node. Moreover, equation (5) assures that virtual network nodes are assigned to network nodes with virtual capabilities.

$$\forall v, m \in S : \sum_{i \in S} x_i^{v,m} = 1 \quad (4)$$

$$\forall v, m \in R : \sum_{i \in R} x_i^{v,m} = 1 \quad (5)$$

Equations (6), (7) and (8) guarantee that the capacity values of physical nodes, i.e. CPU load, memory, and storage, are kept within range. Equations (9) and (10) make sure that the number of cores and CPU frequency requirements are respected, i.e., a physical node selected to host a virtual node has at least the same number of cores and frequency as the virtual node.

$$\forall i : \sum_{m,v} x_i^{v,m} \times C_{v,m}^V \le C_i^{P_{total}} \quad (6)$$

$$\forall i : \sum_{m,v} x_i^{v,m} \times M_{v,m}^V \le M_i^{P_{total}} \quad (7)$$

$$\forall i : \sum_{m,v} x_i^{v,m} \times STG_{v,m}^V \le STG_i^{P_{total}} \quad (8)$$

$$\forall i, v, m : x_i^{v,m} \times Cs_{v,m}^V \le Cs_i^P \quad (9)$$

$$\forall i, v, m : x_i^{v,m} \times F_{v,m}^V \le F_i^{P_{total}} \quad (10)$$

Equation (11) applies the multi-commodity flow constraint [28] with a node-link formulation [29], in order to simultaneously optimize the mapping of virtual links and virtual nodes. Moreover, the notion of direct flows on the virtual links is also used.

$$\forall v, \forall m, n \in N_v^V(m), m < n, \forall i : \sum_{j \in N^P(i)} \left(y_{ij}^{v,mn} - y_{ji}^{v,mn} \right) = x_i^{v,m} - x_i^{v,n} \quad (11)$$

Equation (12) guarantees that each physical link selected has enough available bandwidth to host a virtual link: the sum of the amount of bandwidth of the virtual links that go through *i* must be equal or lower than the amount of free bandwidth in a physical link. Finally, equation (13) guarantees that the virtual link delay constraint is met, i.e. the maximum delay of the physical links assigned to a virtual link does not exceed the virtual link's maximum delay.

$$\forall i,j \in N^P(i), i<j: \sum_{v,m,n \in N^V(m), m<n} B^V_{v,mn} \times \left(y^{v,mn}_{ij} + y^{v,mn}_{ji}\right) \leq B^{P_{free}}_{ij} \quad (12)$$

$$\forall v, \forall m,n \in N^V_v(m), m<n: \sum_{\forall i,j \in N^P(i), i<j} D^P_{ij} \times \left(y^{v,mn}_{ij} + y^{v,mn}_{ji}\right) \leq D^V_{v,mn} \quad (13)$$

D.4.2 Maximum Node and Link Load Formulation

This sub-section presents a node and link load formulation.

The maximum load consumption of server nodes, i.e. the load consumption of the server node that is more loaded among all server nodes ($S^{\max}_{load}$), is presented in equation (14). This value is given by the sum of CPU, memory and storage load fractions. Moreover, a fraction of the CPU frequency is taken into account in tiebreak situations (when there are two server nodes with the same load), in a normalized function. In this way, physical nodes with lower frequency are the first to be used, preserving the remaining for eventual future nodes with higher frequency demands. δ_1, δ_2, and δ_3 represent the resource component weights, allowing us to define which resource (CPU, memory or storage) is more relevant to the node overall load. ε is a very small value, so that the CPU frequency factor is only relevant in tiebreak situations. The sum of resource component weights is 1.

The maximum load consumption of network nodes ($R^{\max}_{load}$) is presented in equation (15) - similar to the maximum load consumption of server nodes. It only differs from the previous one by not taking into account storage (*STG*). μ_1 and μ_2 represent the resource components weights (CPU load and memory load), and ε is again a very small value so that the CPU frequency is only relevant in tiebreak situations. Again, the sum of resource component weights is 1.

$$\forall i \in S: \left[\begin{array}{l} \delta_1 \dfrac{\sum_{v,m}\left(x^{v,m}_i \times C^V_{m,i}\right)}{C^{P_{total}}_i} + \delta_2 \dfrac{\sum_{v,m}\left(x^{v,m}_i \times M^V_{m,i}\right)}{M^{P_{total}}_i} + \\ \delta_3 \dfrac{\sum_{v,m}\left(x^{v,m}_i \times STG^V_{m,i}\right)}{STG^{P_{total}}_i} + \varepsilon \dfrac{F^P_i}{F^P_{\max}} \end{array} \right] \leq S^{\max}_{load} \quad (14)$$

$$\forall i \in R: \left[\begin{array}{l} \mu_1 \dfrac{\sum_{v,m}\left(x^{v,m}_i \times C^V_{m,i}\right)}{C^{P_{total}}_i} + \\ \mu_2 \dfrac{\sum_{v,m}\left(x^{v,m}_i \times M^V_{m,i}\right)}{M^{P_{total}}_i} + \varepsilon \dfrac{F^P_i}{F^P_{\max}} \end{array} \right] \leq R^{\max}_{load} \quad (15)$$

Equation (16) defines the maximum load consumption of a physical link, among all links ($L^{\max}_{load}$). $B^V_{v,mn}$ refers to the bandwidth of virtual link *mn* (i.e. the virtual link connecting virtual node *m* and *n*) of a certain VI (*V*). $B^{P_{Total}}_{ij}$ refers to the total bandwidth capacity of physical link *ij* (i.e. the physical link between physical node *i* and *j*). The variable $y^{v,mn}_{ij}$ assumes the value 1 if virtual link *mn* is crossing physical link *ij*, and 0 otherwise. In this way, the formula sums the bandwidth of all virtual links ($B^V_{v,mn}$) that cross a certain physical link (*ij*) and divides that value by the total capacity of the physical link ($B^{P_{Total}}_{ij}$). In the end, $L^{\max}_{load}$ assumes the load value of the most loaded physical link.

$$\forall i,j \in N^P(i), i<j: \left[\frac{\sum_{v,m,n \in N^V(m), m<n} B^V_{v,mn} \times \left(y^{v,mn}_{ij} + y^{v,mn}_{ji}\right)}{B^{P_{Total}}_{ij}} \right] \leq L^{\max}_{load} \quad (16)$$

D.4.3 Bandwidth Consumption Formulation

Equation (17) defines the percentage of the total bandwidth consumed by a VI in the physical infrastructure. The amount of bandwidth consumed by each virtual links ($B^V_{v,mn}$) in all physical links ($y^{v,mn}_{ij}$) is summed and divided by the total bandwidth capacity of the entire physical infrastructure.

$$\frac{\sum_{v\in V,m,n\in N^V(m),m<n} y_{ij}^{v,mn} \times B_{v,mn}^{V}}{\sum_{i,j\in N^P(i),i<j} B_{ij}^{P_{total}}} = B_{cons} \quad (17)$$

D.4.4 Node and Link State Formulation

We present a formulation for node and link state with two extra (binary) variables, x_i^{active} and y_{ij}^{active}. x_i^{active} denotes if physical node *i* is active ($x_i^{active}=1$) or not ($x_i^{active}=0$). A node is considered to be active if it hosts at least one virtual node or if a virtual link traverses this node. Similarly, y_{ij}^{active} denotes if physical link *ij* is active ($y_i^{active}=1$) or not ($y_i^{active}=0$). Equation (18) guarantees that, if a physical link is occupied by at least one virtual link, variable y_{ij}^{active} will be set to 1 (0, otherwise). Moreover, equations (19) and (20) guarantee that, if a physical node has at least one active link variable, x_i^{active} takes value 1 and 0 otherwise. (Note: variable *K* is a high value constant used to assure that variable x_i^{active} does not assume a value higher than 1)

$$\forall i,j\in N^P(i),i<j: \sum_{v,m,n\in N^V(m),m<n} B_{v,mn}^{V} \times \left(y_{ij}^{v,mn} + y_{ji}^{v,mn}\right) \le B_{ij}^{P} \times y_{ij}^{Active} \quad (18)$$

$$\forall i\in N^P: \sum_{i,j\in N^P} \left(y_{ij}^{Active} + y_{ji}^{Active}\right) \ge x_i^{Active} \quad (19)$$

$$\forall i\in N^P: \sum_{i,j\in N^P} \left(y_{ij}^{Active} + y_{ji}^{Active}\right) \le x_i^{Active} \times K \quad (20)$$

D.5 Optimal Embedding Strategies - (Multi-objective) Objective Functions

The mathematical formulations presented in the previous section are the foundation for the definitions of different embedding strategies. In this section we propose a set of embedding strategies based on those formulations.

We divide the strategies in two main categories: base strategies and dynamic strategies. The proposal of these strategies relies on the well-known approach to multi-objective problems, the aggregation (or weighted) method. Furthermore, we use an approach with the weights (w_i, z_i) defined *a priori* [27].

D.5.1 Base Strategies

Herein we propose three base strategies that will be presented within the remaining categories.

D.5.1.1 Load Balancing (Node and Link) strategy – VIE-Opt-LB

Based on a strategy presented in [11], this strategy aims to minimize the maximum load of node and link resources. Equation (21) represents the strategy. Weights w_1 and w_2 rule the importance of the several loads, and ε is a very small value so that the total bandwidth consumption (B_{cons}) is only considered in tiebreak situations. Note that the maximum load of server ($S_{load}^{\max}$) and network ($R_{load}^{\max}$) nodes is under the effect of the same weight, w_1, and the maximum link load ($L_{load}^{\max}$) is under the effect of w_2. This evens the importance given to the maximum load of nodes and links.

$$min\ \omega_1\left(S_{load}^{\max} + R_{load}^{\max}\right) + \omega_2 L_{load}^{\max} + \varepsilon B_{cons} \quad (21)$$

This strategy will be used as the reference one, since in [11] it is shown that it outperforms several state-of-the-art embedding algorithms (D-Vine [15], R-Vine [15], etc).

In contrast to these approaches, in our approach we consider two types of nodes, more parameters in the load of a node, and we do not restrict a physical node to host only one virtual node per VI. Moreover,

we will be able to improve the VI embedding performance when using load balancing formulation. This improvement is achieved with the next strategy.

D.5.1.2 VIE-Opt-LB with Shortest Path (Opt-LB+SP)

This strategy differs from the previous one, as it does not take into account the total bandwidth consumption of a VI merely as a tiebreaker. In this case, B_{cons} is under the effect of weight w_2. We argue that the bandwidth consumption shall be considered as an active aspect to be taken into account while mapping VIs.

$$min\ \omega_1\left(S_{load}^{\max} + R_{load}^{\max}\right) + \omega_2\left(L_{load}^{\max} + B_{cons}\right) \quad (22)$$

D.5.1.3 Green strategy - VIE-Opt –Green

This strategy minimizes the energy consumption of the physical infrastructure. The energy consumption of a device is deeply related to its load. However, the relation between these two parameters is intimately dependent on the specific device. Moreover, in most devices the energy consumption in idle mode (i.e. powered on but with minimal load) is considerably high compared to the maximum energy consumption at full power (i.e. with maximum load). In this sense, we consider that it is better to have one node at full power than two nodes in idle mode. Moreover, we assume that reducing the number of active links in an active node will reduce the node's overall energy consumption.

With these assumptions in mind, we draw the objective to map a VI in such a way that it has a minimum number of active physical resources. In equation (23) the main objective is to minimize the number of active nodes, since they are the main energy consumption source. The second objective is to minimize the number of active links in order to reduce the overall energy consumption of the active nodes. In this case w_1 is higher than w_2, and ε is a very small value, so that another objective can be considered as a tiebreak objective, presented in the equation as *Strat*.

The objective considered in *Strat* in the evaluation section is the load balancing with shortest path (*VIE-Opt-LB+SP*). The load balancing of the active elements of the infrastructure will ease the future embedding of VIs without having to increase the number of active elements.

$$min\ \omega_1\left(\sum_{i\in N^P} x_i^{Active}\right) + \omega_2\left(\sum_{i,j\in N^P, i<j} y_{ij}^{Active}\right) + \varepsilon(Strat) \quad (23)$$

D.5.2 Dynamic Strategies - Adjusting Weights

In this category we study the combination of different objectives in a dynamic way. In the previous cases, different objectives were considered in a strategy. However, there was always a fixed hierarchy of objectives. In this case, we do not keep that hierarchy fixed, and change it depending on the infrastructure state.

D.5.2.1 Green + Load Balancing - VIE-Opt-D-Green-LB

This strategy uses the green strategy (*VIE-Opt –Green*) and the load balancing with shortest path strategy (*VIE-Opt-LB+SP*), as presented in equation (24). Here, weights (z_1 and z_2) are not fixed along time, but change depending on the current state of the infrastructure at the time of each VI arrival. More precisely, we consider that, if the infrastructure is loaded over a certain limit, the load balancing strategy is applied. If the load is below or equal to the limit, the green strategy is applied.

$$min\ z_1(Strat\_1) + z_2(Strat\_2) \quad (24)$$

D.6 Heuristic Approach

In this section we present a heuristic approach to solve the VI embedding problem. A high level description of the algorithm is provided. Further, two approaches to the virtual link assignment process are considered. A comparative analysis of the two approaches will be performed in section D.7.

D.6.1 The Algorithm

Herein we detail the different parts of the algorithm.

Candidates' Selection **–** The algorithm starts to compare the hardware features of all virtual nodes $m \in N^V$ with all physical nodes $i \in N^P$. In order for a physical node to be a candidate, the characteristics of node *i* have to be higher or equal to the characteristics requested by virtual node *m* ($C_i^P \geq C^V_m$, $F_i^P \geq F^V_m$, $M_i^P \geq M^V_m$, $STG_i^P \geq STG^V_m$). Additionally, $\forall i \in Candidates(\forall m)$ need to have a physical connection $ij \in A^P$ with at least one candidate of the virtual node neighbor $mn \in A^V$. In order to strengthen the quality of the candidates, we introduced an individual selection of $\forall i \in Candidates(m)$ from all the virtual nodes $m \in N^V$ to find the candidates that will certainly lead to a mapping failure.

Resource Stress Calculation **–** This part includes the stress calculation of the infrastructure elements that will be used in the selection process. Stress is an indicator of how likely a resource should host a virtual resource in comparison with other physical resources (resources with less stress are more prone to accepting new virtual resources). The link stress, $S_{LS_{(ij)}}$ (equation (25)) is calculated for $\forall ij \in A^P$. Since the substrate has two types of nodes, each stress depends on different characteristics according to the type of the physical nodes. If node $i \in S^P$, the STG^P will be considered in S_{Ni} (equation (26)). If node $i \notin S^P$, F^P will be considered instead of the STG^P in S_{Ni}. After the stress calculation, we introduce a routine to assure that S_{Ni} is never zero. In the cases $S_{Ni}=0$, the algorithm assigns the *MinStressValue*($S_{Ni}!=0$) to $\forall i \in N^p$ that already hosts $\forall m$, in order to not compromise the node potential calculation in equation (27) ($\pi(i)$ - the potential of a candidate is not calculated only using server stress, but also considering the stress of the physical links which might be used to host virtual links). If at this moment there are virtual nodes with only one candidate, the algorithm will simulate an update to the substrate occupied resources of that candidate, in order to simulate its future occupation, and check if that candidate is still a valid candidate for the rest of the virtual nodes that have it as an option.

$$S_{LS_{(ij)}} = \sum_{v \in V, m, n \in N^V(m), m<n} \left(S_{LV}(mn) \times y_{ij}^{v,mn} \right) \tag{25}$$

$$S_{N_i} = \begin{bmatrix} N.ActiveVMs.(1 + \frac{k.C^V{}_{med\,\mathrm{Req}}}{C_i^{P_{free}} + \delta}).(1 + \frac{k.M^V{}_{med\,\mathrm{Req}}}{M_i^{P_{free}} + \delta}) \\ (1 + \frac{k.STG^V{}_{med\,\mathrm{Req}}}{STG_i^{P_{free}} + \delta}) \end{bmatrix} \tag{26}$$

Selection Process **–** The selection of $i \in Candidates(\forall m)$ is performed in this part. At the beginning of the selection process, a verification process of the candidates due to the future occupation status is introduced. After this analysis, it starts the selection process (which is sequential), i.e. a node $m \in N^V$ is selected in order to calculate $\pi(i)$ (equation (27)) for $\forall i \in Candidates(m)$. S_{N_i} is multiplied by the path stress average of $\forall j \in Candidates(n)$, where $mn \in A^V$. The formula for the path stress (*PathStress*) is described in section D.6.2. The candidate $i \in Candidates(m)$ that presents the lower value of π is chosen to host *m*.

$$\pi(i) = S_{N_i} \frac{\sum_j PathStress(i,j)}{totalCand} \tag{27}$$

Substrate Links mapping **–** The last part of the algorithm is executed if every virtual node $m \in N^V$ has a candidate. This process comprises the mapping of all virtual links (further details are provided in section D.6.2).

D.6.2 Path Find and Path Stress approaches

In the decision process, the candidate that presents the highest potential is chosen to host the virtual node. The candidate potential is inversely proportional to the multiplication of the candidate stress with the average of the path stress necessary to link the candidate with all candidates of the neighbor's virtual nodes – equation (27). Note that the path stress calculation does not influence the path finding process, but it is dependent on the latter.

The process of calculating the path stress is preceded by the finding of a proper path. The approach followed in [17] used the Constrained Shortest Path First (CSPF) Dijkstra algorithm [30] to find a path. However, just like the optimal strategy VIE-Opt-LB, this approach tries to reduce the physical link's load, neglecting to a certain extent the amount of bandwidth consumed by a virtual link. In this sense, the Breadth-First Search (BFS) algorithm [31] is considered as a credible alternative to be evaluated. The BFS looks for the path between source and destination that requires less hops. Originally, the algorithm stops as soon as it reaches the first solution (which guarantees the minimum number of hops), or when it does not find a solution. In our case, we let the algorithm find all solutions that guarantee the minimum number of hops, and the tiebreak is done using the Dijkstra algorithm. In this sense, we guarantee the best path among those with minimum hops.

After finding a suitable path, the path stress is calculated. In the baseline approach (that uses only Dijkstra), the path stress is retrieved directly from the path find algorithm. However, having in mind that the bandwidth of the physical links was already identified in the past as a main bottleneck to the embedding of VIs [18], the approach now imposes weights to the hops in addition with the substrate occupation by a factor *K* of the virtual link. The weight added by hop follows an exponential behavior. Equation (28) reflects the stress calculation of mapping a virtual link in a certain physical path. *PathStress* represents the stress, *H* represents the number of hops, $S_{LS}(ij)$ represents the link stress (reserved substrate bandwidth on the substrate link) between the node *i* and the node *j*, *P* represents the sorted list of nodes traversed by the link, *K* represents a constant value, and B^{mn} is the virtual link bandwidth.

$$PathStress = \sum_{h=1}^{H} S_{LS}(P_h, P_{h+1}) + \sum_{h=1}^{H} \left[K.(h-1).B^{\mathrm{mn}} + B^{\mathrm{mn}} \right] \quad (28)$$

In summary, this approach does not allow to choose candidates that will consume a high number of links, even if the physical substrate in that zone has a lower stress value. This approach is also used in the final link mapping.

D.7 Performance Evaluation

This section evaluates the proposed approaches. Here, we provide a thorough evaluation of the different optimal embedding strategies presented in section D.5, along with an evaluation of the proposed heuristic. Note that, in this section, the term *Heuristic* refers to the approach in which the Dijkstra algorithm is used in the path finding process, while the term *E-Heuristic* refers to the BFS case.

D.7.1 Evaluation scenario description

To evaluate the performance of the proposed approaches, a Matlab [32] simulator is used. For each run, the program designs a random physical infrastructure of 20 nodes based on the *Waxman* network topology generator [33], and it simulates a set of requests of VIs (with a number of nodes between 4 and 14) with Markov-modulated inter-arrival and inter-departure times. Both physical infrastructure and the generated VIs have 70% of the nodes as servers (rounded to the higher integer), and the remaining 30% as network nodes. Details on the nodes and link parameters can be seen in Table D-2. Moreover, the VI request rates (λ) are between 2 to 5 VIs per time unit (Poisson arrivals), and the average duration ($1/\mu$, where μ is the average service rate) is 20 time units (exponentially distributed).

Table D-2: Physical and virtual infrastructure parameters

		Physical Infrastructure	Virtual Infrastructures
Network Nodes	Cs	{2; 4; 6; 8}	{1; 2; 3; 4 }
	F(Hz)	{2.0-3.2 / 0.2 steps}	{2.0-3.2 / 0.1 steps}
	Memory	{2; 4; 6}(GB)	{64; 128; 256; 512}(MB)
Server Nodes	Cs	{8; 16; 32; 64}	{1; 2; 4; 8; 16; 32; 64}
	F(Hz)	{2.8-3.2 / 0.2 steps}	{2.8-3.2 / 0.1 steps}
	STG (GB)	{6400; 12800; 25600}	{100; 200; 400; 800; 1600}
	M (GB)	{256; 512; 1024}	{2; 4; 8; 16; 32; 64}
Links	B (Mbps)	{500-2000 / 500 steps}	{10-100 / 10 steps}
	D (ms)	{5-10 / 1 steps }	{0-40 / 5 steps}

In order to solve the ILP, we have used CPLEX [34] version 12.3, integrating a plug-in for our Matlab simulator and setting a time limit of 600 seconds (i.e. 10 minutes) for each VI mapping. The weight values used for the optimal strategies are the following:

- w_1 and w_2 are 0.5 (50%);
- ε is 0.001;
- z_1 and z_2 have dynamic values which change depending on the state of the physical infrastructure. If the maximum load of any resource (server node, network node or link) is below 75%, *Opt-Green* is applied; otherwise, *Opt-LB+SP* is applied. The 75% value was chosen empirically due to its good results in the run experiments.

We analyze the VI acceptance ratio as one of the main indicators of the algorithms' performance. Moreover, the energy consumption of the physical infrastructure is also analyzed, based on the analysis of the active infrastructure resources along time. All values in the following graphics present a mean of 10 runs (with different substrate) with a confidence interval of 95%.

D.7.2 Virtual Infrastructure Acceptance Ratio

Figure D-4 presents the VI acceptance ratio of the strategies. It is important to note that, given the amount of constraints involved in the embedding problem (delay, bandwidth, processing, memory, storage), many VIs are rejected due to the impossibility of the infrastructure to simultaneously meet all requirements. Naturally, the values in all strategies present a decreasing linear behavior as the number of VIs per time unit increases. As the VI arrival rate increases, the substrate reaches a saturation point much faster, which leads to a decreasing behavior in the acceptance.

The difference between the VIE-Opt-LB (values between 12% and 22%) and the remaining strategies is clear, approximately above 10%. This shows that actively considering the minimization of the bandwidth consumed by VIs improves the acceptance ratio.

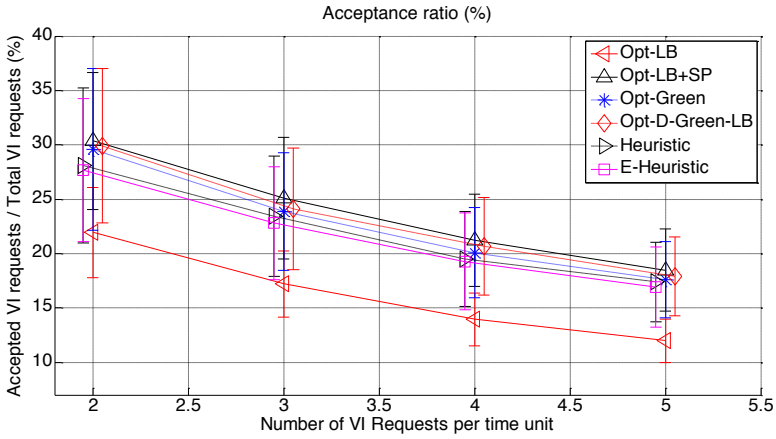


Figure D-4: VI acceptance ratio

The remaining strategies present very close values. The VIE-Opt-LB+SP is the strategy that overall presents better results – between 31% with 2 VIs per time unit, and 18% with 5 VIs per time unit. The VIE-Opt-Green presents lower values. This is an expected behavior: minimizing the number of active resources will lead to a faster saturation of these resources, making them inadequate for future VIs. Nevertheless, the decrease is low, approximately 3%. In-between these latter two strategies is the Opt-D-Green-LB. This is also an expected result, since the interplay between the green strategy and the load balancing naturally results in a position between these two.

With respect to the heuristic approaches, they present results very close, but it is noticeable the slight better performance of the Heuristic over the E-Heuristic.

D.7.3 Delay and Physical Infrastructure size

In this sub-section we briefly analyze the impact of adding a new constraint to the problem formulation. More specifically, it is analyzed the delay constraint impact in the VI acceptance results. Furthermore, the impact of the size of the physical infrastructure is analyzed.

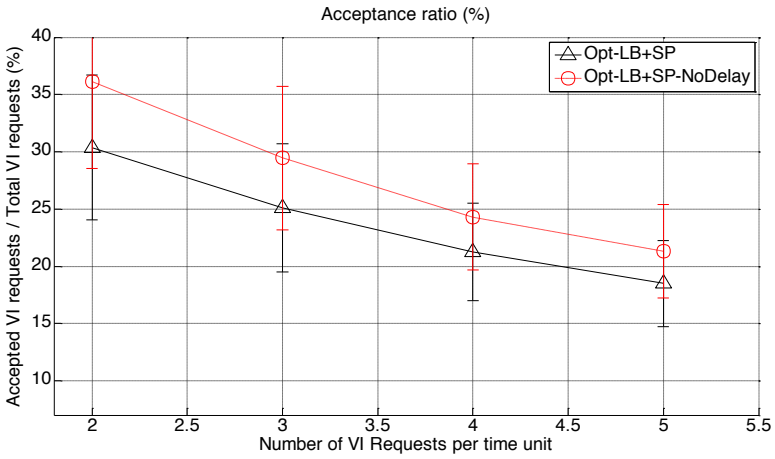


Figure D-5: Delay constraint impact

Figure D-5 shows the impact that the delay constraint brings to the acceptance of VIs. The delay factor decreases the acceptance ratio from 3% to 6% depending on the VI arrival rate. In this analysis only one strategy (*VIE-Opt-LB+SP*) is considered, since the comparison between different strategies was already provided in the previous sub-section.

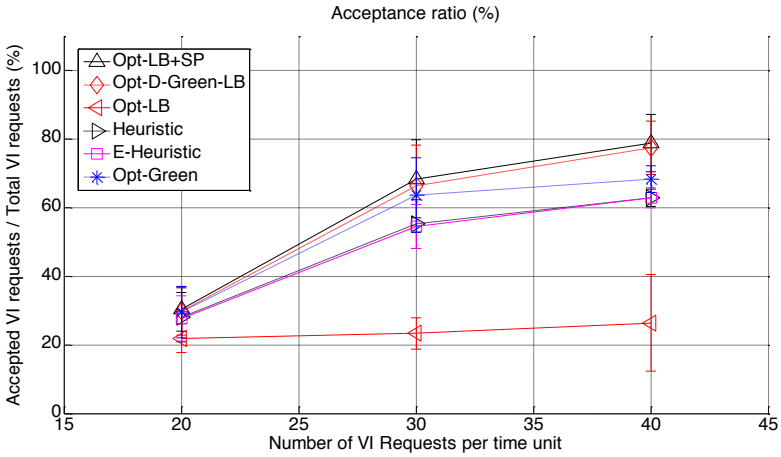


Figure D-6: Physical Infrastructure size impact

The impact of the physical infrastructure size can be observed in Figure D-6. Naturally, by increasing the physical infrastructure size, the acceptance rate increases. The overall ranking between strategies remains the same as in Figure D-4.

D.7.4 Physical Infrastructure Energy Consumption

This sub-section presents the performance of the different strategies in terms of energy consumption of the physical infrastructure. While in terms of acceptance ratio some strategies presented values relatively close, the differences with respect to energy consumption are more pronounced.

The analysis is performed taking into account the cumulative time of active resources (node and link), and the results are depicted in Figure D-7 and Figure D-8. The values are presented in percentage, where 100% means that all resources were active during the entire simulation time. First, we analyze the values related to nodes, Figure D-7, as they are the lead indicators of the energy consumed.

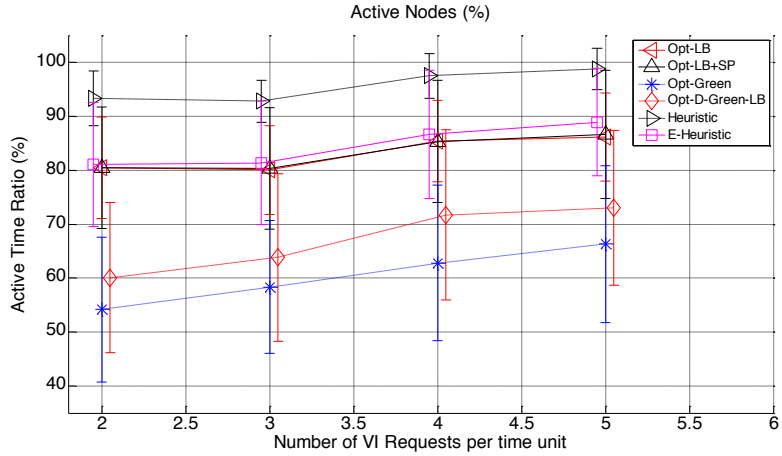


Figure D-7: Cumulative time of active nodes

All strategies that consider the green policy perform better than the remaining ones in terms of energy consumption. The VIE-Opt-Green strategy clearly outperforms the remaining ones, with values between 53% for arrivals of 2 VIs per time unit, and 66% for 5 VIs per time unit. The next performing strategy is the VIE-Opt-D-Green-LB strategy, with values between 60% and 72%. It is important to highlight that, although VIE-Opt-D-Green-LB requires more active nodes, it presents a higher rate in terms of acceptance ratio.

The remaining two optimal strategies, those that do not apply the green strategy, present higher and very close values, between 80% and 87%. It is of relevance to highlight here the fact that the VIE-Opt-LB,

used as reference point, presents values 10% lower in terms of acceptance ratio in relation to the remaining strategies, and it also presents equal values in terms of active nodes.

Finally, with respect to the heuristic approaches, the E-Heuristic (82-89%) clearly outperforms the Heuristic (93-99%), with improvements above 10%. Also note that, in terms of acceptance ratio, they present very close values.

In terms of active links, the strategies ranking has a lower variation (Figure D-8). The optimal strategies with green approaches perform better, with values of active time ratio between 17% and 28%. The Heuristic is the one presenting higher values of active time ratio, between 42% and 49%. The E-Heuristic and the optimal strategies with load-balancing policies have performance values in between the previous ones.

Note that, just like the acceptance ratio indicator, the values here present a linear behavior as the VI arrival rate increases. However, as opposed to the acceptance ratio, here they increase as the VI arrival rate increases. This is due to the fact that, although the acceptance ratio decreases with the increase of the VI arrival rate, the overall number of VIs embedded in the physical infrastructure increases, which leads to a higher percentage of active resources in the substrate.

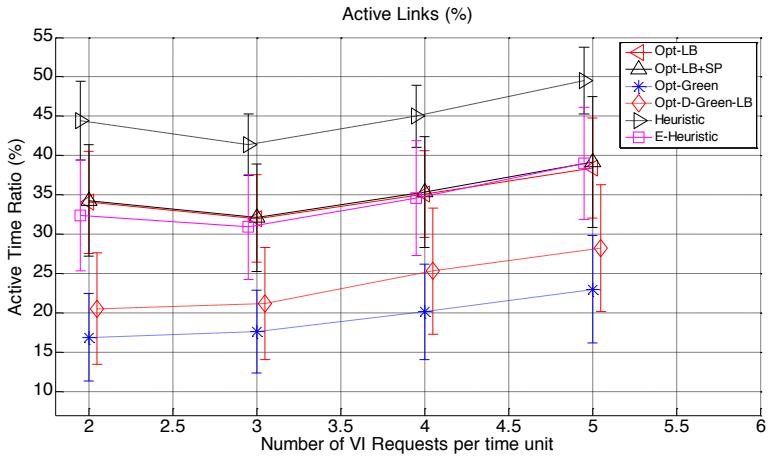


Figure D-8: Cumulative time of active links

In summary, considering the VI bandwidth consumption an active factor in the embedding strategy (VIE-Opt-LB+SP) clearly improves the acceptance without affecting the cumulative time of active resources (i.e. there is no significant variation of the energy consumption). Moreover, adopting a green approach (VIE-Opt-Green) presents nearly as good results as the best strategy. Finally, its positive impact in terms of energy consumption is considerably high.

D.8 Conclusion

This paper proposed optimal strategies to solve the VI allocation problem. The formulation of these strategies was performed using ILP. A formulation that keeps a balance between the nodes and the overall link consumption was proposed, along with a formulation that looks at reducing the energy consumption of the physical infrastructure. Through the different formulations, different strategies were defined and studied. Moreover, a heuristic algorithm that takes into account the load of physical resources was presented in detail. Having in mind that the link consumption is in several cases a bottleneck, the algorithm favors to reduce the overall link consumption of the infrastructure.

A comparative performance analysis between the different strategies was performed considering two different aspects: acceptance ratio and physical infrastructure energy consumption. The study of the optimal strategies shows that there is a clear tradeoff between improving acceptance with reducing the physical infrastructure usage. The use of different strategies depending on the physical infrastructure resources load can be a way to deal with this tradeoff. With respect to the heuristic algorithm, it improves

the VI acceptance compared to a former one, and it also reduces the energy consumption of the physical infrastructure.

As future work, the extension of the energy consumption analysis will be performed, by means of calculating the approximate energy consumption of node depending on its load. Finally, an extension of the formulation to consider reconfiguration and elastic VIs (VIs that vary the amount of resources along time) will also be addressed.

References

[1] ETSI, Network Functions Virtualisation (NFV): Architectural Framework, Technical Report ETSI GS NFV 002 v1.1.1, Oct. 2013.

[2] N. Yong; X.C. Liang; Z. Kai, Connectivity as a Service: Outsourcing Enterprise Connectivity over Cloud Computing Environment, Computer and Management (CAMAN), 2011 International Conference on , vol., no., pp.1,7, 19-21 May 2011.

[3] W.H. Liao; S.C. Su, A Dynamic VPN Architecture for Private Cloud Computing, Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on , vol., no., pp.409,414, 5-8 Dec. 2011.

[4] A. Mahimkar, A. Chiu, R. Doverspike, M.D. Feuer, P. Magill, E. Mavrogiorgis, J. Pastor, S.L. Woodward, and J. Yates. 2011. Bandwidth on demand for inter-DC communication. In Proceedings of the 10th ACM Workshop on Hot Topics in Networks (HotNets-X). ACM, New York, NY, USA, , Article 24 , 6 pages.

[5] ETSI, Network Functions Virtualisation (NFV): Use-cases, Technical Report ETSI GS NFV 001 v1.1.1, Oct. 2013.

[6] Sophia Antipolis, Leading operators create ETSI standards group for network functions virtualization, 22 January 2013 - http://www.etsi.org/news-events/news/644-2013-01-isg-nfv-created.

[7] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. H. Katz, A. Konwinski, et al.. Above the Clouds: A Berkeley View of Cloud Computing. Tech. rep., EECS Department, University of California, Berkeley.

[8] Akamai, Can Cloud and High Performance Co-Exist?, May 2011 - http://www.akamai.com/dl/akamai/Can_Cloud_High_Performance_Co-Exist.pdf.

[9] M. Melo, J. Carapinha, S. Sargento, L. Torres, P.N. Tran, U. Killat, et al., Virtual Network Mapping - An Optimization Problem. In K. Pentikousis, R. Aguiar, S. Sargento, R. Aguéro, O. Akan, P. Bellavista, et al. (Edits.), Mobile Networks and Management (Vol. 97, pp. 187-200). Springer Berlin Heidelberg, 2012.

[10] J. Nogueira, M. Melo, J. Carapinha, S. Sargento, Virtual network mapping into heterogeneous substrate networks. Computers and Communications (ISCC), 2011 IEEE Symposium on, (pp. 438-444).

[11] M. Melo; S. Sargento; U. Killat; A. Timm-Giel; J. Carapinha, "Optimal Virtual Network Embedding: Node-Link Formulation," Network and Service Management, IEEE Transactions on , vol.10, no.4, pp.356,368, December 2013.

[12] M. Yu, Y. Yi, J. Rexford, M. Chiang, Rethinking virtual network embedding: substrate support for path splitting and migration. SIGCOMM Comput. Commun. Rev. , 38 (2), 17-29, 2008.

[13] P.N. Tran; L. Casucci; A. Timm-Giel, Optimal mapping of virtual networks considering reactive reconfiguration, Cloud Networking (CLOUDNET), 2012 IEEE 1st International Conference on , vol., no., pp.35,40, 28-30 Nov. 2012

[14] N. Chowdhury, M. Rahman, R. Boutaba, Virtual Network Embedding with Coordinated Node and Link Mapping. INFOCOM 2009, IEEE, (pp. 783-791), 2009.

[15] N. Chowdhury, M. Rahman, R. Boutaba, ViNEYard: Virtual Network Embedding Algorithms With Coordinated Node and Link Mapping. Networking, IEEE/ACM Transactions on , 20 (1), 206-219, 2012.

[16] J.F. Botero, X. Hesselbach, M. Duelli, D. Schlosser, A. Fischer, A.; De Meer, H., Energy Efficient Virtual Network Embedding, Communications Letters, IEEE , vol.16, no.5, pp.756,759, May 2012

[17] J. Soares, J. Carapinha, M. Melo, R. Monteiro, S. Sargento. Resource allocation in the network operator's cloud: A virtualization approach. Computers and Communications (ISCC), 2012 IEEE Symposium on, (pp. 800-805), 2012.
[18] J. Soares, R. Monteiro, M. Melo, S. Sargento, J. Carapinha, The Cloud inside the Network: a virtualization approach to resource allocation, in: H. Mouftah and B. Kantarci (Eds.), Communication Infrastructures for Cloud Computing: Design and Applications, IGI Global, 2013.
[19] J. Soares, S. Sargento, "Optimizing the Embedding of Cloud and Network Virtual Infrastructures", The twenty first IEEE International Conference on Telecommunications (ICT 2014), Lisbon, Portugal, 4-7 Maio, 2014.
[20] B. Kantarci, H. Mouftah, Minimizing the provisioning delay in the cloud network: Benefits, overheads and challenges. Computers and Communications (ISCC), 2012 IEEE Symposium on, (pp. 806-811), 1-4 July 2012.
[21] B. Kantarci, H. Mouftah, Optimal Reconfiguration of the Cloud Network for Maximum Energy Savings. Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on, (pp. 835-840), 13-16 May 2012.
[22] B. Kantarci, H. Mouftah, Overcoming the energy versus delay trade-off in cloud network reconfiguration, Computers and Communications (ISCC), 2012 IEEE Symposium on , vol., no., pp.000053,000058, 1-4 July 2012.
[23] 4WARD, The 4WARD project, http://www.4ward-project.eu/, 2010.
[24] SAIL, The SAIL project, http://www.sail-project.eu/, 2013.
[25] Geysers, The GEYSERS project, www.geysers.eu/, 2012.
[26] MCN, The Mobile Cloud Networking project, https://www.mobile-cloud-networking.eu/, 2013.
[27] Hoboken, Metaheuristics: from design to implementation . John Wiley \& Sons, 2009.
[28] S. Even, A. Itai, A. Shamir, On the complexity of time table and multi-commodity flow problems, Foundations of Computer Science, 16th Annual Symposium on, (pp. 184-193), 1975.
[29] M. Pioro, D. Medhi. Routing, Flow, and Capacity Design in Communication and Computer Networks. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc, 2004.
[30] E.W. Dijkstra, A note on two problems in connexion with graphs. Numerische Mathematik 1: 269–271, 1959.
[31] S. Russel and P. Norvig, Artificial Intelligence, a modern approach, 2003.
[32] MATLAB Release 2010, The MathWorks, Inc., Natick, Massachusetts, United States.
[33] MATLAB, Waxman Network Topology Generator, http://www.mathworks.com/matlabcentral/fileexchange/2517-waxman-network-topology-generator/content/waxtop.m
[34] IBM. IBM ILOG Optimization Products. www-01.ibm.com/software/websphere/products/optimization. Retrieved 13 May 2013.

Paper E. Re-Optimizing the Embedding of Virtualized Cloud Network Infrastructures

João Soares, Susana Sargento

submitted to IEEE Transactions on Network and Service Management, 2014

The format has been revised

Re-Optimizing the Embedding of Virtualized Cloud Network Infrastructures

João Soares and Susana Sargento

Abstract

Today, in the cloud computing context, the network is starting to arise as a true resource itself and not just as a required connectivity add-on. The ability to define network resources (e.g. routing/switching elements, bandwidth, delay) in this context is still scarce, but there are clear evidences that this is a future reality. In this article we consider that cloud infrastructure services will allow the definition of complete infrastructures, to which we refer as Virtual Infrastructures (VIs). These VIs comprise both computing, storage and network resources.

This article specifically tackles the VI embedding problem, which is known to be NP-hard. Having in mind this fact, different embedding strategies based on Integer Linear Programming (ILP) formulation are presented. The problem is addressed by proposing strategies that target the load balancing and energy consumption of physical resources. Moreover, re-optimization strategies are also considered, which take special attention to the costs associated to such re-optimizations.

Finally, a thorough evaluation of the different strategies is performed, considering the VI acceptance ratio, physical infrastructure energy consumption, and the impact of virtual resources reconfiguration. The obtained results show that there is a clear tradeoff between improving the VI acceptance ratio and reducing the physical infrastructure usage without allowing re-optimizations. Moreover, allowing virtual resources reconfiguration can considerably increase the acceptance ratio, as well as reduce the energy consumption of the physical infrastructure.

Keywords - Cloud Networking, Integer Linear Programming, Virtualization, Virtual Infrastructure.

E.1 Introduction

The network has a fundamental role in the cloud computing context, since it is able to guarantee performance, reliability and security to cloud services both inside DCs (DCs) as well as outside, i.e. on the Wide Area Network (WAN). The momentum around the Network Functions Virtualization (NFV) [1] [2] concept can be seen as the most recent catalyst to this uprising need for an active role of the network in the cloud. The Telco sector seems eager to "cloudify" some of its solutions, which have more than just computing requirements: they have very strict requirements in terms of network guarantees. In fact, some of these solutions are already "cloud-ready", i.e. virtualized and prepared to be deployed in the cloud. Examples of these solutions are the Evolved Packet Core (EPC) and IP Multimedia Subsystem (IMS). However, today's cloud solutions cannot meet the requirements imposed by these systems, mainly in terms of network aspects.

These recent events have re-leveraged the concept of Network-as-a-Service (NaaS), which can go from providing simple connectivity [3], connectivity with bandwidth and security requirements [4] [5], up to providing a more complete approach with network and connectivity elements (nodes and links). Nevertheless, the key enabler behind these concepts lays upon virtualization techniques. The "as-a-Service" concept is, among other aspects, related with dynamics and agility, which is on the birth of the concept of Software Defined Networking (SDN). Among the opportunities that SDN brings is the effective "unlock" of the NaaS concept. In this scenario of revolution and evolution, the IaaS and NaaS concepts will naturally tend to merge, which will require an integrated management of resources across cloud and network providers.

This article addresses the joint embedding of cloud and network resources, a NP-hard problem, through ILP. We consider the provisioning of cloud and network resources through virtualization (virtualized computing, storage, and network), enabling the support of Virtual Infrastructures (VIs). The VI embedding problem is addressed in a single domain perspective. The single domain comprises the case in which a pool

of computing and network resources (i.e. server and network nodes) is managed in an integrated way by a single entity. An example of such entity can be a Cloud Provider DC or a network operator with computing resources spread throughout its network. The scenario is further detailed ahead in the document.

We propose different VI embedding strategies that take into account the load balancing in the physical infrastructure, the energy consumption, and the impact of re-optimization processes. The different strategies are submitted to a thorough evaluation process. It is shown that allowing reconfiguration improves the VI acceptance ratio as well as the energy consumption, but it may also have an impact on the running services. Therefore, our proposed strategy aims to also minimize the costs associated with the reconfiguration process.

This work extends the formulations of our previous work presented in [6] to enable VI reconfigurations. We thoroughly explore how enabling VI reconfigurations can benefit the VI embedding problem, without neglecting the impact of the reconfiguration processes. Compared to [6], this paper has the following contributions:

- ILP formulation for the re-optimization of VIs and reconfiguration of resources;
- Formulation of the costs associated with re-optimization;
- Re-optimization embedding strategy for both load-balancing, green and dynamic load and green approaches;
- Evaluation of the re-optimization approaches with respect to their performance and associated costs.

The remainder of this article is organized as follows. Section E.2 presents related work. Section E.3 details the concept of a cloud network virtual-enabled infrastructure in single-domain scenarios. Furthermore, it elaborates on the resource allocation problem. Section E.4 proposes the mathematical formulations for the embedding, and section E.5 proposes the different strategies based on the formulations. Further, section E.6 depicts and discusses the evaluation of the different proposed strategies. Finally section E.7 provides final conclusions and future work.

E.2 Related Work

This section presents works that closely relate to the VI allocation problem.

In [6], we have first addressed the VI embedding problem. Different optimal strategies were proposed, considering aspects like load balancing and energy consumption. Furthermore, a heuristic algorithm was also presented. In this work we extend the formulation of [6] to enable VI reconfigurations and improve the performance of the strategies, both in terms of acceptance ratio and energy consumption of the physical infrastructure.

The work in [7] presents a strategy that considers the ability to reconfigure currently mapped Virtual Networks (VNs) when trying to map a new one. The acceptance ratio of the VNs is evaluated, as well as the impact of the reconfigurations on the running services. The authors formulate the mapping problem as an ILP problem and as a heuristic to speed up the solving time of the optimization problem. In this work VNs are characterized only by node CPU and link bandwidth. Nevertheless, this work was an important reference to the ILP formulations presented in this work with respect to the re-optimization process formulation.

Other works have looked to the cloud network field, such as Kantarci in [9], [10] and [11]. The work in [9] studies the delay minimization in the cloud network through a Mixed ILP (MILP) formulation. In [10] the authors address the reconfiguration of the cloud network in order to maximize the energy savings, and propose two heuristic approaches benchmarked by MILP approaches. The trade-off between energy savings and delay minimization was later studied in [11]. However, just like in the previous works, the interplay between CPU, memory, storage and network resources is not considered.

The joint manipulation of cloud and network virtual resources has not yet been widely explored. Nevertheless, there is a considerable amount of work in the VN embedding field, a problem that closely relates to the VI one, and that is a good starting point for inspiration to solve the VI embedding problem. Although there is some work in the VN scope with respect to re-optimization processes, the definition and study of re-optimization processes in the scenarios envisioned in this work (joint manipulation of cloud and network resources – section E.3) have not yet been addressed.

E.3 Cloud Network Virtual-Enabled Infrastructure

It is well known that virtualization is a key enabler in cloud scenarios, as it provides the ability of resources to be elastic, scale and even be moved (i.e. migrated). We consider the concept of cloud network virtual-enabled infrastructure, where cloud and network resources are part of a pool of resources, can be virtualized, and are managed in an integrated way.

In this work we look at the scenario where cloud and network resources are hosted within a DC or spread through an Operator Network.

This section describes the abovementioned scenario as well as the notation used. Moreover, it details the embedding allocation problem.

E.3.1 Scenario

This work considers two types of nodes, network nodes and server nodes, each with its specific set of associated characteristics. Network nodes are characterized by the number of CPU, denoted by Cs, the clock CPU frequency, F, and by the amount of memory, M. Two types of network nodes are considered: those which have virtualization capabilities and can therefore host virtual network entities; and those that do not have virtualization capabilities, considered network transport elements only. Server nodes are characterized by the same parameters as network nodes (Cs, F, and M) plus storage capabilities, denoted by STG. With respect to links, these are characterized by bandwidth capacity (B) and assumed to be bidirectional and with a maximum delay (D). Moreover, associated to the number of CPUs of a node (Cs), we consider another parameter that reflects the CPU load (or capacity) denoted by C.

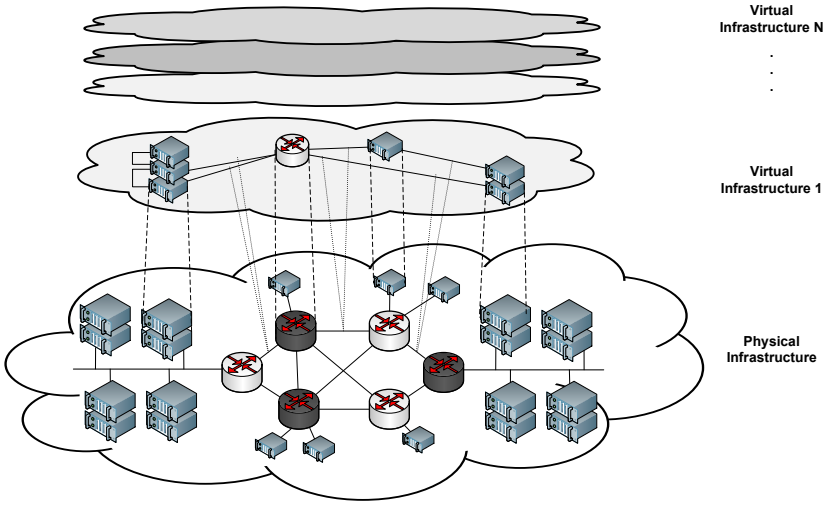


Figure E-1: Single Domain Physical Infrastructure view.

The cloud network infrastructure can host multiple VIs, as it is illustrated in Figure E-1. These VIs are described in the same way as physical infrastructures. Note that we consider that VI nodes of a certain type (server or network) can only be accommodated in the physical infrastructure by nodes of the same type.

Table E-1 summarizes the notation used. The reference to physical resources uses letter *P*, e.g., N^P, and virtual resources use letter *V*, e.g., $N^V{}_v$, where *v* refers to a specific VI. With respect to the connectivity of an infrastructure, an adjacent matrix is used: $A^P = N^P \rightarrow N^P$ when referring to a physical infrastructure. The convention used for the index notation is the following: *i*, *j* for nodes and links in the physical network, and *m*, *n* for nodes and links in the VI.

Table E-1: VI Assignement Problem Notation – Single Domain

Symbol	Description
G^P	Physical Infrastructure
N^P	Set of Physical Nodes
S^P	Set of Server Nodes
R^P	Set of virtual-enabled Network Nodes
Rt^P	Set of transport Network Nodes
L^P	Set of Physical Links
i, j	Physical Nodes
ij	Physical Link
$Cs^P{}_i$	Number of CPUs of Physical Node i
$C^P{}_i$	Total CPU of Physical Node i
$M^P{}_i$	Total Memory of Physical Node i
$STG^P{}_i$	Total Storage of Physical Node i
$F^P{}_i$	CPU Frequency of Physical Node i
$B^P{}_{ij}$	Total Bandwidth of Physical Link ij
$D^P{}_{ij}$	Delay of Physical Link ij
$C^{Ptotal}{}_i$	Total CPU of Physical Node i
$M^{Ptotal}{}_i$	Total Memory of Physical Node i
$STG^{Ptotal}{}_i$	Total Storage of Physical Node i
$C^{Pfree}{}_i$	Free CPU of Physical Node i
$M^{Pfree}{}_i$	Free Memory of Physical Node i
$STG^{Pfree}{}_i$	Free Storage of Physical Node i
$G^V{}_v$	VI request v
$N^V{}_v$	Set of Virtual Nodes of VI request v
$S^V{}_v$	Set of virtual Server Nodes of VI request v
$R^V{}_v$	Set of virtual Network Nodes of VI request v
$L^V{}_v$	Set of Virtual Links of VI request v
m, n	Virtual Nodes
mn	Virtual Links
Cs	Number of CPUs of Virtual Node m of VI request v
$C^V{}_{v,m}$	CPU of Virtual Node m of VI request v
$M^V{}_{v,m}$	Memory of Virtual Node m of VI request v
$STG^V{}_{v,m}$	Storage of Virtual Node m of VI request v
$F^V{}_{v,m}$	CPU Frequency of Virtual Node m of VI request v
$B^V{}_{v,mn}$	Bandwidth of Virtual Link mn of VI request v
$D^V{}_{v,mn}$	Maximum Delay of Virtual Link mn of VI request v

E.3.2 Virtual Embedding Problem

The VI embedding problem is known to be NP-hard. A VI can be mapped according to one or multiple objectives, e.g.: occupy the lowest bandwidth possible, balancing the occupation of the physical nodes, or considering both previous objectives. Combining both objectives is not simple as they are not standalone, i.e. the choice of an ideal physical node may not allow the choice of the ideal physical link. This is considered to be a multiple-objective optimization problem (MOP) [12], where two or more conflicting objectives subject to constraints need to be simultaneously optimized.

In the following sections we will present the mathematical formulation for the optimal embedding as well as different strategies to solve the problem (sections E.4 and E.5).

E.4 Mathematical Formulation

In this section we present the mathematical formulation to solve the VI embedding problem. Following the optimization approach presented, special attention is given to the new formulation introduced that allows the re-optimization of VIs.

Assignment Variables

The formulation considers two binary assignment variables, *x* and *y*, one for the virtual nodes and another for virtual links as equations (1) and (2) show.

$$x_i^{v,m} = \begin{cases} 1, \text{virtual node } m \text{ of VI } v \text{ is allocated at physical node } i \\ 0, \text{else} \end{cases} \quad (1)$$

$$y_{ij}^{v,mn} = \begin{cases} 1, \text{virtual link } mn \text{ of VI } v \text{ is uses physical link } ij \\ 0, \text{else} \end{cases} \quad (2)$$

E.4.1 Generic Constraints

As mentioned in section E.3, virtual servers are assigned to physical server nodes. Equation (3) reflects this constraint plus the fact that a virtual node is assigned just to one physical node. Moreover, equation (4) assures that virtual network nodes are assigned to network nodes with virtual capabilities.

$$\forall v,m \in S^P : \sum_{i \in S^P} x_i^{v,m} = 1 \quad (3)$$

$$\forall v,m \in R^P : \sum_{i \in R^P} x_i^{v,m} = 1 \quad (4)$$

Further, equations (5), (6) and (7) guarantee that the capacity values of physical nodes, i.e. CPU load, memory, and storage, are kept within range. Equations (8) and (9) make sure that the number of cores and CPU frequency requirements are respected, i.e., a physical node selected to host a virtual node has at least the same number of cores and frequency as the virtual node.

$$\forall i : \sum_{m,v} x_i^{v,m} \times C_{v,m}^V \le C_i^{P_{total}} \quad (5)$$

$$\forall i : \sum_{m,v} x_i^{v,m} \times M_{v,m}^V \le M_i^{P_{total}} \quad (6)$$

$$\forall i : \sum_{m,v} x_i^{v,m} \times STG_{v,m}^V \le STG_i^{P_{total}} \quad (7)$$

$$\forall i,v,m : x_i^{v,m} \times Cs_{v,m}^V \le Cs_i^P \quad (8)$$

$$\forall i,v,m : x_i^{v,m} \times F_{v,m}^V \le F_i^{P_{total}} \quad (9)$$

Equation (10) applies the multi-commodity flow constraint [13] with a node-link formulation [14] in order to simultaneously optimize the mapping of virtual links and virtual nodes. Moreover, the notion of direct flows on the virtual links is also used.

$$\forall v, \forall m,n \in N_v^V(m), m < n, \forall i : \sum_{j \in N^P(i)} \left(y_{ij}^{v,mn} - y_{ji}^{v,mn} \right) = x_i^{v,m} - x_i^{v,n} \quad (10)$$

Equation (11) guarantees that each physical link selected has enough bandwidth available to host a virtual link. In other words, the sum of the amount of bandwidth of the virtual links that go through *i* must be equal or lower than the amount of free bandwidth in a physical link. Finally, equation (12) guarantees that the virtual link delay constraint is met, i.e. the maximum delay of the physical links assigned to a virtual link does not exceed the virtual link's maximum delay.

$$\forall i,j \in N^P(i), i < j : \sum_{v,m,n \in N^V(m), m < n} B_{v,mn}^V \times \left(y_{ij}^{v,mn} + y_{ji}^{v,mn} \right) \le B_{ij}^{P_{free}} \quad (11)$$

$$\forall v, \forall m,n \in N_v^V(m), m < n : \sum_{\forall i,j \in N^P(i), i < j} D_{ij}^P \times \left(y_{ij}^{v,mn} + y_{ji}^{v,mn} \right) \le D_{v,mn}^V \quad (12)$$

E.4.2 Formulations

In this section we present our specific formulations: maximum node and link load formulation, bandwidth consumption, and node and link state formulation. The maximum load formulation provides the load consumption of each type of resource (server node, network node and link) that is more loaded among all other resources. The bandwidth consumption formulation calculates the total bandwidth that a VI consumes. The node and link state formulation keeps track of the state of all resources, i.e. if they are active or not. In this section we specifically focus on the re-optimization formulation.

The ability to completely reconfigure (re-arrange) currently mapped VIs when trying to map a new VI increases the likelihood of a successful mapping. In this sense, we keep record of the currently mapped VIs in the physical infrastructure in variables *X* and *Y*. $X_i^{v,m}$ denotes if virtual node *m* of VI *v* is mapped at the physical node *i* ($X_i^{v,m}=1$) or not ($X_i^{v,m}=0$). $Y_{ij}^{v,mn}$ denotes if virtual link *mn* of the VI *v* uses the physical link *ij* ($Y_{ij}^{v,mn}=1$) or not ($Y_{ij}^{v,mn}=0$).

Completely reconfiguring VIs can give an upper bound for the VI acceptance ratio/revenue; however, this is done at a high cost. The disruption caused is extremely high, especially when virtual nodes are migrated. When considered, equation (13) guarantees that VI nodes already embedded have to remain in the same physical hosts. Equation (14) guarantees that virtual links already embedded remain using the same physical path. In this sense, when equations (13) and (14) are not considered, they allow the ILP to reconfigure nodes and/or links respectively.

$$\forall v \in V^{old}, \forall m \in N^V, \forall i \in N^P : x_i^{v,m} = X_i^{v,m} \tag{13}$$

$$\forall v \in V^{old}, \forall m,n \in N^V, \forall i \in N^P, m<n : y_{ij}^{v,mn} + y_{ji}^{v,mn} = Y_{ij}^{v,mn} \tag{14}$$

Moreover, re-optimizations of VIs are registered. We keep track of node and link changes within each VI. $X_{v,m}^{change}$ registers if virtual node *m* in VI *v* is moved ($X_{v,m}^{change}=1$) or not ($X_{v,m}^{change}=0$). This constraint is reflected in equation (15). $Y_{v,mn}^{change}$ registers if virtual link *mn* in VI *v* was reconfigured ($Y_{v,mn}^{change}=1$) or not ($Y_{v,mn}^{change}$=0), which is guaranteed by equations (16), (17) and (18). Note that we assume that the number of changes in the physical links used by a virtual link is counted only once. Variables $Y_{v,mn}^{changeAUX}$ and *K* are auxiliary variables that help to keep track of virtual link changes. $Y_{v,mn}^{changeAUX}$, in equation (16), provides the number of physical links affected by changing (either by supporting or no longer supporting) the embedding of virtual link *mn*. Equation (17) guarantees that, if $Y_{v,mn}^{changeAUX}=0$, then $Y_{v,mn}^{change}=0$. In equation (18) variable *K* is a sufficient large number to ensure that $Y_{v,mn}^{change}=1$ if $Y_{v,mn}^{changeAUX} \geq 1$.

$$\forall v \in V^{old}, \forall m \in N_v^V : \sum_{i \in N^P} \left(X_i^{v,m} - x_i^{v,m}\right) \times X_i^{v,m} = X_{v,m}^{change} \tag{15}$$

$$\forall v \in V^{old}, \forall m,n \in N_v^V, m<n : \sum_{i,j \in N^P} \left(Y_{ij}^{v,mn} - (y_{ij}^{v,mn} + y_{ji}^{v,mn})\right) \times Y_{ij}^{v,mn} = Y_{v,mn}^{changeAUX} \tag{16}$$

$$\forall v \in V^{old}, \forall m,n \in N_v^V, m<n : Y_{v,mn}^{change} \leq Y_{v,mn}^{changeAUX} \tag{17}$$

$$\forall v \in V^{old}, \forall m,n \in N_v^V, m<n : \frac{Y_{v,mn}^{changeAUX}}{K} \leq Y_{v,mn}^{change} \tag{18}$$

Furthermore, we also define V_v^{Change} to count the number of affected VIs. Similarly to what was previously explained, equations (19), (20) and (21) reflect the associated constrains. V_v^{Change} assumes value 1 if VI *v* was subject of a re-optimization process, and 0 otherwise. Variables $V_v^{ChangeAUX}$ and *K* are auxiliary

variables that help to keep track of VI changes. $V_v^{ChangeAUX}$, in equation (19), provides the number of virtual nodes and links in VI *v* affected by changing (either by no longer being embedded in a certain physical resource, or by embedding in a new one) the embedding of these resources. Equation (20) guarantees that if $V_v^{ChangeAUX} = 0$, then $V_v^{Change} = 0$. In equation (21) variable *K* is a sufficient large number to ensure that $V_v^{Change} = 1$ if $V_v^{ChangeAUX} \geq 1$.

$$\forall v \in V^{old} : \left(\sum_{m \in N_v^V} X_{v,m}^{change} \right) + \left(\sum_{m,n \in N_v^V : m<n} Y_{v,mn}^{change} \right) = V_v^{ChangeAUX} \quad (19)$$

$$\forall v \in V^{old} : V_v^{Change} \leq V_v^{ChangeAUX} \quad (20)$$

$$\forall v \in V^{old} : \frac{V_v^{ChangeAUX}}{K} \leq V_v^{Change} \quad (21)$$

Since moving nodes and links has a significantly different impact, we define two other counters for affected VIs, one based only on virtual node changes – equations (22), (23) and (24) – and another based only on virtual link changes – equations (25), (26) and (27). $V_v^{NodeChange}$ assumes value 1 if VI *v* was affected by a virtual node change, or 0 otherwise. The same is true for $V_v^{LinkChange}$ with respect to virtual link changes. Variables $V_v^{NodeChangeAUX}$, $V_v^{LinkChangeAUX}$, and *K* are used for the same purpose as in the cases before, i.e. to help keeping the values of $V_v^{NodeChange}$ and $V_v^{LinkChange}$ within their boundaries (0 or 1).

$$\forall v \in V^{old} : \sum_{m \in N_v^V} X_{v,m}^{change} = V_v^{NodeChangeAUX} \quad (22)$$

$$\forall v \in V^{old} : V_v^{NodeChange} \leq V_v^{NodeChangeAUX} \quad (23)$$

$$\forall v \in V^{old} : \frac{V_v^{NodeChangeAUX}}{K} \leq V_v^{NodeChange} \quad (24)$$

$$\forall v \in V^{old} : \sum_{m,n \in N_v^V : m<n} Y_{v,mn}^{change} = V_v^{LinkChangeAUX} \quad (25)$$

$$\forall v \in V^{old} : V_v^{LinkChange} \leq V_v^{LinkChangeAUX} \quad (26)$$

$$\forall v \in V^{old} : \frac{V_v^{LinkChangeAUX}}{K} \leq V_v^{LinkChange} \quad (27)$$

E.5 Embedding Strategies

The mathematical formulations presented in the previous section are the foundation for the definitions of different embedding strategies. In this section we present a set of embedding strategies based on those formulations.

We divide the presented strategies in three main categories: base strategies, dynamic strategies, and reconfiguration strategies. The elaboration of these strategies relies on the well-known approach to multi-objective problems (MOP), the aggregation (or weighted) method. Furthermore, we use an *a priori* approach that consists in having the weights (w_i) defined *a priori* [12].

E.5.1 Base Strategies

These strategies are considered base ones because they will be part of the strategies presented in the following categories.

1) Load Balancing (Node and Link) + Shortest Path - VIE-Opt-LB+SP

This strategy aims to minimize the maximum load of node and link resources by also taking into account the total bandwidth consumption of a VI. $S_{load}^{\max}$ and $R_{load}^{\max}$ refer to the maximum load consumption of server and virtual-enabled network nodes, respectively. $L_{load}^{\max}$ refers to the maximum load consumption of links, and B_{cons} refers to the total amount of bandwidth consumed by a VI. In this case, the weights w_1 and w_2 have equal values so that there is an equal balance between maximum node load and overall link load (by considering both maximum link load as well as total bandwidth consumed).

$$min\ \omega_1\left(S_{load}^{\max} + R_{load}^{\max}\right) + \omega_2\left(L_{load}^{\max} + B_{cons}\right) \quad (28)$$

2) Green strategy - VIE-Opt –Green

This strategy aims to minimize the energy consumption of the physical infrastructure. The main objective is to minimize the number of active nodes (x_i^{Active}) as they are the main energy consumption source. The second objective is to minimize the number of active links (y_{ij}^{Active}) in order to reduce the overall energy consumption of the active nodes - equation (29). In this case, w_1 is sufficiently higher than w_2 so that the primary objective is to minimize the number of active nodes, and ε is a very small value so that another objective can be considered as a tiebreak objective, presented in the equation as *Strat*.

The objective considered in *Strat* in the evaluation section is the load balancing with shortest path (*VIE-Opt-LB+SP* – equation (28)). Load balancing the active elements of the infrastructure will ease the future embedding of VIs without having to increase the number of active elements.

$$min\ \omega_1\left(\sum_{i\in N^p} x_i^{Active}\right) + \omega_2\left(\sum_{i,j\in N^p, i<j} y_{ij}^{Active}\right) + \varepsilon\left(Strat\right) \quad (29)$$

E.5.2 Dynamic Strategies - Adjusting Weights

This strategy combines different objectives in a dynamic way. In the previous cases there was always a fixed hierarchy of objectives, i.e. a first one, a second one, and eventually a third one as presented in equation (29). In this case, we do not keep that hierarchy fixed and change it depending on the physical infrastructure load.

This strategy uses the green strategy (*VIE-Opt –Green*) and the load balancing with shortest path strategy (*VIE-Opt-LB+SP*). In equation (30) weights (z_1 and z_2) are not fixed along time, but change depending on the infrastructure state at the time of each VI arrival. More precisely, we consider that, if the infrastructure is loaded over a certain limit, the load balancing strategy is applied. If the load is below or equal to the limit, the green strategy is applied.

$$min\ z_1\left(Strat\_1\right) + z_2\left(Strat\_2\right) \quad (30)$$

E.5.3 Re-optimization Strategies

The strategies defined in this category allow the reconfiguration of already deployed VIs. We present two types of strategies: those that allow only link re-optimization; and those that allow both link and node re-optimization.

Taking advantage of the strategies presented in the previous subsection, the following strategies are defined:

- Load Balancing (Node and Link) + Shortest Path strategy with Link Re-optimization - VIE-ReOpt-LB+SP
- Load Balancing (Node and Link) + Shortest Path strategy with Node and Link Re-optimization - VIE-LinkReOpt-LB+SP
- Green strategy with Link Re-optimization - VIE-LinkReOpt-LB-R
- Green strategy with Node and Link Re-optimization- VIE-ReOpt –Green

- Green + Load Balancing with Link Re-optimization - VIE-LinkReOpt-D-Green-LB
- Green + Load Balancing with Node and Link Re-optimization - VIE-ReOpt-D-Green-LB

Node reconfigurations are considered to be the most expensive ones, while link reconfigurations can be easily achieved and, to a certain extent, they can be neglected when considering the use of technologies like SDN. With this assumption in mind, we do not penalize link reconfigurations in the refered strategies.

The strategies that allow only the re-optimization of links keep the original objective functions. However, those that allow also node re-optimization suffer a change, presented in equation (2). Since node reconfigurations are considered very expensive, the objective functions first try to minimize the number of virtual node reconfigurations. ε is a very small value so that *Strat* is considered as the last objective.

$$min \left(\sum_{v \in V} \sum_{m \in N_v^V} X_{v,m}^{change} \right) + \varepsilon(Strat) \quad (2)$$

Depending on the strategy used, *Strat* will assume different forms. For example, in the strategy VIE-ReOpt-LB+SP, *Strat* will assume the form of equation (28), which leads to the objective function presented in equation (32). In this case, since both virtual node and link reconfigurations are allowed, equations (13) and (14) are not considered. In the case of strategy VIE-LinkReOpt-LB+SP, equation (13) is considered to prevent virtual node reconfigurations.

$$min \left(\sum_{v \in V} \sum_{m \in N_v^V} X_{v,m}^{change} \right) + \varepsilon\left(\omega_1\left(S_{load}^{\max} + R_{load}^{\max}\right) + \omega_2\left(L_{load}^{\max} + B_{cons}\right)\right) \quad (3)$$

E.6 Evaluation

This section evaluates the proposed embedding approaches. Here, we provide a thorough evaluation of the different optimal embedding strategies presented in section E.5.

Scenario description

To evaluate the performance of the proposed approaches, a Matlab [15] simulator is used. For each run, the program designs a random physical infrastructure of 20 nodes based on the Waxman network topology generator [16], and it simulates a set of requests of VIs (with a number of nodes between 4 and 14) with Markov-modulated inter-arrival and inter-departure times. Both physical infrastructure and the generated VIs have 70% of the nodes as servers (rounded to the higher integer), and the remaining 30% as network nodes. These values were chosen because, in a cloud computing context, the required computing node capacity is much higher than the network one. The physical infrastructure has a size of 20 nodes, while the size of VIs varies between 4 and 14 nodes. The reason for these values lays on the fact that these dimensions allow for the physical infrastructure to reach a saturation point, which will show the gains of the different strategies, if they exist.

Details on the nodes and link parameters can be seen in Table E-2. Moreover, the VI request rates (λ) vary between 2 and 5 VIs per time unit (Poisson arrivals), and the average duration of the VIs ($1/\mu$, where μ is the average service rate) is 20 time units (exponentially distributed duration).

Table E-2: Physical and virtual infrastructure parameters

		Physical Infrastructure	Virtual Infrastructures
Net Nodes	**Cs**	{2; 4; 6; 8}	{1; 2; 3; 4 }
	F(Hz)	{2.0-3.2 / 0.2 steps}	{2.0-3.2 / 0.1 steps}
	Memory	{2; 4; 6}(GB)	{64; 128; 256; 512}(MB)
Server Nodes	**Cs**	{8; 16; 32; 64}	{1; 2; 4; 8; 16; 32; 64}
	F(Hz)	{2.8-3.2 / 0.2 steps}	{2.8-3.2 / 0.1 steps}
	STG (GB)	{6400; 12800; 25600}	{100; 200; 400; 800; 1600}
	M (GB)	{256; 512; 1024}	{2; 4; 8; 16; 32; 64}
Links	**B (Mbps)**	{500-2000 / 500 steps}	{10-100 / 10 steps}
	D (ms)	{5-10 / 1 steps }	{0-40 / 5 steps}

In order to solve the ILP, we have used CPLEX [17] version 12.3, integrating a plug-in for our Matlab simulator, and setting a time limit of 600 seconds (i.e. 10 minutes) for each VI mapping.

The weight values used for the optimal strategies are the following:

- w_1 and w_2 are 0.5 (50%);
- ε is 0.001;

z_1 and z_2 have dynamic values which change depending on the state of the physical infrastructure. If the maximum load of any resource (server node, network node or link) is below 75%, *Opt-Green* is applied; otherwise, *Opt-LB+SP* is applied. The 75% value was chosen empirically due to its good results in a large set of experiments.

We analyze the VI acceptance ratio as one of the main indicators for the performance of the algorithms. Moreover, the energy consumption of the physical infrastructure is also analyzed, based on the active infrastructure resources along time. Finally, the impact of the re-optimization processes in the VI is carefully evaluated.

All values in the following graphics present a mean of 10 runs (with different substrate) with a confidence interval of 95%.

E.6.1 Virtual Infrastructure Acceptance Ratio analysis

This section analyses the different strategies in terms of acceptance ratio. Figure E-3 compares the results of the strategies with and without link re-optimization. It is clear that, in this case, the strategies allowing only link reconfiguration do not present significant differences as compared to the ones without reconfiguration. This happens because the strategies allowing only link re-optimization are limited in terms of virtual link mapping alternative solutions that make a reconfiguration viable. This lack of alternatives is mostly due to the fact that virtual nodes cannot be moved.

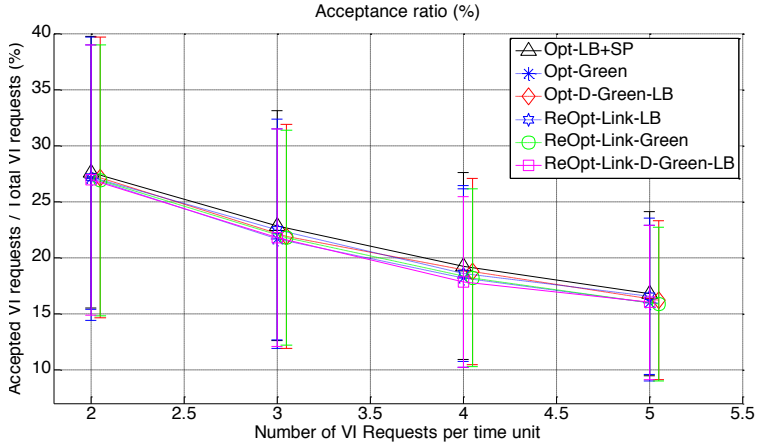


Figure E-2: VI acceptance ratio – with link re-optimization

On the other hand, the strategies allowing both node and link re-optimization outperform all others – see Figure E-3. The values are, in average, up to 5% higher. The ReOpt-Green strategy is the one presenting the best performance (with values between 32% for a VI arrival rate of 2 and 19% for an arrival rate of 5), followed by the ReOpt-D-Green-LB and the ReOpt-Lb+SP. The difference between the three strategies is low (a gap between strategies of approximately 1%). We notice that, in theory, all three strategies should present the exact same values in terms of acceptance ratio as they all allow full reconfiguration of resources. This does not happen due to the time constraint set to solve the ILP problem (600 seconds). In other words, each strategy takes different time to find a VI embedding solution, and in some cases no solution is found within the defined timeframe.

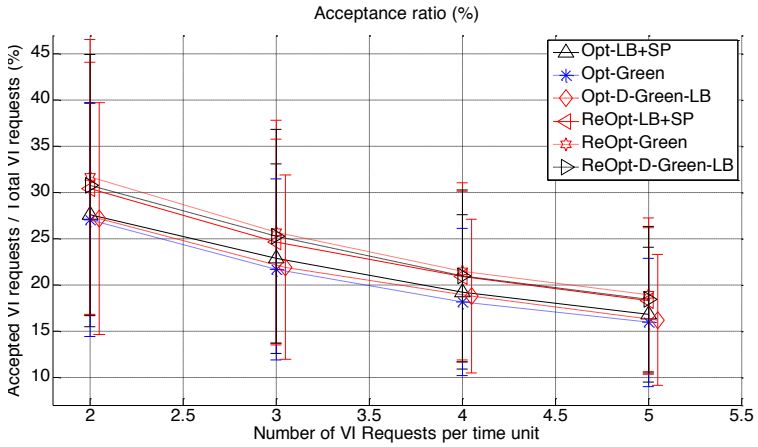


Figure E-3: VI acceptance ratio – with node and link re-optimization

E.6.2 Physical Infrastructure Energy Consumption (cumulative time of active resources) analysis

Herein it is analyzed the performance of the different strategies in terms of the physical infrastructure energy consumption. While in terms of acceptance ratio some strategies present values relatively close, the differences with respect to energy consumption are more pronounced. The analysis is done taking into account the cumulative time of active resources (node and link), and the results are depicted in Figure E-4 and Figure E-5. The values are presented in percentage, where 100% means that all resources were active during the entire simulation time.

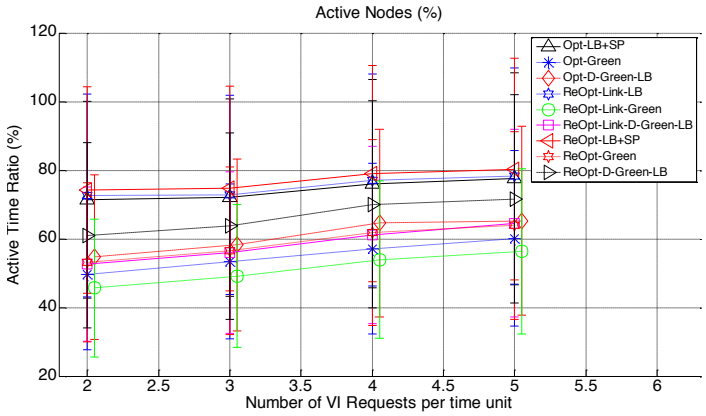


Figure E-4: Cumulative time of active nodes

First we analyze the values related to nodes, Figure E-4, as it is the lead indicator of the energy consumed. Not surprisingly, all strategies that consider the green policy outperform the remaining ones in terms of energy consumption, i.e. they present lower values. The VIE-ReOpt-Link-Green strategy clearly outperforms the remaining ones, with values between 47% for arrivals of 2 VIs per time unit, and 57% for 5 VIs per time unit. The next strategy is the VIE-Opt-Green strategy, with values between 50% and 60%. It is important to highlight that these two strategies perform very similarly in terms of acceptance ratio, but by allowing link reconfiguration, the VIE-ReOpt-Link-Green presents gains up to 3% with respect to active nodes.

These strategies are then followed by VIE-ReOpt-Green and the VIE-ReOpt-Link-D-Green-LB, followed by the VIE-Opt-D-Green-LB, with the first two with values between 53% and 63%, and the latter one between 55% and 64%. Although having close values, it is important not to forget that the VIE-ReOpt-Green strategy presents gains up to 5% in terms of acceptance and still manages to present lower values in terms of active

nodes. The VIE-ReOpt-D-Green-LB is the following strategy presenting better performance, with values between 61% and 71%.

The remaining optimal strategies, those that do not apply the green strategy, present higher and very close values, between 71% and 81%.

We highlight here the fact that, among the strategies that consider energy consumption, those that allow only link re-optimization present similar acceptance ratio values to those not allowing any re-optimization. However, the former ones do improve the node energy consumption indicators. If the impact of link reconfiguration is considered to be low to the point that it can be neglected, these strategies should indeed be considered as better approaches than those performing no reconfiguration.

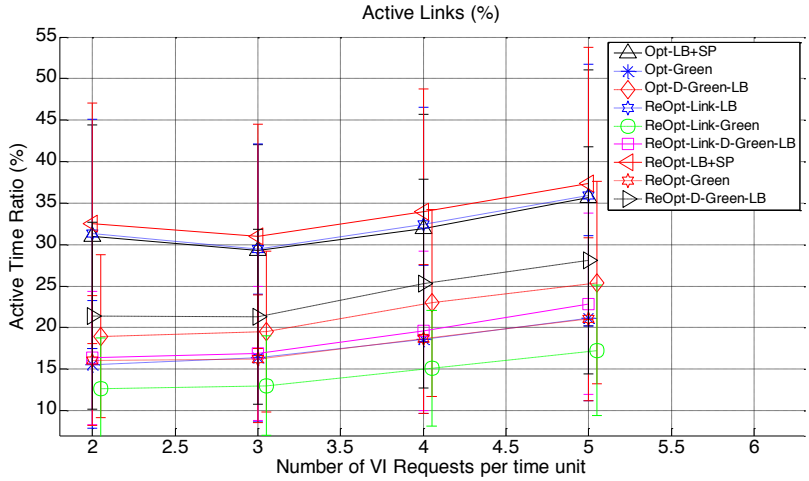


Figure E-5: Cumulative time of active links

In terms of active links (Figure E-5), the strategies have similar performance. The optimal strategies with green approaches perform better, with values of active time between 12% and 27%. The optimal strategies with load-balancing policies present values between 29% and 37%.

Note that, just like the acceptance ratio indicator, the values here present a linear behavior as the VI arrival rate increases. However, as opposite to the acceptance ratio, here they increase as the VI arrival rate increases. This is due to the fact that, although the acceptance ratio decreases with the increase of the VI arrival rate, the overall number of VIs embedded in the physical infrastructure increases, which leads to a higher percentage of active resources in the substrate.

Furthermore, it is clear that, allowing resource optimization in the strategies that consider energy consumption helps reducing the energy consumption of the physical infrastructure without compromising the acceptance ratio. In strategies where node and link reconfiguration are allowed, the acceptance ratio is increased.

E.6.3 Re-optimization impact analysis

In this subsection, it is performed an analysis to the different strategies that consider re-optimization of resources in terms of the impact of changes in VIs that they naturally lead to. The analysis is done in five different perspectives: i) VIs affected by reconfiguration, whether by link or node; ii) VIs affected by link reconfiguration; iii) VIs affected by node reconfiguration; iv) overall number of link reconfigurations; v) overall number of migrated nodes.

Figure E-6 shows the number of VIs affected by reconfigurations (due to node and/or link). The VIE-ReOpt-D-Green-LB presents the highest value with a mean of 1.8 VIs affected per re-optimization process, followed by the VIE-ReOpt-LB+SP with a mean of 1.6. Then, it follows the strategies VIE-ReOpt-Green, VIE-ReOpt-Link-LB and VIE-ReOpt-Link-D-Green-LB with mean values of affected VIs of 1.3, 1.2 and 1, respectively. The strategy that shows lower impact is the VIE-ReOpt-Link-Green, with a mean of 0.6 affected VIs.

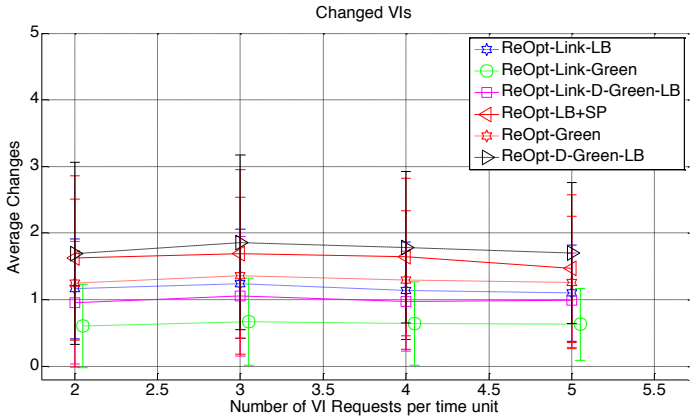


Figure E-6: VIs affected by changes

The gap between the VIE-ReOpt-Link-Green and the other two strategies that allow only link reconfiguration (i.e. VIE-ReOpt-Link-LB and VIE-ReOpt-Link-D-Green-LB) is due to the fact that the former is a pure green strategy that tends to load active resources (nodes and links), which leads resources to a point of saturation much faster than the load balancing strategy. Therefore, with a higher probability of saturating nodes, trying to reconfigure virtual links to map a new VI will be in some cases inglorious. On the other hand, in the load balancing strategy, the probability of nodes reaching a saturation point is much lower than in the former case, which will allow more fruitful reconfigurations of virtual links.

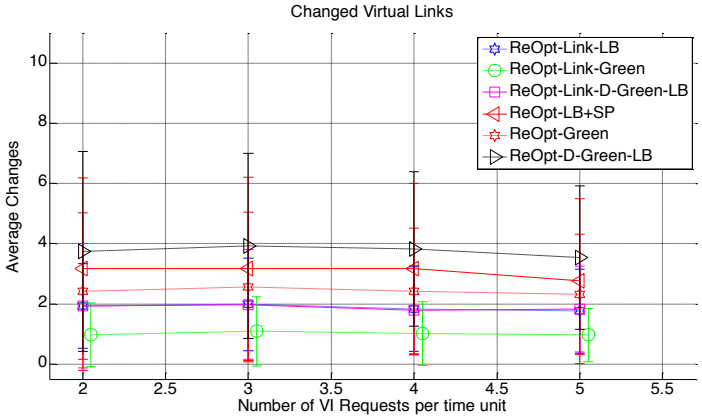


Figure E-7: Virtual link changes

Figure E-7 shows the values related to the overall number of links reconfigured. The ranking here is the same as before. VIE-ReOpt-D-Green-LB (average of 3.9) is the strategy that triggers more link reconfigurations, followed by the VIE-ReOpt-LB-SP (average of 3.1), VIE-ReOpt-Green (average of 2.5), VIE-ReOpt-Link-LB (average of 1.9), VIE-ReOpt-Link-D-Green-LB (average of 1.9), and finally VIE-ReOpt-Link-Green (average of 1).

The number of VIs affected by the reconfiguration of links is depicted in Figure E-8. Here the ranking remains the same as before.

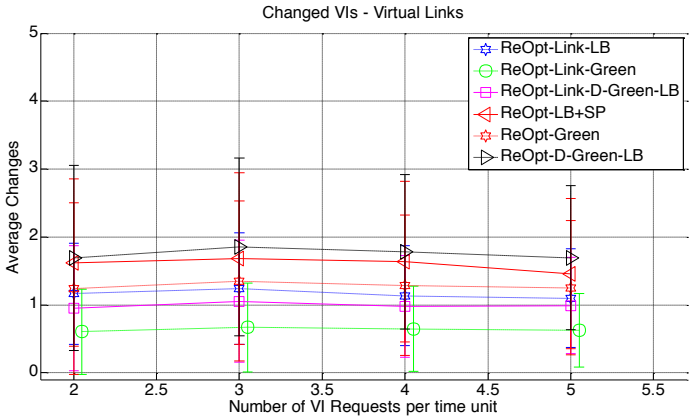


Figure E-8: VIs affected by virtual link changes

With respect to the number of nodes reconfigured, the values between the three strategies that allow node reconfiguration are close – see Figure E-9. The mean values are between 0.5 and 0.6 of virtual nodes that change. However, the number of VIs affected by the migration of nodes is only nearly 0.15 as shown in Figure E-10.

These results show that the node re-optimization impact can be low. However, it is considered as future work the characterization of VI nodes in terms of their ability/cost to me migrated. In this sense, the embedding strategy can better assess which nodes it should first try to migrate.

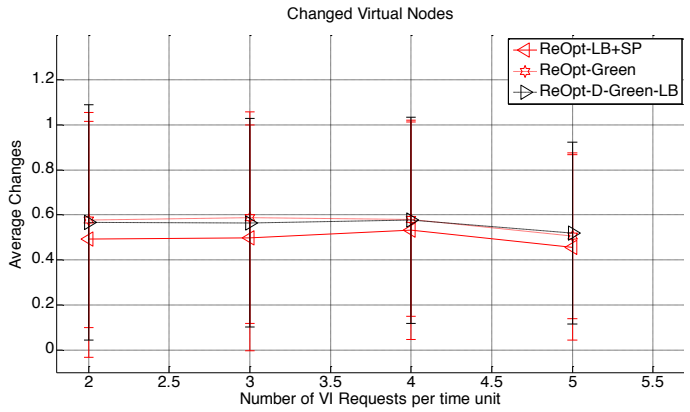


Figure E-9: Virtual node changes

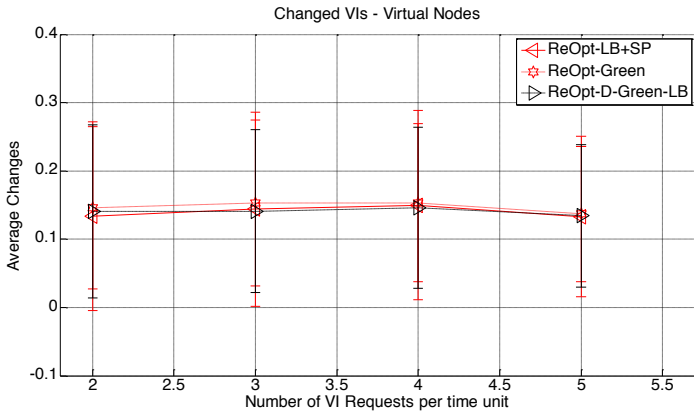


Figure E-10: VIs affected by virtual node changes

E.6.4 Discussion

Strategies allowing full re-optimization of resources clearly outperform the remaining ones in terms of acceptance ratio. However, the adoption of these strategies needs to take into consideration the costs of reconfiguring resources. The values in terms of reconfiguration impact presented in this work are optimistic. Furthermore, since each VI and resource can have specific requirements, an evolvement of these strategies could be to classify resources with respect to their ability/cost to be migrated. This aspect can be especially important in virtual nodes, since they are the ones more difficult to move. We believe that the strategies presented in this work provide a solid base foundation that can be extended for more specific and detailed scenarios as the one just mentioned.

Moreover, the improvements achieved are not strict to the acceptance ratio. Energy consumption can also be improved by allowing re-optimization in strategies with green objectives.

In the cases of strategies allowing only link re-optimization, the values of acceptance ratio remain nearly the same; however, the energy consumption can be improved with green objectives. These are definitely strategies worth taking into account, since the cost of reconfiguring links is much lower than those of reconfiguring nodes.

E.7 Conclusion and Future Work

This article considered a future approach where cloud infrastructure services will allow the definition of complete infrastructures, to which we refer to as VIs. These VIs comprise both computing, storage and network resources. The focus of this work was on the VI embedding problem in a single domain perspective.

Optimal formulations and strategies based on ILP to solve the embedding and re-optimization problem were proposed. These strategies take into account the load balancing of physical resources, the energy consumption of the physical infrastructure, and the impact of the VI re-optimization process.

A comparative performance analysis between the different strategies was performed taking into account the acceptance ratio, the energy consumption of the physical infrastructure and the impact of re-optimizations. The study of the optimal strategies shows that there is a clear tradeoff between improving acceptance ratio and reducing the physical infrastructure usage without allowing re-optimizations. The use of different strategies depending on the load of the physical infrastructure resources can be a way to deal with this tradeoff. Moreover, allowing virtual resources to be reconfigured can considerably increase the acceptance ratio, as well as reduce the energy consumption of the physical infrastructure. This is also clear in the case where only virtual links are allowed to be reconfigured.

As future work, further study regarding the combination of different strategies depending on the status of the physical infrastructure is required. Moreover, the extension of the energy consumption analysis by means of calculating the approximate energy consumption of node depending on its load would improve the formulation. Finally, extensions to the formulations can be accommodated, such as the one of dynamic/elastic VIs (VIs that vary the amount of resources along time).

References

[1] ETSI, Network Functions Virtualisation (NFV): Architectural Framework, Technical Report ETSI GS NFV 002 v1.1.1, Oct. 2013.

[2] ETSI, Network Functions Virtualisation (NFV): Use-cases, Technical Report ETSI GS NFV 001 v1.1.1, Oct. 2013.

[3] N. Yong; X.C. Liang; Z. Kai, Connectivity as a Service: Outsourcing Enterprise Connectivity over Cloud Computing Environment, Computer and Management (CAMAN), 2011 International Conference on , vol., no., pp.1,7, 19-21 May 2011.

[4] W.H. Liao; S.C. Su, A Dynamic VPN Architecture for Private Cloud Computing, Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on , vol., no., pp.409,414, 5-8 Dec. 2011.

[5] A. Mahimkar, A. Chiu, R. Doverspike, M.D. Feuer, P. Magill, E. Mavrogiorgis, J. Pastor, S.L. Woodward, and J. Yates. 2011. Bandwidth on demand for inter-DC communication. In

Proceedings of the 10th ACM Workshop on Hot Topics in Networks (HotNets-X). ACM, New York, NY, USA, , Article 24 , 6 pages.
[6] J. Soares, J. Aparicio, S. Sargento, Optimizing the Embedding of Virtualized Cloud Network Infrastructures, submitted to IEEE Transactions on Cloud Computing, June 2014.
[7] P.N. Tran, L. Casucci, A. Timm-Giel, Optimal mapping of virtual networks considering reactive reconfiguration, Cloud Networking (CLOUDNET), 2012 IEEE 1st International Conference on , vol., no., pp.35,40, 28-30 Nov. 2012
[8] I. Houidi, W. Louati, W.B. Ameur, D. Zeghlache, Virtual network provisioning across multiple substrate networks, Computer Networks, Volume 55, Issue 4, 10 March 2011, Pages 1011-1023, ISSN 1389-1286, http://dx.doi.org/10.1016/j.comnet.2010.12.011.
[9] B. Kantarci, H. Mouftah, Minimizing the provisioning delay in the cloud network: Benefits, overheads and challenges. Computers and Communications (ISCC), 2012 IEEE Symposium on, (pp. 806-811), 1-4 July 2012.
[10] B. Kantarci, H. Mouftah, Optimal Reconfiguration of the Cloud Network for Maximum Energy Savings. Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on, (pp. 835-840), 13-16 May 2012.
[11] B. Kantarci, H. Mouftah, Overcoming the energy versus delay trade-off in cloud network reconfiguration, Computers and Communications (ISCC), 2012 IEEE Symposium on , vol., no., pp.000053,000058, 1-4 July 2012.
[12] Hoboken, Metaheuristics: from design to implementation . John Wiley & Sons, 2009.
[13] S. Even, A. Itai, A. Shamir, On the complexity of time table and multi-commodity flow problems, Foundations of Computer Science, 16th Annual Symposium on, (pp. 184-193), 1975.
[14] M. Pioro, D. Medhi. Routing, Flow, and Capacity Design in Communication and Computer Networks. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc, 2004.
[15] MATLAB Release 2010, The MathWorks, Inc., Natick, Massachusetts, United States.
[16] MATLAB, Waxman Network Topology Generator, http://www.mathworks.com/matlabcentral/fileexchange/2517-waxman-network-topology-generator/content/waxtop.m
[17] IBM. IBM ILOG Optimization Products. www-01.ibm.com/software/websphere/products/optimization. Retrieved 13 May 2013.

Paper F. Optimizing the Embedding of Virtualized Cloud Network Infrastructures in Multi-Domain Scenarios

João Soares, Susana Sargento

in IEEE International Conference on Communications (ICC), 2015

The format has been revised

Optimizing the Embedding of Virtualized Cloud Network Infrastructures across Multiple Domains

João Soares and Susana Sargento

Abstract

Network is now a main part of the cloud resources, becoming more than a connectivity add-on. The ability to define network resources (e.g. routing/switching elements, bandwidth, delay) in this context is still scarce, but there are clear evidences that this is a future reality, as is the case of Network Functions Virtualization (NFV). In this paper we consider that cloud infrastructure services will allow the definition of complete infrastructures, to which we refer as Virtual Infrastructures (VIs). These VIs comprise both computing, storage and network resources.

This paper specifically tackles the VI embedding problem in scenarios with multiple domains. These scenarios consider multiple Data Center (DC) locations inter-connected through an operator network. The embedding problem is known to be NP-hard, and therefore we present an embedding strategy based in Integer Linear Programming (ILP) formulation. The proposed strategy aims to balance the load among the different domains, and considers the location as a key constraint for virtual resources. Finally, a thorough evaluation of the formulation is performed, analyzing the VI acceptance ratio and the occupation of the physical infrastructure. The obtained results show that the location constraint has a high impact on both the acceptance ratio and occupation of physical resources.

Keywords - Cloud computing, infrastructure-as-a-serivce, network-as-a-service, integer linear programming, virtualization, virtual infrastructure, multiple domains.

F.1 Introduction

The momentum around the Network Functions Virtualization (NFV) [1] [2] concept can be seen as the most recent catalyst to the evolution of cloud infrastructure solutions and services. From a high-level perspective, NFV envisions accelerating the innovation of networks and services, allowing, among other things, new operational approaches, novel services, faster service deployment (shorter time to market), service assurance and security. From a low-level perspective, it proposes to use standard IT virtualization related technologies to accommodate network functions in order to accomplish the above mentioned high-level benefits.

The cloud evolution towards NFV brings the need for an active role of the network in the cloud as a mean to guarantee performance, reliability, and security to cloud services both inside DCs as well as outside, i.e. on the Wide Area Network (WAN).

The ongoing cloud evolution towards NFV compliance has re-leveraged the concept of Network-as-a-Service (NaaS), which can go from providing simple connectivity [3], connectivity with bandwidth and security requirements [4] [5], up to providing a more complete approach with network and connectivity elements (nodes and links). Just like in today's cloud, the key technology enabler behind these concepts lays upon virtualization techniques. The "as-a-Service" concept is, among other aspects, related with dynamics and agility. This need for evolution is on the birth of concepts like Software Defined Networking (SDN). Among the opportunities that SDN brings is the effective "unlock" of the NaaS concept. In this scenario of evolution, the IaaS and NaaS concepts are naturally tending to merge, which requires an integrated management of resources across cloud and network providers.

This paper addresses the joint embedding of cloud and network resources belonging to multiple domains. We consider the provisioning of cloud and network resources through virtualization (virtualized computing, storage, and network), enabling the support of Virtual Infrastructures (VIs). The VI embedding problem is addressed through an Integer Linear Programming (ILP) approach, in a multi-domain perspective, where a set of cloud facilities (i.e. DCs) are interconnected via an operator network. The vision

of this scenario is becoming extremely common, especially within the NFV scope - e.g. an NFV Infrastructure (NFVI) [1] [2].

In our previous works [6] and [7], we addressed the single domain scenarios with respect to VI embedding and re-optimization. In this work it is introduced, for the first time, the formulation for multi-domain scenarios. We propose a VI embedding strategy that takes into account the load balance of the physical infrastructure. The strategy is submitted to a thorough evaluation process, where it is shown that considering location as a constraint has a high impact on the embedding strategy's performance.

The remainder of this paper is organized as follows. Section F.2 presents the related work. Section F.3 details the concept of a cloud network virtual-enabled infrastructure in multi-domain scenarios. Furthermore, it elaborates on the resource allocation problem. Section F.4 presents the mathematical formulations for the embedding, and section F.5 proposes an embedding strategy based on this formulation. Further, section F.6 depicts and discusses the evaluation of the solution. Finally, section F.7 provides final conclusions and future work.

F.2 Related Work

This section presents works that closely relate to the VI embedding problem.

In [6], we have first addressed the VI embedding problem for the single domain case. Different optimal strategies were proposed, considering aspects like load balancing and energy consumption. Furthermore, a heuristic algorithm was also presented.

Other works have looked to the cloud network field, such as Kantarci in [9], [10] and [11]. The work in [9] studies the delay minimization in the cloud network through a Mixed ILP (MILP) formulation. In [10] the authors address the reconfiguration of the cloud network in order to maximize the energy savings, and propose two heuristic approaches benchmarked by MILP approaches. The trade-off between energy savings and delay minimization was later studied in [11]. However, just like in the remaining works, the interplay between CPU, memory, storage and network resources is not considered.

With the latter fact in mind, in [7] we extended the single domain formulation of [6] to enable VI reconfigurations and improve the performance of strategies both in terms of acceptance ratio and physical infrastructure energy consumption.

The work in [8] studies the embedding of VNs in a multiple infrastructure provider's scenario. The embedding is formulated and solved as a MILP with the focus of decreasing the embedding cost for providers while increasing the VN acceptance ratio. This is an interesting work in the multi-domain area, but it is limited to VNs and not fully compliant with the scenario envisioned in this work, where there is one network domain and multiple DC domains.

The joint manipulation of cloud and network virtual resources has not yet been widely explored. In our previous works we have used this inspiration to work on the VI embedding for single domain scenarios. However, to the best of our knowledge, the multi-domain perspective that we envision in this work, which is the joint manipulation of cloud and network resources across DCs and the operator network, has not been addressed yet in the literature.

F.3 Cloud Network Virtual-Enabled Infrastructure

We consider the concept of cloud network virtual-enabled infrastructure, where cloud and network resources are part of a pool of resources, can be virtualized, and are managed in an integrated way.

In this work we look at a multi-domain perspective, where cloud and network resources are hosted in specific domains (e.g. DCs or Points of Presence -PoPs) that are interconnected by a network domain (e.g. an Operator Network). This section describes this perspective as well as the notation used. Moreover, it details the embedding allocation problem.

F.3.1 Multi-Domain scenario

The multi-domain perspective considers an infrastructure where multiple DCs of varied size and capacity are geographically dispersed and interconnected by a network, which can be an Operator Network or an

overlay network within the Operator's Network. In both cases, this network is refered to as Inter-DC Network. Figure F-1 illustrates this perspective, which resembles the ETSI's view of a NFV Infrastructure [1].

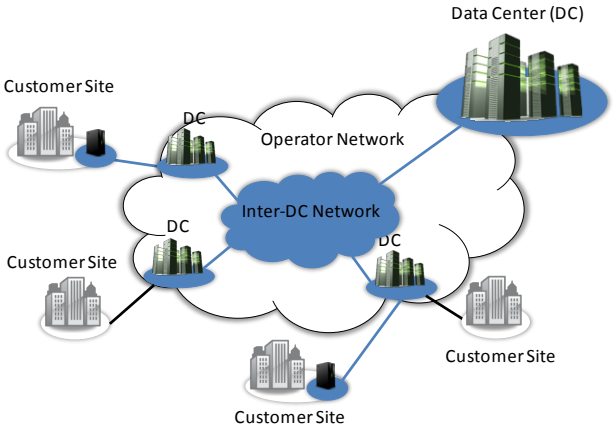


Figure F-1: Multi-Domain Physical Infrastructure view.

At the VI level two types of nodes are considered, network nodes and server nodes, each with its specific set of associated characteristics. Network nodes are characterized by the number of CPU, denoted by *Cs*, and by the amount of memory, denoted by *M*. Server nodes are characterized by the same parameters as network nodes (*Cs*, and *M*) plus storage capabilities, denoted by *STG*. With respect to links, these are characterized by bandwidth capacity (*B*) and assumed to be bidirectional and with a maximum delay (*D*). Moreover, associated to the number of CPUs of a node (*Cs*), we consider another parameter that reflects the CPU load (or capacity), denoted by *C*.

At the physical infrastructure, two types of nodes are considered: DC nodes (denoted by *DC*) and transport network nodes (part of the Inter-DC Network, denoted by *Rt*). In this case, server nodes and virtual-enabled network nodes are considered to be part of DC nodes; however, the resource allocation within DC nodes is transparent in this perspective, since it is considered to be the responsibility of the DC domain management system (using single domain approaches like [6] and [7]).

DC nodes are characterized by their total amount of CPU, memory and storage. This amount does not need to reflect the actual total amount of resources of a DC, and can be just a subset of those resources. A good example is the case of a DC from a third party, where the total amount of available resources is a reflection of a business agreement between the owner of the DC and the VI provider. Moreover, in this case the geographical factor is very important; therefore, a DC is also characterized by its location, denoted by *Loc*.

From a VI perspective, network or server nodes can only be accommodated in DC nodes. The transport network is used for hosting virtual links that connect virtual nodes in different DCs. Table F-1 summarizes the notation used.

Table F-1: VI Assignement Problem Notation – Multi-Domain

Symbol	Description
G^P	Physical Infrastructure
N^P	Set of Physical Nodes
DC^P	Set of DC Nodes
Rt^P	Set of transport Network Nodes (Inter-DC network)
L^P	Set of Physical Links (Inter-DC network)
i, j	Physical Nodes
ij	Physical Link
$C^P{}_i$	Total CPU of DC Node *i*
$M^P{}_i$	Total Memory of DC Node *i*
$STG^P{}_i$	Total Storage of DC Node *i*
$B^P{}_{ij}$	Total Bandwidth of Physical Link *ij*
$D^P{}_{ij}$	Delay of Physical Link *ij*
$Loc^P{}_i$	Location of physical Node *i*
$C^{Ptotal}{}_i$	Total CPU of DC Node *i*
$M^{Ptotal}{}_i$	Total Memory of DC Node *i*
$STG^{Ptotal}{}_i$	Total Storage of DC Node *i*
$B^{Ptotal}{}_{ij}$	Total Bandwidth of Physical Link *ij*
$C^{Pfree}{}_i$	Free CPU of DC Node *i*
$M^{Pfree}{}_i$	Free Memory of DC Node *i*
$STG^{Pfree}{}_i$	Free Storage of DC Node *i*
$B^{Pfree}{}_{ij}$	Free Bandwidth of Physical Link *ij*
$DC^{max}{}_{load}$	Load of the DC with maximum load
$L^{max}{}_{load}$	Load of the physical link with maximum load
B_{cons}	Bandwidth being consumed in the physical links
$Loc^V{}_m$	Location of Virtual Node *m*
$G^V{}_v$	VI request *v*
$N^V{}_v$	Set of Virtual Nodes of VI request *v*
$S^V{}_v$	Set of virtual Server Nodes of VI request *v*
$R^V{}_v$	Set of virtual Network Nodes of VI request *v*
$L^V{}_v$	Set of Virtual Links of VI request *v*
m, n	Virtual Nodes
mn	Virtual Links
Cs	Number of CPUs of Virtual Node *m* of VI request *v*
$C^V{}_{v,m}$	CPU of Virtual Node *m* of VI request *v*
$M^V{}_{v,m}$	Memory of Virtual Node *m* of VI request *v*
$STG^V{}_{v,m}$	Storage of Virtual Node *m* of VI request *v*
$F^V{}_{v,m}$	CPU Frequency of Virtual Node *m* of VI request *v*
$B^V{}_{v,mn}$	Bandwidth of Virtual Link *mn* of VI request *v*
$D^V{}_{v,mn}$	Maximum Delay of Virtual Link *mn* of VI request *v*

F.3.2 Virtual Embedding Problem

The VI embedding problem is known to be NP-hard. A VI can be mapped according to one or multiple objectives, e.g.: occupy the lowest bandwidth possible, balancing the occupation of the physical nodes, or considering both previous objectives. Combining both objectives is not simple as they are not standalone, i.e. the choice of an ideal physical node may not allow the choice of the ideal physical link. This is considered to be a multiple-objective optimization problem (MOP) [12], where two or more conflicting objectives subject to constraints need to be simultaneously optimized.

In the following sections we will present the mathematical formulation for the optimal network and cloud embedding in multi-domain scenarios (sections F.4 and F.5).

F.4 Mathematical Formulation

In this section the mathematical formulation to solve the VI embedding problem for multi-domain scenarios is presented.

F.4.1 Assignment Variables

The formulation considers two binary assignment variables, *x* and *y*, one for the virtual nodes and another for virtual links as equations (1) and (2) show.

$$x_i^{v,m} = \begin{cases} 1, \text{virtual node } m \text{ of VI } v \text{ is allocated at Data Center } i \\ 0, \text{else} \end{cases} \quad (1)$$

$$y_{ij}^{v,mn} = \begin{cases} 1, \text{virtual link } mn \text{ of VI } v \text{ is uses physical link } ij \\ 0, \text{else} \end{cases} \quad (2)$$

F.4.2 Generic Constraints

As mentioned earlier in section F.3.1, virtual server and network nodes are assigned to DC nodes. Equation (3) reflects this constraint plus the fact that a virtual node is assigned just to one DC node.

$$\forall v,m \in DC : \sum_{i \in S} x_i^{v,m} = 1 \quad (3)$$

Further, equations (4), (5) and (6) guarantee that the capacity values of DC nodes, i.e. CPU load, memory, and storage, are kept within range. Equation (4) sums the CPU load of all virtual nodes hosted in each DC, and guarantees that it is kept below or equal to the total amount of CPU available in the corresponding DC. Equations (5) and (6) do the same for memory and storage resources.

$$\forall i : \sum_{m,v} x_i^{v,m} \times C_{v,m}^V \le C_i^{P_{total}} \quad (4)$$

$$\forall i : \sum_{m,v} x_i^{v,m} \times M_{v,m}^V \le M_i^{P_{total}} \quad (5)$$

$$\forall i : \sum_{m,v} x_i^{v,m} \times STG_{v,m}^V \le STG_i^{P_{total}} \quad (6)$$

Equation (7) applies the multi-commodity flow constraint [13] with a node-link formulation [14], in order to simultaneously optimize the mapping of virtual links and virtual nodes. Moreover, the notion of direct flows on the virtual links is also used.

$$\forall v, \forall m,n \in N_v^V(m), m < n, \forall i : \sum_{j \in N^P(i)} \left(y_{ij}^{v,mn} - y_{ji}^{v,mn} \right) = x_i^{v,m} - x_i^{v,n} \quad (7)$$

Equation (8) guarantees that each selected physical link (of the inter-DC network) has enough bandwidth available to host a virtual link. In other words, the sum of the amount of bandwidth of the virtual links that go through *i* must be equal or lower than the amount of the free bandwidth in a physical link. Finally, equation (9) guarantees that the virtual link delay constraint is met, i.e. the maximum delay of the physical links assigned to a virtual link does not exceed the maximum delay of the virtual link. Virtual links hosted within a single DC are not subject to these constraints, since it is considered that the DC is able to internally fulfill the requirements.

$$\forall i,j \in N^P(i), i < j : \sum_{v,m,n \in N^V(m), m<n} B_{v,mn}^V \times \left(y_{ij}^{v,mn} + y_{ji}^{v,mn} \right) \le B_{ij}^{P_{free}} \quad (8)$$

$$\forall v, \forall m,n \in N_v^V(m), m < n : \sum_{\forall i,j \in N^P(i), i<j} D_{ij}^P \times \left(y_{ij}^{v,mn} + y_{ji}^{v,mn} \right) \le D_{v,mn}^V \quad (9)$$

Finally, the virtual node location factor (refered in section F.3) is reflected in equation (10). It ensures that a virtual node can only be located in a DC that respects its location constraint.

$$\forall v,m \in DC : \sum_{i \in S} \left\lfloor x_i^{v,m} \times Loc_i^V \right\rfloor = Loc_m^V \quad (10)$$

F.4.3 Formulations

This section presents the proposed formulations. Three formulations are considered: maximum DC node, maximum link load, and bandwidth consumption.

Equation (11) shows the formulation for the DC maximum load, which looks for the DC node more loaded among all DCs. This value is given by the sum of CPU, memory and storage load fractions. Equal weights are considered for each resource type - δ_1, δ_2 and δ_3 - as they are equally important. Nevertheless, depending on the real scenario, adjustments can be made to these weights, e.g. if VI requests are more processing intensive, a higher weight can be given to δ_1.

$$\forall i \in DC: \left[\begin{array}{l} \delta_1 \dfrac{\sum_{v,m} x_i^{v,m} \times C_{m,i}^V}{C_i^{P_{total}}} + \delta_2 \dfrac{\sum_{v,m} x_i^{v,m} \times M_{m,i}^V}{M_i^{P_{total}}} + \\ \delta_3 \dfrac{\sum_{v,m} x_i^{v,m} \times STG_{m,i}^V}{STG_i^{P_{total}}} \end{array} \right] \leq DC_{load}^{\max} \quad (11)$$

The formulations for maximum link load and bandwidth consumption concern the resources in the inter-DC network domain. Equation (12) defines the maximum load consumption of a physical link among all links ($L_{load}^{\max}$). $B_{v,mn}^V$ refers to the bandwidth of the virtual link *mn* (i.e. the virtual link connecting the virtual node *m* and *n*) of a certain VI (*V*). $B_{ij}^{P_{Total}}$ refers to the total bandwidth capacity of the physical link *ij* (i.e. the physical link between physical node *i* and *j*). The variable $y_{ij}^{v,mn}$ assumes the value 1 if virtual link *mn* is crossing physical link *ij*, and 0 otherwise. In this way, the formula sums the bandwidth of all virtual links ($B_{v,mn}^V$) that cross a certain physical link (*ij*), and divides that value by the total capacity of the physical link ($B_{ij}^{P_{Total}}$). In the end, $L_{load}^{\max}$ assumes the load value of the most loaded physical link.

$$\forall i,j \in N^P(i), i < j: \left[\frac{\sum_{v,m,n \in N^V(m), m<n} B_{v,mn}^V \times \left(y_{ij}^{v,mn} + y_{ji}^{v,mn} \right)}{B_{ij}^{P_{Total}}} \right] \leq L_{load}^{\max} \quad (12)$$

Equation (13) defines the percentage of the total bandwidth consumed by a VI in the physical infrastructure. The amount of bandwidth consumed by each virtual link ($B_{v,mn}^V$) in all physical links ($y_{ij}^{v,mn}$) is summed and divided by the total bandwidth capacity of the entire physical infrastructure.

$$\frac{\sum_{v \in V, m, n \in N^V(m), m<n} y_{ij}^{v,mn} \times B_{v,mn}^V}{\sum_{i,j \in N^P(i), i<j} B_{ij}^{P_{total}}} = B_{cons} \quad (13)$$

F.5 Embedding Strategies

Taking into account our findings in the definition of embedded strategies for single domain scenarios, a load balancing strategy is considered the most adequate for multi-domain scenarios. The objective of this strategy is to balance the load among the different DCs, and at the same time reduce the impact in the inter-DC network, taking into account the location of each domain and the VI location restrictions.

Equation (14) presents the strategy's objective function. The weights v_1 and v_2 are considered equal and its sum is 1. Note that the maximum link load ($L_{load}^{\max}$) and bandwidth consumption (B_{cons}) are under the

effect of the same weight, v_2, and the maximum DC load ($DC_{load}^{\max}$) is under the effect of v_1. This evens the importance given to the DC load and inter-DC network occupation.

The approach that considers the inter-DC network load as a joint analysis of the total bandwidth consumption and the maximum link consumption has proved to be the best approach so far – see [6].

$$min\ v_1\left(DC_{load}^{\max}\right)+ v_2\left(L_{load}^{\max} + B_{cons}\right) \quad (14)$$

In contrast with the single domain case, embedding strategies that take into account the energy consumption are not considered in the multi-domain approach. These strategies are considered to be adopted inside each DC domain and are transparent to the multi-domain perspective. On the inter-DC network side, since it is a transport network, we assume that the network nodes are always active.

F.6 Evaluation

This section provides a thorough evaluation of the optimal embedding solution presented in this work in a multi-domain scenario.

F.6.1 Scenario description

To evaluate the performance of the proposed approaches, a Matlab [15] simulator is used. For each run, the program designs a random physical infrastructure of 20 nodes based on the Waxman network topology generator [16], and it simulates a set of requests of VIs (with a number of nodes between 4 and 14) with Markov-modulated inter-arrival and inter-departure times. The program builds a random physical infrastructure of 20 nodes, where 40% of the nodes are DCs and 60% are transport network nodes of the inter-DC network. The choice of these values is due to the fact that: the envisioned scenario considers a wide geographical area with several DCs (in this case with 8 DCs); the transport network does not (have to) match an entire operator network, but only a subset of it, exclusively dedicated for interconnecting the DCs (in this case with 12 nodes). The reason for these values also lays on the fact that these dimensions allow for the physical infrastructure to reach a saturation point, which will allow a better analysis of the solution.

Details on the nodes and link parameters can be seen in Table F-2. Moreover, the VI request rates (λ) vary between 2 and 5 VIs per time unit (Poisson arrivals), and the average duration of the VIs ($1/\mu$, where μ is the average service rate) is 20 time units (exponentially distributed duration).

In order to solve the ILP, we have used CPLEX [17] version 12.3, integrating a plug-in for our Matlab simulator and setting a time limit of 600 seconds (i.e. 10 minutes) for each VI mapping.

Table F-2: Physical and virtual infrastructure parameters

		Physical Infrastructure	Virtual Infrastructures
Net Nodes	**Cs**	-	{1; 2; 3; 4 }
	Memory	-	{64; 128; 256; 512}(MB)
DC Nodes	**Cs**	{32; 64}	{1; 2; 4; 8; 16; 32; 64}
	STG (GB)	{6400; 12800; 25600}	{100; 200; 400; 800; 1600}
	M (GB)	{256; 512; 1024}	{2; 4; 8; 16; 32; 64}
Links	**B (Mbps)**	{500-2000 / 500 steps}	{10-100 / 10 steps}
	D (ms)	{5-10 / 1 steps }	{0-40 / 5 steps}

The weight values used for the optimal strategy,v_1 and v_2, are 0.5 (50%); as it was highlighted in the strategy definition, these values are used to even the maximum load on DCs and inter-DC network.

The analysis is focused on the acceptance ratio, since the energy and re-optimization strategies are considered to be applied within each DC. In this case we analyze the impact of location as a constraint in the allocation of virtual resources. We consider that each virtual node resource has a location restriction with an associated range: the higher the range, more DC locations are likely to satisfy the virtual resource location restriction. In this work we present the values for two different location ranges: 25% and 50%. The range percentage is calculated based on the geographical size of the physical infrastructure, for example: for a physical infrastructure spanning 600 square kilometers, a range of 25% corresponds to 150 square

kilometers, and a range of 50% corresponds to 300 square kilometers. Moreover, this variation is also analyzed with respect to its impact on the physical infrastructure.

All values present a mean of 10 runs (with different substrate) with a confidence interval of 95%.

F.6.2 Virtual Infrastructure Acceptance Ratio analysis

Figure F-2 presents the VI acceptance ratio of the multi-domain embedding strategy for the two different location ranges (25% and 50%). It is clear that the impact of location is significantly high. For a VI arrival rate of 2, the acceptance ratio for the range of 50% drops approximately to half (18%) of the acceptance for the range of 25% (35%).

This result is due to the fact that, as the location range is reduced, the number of DCs able to embed virtual nodes is also reduced. This automatically reduces the probability of finding an embedding solution for the VIs.

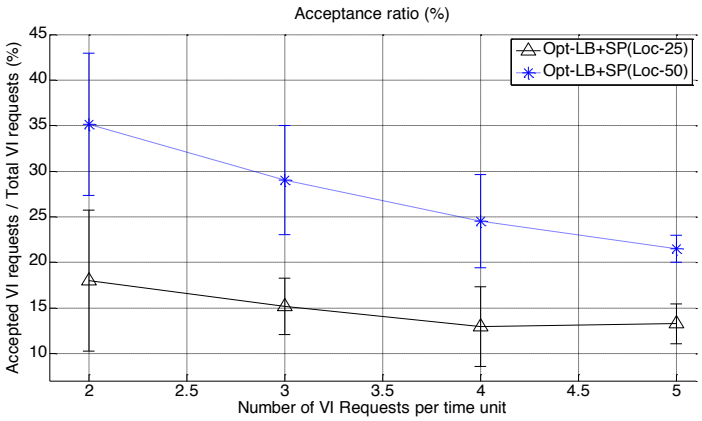


Figure F-2: VI acceptance ratio vs arrival rate and location restriction

F.6.3 Physical Infrastructure Impact analysis

The impact of the number of VI requests and the location restriction on the physical infrastructure are analyzed in terms of inter-DC network and DC occupation. Figure F-3 shows the average number of embedded virtual nodes. The embedding strategy with a location restriction of 50% has, in average, the double of virtual nodes embedded when compared with the strategy with a location restriction of 25%. This happens because the range of available virtual node embedding solutions in the former case is significantly higher than in the latter one. Moreover, the number of embedded virtual nodes in both cases increases linearly as the arrival rate increases. Although the acceptance ratio decreases while increasing the number of VI requests, the number of virtual nodes increases because the total number of VIs embedded still increases (even with a decrease in the acceptance ratio).

Figure F-4 shows the average bandwidth occupation of the Inter-DC network. This value is obtained by dividing the total amount of occupied bandwidth for the total amount of virtual bandwidth effectively requested. In this case, compared to the virtual node one, the ranking is inverted: the strategy with a location restriction range of 25% presents higher occupation than the one with a restriction range of 50%. This happens because, with a lower location range, the number of embedding solutions decreases, which prevents the finding of a better collocation of virtual nodes of a same VI in order to reduce the impact on the inter-DC network. In summary, the results presented in both cases are a reflection of the applied strategies.

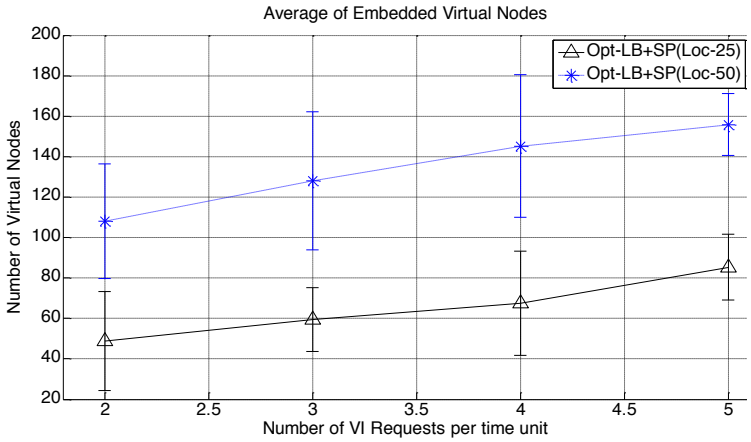


Figure F-3: Average number of Embedded Virtual Nodes vs arrival rate and location restriction

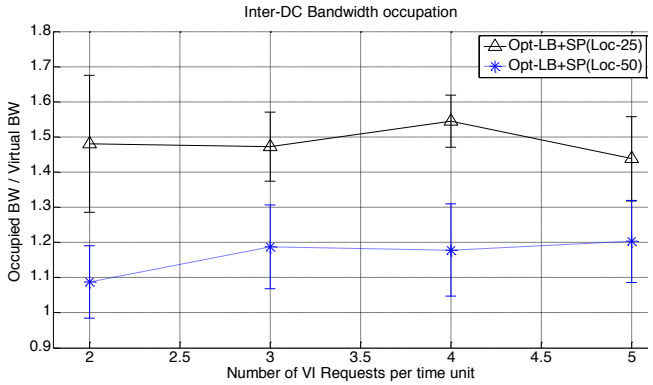


Figure F-4: Average Inter-DC network bandwidth occupation vs arrival rate and location restriction

F.7 Conclusion and Future Work

This paper proposed a VI embedding approach for multi-domain scenarios. An optimal formulation and an embedding strategy based on ILP were presented. The proposed strategy takes into account the load balancing of DC and inter-DC network domains. It is shown that location requirements are an important factor that brings a high impact on the VI acceptance ratio and on the occupation of the physical infrastructure. By reducing the location range, the VI acceptance ratio reduces, since less DCs become eligible for accommodating virtual nodes. The same happens with the DC load: with less VIs accepted, the DC's load is lower. However, in terms of inter-DC network, by reducing the location range, the inter-DC network load increases. This happens because the ability to co-locate virtual nodes in the same DC is lower, leading to more disperse nodes across DCs and a larger virtual link accommodation in the inter-DC network.

The formulation here provided is a solid base foundation that can be easily extended to accommodate further constraints, depending on the specific scenario in which it can be applied.

References

[1] ETSI, Network Functions Virtualisation (NFV): Architectural Framework, Technical Report ETSI GS NFV 002 v1.1.1, Oct. 2013.

[2] ETSI, Network Functions Virtualisation (NFV): Use-cases, Technical Report ETSI GS NFV 001 v1.1.1, Oct. 2013.

[3] N. Yong; X.C. Liang; Z. Kai, Connectivity as a Service: Outsourcing Enterprise Connectivity over Cloud Computing Environment, Computer and Management (CAMAN), 2011 International Conference on , vol., no., pp.1,7, 19-21 May 2011.
[4] W.H. Liao; S.C. Su, A Dynamic VPN Architecture for Private Cloud Computing, Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on , vol., no., pp.409,414, 5-8 Dec. 2011.
[5] A. Mahimkar, A. Chiu, R. Doverspike, M.D. Feuer, P. Magill, E. Mavrogiorgis, J. Pastor, S.L. Woodward, and J. Yates. 2011. Bandwidth on demand for inter-DC communication. In Proceedings of the 10th ACM Workshop on Hot Topics in Networks (HotNets-X). ACM, New York, NY, USA, Article 24.
[6] J. Soares, J. Aparicio, S. Sargento, Optimizing the Embedding of Virtualized Cloud Network Infrastructures, submitted to IEEE Transactions on Cloud Computing, June 2014.
[7] J. Soares, S. Sargento, Re-Optimizing the Embedding of Virtualized Cloud Network Infrastructures, submitted to Conference/Magazine name, Month 2014.
[8] I. Houidi, W. Louati, W.B. Ameur, D. Zeghlache, Virtual network provisioning across multiple substrate networks, Computer Networks, Volume 55, Issue 4, 10 March 2011, Pages 1011-1023, ISSN 1389-1286, http://dx.doi.org/10.1016/j.comnet.2010.12.011.
[9] B. Kantarci, H. Mouftah, Minimizing the provisioning delay in the cloud network: Benefits, overheads and challenges. Computers and Communications (ISCC), 2012 IEEE Symposium on, (pp. 806-811), 1-4 July 2012.
[10] B. Kantarci, H. Mouftah, Optimal Reconfiguration of the Cloud Network for Maximum Energy Savings. Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on, (pp. 835-840), 13-16 May 2012.
[11] B. Kantarci, H. Mouftah, Overcoming the energy versus delay trade-off in cloud network reconfiguration, Computers and Communications (ISCC), 2012 IEEE Symposium on , vol., no., pp.000053,000058, 1-4 July 2012.
[12] Hoboken, Metaheuristics: from design to implementation . John Wiley & Sons, 2009.
[13] S. Even, A. Itai, A. Shamir, On the complexity of time table and multi-commodity flow problems, Foundations of Computer Science, 16th Annual Symposium on, (pp. 184-193), 1975.
[14] M. Pioro, D. Medhi. Routing, Flow, and Capacity Design in Communication and Computer Networks. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc, 2004.
[15] MATLAB Release 2010, The MathWorks, Inc., Natick, Massachusetts, United States.
[16] MATLAB, Waxman Network Topology Generator, http://www.mathworks.com/matlabcentral/fileexchange/2517-waxman-network-topology-generator/content/waxtop.m
[17] IBM. IBM ILOG Optimization Products. www-01.ibm.com/software/websphere/products/optimization. Retrieved 13 May 2013.

Paper G. SDN Framework for Connectivity Services

Rafael Gouveia, João Aparício, João Soares, Bruno Parreira, Susana Sargento, Jorge Carapinha

in IEEE International Conference on Communications, 2014

The format has been revised

SDN Framework for Connectivity Services

Rafael Gouveia, João Aparício, João Soares, Bruno Parreira, Susana Sargento, Jorge Carapinha

Abstract

The momentum around Software-defined Networking (SDN) is increasing. It has become clear that the network architecture needs to evolve to be able to provide current and next generation networking services, both from a performance and implementation point of view. However, most of the current SDN research has looked to specificities of SDN and few have looked from a full stack perspective. Although the former are essential, also the latter perspective needs to be taken into account. In this sense, our work proposes a complete and modular SDN framework targeted at connectivity services. This framework allows the creation and management of network connectivity services over an OpenFlow based network, with mechanisms of fault-management, as well as the optimal usage of the infrastructure. The performance results obtained from the SDN framework evaluation in a real environment show that the three different scenarios, service activation, link loss, and reaction to a new link, are dynamically supported with fast reaction to the network changes.

Keywords - Software-defined networking; OpenFlow; cloud computing; connectivity services.

G.1 Introduction

The explosion of connected devices and applications in the last years revealed a series of limitations in the development and management of new services in the network; on the end-user side, due to the fact that a large part of these devices is mobile and there are well inherent challenges to keep these devices permanently connected; on the other side, due to the dynamics that new paradigms such as cloud computing are bringing to the network. In the latter case there is the clear need of a more scalable and flexible network architecture, starting within data centre (DC) domains [1]. Cloud environments have continuous allocation and re-optimization processes of virtual resources in the network, which requires constant configurations in the network. However, the impact is not limited to DC networks as the transport network also plays a fundamental role, from connecting DCs to the actual delivery of these services to the end-user. Thereby, the complexity of network management increases exponentially both inside and outside DCs, i.e. on the operator network.

Taking into account the abovementioned factors, it is not feasible to have a static network (as today) that does not have dynamic capacity. The network architecture needs to evolve to be able to provide current and next generation networking services, both from a performance and implementation point of view. In other words, it needs to be more service oriented.

Software-Defined Networking (SDN) [2] is an emergent paradigm that proposes a change in the current network architecture, based on the separation of the data plane and control plane. From a logical point of view, this separation allows the placement of the network intelligence into a higher layer, reducing the current complexity of network elements and making the network more programmable. Therefore, network elements just need to ensure connectivity services as the intelligence is now at an upper layer [3][4]. The operation method is based on the exchange of messages between the control plane and network elements (data plane), instead of statically programming those elements. This allows the network to become more dynamic and for resources to be more efficiently used, for example in terms of Quality of Service (QoS) [3]. The most widely accepted protocol for the communication between these two planes is OpenFlow [4].

SDN, with OpenFlow, offers other advantages to the operator, among which: the ability, from a control perspective, to be agnostic to the hardware, which reduces vendor lock-in; the displacement of network intelligence that allows for a higher layer of abstraction from the lower layer, i.e., an abstraction to the individual network elements. A logically centralized management entity also enables faster deployments of new services.

Although the application of SDN has been mostly linked to DC networks, there are network operators already using it beyond DCs, such as NTT [5]. Apart from the known advantages of SDN, currently there is no framework available that contains a complete and modular SDN architecture. Ideally, with a complete framework the network should be seen from a service perspective, leaving the work of provisioning and managing the network connectivity to the framework itself. The latter should have monitoring and management mechanisms that guarantee the normal operation of the network and non-optimal data paths.

In this paper we present a service-oriented SDN framework that enables the automatic provisioning of connectivity services over an OpenFlow network. Moreover, we undertake the management and maintenance of the network, by presenting a set of self-management mechanisms to ensure the optimal functioning of the network. This framework allows studying the potential of SDN as a solution for the future Internet. A thorough evaluation of the framework is performed in three different scenarios, service activation, link loss, and reaction to a new link, presenting encouraging results towards a future deployment.

The remainder of this paper is organized as follows. Section G.2 briefly presents related work, and Section G.3 presents the architecture of the SDN framework with a thorough description of its internal modules, mechanisms and functionalities. Section G.4 presents the implementation aspects of those functionalities, while Section G.5 describes the developed testbed and the experimental results. Finally, Section G.6 presents conclusions and points out future work.

G.2 Background and Related Work

SDN has been widely explored both on the industrial side as well as on the academic side.

On the industrial side the Open Networking Foundation (ONF) [6] is a well-known initiative currently focused on the analysis of SDN requirements and the evolvement of the OpenFlow standard. Another initiative is the OpenDayLight project [7], which intends to create a common and open SDN platform enabling network control and programmability. Its ultimate goal is to provide a complete SDN framework based on OpenFlow. Although far from being completed, this initiative is most probably the one closest to the focus of our work. Moreover, there is the European project Ofelia [8], which aims at creating an unique experimental infrastructure that allows researchers to, not only experiment on a test network, but also to control and extend it dynamically. It consists of autonomous OpenFlow enabled islands where each will serve as a nucleus for an OpenFlow enabled campus.

In the academia there are several works that tackle SDN and OpenFlow related topics. Kempf [9] proposes an efficient method to locate the link failures in the network. Through the implementation of a monitoring function in the network switches, the scalability limitation is overcome, which allows a fast recovery of the system. Yu [10] proposes a framework that uses OpenFlow to handle the transient link failure. This still uses the legacy routing protocols, shifting for the OpenFlow to tackle the problem until the system is recovered. These works focuses on resolving the fault management: detection and recovery. However, the processes are neither integrated in a single framework nor only based in a fully OpenFlow network.

Sharma [11] proposes a network management and control system framework that allows operators to run their networks in a hybrid mode, i.e. composed by the legacy network protocols and a controller with OpenFlow. With this approach, it automates and simplifies network management while increasing the dynamic control of individual flows. Although an improvement, this architecture is still hindered by legacy management tools and protocols, limiting the potential of a complete SDN solution. Furthermore, Egilmez [12] proposes a different approach of QoS architectures for rerouting capability, using non-shortest paths for lossless and lossy QoS flows. In a latter work [13], the author proposes a framework for dynamic rerouting of QoS flows to stream scalable coded videos. This work aims to resolve the optimization problem of the flows concerning the QoS requirements. Although it is similar to the framework here presented, it is limited to video streaming services and with limited expandability due to its single module architecture. Moreover, Sonkoly [14] proposes an integrated OpenFlow framework capable of running with different OpenFlow versions simultaneously and specifying QoS parameters.

Most of the current works tackle specific SDN subjects and only a very small set looks into SDN from a full solution perspective. Although the former are essential, also the latter perspective needs to be taken into account. In this sense, our work focuses on a complete and modular framework targeted at connectivity services.

G.3 SDN Framework Architecture

In this section we present the proposed framework. The framework allows the creation and management of network connectivity services over an OpenFlow based network. Moreover, it assures the integrity of these services (fault-management), as well as the optimal usage of the infrastructure (run-time management).

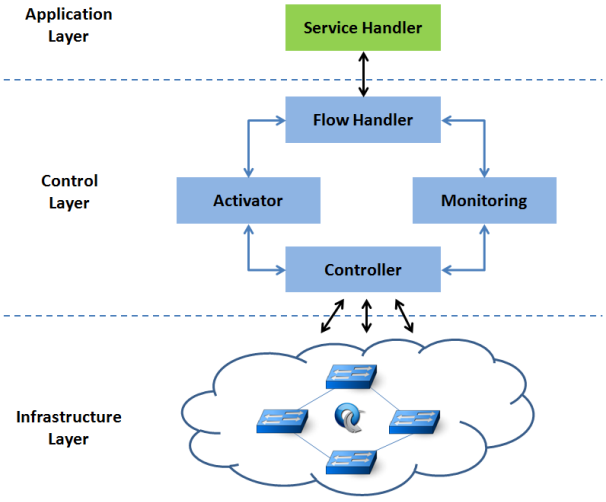


Figure G-1: SDN Framework Architecture

In the framework, a network connectivity service is defined by four parameters: communication type, service type, endpoints and rule type. The first defines the communication type, for example, Multicast or Full-Mesh communication. The second defines the type of network service, for example, TCP, UDP, ICMP and ARP communication. The third defines a set of endpoints that are part of the service, which are identified using MAC and/or IP addresses. Moreover, single ports (i.e. switch interfaces) are also an option. The last parameter, rule type, defines the type of rule that is applied in the network, i.e. if apart from the service type, also IP and/or MAC are taken into account. It is important to note that QoS parameters were not yet taken into account at the moment due to implementation restrictions of Openflow 1.0.

The framework's architecture is presented in Figure G-1. The control grid of the framework is composed by four modules: Flow Handler, Activator, Monitoring and Controller. It is responsible for the connection between the data plane, where the forwarding elements are, and the applications plane, the Service Handler. Moreover, it is also responsible for assuring the optimal functioning of the network. The individual functionalities of the different modules are described below.

• ***Service Handler***: provides a mean for users (i.e. applications) to make connectivity requests, and it can be seen has a frontend for the framework. It has three functionalities: *Service to Flow Translation*, *Service Repository* and *Flow Repository*. The *Service to Flow Translation* functionality translates connectivity requests into individual flows, which are then persistently stored using the respective *Service Repository* and *Flow Repository* functionalities.

• ***Flow Handler***: this module comprises the network intelligence by using algorithms to find data paths for individual flows. It has three functionalities*: Flow Allocation*, *Event/Fault Detection*, and *Flow Optimization*. The first refers to the ability to find data paths for flows. The second functionality refers to the evaluation of network changes that can eventually trigger optimization processes. The third consists on reacting to generated events (e.g. a link failure event may lead to flow reallocations). Moreover, the flow allocation algorithm used is the Dijkstra algorithm considering link bandwidth capacity - equation 1. The set of nodes in the physical infrastructure is denoted by N^P, N^{flow} denotes the two endpoints of a flow, bw^{ij}

denotes the bandwidth capacity of link $ij \in N^P$, and finally, ${Cost_{bw}}^{mn}$ refers to the cost to allocate flow *mn* in the network.

$$ {Cost_{bw}}^{mn} = \min\left(\frac{1}{\sum_{ij}^{N^P} {bw}^{ij}}\right) mn \in N^{flow} \qquad (1) $$

• ***Monitoring***: this element identifies monitors, saves and provides information regarding network elements. Two of the functionalities, *Real Time Network Discovery* and *Event Generation*, are subscription based. The first one uses information provided by the controller to retrieve real-time information regarding the network topology and individual links bandwidth. The *Event Generation* functionality relates to the *Network Statistics* functionality, which gathers network statistics to generate events regarding link and switch usage. These event messages are forwarded to subscribers. It should be noted that the module supports multi-tenancy.

• ***Activator***: this module uses two functionalities, *Flow Translation* and *Flow Enforcement* to achieve its goal: store rules in the network elements. The first one consists in the translation of flows sent by the *Flow Handler* into rules for individual network elements. The second one is accomplished by communicating to the controller the rules to be enforced in individual switches.

• ***Controller***: this element provides the necessary abstraction from individual network elements to upper layers. To fulfill this objective one functionality is used, *Network Device Mediation*, which establishes the communication between the framework and the OpenFlow network.

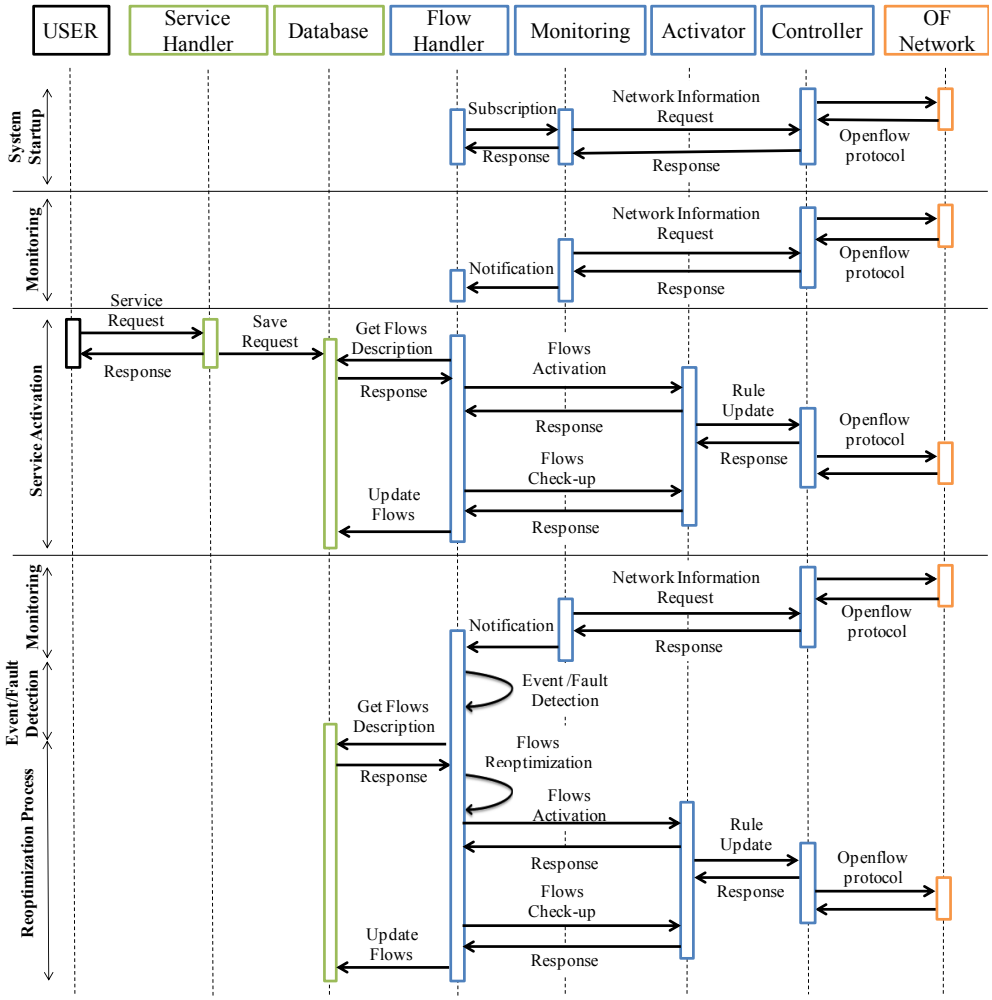


Figure G-2 – Modules' interaction (System Startup, Service Activation and Reoptimization Process)

Figure G-2 shows the interaction between the modules in its various functionalities: system startup; service activation; event detection; fault detection; and re-optimization process. Throughout the iteration between modules, there is a continuous communication between *Monitoring* and *Controller* and also

between the *Controller* and the *OpenFlow network*. When the system starts, the *Flow Handler* module subscribes to the services provided by the *Monitoring*. Simultaneously, the *Monitoring* collects network information from the *Controller* (network element list and capabilities), which will then be used to build the network topology and associated information to be transmitted to the *Flow Handler*. After this, the platform is available to receive and activate connectivity services.

On the activation of a service, the user communicates to the *Service Handler* which service he wants to activate on the network. This module validates the request (i.e. verifies if parameters are consistent with what the platform has to offer) and stores it in a database. The *Flow Handler* is in continuous communication with the database looking for new flows to activate. When this module finds new flows in the database, it retrieves its description to later associate them a path that will be mapped on the network. Finally, it transmits all necessary information to the *Activator* module to insert these flows in the network. The activation flows are updated on the table of flows of the network elements, via the *Controller*.

The reoptimization process starts when the *Flow Handler*, after receiving an event from the *Monitoring* module, detects a change in the network by comparing the new and the previous topology. In the case there are new links, the *Flow Handler* will search for possible optimization of active services. For this purpose, the module executes the flow allocation algorithm (Dijkstra) for every flow stored in the database in an attempt of finding shorter paths. In the case some links have been removed, the module will search for possible failures in active services - *Fault Detection* functionality previously mentioned. In this case, all data paths of active flows are analysed and, in the case the failure affects a flow, the flow allocation algorithm is applied in order to reallocate the flow.

G.4 Implementation

This section describes the internal characteristics and options made in the development of the different modules.

The *Controller* that is chosen to operate with the OpenFlow protocol is the *Floodlight*, version 0.9. This controller is open source and uses the RESTful web API principles [15]. These characteristics make the *Floodlight* suitable to be used in the framework. The rest of the modules were developed using the version 2.7 of the *Python* language. All modules use a *RESTful web API* and *HTTP* principles to make the exchange of data from and to the servers, using *JSON* to serialize data [16].

The *Monitoring* is in continuous communication with the *Controller* in order to gather all the relevant information of the network elements. This module produces adjacency matrixes with the information of the network topology and link bandwidth. An example is shown in Figure G-3, which relates to the testbed presented in Figure G-5. The first matrix shows the connection between switches, and the third one shows the link bandwidth. Moreover, it is given a list of switches identified through their MAC addresses. The modules that want to receive this information need to perform a subscription, and after that moment, the *Monitoring* knows the addresses of where to send information. Every time there is a change in the network topology, a new adjacency matrix is sent to the subscribers.

$$\begin{array}{c|cccc} \boldsymbol{SW} & \boldsymbol{A} & \boldsymbol{B} & \boldsymbol{C} & \boldsymbol{D} \\ \hline \boldsymbol{A} & 0 & 1 & 0 & 1 \\ \boldsymbol{B} & 1 & 0 & 1 & 0 \\ \boldsymbol{C} & 0 & 1 & 0 & 1 \\ \boldsymbol{D} & 1 & 0 & 1 & 0 \end{array} \qquad \begin{bmatrix} MAC & SW & A \\ MAC & SW & B \\ MAC & SW & C \\ MAC & SW & D \end{bmatrix} \qquad \begin{array}{c|cccc} \boldsymbol{SW} & \boldsymbol{A} & \boldsymbol{B} & \boldsymbol{C} & \boldsymbol{D} \\ \hline \boldsymbol{A} & 0 & 100 & 0 & 10 \\ \boldsymbol{B} & 100 & 0 & 100 & 0 \\ \boldsymbol{C} & 0 & 100 & 0 & 10 \\ \boldsymbol{D} & 10 & 0 & 10 & 0 \end{array}$$

Figure G-3 – Network View (Topology; Switch addresses, BW capacity)

The *Service Handler* allows two types of services: *Full-mesh* that creates bidirectional connections between the list of elements; and *Multicast* that creates unidirectional connections between an element labelled as source and the elements labelled as destinations.

The communication between this module and the *Flow Handler* is performed through a database. The *Flow Handler* uses a polling mechanism to detect if there are flows without a path associated, or if there are inactive flows. This module uses the *Dijsktra* algorithm, previously presented, to discover the shortest path in the network. An activation request sent from the *Flow Handler* to the *Activator* is shown in Figure G-4. It is composed by the flow topology matrix and a list describing the rule and the identification of the source

and destination endpoints. This example shows the mapping of a flow from the endpoint connected to the Node A to an endpoint connected to the Node C. In this case, although paths A->B->C and A->D->C have the same number of hops, path A->B->C has higher capacity (100 + 100 > 10 + 10), making it a better choice.

$$\begin{array}{c} \boldsymbol{SW} \\ \boldsymbol{A} \\ \boldsymbol{B} \\ \boldsymbol{C} \\ \boldsymbol{D} \end{array} \begin{array}{c} \begin{array}{cccc} \boldsymbol{A} & \boldsymbol{B} & \boldsymbol{C} & \boldsymbol{D} \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \begin{bmatrix} Source: \{MAC\ SW\ A, PORT\ SW\ A, IP, MAC\} \\ Destiny: \{MAC\ SW\ C, PORT\ SW\ C, IP, MAC\} \\ Rule: \{UDP, IP + MAC\} \end{bmatrix}$$

Figure G-4 – Flow Activation View (Node A -> Node C)

G.5 Evaluation

This section evaluates the SDN framework. First, it is described the testbed used for the evaluation, and then it is presented and analysed the experimental results are.

G.5.1 Testbed

The testbed built to evaluate the SDN framework is shown in Figure G-5. The entire framework is hosted within the same physical machine. Each network element corresponds to an individual physical machine – see Table G-1.

Table G-1: Testbed Specification

Node	CPU Model (Intel)	CPU Freq. (GHz)	CPU Cores	CPU Threads	HDD Memory (GB)	RAM Amount (GB)	RAM Freq (MHz)
A	PentiumD 950	3.40	2	4	40	6	667 DDR2
B	Core 2Duo E6400	2.13	2	2	145	4	533 DDR2
C	Core 2Duo 6400	2.13	2	2	145	4	533 DDR2
D	PentiumD 950	3.40	2	4	40	6	667 DDR2

The *Open VSwitch* [17] is used to emulate the switches in the nodes. The *Open VSwitch* is an open source application that emulates the behaviour of a real switch that supports the OpenFlow protocol.

G.5.2 Performance Results

This sub-section starts with the analysis of the individual performance of the modules developed in the scope of this framework. Moreover, the overall performance of the framework is analysed in three scenarios: 1) service activation, 2) link loss (event/fault-detection and reoptimization process), and 3) new link (event-detection and reoptimization process). These functionalities are explained in Sections G.3 and G.4 when describing the module's interaction. The evaluation starts with the activation of a connectivity service. Then, the link that connects node A to node D is removed and finally the same link is reconnected. All values in the graphics present a mean of 10 runs with a confidence interval of 95%.

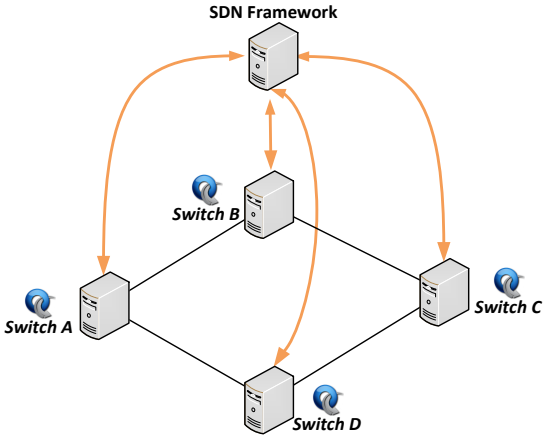


Figure G-5 – Network elements scheme

Regarding to the *Service Handler* and the *Flow Handler*, the performance is analysed for the two communication types (*Full-mesh* and *Multicast*). The aspects analysed are the service translation to independent flows and its storage in the database by the Service *Handler*, and the path association of those flows, which is a role of the *Flow Handler*.

Figure G-6 shows the overall time of the two modules for each mode by varying the number of elements in the request. The time consumption for these modules is directly proportional to the number of flows, which are the result of the service translation. Note that there will be a predictable difference between operation modes: as an example, in the 20 elements case the *Full-mesh* translates the service in 380 independent flows (bidirectional flows), meanwhile for the same number of elements the *Multicast* just translates the service to 19 independent flows (unidirectional).

Regarding the performance of the *Service Handler*, the *Full-mesh* communication type requires approximately 50 ms per flow and the *Multicast* needs approximately 95 ms. The *Flow Handler* is independent of the communication type chosen, and takes approximately 55 ms to perform the path assignment per flow. Moreover, due to the inherent nature of each service (unidirectional *vs* bidirectional), the *Multicast* case shows a linear behaviour while the *Full-Mesh* shows an exponential one. For simplicity purposes, we only analyse in more detail the 20 elements case. The values for the *Multicast* communication (19 flows) are approximately 1.85s for the *Service Handler*, 1.13s for the *Flow Handler*, which gives a total of approximately 2.98s. Regarding the *Full-mesh* communication, the values in this case highly increase due to the number of flows (380 flows): 18.8s for the *Service Handler*, 21.2s for the *Flow Handler*, giving a total of approximately 40s.

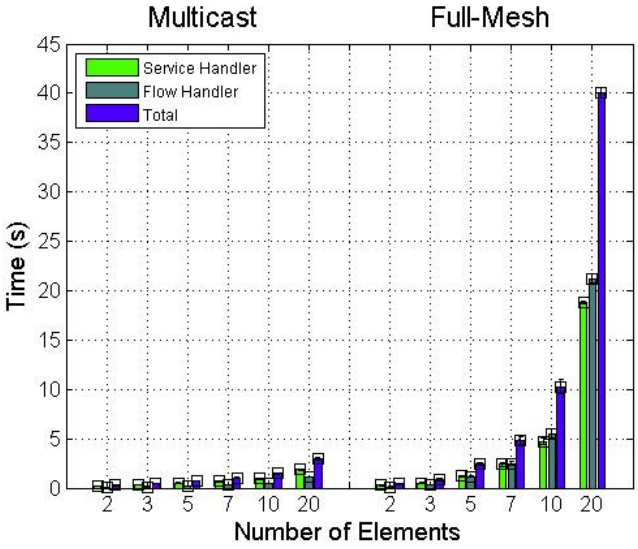


Figure G-6: Service Handler & Flow Handler performance

For the *Activator* module it is analysed the time needed to activate rules in a set of nodes (3, 5, 10 and 20 nodes), for the different connection types (UDP, TCP and ICMP). Figure G-7 shows the activation times for the different cases. The UDP and TCP type takes approximately the same time, 41 ms (3 nodes), 68 ms (5 nodes), 163 ms (10 nodes) and 331 ms (20 nodes), with both presenting lower times than the ICMP type: 62 ms (3 nodes) 100 ms (5 nodes), 224 ms (10 nodes) and 454 ms (20 nodes). This is justified because the ICMP type involves the configuration of two rules on the switches (bidirectional). In order to overcome the size limitation of the *testbed*, substrate topologies were manually injected within the *Monitoring* component to give the impression to upper layers that the substrate is larger than it actually is. This is done by introducing some switches more than once in the substrate elements list. In the end this leads the rules to be activated more than once in the same node of the *testbed*. Note that this does not affect the objectives of this evaluation.

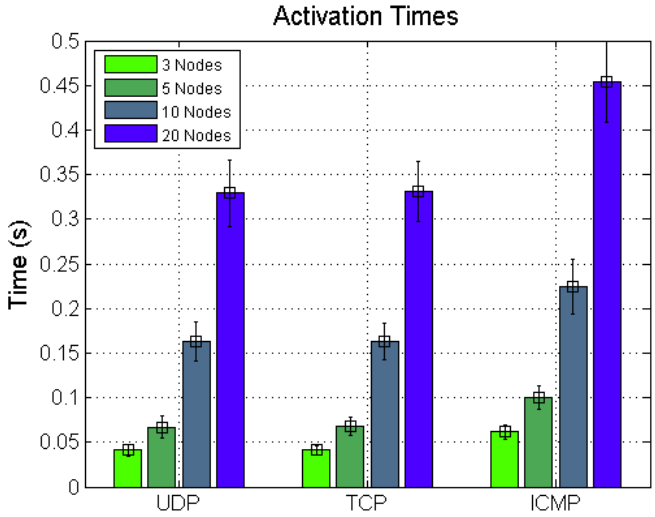


Figure G-7: Activator performance - time to activate the connectivity services

Figure G-8 shows the results of the first scenario. It is activated a connectivity service of the type UDP with the communication type *Multicast*, in which the host source will be always placed at node A. The targets are placed: 1 at node D (service with 2 elements - 1 flow), 2 at node B and 2 at node D (service with 5 elements - 4 flows); 3 at node A, 4 at B and 12 at D (service with 20 elements - 19 flows).

The *Service Handler* and the *Flow Handler* present the behaviour previously analysed. *Service Handler* takes 95ms per flow: 254ms (2 elements/1 flow); 498ms (5 elements/4 flows); and 1.86s (20 elemets/19 flows). *Flow Handler* takes 55ms per flow: 68ms (2 elements/1 flow); 217ms (5 elements/4 flows); and 1.08s (20 elemets/19 flows). Similar to the individual analyses of the *Service Handler* and *Flow Handler*, the activation time also presents a linear behaviour. It takes approximately 140ms to activate each flow. This explains the activation values of 138ms (2 elements/1 flow), 541ms (5 elements/4 flows) and 2.86s (20 elements/19 flows). The total time achieved in the first scenario for 1, 4 and 19 flows are approximately 460ms, 1.256s and 5.8s respectively.

It is important to note that the values presented are affected by the size of the substrate network. This is related with the policy used for finding flow paths (Dijkstra). The difference noticed between the actual results with the average times is justified because the static times as well as the validation times are low, but more influent in the experiments with a lower number of flows. The communications between modules slightly differ from experience to experience (average times are given in order of the flow).

The results of the second and third scenarios are presented in Figure G-9. For both cases, the initial step is the detection of a change in the substrate network by the *Monitoring*. There is a large time difference between the reaction to a link loss, approximately 100 ms, and the reaction to the addition of a link, approximately 3000 ms. This is due to the fact that, in the addition of a link, the switch interface needs to perform an internal reconfiguration, informing the *Open VSwitch*, and only after that, the controller is notified. On the contrary, the loss of a link does not need equipment reconfiguration and the controller is notified right away.

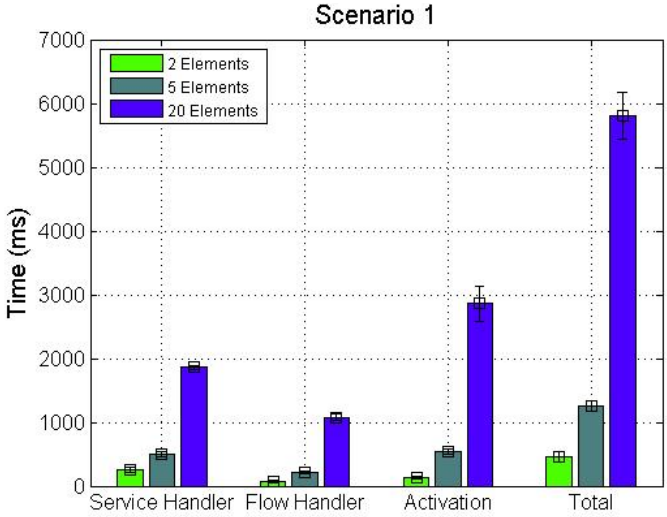


Figure G-8: Overall framework performance - Service activation

The following step is the event/fault-detection and reoptimization process. For the first case, link loss, the reoptimization time will be directly proportional to the flows affected, because it needs to find another path to allocate and activate the flows again. It takes 189ms (2 elements/1 flow/1 flow affected), 373ms (5 elements/4 flow/2 flows affected) and 2.394s (20 elements/19 flow/12 flows affected). For the second case, it is performed a path finding attempt for every flow. This process takes 174ms (2 elements/1 flow/1 flow affected), 402ms (5 elements/4 flow/2 flows affected) and 2.317s (20 elements/19 flow/ 12 flows affected). In our experiments, the time for both cases is almost the same, because the number of flows that get affected with the loss of a link is the same than the ones that get a shorter path with the addition of that same link.

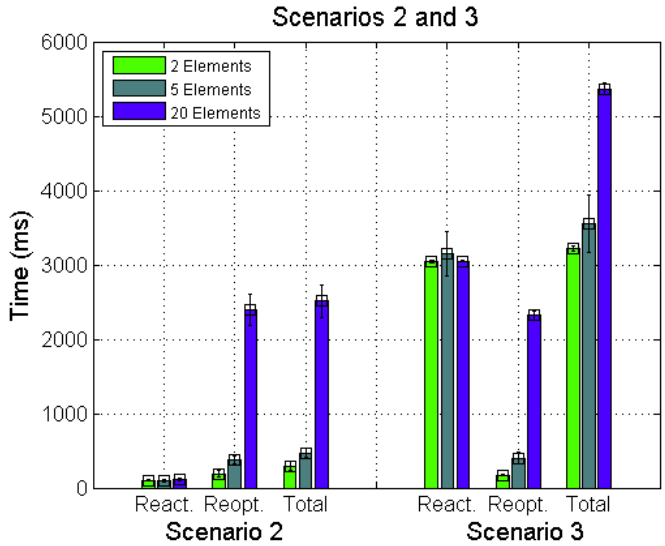


Figure G-9: Overall framework performance - Loss and addition of a link

In addition, we noticed in our experiments that the search for the flows affected when there is a loss of a link consume more time than a path finding try for the entire flows present in the database. However, when the number of nodes in the network is larger, it is expected that the reoptimization process due to the addition of elements will have longer time consumption than the cases when there is a loss of elements, because of the complexity of the Dijkstra algorithm.

G.6 Conclusion

The goal of this paper is to overcome the lack of a complete and modular SDN framework, from the data plane to the application plane, which offers connectivity services to the users in a simple, autonomous form.

This paper proposed a complete SDN framework that allows the creation and management of network connectivity services over an OpenFlow based network. Moreover, it assures the integrity of these services (fault-management), as well as the optimal usage of the infrastructure (run-time management). The performance of the SDN framework is evaluated in a real environment, which gives insight on the times that it takes to perform the configuration and management of connections upon a real OpenFlow network.

As future work, we plan to include other algorithms and policies for flow optimization, which will make the framework more adaptable and optimized to different types of services. Finally, we will extend the framework to include cloud resources integrated with the network.

References

[1] Open Networking Foundation: “Enabled Hybrid Cloud Services Connect Enterprise and Service Provider DCs”, white paper, November, 2012.

[2] Open Networking Foundation: “Software-Defined Networking: The New Norm for Networks”, white paper, April, 2012.

[3] M. Mendonca et al., “A Survey of Software-Defined Networking: Past, Present and Future of Programmable Networks”, HAL-Inria archive, May, 2013.

[4] W. Stallings: “Software-Defined Networks and OpenFlow”, in The Internet Protocol Journal, Cisco, vol. 16, no 1, pp. 2-14, March, 2013.

[5] H. Nagasono: “NTT DATA’s Efforts for OpenFlow/SDN”, in NTT Technical Review, vol 10, no 11, November, 2012.

[6] Open Networking Foundation: “OpenFlow Switch Specification”, August, 2013.

[7] OpenDaylight: “An Open Source Community and Meritocracy for Software-Defined Networking”, April, 2013.

[8] A. Köpsel and H. Woesner: “Test Facility for OpenFlow Experimentation”, In EICT GmbH, Berlin, November, 2011

[9] J. Kempf et al., "Scalable fault management for OpenFlow", IEEE International Conference on Communications (ICC), 10-15 June, 2012.

[10] Y. Yu, C. Shanzhi, L. Xin and W. Yan: "A framework of using OpenFlow to handle transient link failure", Int. Conf. Transportation, Mechanical, and Electrical Engineering (TMEE), 16-18 Dec., 2011.

[11] P. Sharma et al., “Enhancing Network Management Frameworks with SDN-like Control”, IFIP/IEEE Int. Symp. Integrated Network Management (IM 2013), 27-31 May, 2013.

[12] H. Egilmez et al., "Scalable video streaming over OpenFlow networks: An optimization framework for QoS routing", 18th IEEE Int. Conf. on Image Processing (ICIP), 11-14 September, 2011.

[13] H. Egilmez, S. Civanlar and A. Tekalp: "An Optimization Framework for QoS-Enabled Adaptive Video Streaming Over OpenFlow Networks", IEEE Trans. on Multimedia, vol.15 (3), pp.710-715, April, 2013.

[14] B. Sonkoly et al., "OpenFlow Virtualization Framework with Advanced Capabilities", European Work. Software Defined Networking (EWSDN), 25-26 October, 2012.

[15] R. T. Fielding and R. N. Taylor: “Principled design of the modern Web architecture”, ACM Trans. Internet Technol, vol 2, no 2, pp. 115-150, May, 2002.

[16] JavaScript Object Notation, http://www.json.org/

[17] Open vSwitch, http://openvswitch.org/

Paper H. Towards a Telco Cloud Environment for Service Functions

João Soares, Carlos Gonçalves, Bruno Parreira, Paulo Tavares Jorge Carapinha, João Barraca, Rui Aguiar, Susana Sargento

submitted to IEEE Communications Magazine, 2014

The format has been revised

Towards a Telco Cloud Environment for Service Functions

João Soares, Carlos Gonçalves, Bruno Parreira, Paulo Tavares Jorge Carapinha, João Barraca, Rui Aguiar, Susana Sargento

Abstract

Deploying Service Functions (SFs) is an essential action for a network provider. However, the action of creating, modifying and removing network SFs is traditionally very costly in time and effort, requiring the acquisition and placement of specialized hardware devices and its interconnection. Fortunately, the emergence of concepts like Cloud Computing, Software Defined Networking (SDN), and ultimately, Network Functions Virtualization (NFV) is expected to raise new possibilities about the management of SFs with a positive impact in terms of agility and cost. From a telecom operator (Telco) viewpoint these concepts can help to reduce both Operational Expenditure (OPEX) and open the door to new business opportunities. In this article, we identify how Telcos can benefit with the abovementioned paradigms, and explore some of the aspects that still need to be addressed in the NFV domain. We focus on two major aspects: enabling Telco infrastructures to adopt this new paradigm; and orchestrating and managing SFs towards Telco-ready cloud infrastructures. The technologies we describe enable a Telco to deploy and manage SFs in a distributed cloud infrastructure. In this sense, the platform Cloud4NFV is presented. Special attention is given to the way SFs are modeled towards cloud infrastructure resources. In addition, we explore the ability to perform Service Function Chaining (SFC) as one of the fundamental features in the composition of SFs. Finally, we describe a Proof of Concept (POC) that demonstrates how a Telco can benefit from the described technologies.

Keywords - Service Function, Service Function Chaining, Network Function Virtualization, Software-Defined Networking, Cloud Computing.

H.1 Introduction

The emergence of the cloud concept, its ongoing evolution and the opportunities that it brings, has led many businesses to adapt in order to get the most utility out of it. One can say that the Telco sector is today one of the most active business sectors exploring the opportunities offered by the cloud. The relationship and inter-dependency between clouds and telecommunications can be analyzed from two distinct perspectives:

- Telcos *supporting* the cloud: In a cloud environment, communication end points are user devices and Virtual Machines (VMs) that can be hosted in different physical locations, according to varying conditions. Compared to traditional networking environments, network capacity requirements are no longer static, but are likely to change as the associated computing and storage resources expand and reduce. This poses a whole new set of challenges to the network, now jointly including the Data Center (DC) and the Wide Area Network (WAN) segments. To provide assured levels of performance to cloud services, cloud and Telco services need to be provisioned, managed, controlled and monitored in an integrated way.

- Telcos *using* the cloud: Today, the establishment, management and composition of SFs (e.g. router, firewall) follow a rigid, static and time consuming process – e.g. resource overprovisioning is usually necessary to cope with estimated peak demand; a fault in a single function can disrupt an entire network, imposing the need for faster disaster recovery methods. As virtualization technologies reach maturity and are able to provide carrier-grade performance and reliability, it becomes feasible to consolidate multiple network equipment types, traditionally running on specialized hardware platforms, onto industry standard hardware, which minimizes costs, reduces time-to-market and facilitates open innovation. Cloud Computing, coupled with SDN [1] and NFV [2], promises to make SF management processes much more agile. Cloud Computing represents a paradigm for Information Technology (IT) services, which can now be delivered in an on-demand and self-service manner. SDN brings new capabilities in terms of network automation, programmability and agility that facilitate the integration with the cloud. On the other hand, NFV, from a high-level perspective, accelerates the innovation of networks and services, allowing new

operational approaches, novel services, faster service deployment (shorter time to market), increased service assurance and stronger security.

Conceptually, a SF is a functional block responsible for a specific treatment of received packets and has well-defined external interfaces [3]. A SF can be embedded in a virtual instance or directly in a physical element (the usual situation until recently). Virtual SFs offer the opportunity to compose and organize virtual SFs dynamically, opening a new set of business opportunities – and technical challenges. One of the topics that arise from the combination of SFs is service function chaining (SFC). SFC is loosely defined as "*an ordered set of service functions that must be applied to packets and/or frames selected as a result of classification*" [3]. It can be considered as a particular case of service composition. It requires the placement of SFs and the adaptation of traffic forwarding policies of the underlying network to steer packets through an ordered chain of service components. However, the lack of automatic configuration and customization capabilities increases the operational complexity.

In this article, we explore how telecom operators can take advantage of the referred concepts to improve the management of SFs and potentially build new business models. First, we highlight the Telcos privileged position in this area compared to traditional cloud providers. We then present Cloud4NFV, a platform for managing SFs in a Telco cloud environment. Later, we focus on SF modeling towards cloud infrastructure resources. Special attention is given to the ability to perform SFC. To emphasize possible application scenarios of the solution presented in this paper, a POC is then detailed. Finally, we point out future work directions and conclusions.

H.2 The Carrier Cloud Opportunity

Traditional cloud infrastructures are far from being suitable for all types of businesses, especially when referring to network SFs. Most network SFs have carrier grade requirements, from guaranteed Quality of Service (QoS) in terms of IT resources and network connectivity, to high-availability (e.g. perform detection and forecast of operational anomalies, support fault mitigation procedures such as VM migration and network re-planning) and fast fault recovery through redundancy.

Telcos, with their already established distributed network infrastructure and hosting centers, are ideally positioned to take the lead in this area, as they can easily create a compelling end-to-end cloud proposition that integrates their network management capabilities, adapted to a more agile and cloud service-oriented operation model (on-demand, self-service, elastic).

We envision a near-future Telco cloud infrastructure that comprises not only the traditional centralized DC domains, but also the WAN domain. In such scenario, the Telco can take advantage of its already established distributed facilities (sometimes referred as Points of Presence, PoPs) to host small cloud environments. It is also possible for this distributed cloud infrastructure to extend itself into the customer site. Figure H-1 depicts this scenario.

Table H-1: Summary of existing approaches

	Cloud4NFV	UNIFY [4]	T-NOVA [5]	CloNe [6]	StEERING [7]	CloudBand[1]
Distributed Infrastructure	Yes	Yes (conceptually)	Yes (conceptually)	Yes	No	Yes
End-to-End Service Management	Yes	Yes (conceptually)	Yes (conceptually)	Yes	No	Yes
Multi-domain Architecture	Yes	Yes (conceptually)	Yes (conceptually)	Yes	No	No
Network QoS support	Yes	Yes (conceptually)	Yes (conceptually)	Yes	No	Yes (partially)
SF Management	Yes	Yes (conceptually)	Yes (conceptually)	No	Yes (partially)	Yes
Traffic Steering support	Yes	Yes (conceptually)	No	No	Yes	No
SFC support	Yes	Yes (conceptually)	No	No	Yes (partially)	No

Although there are important contributions ongoing in this area, work is still required, namely when it comes to the definition of a true Telco cloud platform and to the details on how to model and actually realize SFCs. Table H-1 presents a summary of the features supported by some existing solutions that more closely relate to the scope of this work. The recent UNIFY [4] and T-NOVA [5] projects seem to share a similar vision; however, these projects have recently started and have only provided conceptual approaches to some extent (the features analysis was done based on documents publicly available). CloNe [6] has support for the infrastructure features; however, it lacks SFs management, traffic steering, and performing SFC. Moreover, StEERING [7] supports traffic steering and partially supports SFC and SF management ("partially" because the SFC service model and SF management features do not seem fully mature). Finally, we also consider the Alcatel Lucent CloudBand1 solution, which supports some of the envisioned features.

H.3 Cloud4NFV Platform

The Cloud4NFV platform builds upon Cloud, SDN and WAN technologies to allow SFs to be managed on an *as-a-Service* basis. The platform is targeted for Telcos to improve the management of SFs within their environment, but can also be used to build new services based on the concept of *Service Function as-a-Service* (SFaaS), in which case SFs or bundles containing a combination of SFs can be offered as a service to customers.

H.3.1 Functionalities

The most relevant functionalities of Cloud4NFV are:

- Automated deployment, configuration and lifecycle management (e.g. instantiation, configuration, update, scale up/down, termination, etc) of SFs.
- Exposure of functionalities such as: service deployment and provisioning; service monitoring and reconfiguration; and service teardown.
- Federated management and optimization of WAN and cloud resources for accommodating SFs.
- Support of SF composition through SFC.

All the above mentioned functionalities are essential in the scope of an NFV platform; however, we highlight the last two due to their novelty. These two functionalities are seen as key differentiation factors from other available solutions, taking this platform closer to being fully carrier-grade compliant. The federated management and optimization of WAN and cloud resources, gives the platform a broad and distributed scope. It allows the establishment of end-to-end services over a distributed physical infrastructure. The ability to perform SFC gives the platform an unprecedented flexibility with respect to the SF management and composition, allowing the definition and establishment of advanced services in a much more efficient and flexible way.

H.3.2 Architecture

Figure H-1 provides an overview of the system, organized in four major planes: Infrastructure Plane, Virtual Infrastructure Management (VIM) Plane, Orchestration Plane, and Service Plane. The Service Plane handles the services that are built on Cloud4NFV, and the Infrastructure Plane comprises all physical resources. Special attention should be given to the VIM and Orchestration Plane, since we consider them to be the major lever for enabling SFC. It is important to note that this architecture is aligned with the ETSI NFV architectural guidelines [8]. This fact is highlighted along the description of the platform.

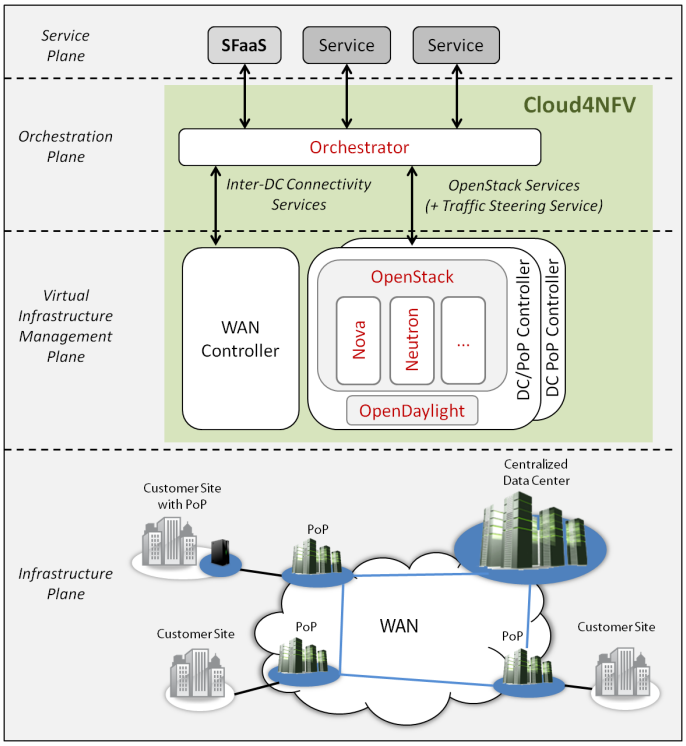


Figure H-1: Cloud4NFV platform – overview

Orchestrator

The Orchestrator is responsible for the automated provision, management and monitoring of SFs over the virtual infrastructure. It exposes the ability to create and delete SFs, as well as the ability to chain SFs. It relies on the VIM Plane to provision the infrastructure resources where SFs run (VMs, virtual networks, etc). Looking to the ETSI NFV reference architectural framework [8], this component considers the *Orchestrator* and the *VNF Manager(s)* entities. The orchestrator has an interface (REST) that exposes the ability to create and delete SFs as well as to chain SFs.

Virtual Infrastructure Management Plane

The VIM Plane includes the components for management of infrastructure resources. It includes cloud DC controllers (one per DC) and a WAN controller that is able to establish inter-DC connectivity services. The VIM Plane can be seen as the *Virtual Infrastructure Manager(s)* in the ETSI NFV reference architectural framework [8]. However, the current ETSI specification does not have into consideration the WAN component. This is considered by ETSI to be subject of future analysis.

DC Controller(s): Although the cloud model may require, to a large extent, the redefinition of SFs and the way they are managed, SFs also require adaptation from today's cloud solutions to cope with their requirements, especially in terms of networking features. A clear evidence of this fact is the OpenStack[10] project, a reference open-source cloud management platform, which has been witnessing a tremendous evolution of its networking features - in its networking project mostly known by the codename *Neutron*. It is also important to note that *Neutron* provides network service logics, and relies on different backends called "drivers" to interact with different networking technologies. Among these drivers is the recent OpenDaylight[11] SDN controller. OpenDaylight is today seen as an initiative equivalent to OpenStack in the SDN domain. With this in mind, our DC Controller is based on OpenStack and OpenDaylight.

[10] OpenStack, http://www.openstack.org/

[11] OpenDaylight, http://www.opendaylight.org/

i) OpenStack: from a networking perspective, OpenStack allows the creation and management of *networks* (L2 network segments) and *ports* (attachment points for devices connecting to networks, e.g. virtual Network Interface Cards (vNICs) in VMs). The OpenStack community has been doing a considerable effort keeping up with users' demand on introducing new *Neutron* network service types - L3 routing, firewall as a service (FWaaS), load balancer as a service (LBaaS) and VPN as a Service (VPNaaS); however, it is unfeasible (and probably unwise) in the long run to keep up with demands at this pace in a timely manner. Therefore, we argue that OpenStack should focus in offering the basic tools for network services to be orchestrated at a higher level and be deployed as VMs.

With the orchestration and composition of SFs in mind, it is easy to identify the need to fill a gap in OpenStack, the one of steering traffic between OpenStack elements (e.g. VMs, routers). We envision a new OpenStack service abstraction that extends and relies on current OpenStack networking features, allowing traffic steering between *Neutron ports* according to classification criteria. New entities are introduced into the OpenStack *Neutron* data model: *Port Steering*, and *Classifier*. Both entities have a set of common OpenStack data model attributes, i.e. *id*, *name*, *description*, and *tenant_id*. The *Port Steering* adds to this common set a list of ports (*ports* attribute), and a list of classifiers (*classifiers* attribute). The former lists the sets of ports that must be targeted of classification and then steered. The *Classifier* entity adds the following attributes: *type*, *protocol*, *port_min*, *port_max*, *src_ip* and *dst_ip*.

This functionality is very useful as it provides the means to realize, among other things, SFC, as described in Section V.

ii) OpenDaylight: OpenDaylight has a module that integrates with OpenStack *Neutron* for the enforcement of services in the infrastructure. This module was extended in order to support and enforce the previously referred OpenStack traffic steering feature. It is important to highlight that this implementation relies on OpenFlow and Open vSwitch Database Management Protocol (OVSDB) for the management of network resources.

Wide Area Network Controller: The WAN Controller is responsible for managing the operator network, and it exposes connectivity services to the upper layers (in this case the orchestrator). In this context, WAN services are used to support SFs (SF is the client of the WAN service). Point-to-point and multi-point connections with guaranteed network QoS are provided. These are exposed through a service interface that, similar to cloud IaaS interfaces, is technology agnostic. The details and mechanisms to manage the automatic establishment of connectivity services across different locations are detailed in [6].

H.4 Service Function Virtualization

This section elaborates on how SFs are modeled towards virtual infrastructure resources. Figure H-2 depicts the correspondent data model and each class is detailed below.

Service Function: represents an instance of a functional block responsible for a specific treatment of received packets.

Service Function Endpoint (SFE): represents an external interface of one SF instance that is always associated to a SF. Each SFE can have associated information regarding layer 1 (e.g. physical/virtual interface), layer 2 (e.g. MAC address) and/or layer 3 (e.g. IP address), or even regarding higher layers (e.g. HTTP).

From an infrastructure perspective, the resources considered to realize a SF are: Compute Instance (i.e. Virtual or Physical Machines), Image (disk image), Compute Flavor (hardware specification of a compute instance, i.e. CPU, memory and root disk), Block Storage (additional disks), Port (i.e. network interface), Network (a network segment), and Link (a connection between two Ports from different Compute Instances) which has an associated Link Flavor (dedicated QoS in terms of bandwidth, delay and jitter). A SF can be associated to multiple Compute Instances, while each Compute Instance has a single Image, a single Flavor and can have multiple Ports and Block Storages. A Port can only be associated to a single Network; however, it can be associated to multiple Links. A SFE is directly associated to a port, but not all ports need to map to SFEs.

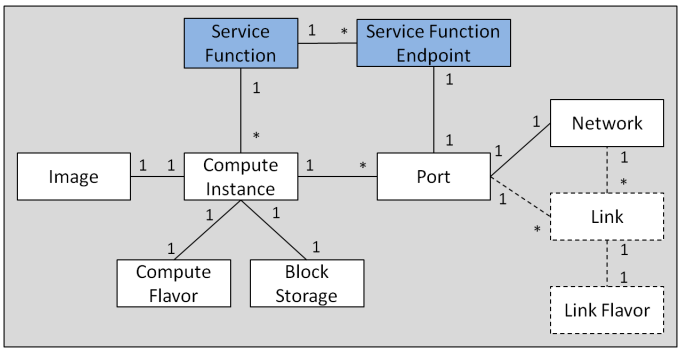


Figure H-2: Service Function data model towards a Cloud Infrastructure

The network QoS, represented in the model by *Link* and *Link Flavor*, is not considered in today's cloud infrastructure systems. However, for a carrier grade cloud this is a must, and OpenStack already has an ongoing project to support it[12].

Figure H-3 presents an example of how several SFs can be composed and organized. Furthermore, it also highlights how SFCs can be built and explored.

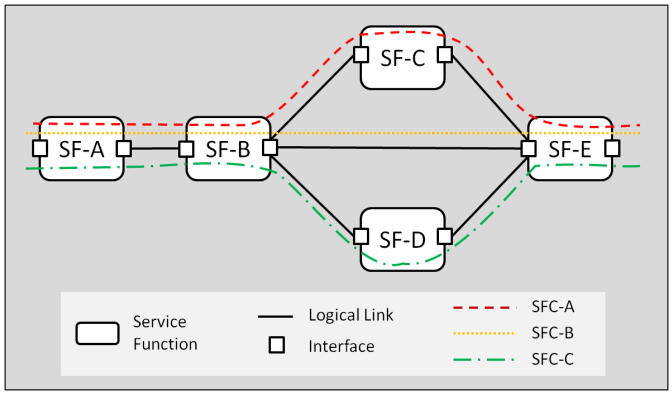


Figure H-3: Service Function composition - example

H.5 Service Function Chaining

In this section we provide insights on the fundamentals and modeling aspects of SFC proposed in this work.

H.5.1 Fundamentals

In SFC two aspects are vital:

Classification: a policy for matching packets (e.g. *HTTP* traffic) used for the identification of appropriate actions (e.g. forwarding). It can be for example an explicit forwarding entry in a network device that forwards packets with a specific IP or MAC address into the SFC. (Re)Classification can also occur at each SF of the SFC independently from the previous SFs. In such cases, multiple classification policy entries should be allowed in an SFC system.

Traffic Steering: ability to manipulate the traffic route at the granularity of subscriber and traffic types [7]. The actual network topology should not be modified to accomplish this.

Moreover, the combination of classification and traffic steering can be done in two ways:

[12] OpenStack Neutron QoS support https://wiki.openstack.org/wiki/Neutron/QoS

Tagged packet approach: classification can occur only at the initial redirection points to a SFC, if upon this classification packets are tagged. After that, packets are steered to the SFC and routed along it according to the embedded tags.

Non-tagged packet approach: classification occurs not only at the redirection points but also at each hop of the SFC. In this case, packets are not tagged and are subject of classification and steering at each SFC hop.

The consequences of following a tagged or non-tagged packet approach are felt at the VIM Plane level. One of the benefits is that this choice is relatively well isolated from the higher planes. We believe the non-tagged approach to be the smoothest approach to follow due to its lower impact on SFs and virtual infrastructure management systems. The advantage of the tagged approach is that the traffic only needs to be classified and tagged (e.g. with a VLAN tag, or another tag) once along the entire SFC. The drawback is the fact that the SFs need to know how to handle the tags (in the simplest case, they should at least ignore them). Although, we can add in the platform the support for a tagged approach (e.g. classify only at one point, tag, and steer traffic according to tag), it will only make sense if there is also support at the SF level. In this sense, in this work we adopt the non-tagged packet approach.

Further aspects should be taken into account when elaborating a SFC solution, such as: i) no assumption should be done on how functions are deployed, i.e. whether they are deployed on physical hardware, as one or more VMs, or any combination thereof; ii) a SF can be part of multiple SFCs; iii) a SF can be network transport independent; iv) a SFC allows chaining of SFs that are in the same layer 3 subnet and of those that are not; v) traffic must be forwarded without relying on the destination address of packets; vi) classification and steering policies should not need to be done by SFs themselves [10].

H.5.2 Service Function Categories

Two categories of SFs have been defined.

Active SFs: those that are in fact part of the main course of a packet, in which case two sub-types are considered: a) functions that may drop packets or forward them, such as a Firewall; b) functions that can actually change packets, e.g. an IPSec VPN server.

Passive SFs: are considered to be out of the main course of the chain. These functions mainly inspect packets, e.g. a monitoring system or a DPI. In practice one can think of a SF in a physical device connected to a hub through a single network interface configured in promiscuous mode. Traffic is considered to be duplicated when having to reach a passive function.

These two categories are important because they impose constraints on how classification and steering can be implemented. In short, passive functions can rely on packet characteristics as packets are not modified, while active functions must be integrated at a service level because ingress and egress packets can be different (e.g., VPN). If a SFC has active functions that change packets, the classification may differ when passing one of these functions.

H.5.3 Service Function Chaining Abstraction Model

The ability to classify and steer traffic accordingly can be enough to implement at the low level the SFC functionality. However, it is important not to forget that the traffic steering functionality is a low level functionality that does not explicitly express a SFC. Having in mind the considerations made so far, a base data model for SFC (that supports both tagged and non-tagged approaches) is now presented. Naturally, other SFC service abstraction proposals may appear in the future, but we consider that this model lays a strong foundation over which other service abstractions can easily be created by extending the model. Figure H-4 depicts the model. Five main classes are considered: *Service Function Chain*; *Service Function; Service Function Endpoint; Packet Flow* and *Classifier*.

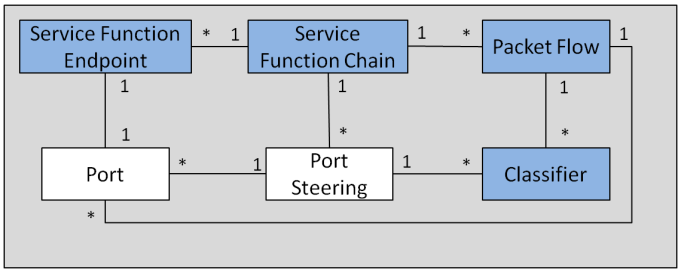


Figure H-4: Service Function Chain data model towards a Cloud Infrastructure

All classes have the following attributes: *id*, *name*, and *description*. The *id* refers to a unique identifier able to identify the class instance within the SFC system. The remaining two, *name* and *description*, are attributes that allow a human-readable characterization of the class instance. Below it is provided further detail about each class.

Service Function Chain (SFC): a SFC has a set of *Service Functions* (SFs) associated and an attribute that defines the ordered sequence of functions (*path*). Since a function can have more than one SFE, the *path* attribute is specified by an ordered list of SFEs organized by hops. For example:

"*path= { hop={SF-A_E2, SF-B_E1};*

hop={SF-B_E2, SF-D_E1}, passive={SF-C_E1} }" - where the chain crosses SF-A, SF-B, and SF-D and has SF-C as a passive function between SF-B and SF-D.

Classifier: a *classifier* represents a classification criteria applied to a packet, which determines if the packet matches that specific criteria or not. In this sense, a *classifier* has an attribute *filter* that contains the classification criteria, e.g.:

"*filter={protocol='6'; port='80-90';*

source_IP='192.168.10.20/32'; destination_IP='192.168.10.40/32'}" – matches all TCP traffic using ports between 80 and 90 with source IP address 192.168.10.20 and destination IP address 192.168.10.40.

Packet Flow: One *classifier* only identifies packets with a certain criteria, while a *packet flow* represents an aggregator of *classifiers*. In this sense, a *packet flow* can have multiple *classifiers*, and a *classifier* can be associated to multiple *packet flows*. Moreover, a *packet flow* has a *source* and *destination port*. The former identifies where the initial classification and redirection of the packet flow to the SFC takes place, while the latter identifies where packets are to be delivered after passing through the SFC. The attributes considered so far would be enough if the system realizing the SFC followed a tagged packet approach. For a non-tagged approach, an additional attribute is considered - *sfc_classifiers*. Due to the possibility of (active) SFs to modify packets, the classification initially done may not be the same along all hops of the SFC, and therefore, the *sfc_classifiers* attribute matches the classification criteria (*classifiers*) at each hop of the SFC. Furthermore, the attribute *direction* is also considered to identify the direction of the SFC which the *packet flow* must traverse – this attribute can assume one of two values: *forward*, *reverse*. We consider that multiple *packet flows* can be associated to a single SFC instance.

Port Steering: this entity refers to the functionality presented above in OpenStack. This feature allows steering traffic between *ports*. Further details about the traffic steering functionality can be found in the OpenStack proposal[13], for which we developed a prototype implementation.

In terms of operations, all classes are considered to allow CRUD (Create, Read, Update and Delete) operations.

[13] OpenStack Traffic Steering blueprint, https://review.openstack.org/#/c/92477/

H.6 Proof of Concept

A POC environment has been deployed to showcase how a Telco can leverage the features described in this work. We highlight one of the most attractive use-cases in the NFV scope and how it has been realized in this POC.

The testbed in place is depicted in Figure H-5, focusing on the PoP setup that is detailed further ahead. At the core of the operator network (Telco Core Network) there is an IP/MPLS backbone composed of four provider (P) routers and four provider edge (PE) routers. The core network is managed by proprietary OSSs that expose connectivity services through a service interface in a technology agnostic manner (the WAN controller). The core network connects to two DC premises (managed by OpenStack *IceHouse* release with the traffic steering functionality), one of which represents a centralized DC and the other a PoP. Finally, the customer premises are represented by switching equipment, which is logically connected to the PoP over an access network (a simple switch based network).

A prototype of the Cloud4NFV orchestrator, which interacts with the WAN and DC controllers, was developed using the Python language. Details regarding the orchestrator implementation (e.g. RESTful API) can be found in [11].

H.6.1 Customer Premises Equipment Use-Case

Customer Premises Equipment (CPE) are often pointed out as one of the most suitable candidates SFs for virtualization [2] [12]. SFC will surely play a particularly relevant role in this case.

The CPE can be seen as a standard routing node enhanced by collection of SFs, such as Network Address Translation (NAT), Firewall (FW), Voice over IP (VoIP) servers, Virtual Private Network (VPN) servers, Network-Attached Storage (NAS), WAN Optimization Controllers (WOC), Deep Packet Inspection (DPI) or Intrusion Prevention System (IPS). These services are deployed for different scenarios, and not all traffic needs to traverse them, leaving room for optimization through SFC. It should be noted that some of the chains can even be temporary, which requires a model that enables the dynamic definition of chains.

H.6.2 Service Function as a Service (SFaaS)

At the service layer we implemented a prototype of the SFaaS concept. This is exposed via a web portal. CPE functions are available in the SFaaS, and the ability to perform SFC is not exposed to the end-user. The user requests CPE SFs, which already have a pre-determined relation with other SFs, and associates them to one of his sites. The instantiation and configuration of the SFs is done in a few minutes and the user is able to control them through a dedicated SF management portal.

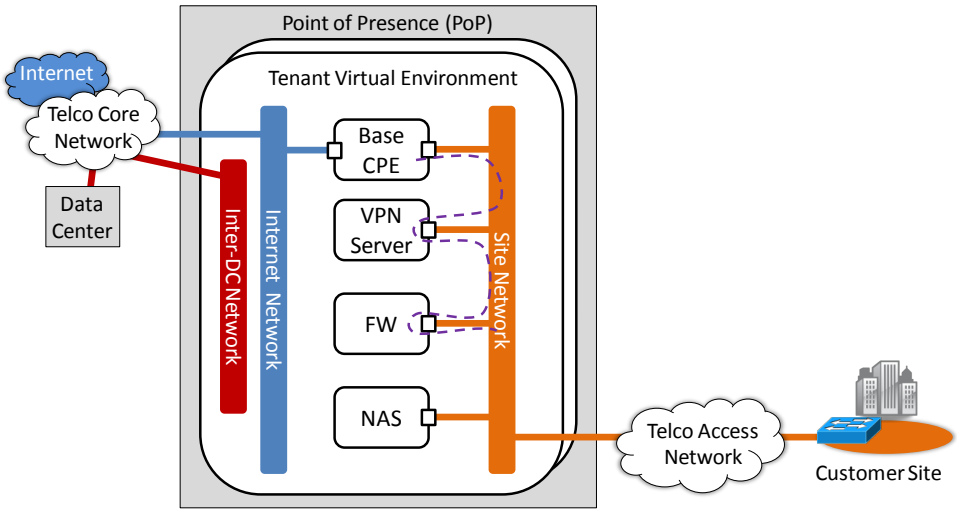


Figure H-5: POC prototype setup

Note that the use of the SFaaS service requires a basic business relationship between the customer and the Telco (customer sites registered and with connectivity services). In other words, the user is a customer

of the Telco who provides connectivity services (e.g. fiber, copper) from the client's sites (e.g. house, enterprise premises). On the site side, a L2 device (or a L3 device in bridge mode) is considered to be in place.

Currently, from a demonstration viewpoint, it is considered that the user, after having the physical connection in place, must first buy a base CPE function with routing, DHCP and NAT functionalities (the POC relies on the OpenStack L3 native device). From that moment on, the user can acquire other CPE functions and services, e.g.: Internet connection, Firewall (POC relies on *iptables*), VPN Server (POC relies on *OpenVPN*), NAS (POC relies on *Samba*) and others.

H.6.3 Prototype setup

Figure H-5 depicts the POC prototype setup with four functions as an example. Special attention is given to the setup at the PoP level.

Each customer has a dedicated virtual private environment in the PoP that is serving his site. This environment allows the creation of virtual networks and VMs (in OpenStack this is known as tenant or project). There is a point to point logical connection between the customer's premises (L2) device (currently we are using VLAN encapsulation to establish this connection, but others can be used). On the PoP side this logical connection is extended to a virtual network in the tenant virtual environment – in the figure "Site Network", which has a private IP range (the OpensStack provider network concept is used to achieve this). Moreover, there is a virtual network that is shared among all tenants, which in the figure is the "Internet Network" (in OpenStack this is achieved using the external network concept). This latter network is then connected to the core network which provides the Internet access. Also depicted in the figure is the "Inter-DC Network" which provides access between the PoP and the DC over a Telco VPN service in the core network (again, on the PoP side we rely on the OpenStack provider network concept to connect to the VPN). The processes explained so far are considered to be in place as soon as the customer establishes the basic business relation with the Telco.

All functions, when deployed upon request, are connected to the "Site Network". When an Internet connection is requested, the base CPE is connected to the "Internet Network" and configured to perform NAT. The figure also highlights a SFC that comprises the base CPE, VPN server and firewall.

H.7 Future Work

Currently, the POC does not support the enforcement of network QoS in DC domains; this is only supported in the WAN connectivity services. We expect to add this support by the time OpenStack officially releases this feature. Furthermore, runtime management operations (such as scaling and migration of SFs) are yet to be included in the platform. On the WAN domain, we are currently adding a SDN-based network. The purpose is to have both legacy and SDN network technologies in place to better evaluate the advantages and disadvantages of each approach. Finally, we are working on exposing the ability of performing SFCs to the end-user.

H.8 Conclusions

The orchestration and management of SFs is today a complex task that takes considerable time and effort. However, concepts like Cloud Computing, SDN and NFV are paving the way to handle SFs in a much more flexible and agile manner. The Telco will play a key role in this scenario, and we have given some insights on how that can be performed in the near future. Special attention has been given to the modeling of SFs towards cloud resources and to the combination of SFs through SFC. Finally, a platform for managing virtual SFs in a Telco cloud infrastructure has been presented and we described a POC that showcases how the platform and the principles here presented can be leveraged in a Telco environment.

References

[1] H. Kim, N. Feamster, "Improving network management with software defined networking," Communications Magazine, IEEE, vol.51, no.2, pp. 114-119, February 2013.

[2] ETSI, Network Functions Virtualisation (NFV): Use Cases, Technical Report ETSI GS NFV 001 v1.1.1, October 2013.

[3] P. Quinn, Kumar, T. Nadeau, "Service Function Chaining Problem Statement," IETF Internet draft, Informational, December. 2013.

[4] A. Császár, W. John, M. Kind, C. Meirosu, G. Pongrácz, D. Staessens, A. Takács, and F.J. Westphal, "Unifying Cloud and Carrier Network: EU FP7 Project UNIFY", IEEE/ACM 6th International Conference on Utility and Cloud Computing (UCC). IEEE Computer Society, Washington DC, USA, 2013.

[5] G. Xilourisl, E. Trouva, F. Lobillo, J.M. Soares, J. Carapinha, M.J. McGrath, G. Gardikis, P. Paglierani, E.Pallis, L. Zuccaro, Y. Rebahi, A. Kourtis, "T-NOVA: A Marketplace for Virtualized Network Functions", European Conference on Networks and Communications (EUCNC 2014), June 2014.

[6] H. Puthalath, J. Soares, et al., "Negotiating On-demand connectivity between clouds and wide area networks," IEEE CloudNet, Paris, November 2012.

[7] Y.Zhang, N. Beheshti, L. Beliveau, G. Lefebvre, R. Manghirmalani, R. Mishra, R. Patney, M. Shirazipour, R. Subrahamaniam, C. Truchan, Tatipamula, M., "StEERING: A Software-Defined Networking for Inline Service Chaining," Proceedings of. IEEE ICNP 2013, Goettingen, Germany, October 2013.

[8] ETSI, Network Functions Virtualisation (NFV): Architectural Framework, Technical Report ETSI GS NFV 002 v1.1.1, October 2013.

[9] ETSI, Network Functions Virtualisation (NFV): Terminology for Main Concepts, Technical Report ETSI GS NFV 003 v1.1.1, October 2013.

[10] W. John, K. Pentikousis, G. Agapiou, E. Jacob., M. Kind, A. Manzalini; F. Risso, D. Staessens, R. Steinert, C. Meirosu, "Research Directions in Network Service Chaining", Future Networks and Services (SDN4FNS), 2013 IEEE SDN for , vol., no., pp.1,7, 11-13 November 2013.

[11] J. Soares, B. Parreira, M. Dias, J. Carapinha, S. Sargento, "Cloud4NFV: A Platform for Virtual Network Functions", IEEE CloudNet, October 2014.

[12] Alcatel Lucent, "Network Functions Virtualization – Challenges and Solutions", White Paper, 2013.