UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

RAFAEL PEREIRA ESTEVES

Application-Aware Adaptive Provisioning in Virtualized Networks

Thesis presented in partial fulfillment
of the requirements for the degree of
Doctor of Computer Science

Prof. Dr. Lisandro Zambenedetti Granville
Advisor

Porto Alegre, December 2014

CIP – CATALOGING-IN-PUBLICATION

Esteves, Rafael Pereira

Application-Aware Adaptive Provisioning in Virtualized Networks / Rafael Pereira Esteves. – Porto Alegre: PPGC da UFRGS, 2014.

104 f.: il.

Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2014. Advisor: Lisandro Zambenedetti Granville.

1. Network Virtualization. 2. Network Management. 3. Resource Allocation. 4. Simulation. I. Granville, Lisandro Zambenedetti. II. Título.

ACKNOWLEDGEMENTS

First of all, I would like to thank my beloved parents that devoted their lives to give me the opportunity to have the education that they could not have and sacrificed themselves so I could pursue my dreams. I love you and I am forever grateful for all you have done for me.

I would like to thank my advisor Lisandro Zambenedetti Granville for his valuable guidance during all these years. His patience, optimism, enthusiasm, and excellence have inspired me to become a better researcher. He has always encouraged me to follow my own ideas and has given crucial feedbacks to improve my research.

I would not be here right now if Antonio Abelem from UFPA had not given me the opportunity to work in a research project back in 2003, when I was a second-year undergraduate student in the distant Belém. This opportunity changed my life forever and inspired me to pursue an academic career. In other words, he is the one to blame!

During the last years, I had the privilege to meet and work with many good friends in Porto Alegre. Thank you for the good times I shared with you in trips around the world, barbecues, soccer matches, and some parties. You helped me to stay sane and get the energy necessary to complete this work. I would like to apologize to my friends that are in Belém or spread around the world for my absence, especially in these last months.

As part of my PhD, I spent one year as a visiting student at the Network and Distributed Systems Group of the David R. Cheriton School of Computer Science of the University of Waterloo. This was by far one of the greatest experiences of my life. I thank professor Raouf Boutaba for hosting me and supporting me during that period. His guidance was essential to define the focus of my research. In Waterloo, I have met some brilliant individuals that helped me with valuable discussions and advices.

I would also like to thank my thesis committee. Their valuable feedback has indeed improved this thesis and made my life way easier.

Finally, I would like to acknowledge CNPq for supporting me with a scholarship during all these years.

*"For everything there is a season,
and a time for every matter under heaven."*
— Ecclesiastes 3:1

ABSTRACT

Network virtualization is a feasible solution to tackle the so-called Internet ossification by enabling the deployment of novel network architectures in a flexible and controlled way. With network virtualization, it is possible to have multiple virtual networks (VNs) running simultaneously on top of a shared physical infrastructure. Network management with virtualization support, however, poses challenges that need to be addressed in order to fully achieve an effective and reliable networking environment.

One of the main aspects related to the management of network virtualization environments is virtual network provisioning. Virtual network provisioning defines how virtual network resources (nodes and links) are allocated in the physical infrastructure. VN provisioning often relies on embedding algorithms that aim to achieve well defined objectives, such as reducing allocation cost, load balancing, or minimizing energy consumption.

Although VNs share the same infrastructure, they typically host diverse applications with different goals. Unfortunately, current provisioning solutions focus on a single or a limited set of objectives that may not simultaneously match the requirements of an increasing number of applications deployed in networks everyday. Novel applications may require different objectives that are not supported by the active provisioning system.

In this thesis, we formulate the Application-Aware Virtual Network Provisioning Problem (AVNP) and propose an adaptive provisioning framework for virtualized networks that takes into consideration the characteristics of multiple applications and their distinct performance objectives. The proposed framework is based on the concept of *allocation paradigm*, which is defined as a set of provisioning policies that guide the resource allocation process. A paradigm translates objectives from both Infrastructure Providers (InPs) and Service Providers (SPs) to individual allocation actions that actually provision VNs. A policy language is also defined to express the relationship between paradigms, objectives, and actions.

To determine the efficiency of a particular paradigm, we propose a virtual network performance computation model based on data measured from existing virtualization benchmarks. The model is able to quantify the performance of allocated VNs and guide paradigm changing decisions. Extensive simulations were performed to verify the viability of the proposed solution and compare different paradigms. Results show the feasibility of allocation paradigms in helping network providers to select the best provisioning strategy given a set of InP/SP objectives.

Keywords: Network Virtualization, Network Management, Resource Allocation, Simulation.

Aprovisionamento Adaptativo Orientado à Aplicação em Redes Virtualizadas

RESUMO

A virtualização de redes é uma solução proposta para superar a chamada ossificação da Internet pois permite o desenvolvimento de novas arquiteturas de rede de forma flexível e controlada. Com a virtualização de redes, é possível criar múltiplas redes virtuais operando simultaneamente em uma infraestrutura física compartilhada. No entanto, o gerenciamento de redes com suporte a virtualização apresenta desafios que precisam ser resolvidos para obter um ambiente de rede confiável e funcional.

Um dos principais aspectos relacionados ao gerenciamento de ambientes de virtualização de redes diz respeito ao aprovisionamento de redes virtuais. O aprovisionamento de redes virtuais define como os recursos de rede virtuais (nós e enlaces) são alocados na infraestrutura física. O aprovisionamento de redes virtuais é comumente baseado em algoritmos de mapeamento que possuem objetivos bem definidos como reduzir o custo de alocação, realizar balanceamento de carga ou minimizar o consumo de energia.

Embora redes virtuais compartilhem a mesma infraestrutura, elas tipicamente são utilizadas para hospedar várias aplicações que possuem diferentes objetivos. Infelizmente, as soluções de aprovisionamento atuais focam em um único ou em um conjunto muito limitado de objetivos que podem não ser capazes de satisfazer os requisitos de um número cada vez mais crescente de aplicações. Novas aplicações podem exigir objetivos diferentes dos que são suportados pelo sistema de aprovisionamento que está em operação em uma infraestrutura de virtualização de redes.

Nesta tese, o problema de Aprovisionamento de Redes Virtuais Orientado à Aplicação é formulado e um arcabouço de aprovisionamento adaptativo para redes virtualizadas que considera as caracteristicas de várias aplicações bem como seus requisitos de desempenho é proposto. O arcabouço proposto é baseado no conceito de *paradigma de alocação*, que é um conjunto de políticas de aprovisionamento que guiam o processo de alocação de recursos. Um paradigma traduz objetivos de Provedores de Infraestrutura e Provedores de Serviço para ações de alocação individuais que criam as redes virtuais. Uma linguagem de políticas para paradigmas é também definida para expressar o relacionamento entre paradigmas, objetivos e ações.

Para determinar a eficiência de um paradigma de alocação, é proposto um modelo para quantificar o desempenho de redes virtuais que é baseado em dados coletados de sistemas de *benchmarking* aplicados no contexto de ambientes virtualizados. O modelo proposto é capaz de calcular o desempenho das redes virtuais alocadas e influenciar mudanças em paradigmas de alocação. Simulações foram conduzidas para verificar a viabilidade da solução proposta e comparar diferentes paradigmas de alocação. Resultados mostram que o uso de paradigmas de alocação pode ajudar administradores de ambientes de virtualização de redes a escolher a melhor estratégia de alocação dado um conjunto de objetivos definidos pelos Provedores de Infraestrutura e pelos Provedores de Serviço.

Palavras-chave: Virtualização de Redes, Gerenciamento de Redes, Alocação de Recursos, Simulação.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

4WARD	Forward
AMS	Autonomic Management System
API	Application Programming Interface
APN	Articulated Private Network
AS	Allocation Service
AVNP	Application-Aware Virtual Network Provisioning
AUTOI	Autonomic Internet
CLI	Command Line Interface
CNRMS	Customer Network Resources Management System
CPU	Central Processing Unit
COST	Continent cOuntry State ciTy
FEDERICA	Federated E-infrastructure Dedicated to European Researchers Innovating in Computing Network Architectures
FP7	Seventh Framework Programme
FPGA	Field Programmable Gate Array
GENI	Global Environment for Network Innovations
GRE	Generic Routing Encapsulation
HSU	High Server Utilization
HTTP	Hypertext Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
INM	In-Network Management
InP	Infrastructure Provider
IP	Internet Protocol
ISP	Internet Service Provider
L3VPN	Layer 3 Virtual Private Network

LAN	Local Area Network
LAP	Location Awareness Protocol
LB	Load Balancing
LCC	Low Communication Cost
LSP	Label Switched Path
MIB	Management Information Base
MIP	Mixed Integer Programming
MPLS	Multiprotocol Label Switching
MS	Monitoring Service
NETCONF	Network Configuration Protocol
NF	Non Functional
NOC	Network Operation Center
NVE	Network Virtualization Environment
PE	Provider Edge
PMS	Paradigm Management Subsystem
PSO	Particle Swarm Optimization
QEMU	Quick Emulator
QoS	Quality of Service
RA	Resource Agent
RCLI	Remote Command Line Interface
SDH	Synchronous Digital Hierarchy
SDN	Software Defined Network
SLA	Service Level Agreement
SNMP	Simple Network Management Protocol
SOA	Service-oriented Architecture
SOAP	Simple Object Access Protocol
SONET	Synchronous Optical Networking
SP	Service Provider
ToR	Top of Rack
UCLP	User Controlled Lightpaths
UCVS	User Controlled Virtual Services
vCPI	Virtualization Component Programmability Interface
VDC	Virtual Data Center
VLAN	Virtual LAN

VM	Virtual Machine
VMI	Virtualisation Management Interface
VN	Virtual Network
VNARMS	Virtual Network-based Autonomic Network Resource Control and Management System
VNE	Virtual Network Embedding
VNRM	Virtual Network Resource Manager
VR-MIB	Virtual Router Management Information Base
VRRP	Virtual Router Redundancy Protocol
VRRP-MIB	Virtual Router Redundancy Protocol Management Information Base
vSPI	Virtualization System Programmability Interface
XML	Extensible Markup Language
XML-RPC	Extensible Markup Language-Remote Procedure Call

LIST OF SYMBOLS

A	Paradigm action
$Alloc$	Allocator
G	Paradigm goal
$\mathcal{P}$	Paradigm
W	Paradigm window
$s(W)$	Size of the paradigm window
N^p	Physical network
M^p	Set of physical machines
m^p	Physical machine
$c(m^p)$	CPU capacity of the physical machine $c(m^p)$
R^P	Set of physical network elements
L^p	Set of physical links
l^p	Physical link
$b(l^p)$	Bandwidth of the link l^p
O^p	Set of InP objectives
o^p	InP objective
$t(o^p)$	Target index of the objective o^p
N^v	Virtual network
M^v	Set of virtual machines
m^v	Virtual machine
$c(m^v)$	CPU requirement of the virtual machine m^v
L^v	Set of virtual links
l^v	Virtual link
$b(l^v)$	Bandwidth requirement of the virtual link l^v
P^v	Set of desired properties
p^v	VN property

$t(p^v)$	Target index of the property p^v
$\mathcal{R}$	Number of rounds
$\sum m^v + \sum l^v$	Total number of virtual resources
$\sum_{i=1}^{al} s(W_i)$	Maximum number of allocation actions allowed per round
al	Number of allocators
c	Policy condition
o	Policy objective
a	Policy action
p	Policy paradigm
w	Policy window
$\mathcal{S}$	Paradigm score
$\mathcal{T}$	Tile performance
τ	Number of tiles
App	Application performance
Ref	Reference value
n	Number of applications
$\mathcal{S}'$	Predicted score
$\mathcal{Q}$	Paradigm quality
$\mathcal{U}$	Reference score
$\mathcal{E}$	Paradigm efficiency

CONTENTS

1 INTRODUCTION

Network virtualization has been considered a viable alternative to guide the development of new network architectures and to overcome the Internet ossification (CHOWDHURY; BOUTABA, 2009) (CARAPINHA; JIMÉNEZ, 2009) (KHAN et al., 2012). Network virtualization thus emerges as an enabling technology to support innovation in the networking area. In a network virtualization environment (NVE) the underlying physical infrastructure, commonly referred to as substrate, is shared among different users who create virtual networks (VNs) that employ isolated protocol stacks. Future Internet advocates claim that NVEs enable diverse network architectures to coexist in a single infrastructure without affecting production services. Moreover, network virtualization is gaining attention from major industry players as commodity network devices (*i.e.*, routers, switches) with virtualization support become popular in the marketplace. In recent years, several network virtualization testbeds have been deployed, allowing researchers to propose and evaluate new solutions at large-scale with real traffic. There are two main opinions shared by Future Internet enthusiasts. The first group states that network virtualization is a transitory step towards the definition of a reference Future Internet architecture, while the other declares that the Future Internet will be an NVE itself (ANDERSON et al., 2005).

Among the technical challenges to enable NVEs, management has a special importance. Management of NVEs is crucial to guarantee the proper operation of the physical infrastructure, of the hosted virtual networks, and of the services supported by the virtual networks. In an NVE, there exist different virtualized components (*e.g.*, links, nodes, domains, services) that present, each one, specific management challenges. For example, the management of virtualized routers must guarantee the isolation of virtual routers's CPUs, memory, and routing tables. The management of virtualized links, in turn, has to deal with performance guarantees for each virtual link. Also, from an end-to-end perspective, inter-domain addressing and service level agreements (SLAs) across multiple domains involved in supporting a virtual network need to be properly managed.

In an NVE, multiple management solutions can share the same physical infrastructure to support different types of virtual networks and hosted applications. For example, virtual networks may require different allocation schemes suited to particular performance objectives defined by service providers. Likewise, a variety of management protocols and models can be deployed in the same infrastructure. Recently, the management of NVEs has started receiving special attention from the research community, motivating a variety of research projects over the past few years, such as 4WARD (4WARD, 2008), AUTOI (AUTOI, 2008), FEDERICA (SZEGEDI et al., 2009), and GENI (GENI, 2009). These projects share the common goal of considering management at the network design phase, as opposed to addressing it after network deployment. This allows to evaluate and test new management architectures before deployment and facilitates the development of customized management solutions for different types of users and applications.

The rest of this chapter is organized as follows. The formal problem definition is introduced in Section 1.1. The main hypothesis and a set of fundamental research questions used to guide this thesis are presented in Section 1.2. In Section 1.3, the main contributions of the thesis are listed. Finally, the organization of the thesis is outlined in Section 1.4.

1.1 Problem Definition

Management in NVEs includes provisioning, monitoring, and interfacing (ESTEVES; GRANVILLE; BOUTABA, 2013). Provisioning consists of allocating VNs to SPs by defining the mapping of VN resources to the physical counterparts. Monitoring involves gathering updated status of physical resources and their associated VNs. Interfacing is required for InPs and SPs to respectively access, operate, maintain, and administer the physical and virtual nodes and links. In this thesis we focus on the provisioning aspect from an application-oriented perspective. The provisioning problem is divided in two subproblems. The first one relates to including application awareness to VN provisioning, which ultimately will determine how VNs are mapped to the substrate. The second one relates to make VN provisioning adaptable to the dynamics of the substrate, where VNs are constantly created and terminated.

Typically, in an NVE, an infrastructure provider (InP) offers virtual network resources to multiple service providers (SPs) that deploy a variety of applications on top of virtual networks (VNs). The provisioning of VNs must consider requirements of both InPs and SPs. While the main objective of InPs is to generate revenue by accommodating a large number of VNs, SPs, on the other hand, have specific needs, such as guaranteed bandwidth among virtual machines, load balancing, and high availability. Inefficiencies in the provisioning process can lead to disastrous consequences for infrastructure providers including reduced number of SPs, monetary penalties (*e.g.*, financial credit) when Service Level Agreements (SLAs) are not satisfied, and low utilization of the physical infrastructure.

SPs request VNs to deploy a variety of applications. SP applications have different characteristics that could determine how VNs are allocated in the physical substrate. Different applications may require, for example, different performance requirements (*i.e.*, belong to different service classes), different high-level objectives (*e.g.*, energy efficiency, load balancing), and different types of resources (*e.g.*, servers, routers, switches, storage). For example, business-critical applications (*e.g.*, ticket reservation, order processing) may require that virtual machine (VM) replicas are placed in distinct locations (*i.e.*, different physical servers). On the other hand, network-sensitive applications benefit if VMs are placed in a single machine, to avoid network bottlenecks. In this respect, one limitation of current VN provisioning schemes is that the specifics of the applications to be deployed over a VN are commonly ignored in the provisioning process because InPs do not know in advance which applications SPs will deploy over a VN.

An SP can deploy a single application on a VN or use a VN to host multiple applications. The choice depends on factors such as cost and performance. While deploying one application per VN provides predictable performance for an SP's applications, one SP may not be able to afford one VN per application due to cost constraints. From the InP perspective, high utilization of the substrate is an important goal because it reflects in increased revenue. If applications can be grouped in a single VN without significant performance degradation the InP operator will be able to deploy more VNs in the same infrastructure. In this case, isolating one VN per application cannot guarantee high utilization of the substrate. Therefore, there is a trade-off between the number of applications a VN can host and the performance requirements of each applications that need to be managed by an NVE provisioning system.

Current NVE provisioning systems allow SPs to request different resource configurations (*e.g.*, CPU, memory, disk) to build a VN. The SP is the main responsible for requesting resources that will better fit its applications. The InP then either allocates resources for the VN in the physical network or rejects the allocation if there are not enough resources to satisfy the SP's request. InPs run allocation algorithms to find the best mapping of VNs onto the physical substrate according to well-defined objectives, such as minimizing the allocation cost, reducing energy consumption, or maximizing the residual capacity of the infrastructure. Mapping virtual to physical resources is commonly referred to as *embedding* and has been extensively studied in recent years (CHOWDHURY; RAHMAN; BOUTABA, 2012) (YU et al., 2008) (CHENG et al., 2012).

Another important issue regarding VN embedding is the fact that most VN embedding solutions map all virtual resources of a VN request as a single enclosed task, *i.e.*, upon receiving a VN request, the InP runs an embedding algorithm that generates as output the complete mapping of virtual resources to their physical counterparts. Such enclosed, single-operation mapping presents limitations. For example, previously allocated VNs expire, releasing resources that could result in better mapping options for an ongoing VN because they satisfy a high-level objective (*e.g.*, map virtual nodes as closest as possible to each other). In this atomic mapping approach, such resources cannot be allocated during VN instantiation because embedding is defined *before* VN deployment. In order to optimize resource allocation and leverage the existence of better mapping alternatives, the VN will have to be migrated, which can result in additional costs for the InP. Moreover, two simultaneous VN requests may end up competing for the same physical resource during their instantiation; because no proper solutions for this race conditions exist, resulting in failed VN requests.

In this thesis, we introduce the concept of *allocation paradigms* to guide resource provisioning in virtualized environments. A paradigm encompasses goals representing with the high-level objectives of the InP and of the SPs. Each objective is achieved through actions (*e.g.*, allocation) executed within a window (one per goal). The action is defined at run-time according to the objectives and the current status of the substrate network. In addition, VN allocation can be performed in several rounds instead of mapping all virtual resources altogether. At each round, actions of each objective of the current active paradigm are included in the allocation window and are executed. This approach allows a quick adaptation of the provisioning process to the dynamics of the substrate and to the characteristics of the deployed applications. Besides, allocation paradigms add flexibility to VN allocations because objectives can be easily added or removed from a paradigm. In this way, VN allocation is rapidly adjusted to reflect changes in either InP or SP objectives. If some of the targeted objectives are not satisfied or if applications running on a particular VN exhibit poor performance, the current allocation paradigm may need to be changed. The concept of allocation paradigms will be detailed through the rest of this thesis.

1.2 Main Hypothesis and Research Questions

Given the limitations in current VN embedding approaches, the main objective of this thesis is to answer the following research question.

Main question: **How to manage resource provisioning in virtualized networks in order to meet both requirements of the multiple applications and of the infrastructure providers by taking advantage of the dynamics of the substrate?**

To answer the above question this thesis presents the following hypothesis:

Hypothesis: **Limitations in virtual network provisioning can be reduced through the use of *allocation paradigms***

In order to guide the investigation of this thesis, additional research questions associated with the hypothesis are defined and presented as follows.

Research question I. *How allocation paradigms can improve virtual network provisioning?*

Research question II. *What methods could be employed to calculate the efficiency of an allocation paradigm?*

Research question III. *What are the possible disadvantages of providing a high level of flexibility to VN provisioning?*

1.3 Contributions

The major contributions of this thesis are outlined below.

- **A model for the management of network virtualization environments.** Because of the importance of management as a first-class requirement for network virtualization we define a conceptual network virtualization management model. The model is generic and provides an overview on how management can be tackled in current NVE implementations by describing the main management entities typically in NVEs and how they interact in order to accomplish management tasks;

- **A survey of management of NVEs.** In order to place this thesis in the overall spectrum of NVE management we first present a comprehensive survey on the management of NVEs. A number of representative research projects addressing diverse NVE management aspects is presented and, most importantly, research challenges and open opportunities in the area are discussed;

- **A framework for multi-objective, application-aware adaptive VN provisioning using allocation paradigms.** From the open opportunities, we focus on the problem of provisioning virtual networks considering multiple and dynamic objectives and propose the concept of allocation paradigms to tackle it. An allocation paradigm encompasses a set of goals representing the high-level objectives of the InP and the SPs. Each objective is achieved through actions (*e.g.*, allocation) executed within a window (one per goal). The action is defined at run-time according to the objectives and the current status of the substrate network. This approach allows a quick adaptation of the provisioning process to the dynamics of the substrate (*e.g.*, VN arrivals and departures) and to the characteristics of the deployed applications. If some of the targeted objectives are not satisfied or if applications running on a particular VN exhibit poor performance, the current allocation paradigm may need to be changed;

- **A language to express application objectives and translate objectives to allocation actions.** A policy language to allow InP operators to specify the relationship between paradigms, objectives, and allocation actions is proposed. The paradigm policies are used by InP operators to define proper allocation actions;
- **A model to quantify the efficiency of allocation paradigms.** A VN computation model that measures the performance of an allocation paradigm based on the applications running on the embedded VNs is proposed. The model is based on three main aspects: paradigm quality, provisioning time of VNs, and provisioning cost. These aspects allow to evaluate the performance of paradigms so as to help InPs to better define provisioning approaches.

1.4 Thesis Roadmap

The thesis is organized as follows.

In Chapter 2, the state of the art on the management of NVEs is presented. The goal is to provide an overview of management network virtualization and enumerate open research opportunities in the area.

In Chapter 3, the problem of provisioning virtual networks considering multiple and dynamic InP and SP objectives is defined and the main concept of this thesis to address the problem (*i.e.*, allocation paradigms) is introduced. First, a conceptual architecture of an adaptive provisioning system based on allocation paradigms is depicted. Then, basic definitions associated with allocation paradigms are described. Next, a methodology to map application high-level goals to allocation paradigms and a policy language used to describe allocation paradigms are presented. A methodology to determine the efficiency of an allocation paradigm in terms of provisioning quality is described and a VN computation model that measures the performance of an allocation paradigm based on that of the applications running on the embedded VNs is proposed.

In Chapter 4, an overall evaluation of the proposed paradigm-based framework is conducted and its advantages and limitations are discussed. Results regarding the efficiency of allocation paradigms for applications, benefits and shortcomings of adaptive provisioning compared to traditional provisioning approaches, and impact of paradigm changes on provisioning are presented.

In Chapter 5, final remarks and conclusions are discussed. The answers for the research questions and contributions are exposed and justified. In addition, opportunities for future research are identified and detailed.

2 MANAGEMENT OF NETWORK VIRTUALIZATION

The goal of this chapter is to provide a comprehensive background on the main topics discussed along the thesis. We start with an overview of network virtualization and discuss how management is currently tackled in such environments. Next, we introduce a conceptual NVE management model and survey the most prominent solutions found in the literature related to NVE management highlighting their main features, benefits, and limitations. Finally, we discuss open research challenges related to NVE management.

2.1 Network Virtualization Environments

In this section, the main concepts of network virtualization are presented. We start by describing the main components of an NVE. Next, we discuss the business model typically employed in NVEs along with the relationship among participating entities.

As mentioned before, network virtualization is a promising approach to enable Future Internet. A generic NVE model is depicted in Figure 2.1. In the substrate layer, physical nodes and links from different network administrative domains serve as a substrate for the deployment of virtual networks. Physical nodes, at the core of the physical networks, represent network devices (*e.g.*, routers) that internally run virtual (or logical) routers instantiated to serve virtual networks's routing necessities.

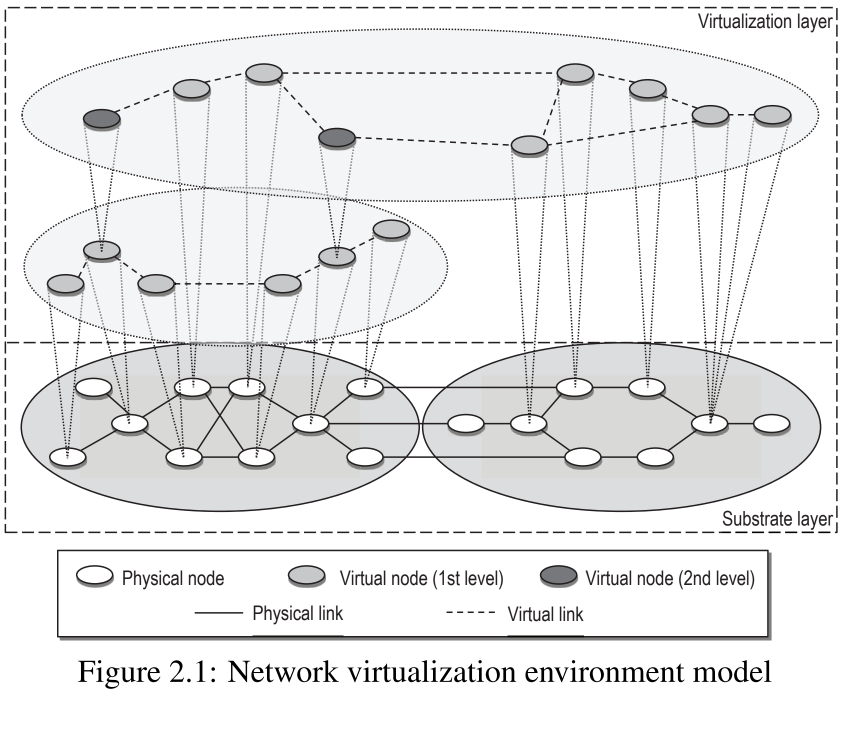


Figure 2.1: Network virtualization environment model

In the virtualization layer, virtual nodes and links are created on the top of the substrate and combined to build virtual networks. Virtual nodes are created on top of a layer of software called hypervisor which is responsible, among other functions, to provide an isolated view for each virtual node sharing the physical one. Virtual links are isolated from each other by enabling technologies such as tunneling or virtual LANs. A virtual network can use resources from different sources, including resources from other virtual networks, which in this case results in a hierarchy of virtual networks. Virtual networks can also be entirely placed into a single physical node (*e.g.*, physical end-host). In this case, since virtual links are not running on top of any physical counterpart, isolation and performance guarantees should be offered, for example, through memory isolation and scheduling mechanisms. In another setup, virtual networks can spread across different adjacent physical infrastructures (*i.e.*, different administrative domains). In this case, network operators, at the substrate layer, must cooperate to provide a consistent view of the underlying infrastructure used by networks from the virtualization layer.

In typical NVE implementations, physical routers can be built employing network processors (GILADI, 2008) (COMER, 2005) or FPGAs (Field Programmable Gate Arrays) (SHAH, 2001) in order to efficiently enable multiple virtual routers running inside a physical one. Such technologies allow packet processing in very high speeds, dynamic device reconfiguration, and the coexistence of multiple processors in a single physical device (TURNER, 2004) (HEPPEL, 2003). Examples of networking devices supporting network processors or FPGAs and that can be virtualized include the Juniper M Series (JUNIPERNETWORKS, 2009) router and the NetFPGA (NetFPGA, 2009). Another alternative is to use physical commodity servers as routers. Instances of routing software such as XORP (XORP, 2012), Quagga (QUAGGA, 2010), Vyatta (VYATTA, 2010), or Open vSwitch (OPENVSWITCH, 2010) can be deployed on virtual machines running on top of hypervisors to enable router virtualization. The advantage of using physical servers is cost reduction because routers and switches are way more expensive than servers. On the other hand, environments deployed using this approach may experience lower performance compared to real routers or switches that are optimized for routing/forwarding tasks.

At the physical end hosts, Xen (XEN, 2013), VMware (VMWARE, 2013), and QEMU (QEMU, 2013) are examples of popular virtualization platforms used to host several virtual machines instances in a single computer. Finally, physical links employ technologies such as Virtual LANs (VLANs) (IEEE, 2006), Generic Routing Encapsulation (GRE) (FARINACCI et al., 2000), and MPLS LSPs (Label Switched Paths) to allow a single physical link to be shared by several virtual ones and to provide isolation among different virtual links belonging to a physical one.

Network virtualization can also be observed from a business point-of-view. The traditional model of an Internet service provider (ISP) offering both infrastructure and services to its customers does not hold in network virtualization anymore. The original roles of ISPs, in the light of virtualization, are now decoupled in two entities when NVEs take place (CHOWDHURY; BOUTABA, 2009):

- **Infrastructure providers (InPs)** are responsible for leasing the physical resources that are used to create virtual networks;

- **Service providers (SPs)** use resources from one or more infrastructure providers to build their own virtual networks and offer services to their customers. SPs can create virtual networks using resources from other SPs too, in a recursive manner.

When mapping InP and SP roles to the general model of NVEs, InPs belong to the substrate layer, while SPs belong to the virtualization layer. In such business model, SPs have flexibility to innovate and deploy new value-added services that ultimately affects their revenue. NVEs also allow InPs to use dynamic pricing schemes by adjusting prices according to supply and demand. Other benefits of network virtualization include:

- **Improved testing and deployment:** multiple VNs can be deployed over a single physical network without affecting its normal operation and production services because VNs are isolated from each other and from the underlying hardware.
- **Scalability:** in a first moment, virtual routers can support a small set of functionalities and be later extended to incorporate new features and adjust capacity accordingly.
- **Reduced costs:** a physical resource can be shared by multiple VNs, thus eliminating the necessity to have one dedicated equipment for each service type.
- **Security:** because each virtual router handles its own addressing space, security leaks, configuration errors, and software errors are less likely to happen.

Similar to traditional networks, NVEs need to be properly managed. In the next subsection, we discuss management of NVEs, based on the three different perspectives: management targets, management functions, and management approaches.

2.2 Management of Network Virtualization Environments

NVE management solutions can be classified according to three different perspectives: *management targets*, *management functions*, and *management approaches* (ESTEVES; GRANVILLE; BOUTABA, 2013). A management target refers to the component being managed, such as physical and virtual nodes, intra-domain links, and inter-domain links. A management function denotes a specific capability supported by a management application, including resource provisioning and monitoring. Management approaches, *i.e.*, the way management components are organized and interact with each other, typically employed on an NVE vary from centralized and distributed management to manual and autonomic/policy-based management (IFIP, 2011).

To allow a better understanding of the management of NVEs in terms of management targets, functions, and approaches we first introduce a conceptual management model for network virtualization describing the entities, relationships, and management operations typically found in NVEs (ESTEVES; GRANVILLE; BOUTABA, 2013). Management operations in NVEs can be classified into InP management and SP management operations. InP management includes, for example, provisioning of virtual networks and monitoring of physical resources. SP management, in turn, deals with the operation of virtual networks and provide services to end-users. Figure 2.2 depicts a conceptual management model for network virtualization considering the relationship between InPs and SPs.

InPs offer their physical nodes to host virtual ones owned by SPs. Several virtual nodes can be created and coexist in an isolated way inside a physical one. Physical nodes are controlled by an *infrastructure manager*, which, using a management protocol, exchanges messages with the agent located at each physical node. Once new virtual nodes are created, they are managed by the SP they belong to. In this respect, a *service manager* communicates with the agents associated with each virtual node to collect information and enforce management actions. SPs can lease resources from different InPs to build their virtual networks. These InPs can be located in different administrative domains (or autonomous systems) and, thus, some level of coordination among different infrastructure managers is required.

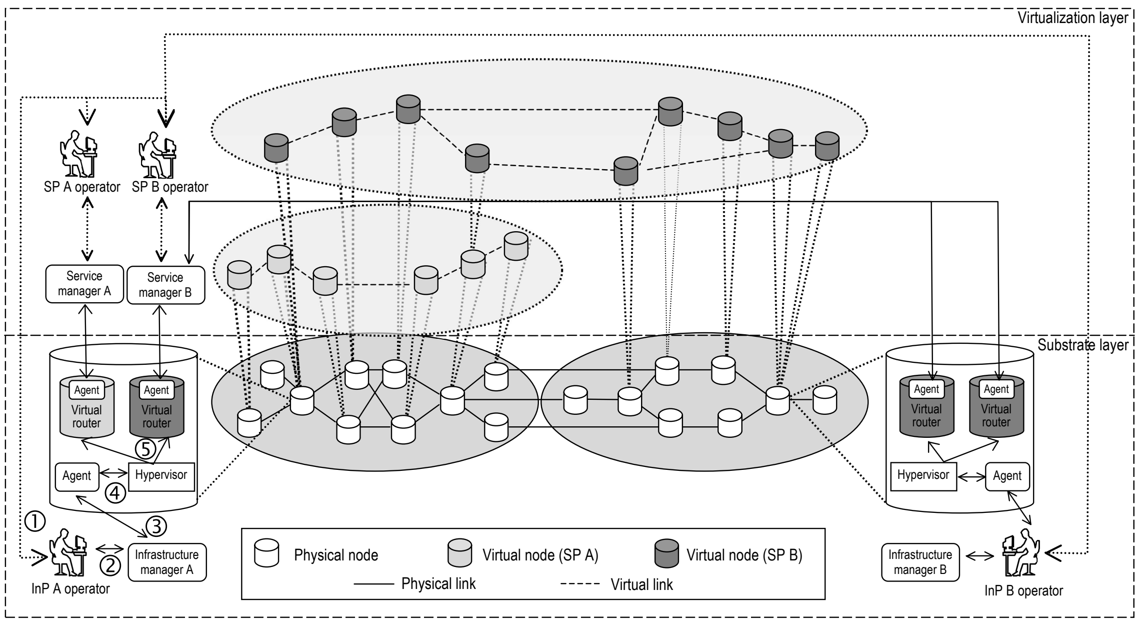


Figure 2.2: Management of NVEs

A virtual node is hosted on a physical one. Virtual node placement can be done either manually by the SP operator or automatically by the InP using an *embedding* algorithm. Once the physical node is selected, the SP operator requests a virtual node creation to the InP operator (*step 1* in Figure 2.2) that, using the infrastructure manager, instantiates the requested node (*step 2* in Figure 2.2). Each physical node has a hypervisor that allows the creation of virtual nodes. When the agent of the physical node receives a request from the infrastructure manager (*step 3* in Figure 2.2), it contacts the hypervisor (*step 4* in Figure 2.2), which then performs the requested action *i.e.*, virtual node creation in the physical node (*step 5* in Figure 2.2). In general, hypervisors provide APIs that allow external programs to call internal operations such as virtual node creation, initialization, removal, and to run scripts to perform fine-grained configuration of the virtual resources.

Similar to virtual nodes, virtual links are created through agents located at physical network nodes. Before contacting the infrastructure manager, the SP operator specifies the desired characteristics of the virtual links. The characteristics of a virtual link include source node, destination node, and bandwidth. Accordingly, the infrastructure manager contacts the agents at the physical nodes hosting the source and destination virtual nodes to create the virtual link. Virtual links can be created by configuring Ethernet VLANs between the physical nodes hosting the virtual ones. MPLS LSPs and GRE tunnels are other candidates to establish virtual links. To complete the creation of a virtual link, virtual network interfaces belonging to source and destination virtual nodes need to be bound to their respective physical network interfaces.

The "owner" of a virtual network is usually a human operator or an entity different from the owner of the physical substrate. Isolated management views have to be provided to different human operators at both substrate and virtualization layers. The isolation at the management plane is dependent on the isolation at the data plane and control plane, which is provided by enabling technologies such as VLAN, MPLS, and hypervisors. In the next subsection, we identify and discuss the criteria used to categorize NVE management solutions.

2.2.1 Characteristics of Management Solutions for NVEs

We classify NVE management projects in terms of management targets, management functions, and management approaches, which combined provide a holistic understanding on how management is currently tackled in modern NVEs (ESTEVES; GRANVILLE; BOUTABA, 2013). These criteria are commonly used to organize network management problems in general and are applicable to NVEs as well (CLEMM, 2006) (IFIP, 2011) (ESTEVES; GRANVILLE; BOUTABA, 2013). Although other criteria (*e.g.*, management lifecycle, management organization (CLEMM, 2006)) could be used to classify the projects, we focused on the ones that are most significant from the technical point of view since they lie in the core of any NVE management system and are critical at this stage of virtual networks design.

2.2.1.1 Management Targets

A management target refers to a managed component of an NVE. Managed components can belong to different layers of an NVE (*i.e.*, physical, virtual, application). Single targets can be combined into more complex ones (*e.g.*, virtual networks) demanding additional management efforts. Here, we classify management targets in node management, link management, and network management, as described below.

- **Node management:** node management deals with the operation of virtual and physical nodes of an NVE, including the initial creation of virtual nodes on the substrate and node migration.

- **Link management:** link management addresses specific aspects related to the configuration and operation of physical and virtual links, such as virtual link isolation and flow scheduling.

- **Network management:** network management encompasses not only a single node or link of the NVE, but an entire virtual network, including virtual networks that span multiple physical networks.

Structuring management activities according to their target (*e.g.*, node, link, and internetwork) can help NVE operators to effectively identify and delegate management tasks (*e.g*, providing isolation among multiple virtual links) based on their target. Next, we enumerate the main management functions that must be supported to realize the virtualization management model presented in Section 2.2.

2.2.1.2 Management Functions

To discuss how network management has been tackled by network virtualization projects, we identify here the main management functions that are essential in any NVE management solution. These functions (*i.e.*, provisioning, monitoring, and interfacing) are already key in traditional networks, but gain more importance in NVEs. Provisioning allows SPs to instantiate and use virtual networks. Monitoring is used to support several other management tasks, such as fault management and billing. Interfacing defines how management applications communicate with NVE resources and enabling interoperability.

- **Provisioning:** resource provisioning in the context of network virtualization consists in defining the mapping of virtual network resources (*e.g.,* nodes, links) to their physical counterparts and giving SPs operators access to their virtual networks.

- **Monitoring:** monitoring large NVEs involves gathering updated status of physical resources and their associated virtual networks. Filtering, correlation, aggregation, and compressing of monitoring information from different sources are required to reduce management overhead.

- **Interfacing:** appropriate management interfaces are required for InPs and SPs to respectively access, operate, maintain, and administer the physical and virtual nodes and links. Physical network devices must present a uniform management interface allowing virtual nodes and links located on heterogeneous physical nodes and links to be part of the same virtual network and easily manageable by the SP. The functionalities that a management interface must support include: registration, creation, removal, copy, initialization, shutdown, and migration of virtual nodes and links; configuration of individual attributes of virtual resources, such as CPU and memory capacity of virtual nodes, bandwidth of virtual links, and routing tables; retrieval of status variables; and notifications support.

Other management functions such as billing and maintenance are also important and needed for the overall management of NVEs. However, we focus on basic management tasks (*i.e.*, provisioning and monitoring) required to enable any NVE. In the following, we discuss the main management approaches employed in NVEs.

2.2.1.3 Management Approaches

Management solutions in NVEs vary in how managers and agents are organized. Some solutions rely on a centralized node responsible for performing all management tasks, while other systems allow multiple distributed nodes to share the task of managing the infrastructure. In contrast to manual network management, *i.e.*, a human operator is responsible for all management tasks, management systems employed in NVEs can also have different levels of automation. Autonomic management helps reducing human intervention and allows dynamic adaptation to changes in the network. Policy-based management is used to enable autonomic management and assists InP administrators to handle the inherent complexity of an NVE by automating resource configuration according to high-level business goals. These approaches are discussed next.

- **Centralized and distributed management:** in the centralized NVE management approach, a single management station located at the InP (respectively the SP) is responsible for overseeing the management of the InP (respectively SP) network resources. On the other hand, in distributed NVE management, multiple nodes work in a cooperative fashion to accomplish management tasks.

- **Manual and autonomic/policy-based management:** autonomic management allows the NVE to manage itself according to the current state of the network as opposed to have a human administrator performing all management tasks. Autonomic management solutions typically rely on high-level policies which are general rules defined to govern the functioning of the underlying network devices. In NVEs, policies are also used by InPs to enforce isolation among virtual networks by controlling access permissions for each SP.

Understanding such management approaches can help NVE administrators to evaluate the tradeoff between the size of the NVE and the complexity of the solution required to manage it. In the next section, we discuss representative projects related to the management of NVEs according to the presented criteria.

2.2.2 Network Virtualization Projects

In this subsection, we have selected twelve projects representative of recent work on network virtualization at the basic research, applied research and testbed deployment levels. The surveyed projects are among the first approaches to consider management as a first-class requirement in their solutions. They also represent consolidated efforts at advanced (or already completed) development stages. In the following, we discuss these projects based on the criteria identified in the previous section.

2.2.2.1 4WARD (VNet)

The FP7 4WARD project (CORREIA et al., 2011) defines a network virtualization framework called *VNet* designed to manage multiple virtual networks hosted on a shared infrastructure. VNet proposes a business model composed of: infrastructure providers (InP) that own the physical resources, the virtual network provider (VNP) responsible for creating virtual networks from multiple InPs, the and virtual network operator (VNO) that manages virtual networks. Figure 2.3 depicts the 4WARD management model along with the relationships among participating entities.

The VNO selects the VNP who actually performs VN provisioning. The VNP provides two special nodes: the Provisioning Node and the Out-of-VNet Management Node that are also present in each InP. The Provisioning Node gathers updated information of the resources and runs embedding algorithms, while the Out-of-VNet Management Node offers a management interface that VNO uses to access virtual networks.

Resource provisioning in VNet includes discovery, embedding, and instantiation. In the discovery phase, the VNet provider generates a list of candidate resources to host the virtual network. The InP uses this list to select the physical resources that will host the virtual ones in the embedding phase. The embedding process employs a greedy algorithm to define the mapping of virtual resources to the physical network. The embedding process consists of two distributed algorithms. The first one is responsible for creating virtual networks while the second one is aimed to handle failures and ensure SLAs. Both algorithms rely on a multi-agent framework where autonomous agents placed at physical nodes interact to achieve the embedding. The instantiation phase consists in actually reserving the selected virtual resources.

VNet agents (or probes) are placed at the physical nodes to provide updated information about physical and virtual resources to the InP. The collected information is used for different purposes, including resource discovery and self-organization of the virtual networks. VNet relies on a *situation awareness framework* that aggregates monitoring information and hides unnecessary details to ensure scalability and efficiency of the monitoring process. VNet offers a management interface based on XML-RPC called VMI

(Virtualisation Management Interface) that defines a set of management operations, including creation, termination, and concatenation of virtual resources. VNet implements a distributed and autonomic management approach, referred to as *In-Network Management (INM)*. In INM, self-managing entities (SEs) embedded inside the network are responsible for the autonomic operation of the physical infrastructure.

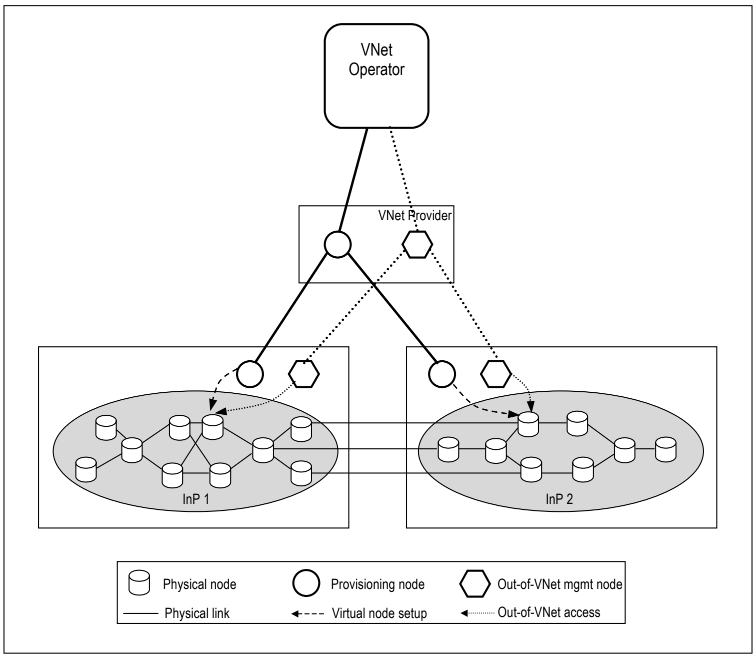


Figure 2.3: 4WARD VNet provisioning scenario

2.2.2.2 AUTOI

The AUTOI (*Autonomic Internet*) initiative (GALIS et al., 2009) aims to develop autonomic management solutions for Future Internet. AUTOI is composed of five conceptual planes: orchestration, service enablers, knowledge, management and virtualization. The orchestration plane governs and controls the overall environment. The service enablers plane provides a programmable environment for the rapid and controlled deployment of new management services. The knowledge plane consists of information models and ontologies used to support autonomic capabilities. Management plane functions (*e.g.,* monitoring, adaptation) are performed by distributed Autonomic Management Systems (AMSs). Different AMSs can cooperate with one another in order to build end-to-end services. The AUTOI virtualization plane is responsible for the provisioning and operation of virtual networks.

Lattice (CLAYMAN; GALIS; MAMATAS, 2010) is the monitoring framework for AUTOI. Lattice was conceived to be used in the monitoring of complex and dynamic virtual environments. Lattice defines the basic building blocks of a monitoring solution for virtual networks. The monitoring system is based on the general concept of *producers* and *consumers* of monitoring data. Monitoring *probes* are responsible for gathering updated information from virtual and physical resources. *Data Sources*, on the other hand, group monitored information collected by probes and send it to interested consumers following a previously defined communication model, such as publish/subscribe or IP multicast. Lattice is designed to be a generic framework allowing customized, specific-purpose probes, consumers, and data sources to be deployed.

Each AUTOI domain is managed by one AMS running a control loop. AMSs can cooperate with one another in order to build end-to-end services. AMSs interact with the virtualization plane through two well-defined interfaces: vSPI (Virtualization System Programmability Interface) and vCPI (Virtualization Component Programmability Interface). vSPI provides a macro view of the virtual resources to the AUTOI orchestration plane, which, in turn, uses vCPI to build and manage virtual networks. vCPI defines basic primitives for management of virtual nodes (registerVM, startVM, shutdownVM, migrateVM, unregisterVM) and virtual links (instantiateLink, removeLink, modifyLink).

AMSs are also involved in self-fault management, self-configuring management, and self-performance management tasks. Both self-fault management and self-performance management rely on the orchestration plane to detect and react to QoS degradation. Self-configuration management is related to the setup and adaptation of virtual infrastructures and service deployment, and involves all AUTOI planes. Authorization, authentication, and accounting are performed at the Service Enablers plane in order to allow end-users to access AUTOI services.

2.2.2.3 FEDERICA

FEDERICA (*Federated E-infrastructure Dedicated to European Researchers Innovating in Computing Network Architectures*) (SZEGEDI et al., 2009) focuses on building a large scale networking infrastructure to enable experimentation of new Internet protocols and architectures.

FEDERICA assumes that a centralized NOC (Network Operation Center) entity performs all administrative management actions in the infrastructure, such as resource discovery, provisioning, and user control. In addition, FEDERICA offers a slice management tool to facilitate resource management. This later allows the NOC operator to create slices (*i.e.*, aggregation of virtual network resources), add virtual resources to a slice, and export slices to the SPs. SPs, in turn, can perform configurations on their assigned slices without affecting other SPs. Monitoring of physical nodes in FEDERICA is performed mainly through SNMP. FEDERICA relies on the VMWare Remote Command Line Interface (RCLI) to monitor virtual nodes.

In FEDERICA, when a researcher requests a slice, she/he contacts the FEDERICA NOC which creates appropriate (*i.e.*, public, private, management) interfaces on a virtual server to allow users to access her/his slices. The NOC then creates credentials and sets the expiration date and time of the slice. The NOC also defines the mapping of the slice on the corresponding physical machines and links and creates VLANs to complete the slice creation.

2.2.2.4 ProtoGENI

ProtoGENI (PROTOGENI, 2011) is a deployed prototype of GENI (Global Environment for Network Innovations) (GENI, 2009). In ProtoGENI, researchers can create *slices* composed of *slivers*. Slivers are instances of virtual computing and networking resources. The main management entities in ProtoGENI are the *clearinghouse*, *slice authorities*, and *component managers*. The clearinghouse is the central management point in ProtoGENI, responsible for registering and tracking all slices, users, and component managers, and enabling the exchange of root certificates between ProtoGENI members. Slice authorities are the entry points for researchers to request slices from several component managers belonging to the participants of the ProtoGENI federation. Component managers control resource provisioning inside a member of the ProtoGENI federation. Figure 2.4 illustrates the main entities of ProtoGENI.

To obtain a slice, a researcher needs to register it at the level of a slice authority and get a corresponding *credential*. The credential allows one to create slivers using component managers belonging to the ProtoGENI federation. Then, the researcher contacts and requests *tickets* from component managers. Tickets are special credentials guaranteeing that requested resources will be bound to a given slice. Both slice authorities and component managers implements an XML-RPC server and provide APIs for managing slices and slivers, respectively.

Resources in ProtoGENI are described through an abstraction called *RSpec*. A RSpec is an XML document describing the components that researchers can use in terms of the available resources and associated constraints. There are three main types of RSpecs: advertisements, requests, and manifests. Advertisement RSpecs list available resources of a component manager. Request RSpecs specify the resources a researcher requested from one or more component managers to build his/her slices. Manifest RSPecs provide additional information about the slivers currently allocated to a researcher, such as dynamically assigned IP addresses, hostnames, and configuration options.

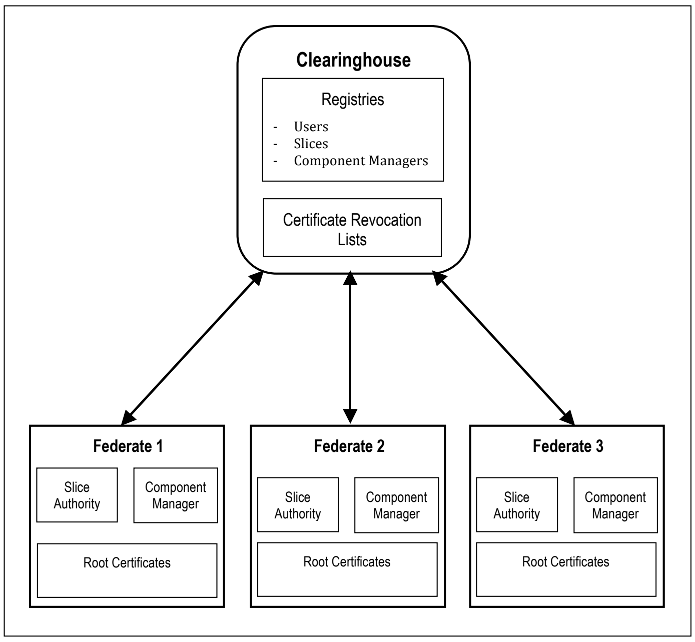


Figure 2.4: ProtoGENI main entities

2.2.2.5 UCLP

UCLP (*User Controlled Lightpaths*) (BOUTABA; GOLAB; IRAQI, 2004) is a management system for provisioning and controlling optical networks across multiple domains. UCLP is based on a service-oriented architecture (SOA) and allows end-users to establish inter-domain *lightpaths* on-demand. Lightpaths can be created, destroyed, advertised, leased, and concatenated using distributed *Web services*. UCLP is structured in three main layers: user access layer, service provisioning layer, and resource management layer, illustrated in Figure 2.5.

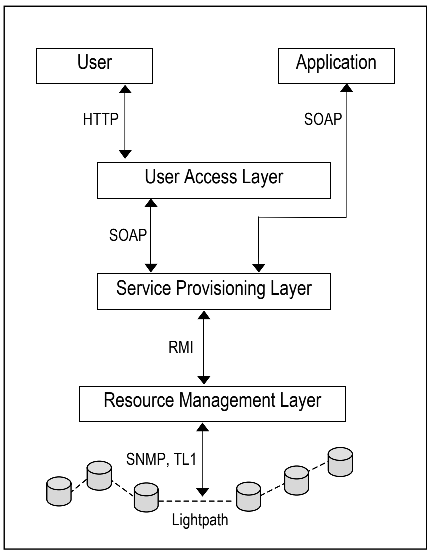


Figure 2.5: UCLP

The user access layer is the entry point from where human users can request and manage lightpath objects through a Web interface. Lightpath operations are implemented by a set of services defined in the service provisioning layer that also acts as an access point for external applications (*e.g.*, Grid). The resource management layer comprises a set of resource agents, responsible for communicating with technology-specific physical devices (*e.g.,* SONET/SDH switches). Monitoring in UCLP is mainly performed through standard SNMP.

UCLP allows SPs to configure their resources (Web services) in an easy and flexible way because SPs do not need to know the internal details of managed devices nor they need to contact the InP for every management task (*e.g.*, provisioning). SPs have full autonomy to perform modifications on lightpaths.

2.2.2.6 VNARMS

VNARMS (*Virtual Network-based Autonomic network Resource control and Management System*) (KIM; LEON-GARCIA, 2007) relies on autonomic management to build virtual networks with performance guarantees. In each virtual network, there are two basic entities: the *Virtual Network Resource Manager (VNRM)*, responsible for managing the virtual network by controlling a set of distributed *Resource Agents (RAs)* that communicate with individual network elements. Both entities are autonomic and monitor the managed resources to identify problems and react accordingly.

VNARMS uses the concept of *root-VN* to abstract the physical network and create virtual networks. The root-VN can be spawned in multiple *child-VNs* that satisfy specific QoS requirements. When an SP requests a new virtual network, the VNRM of the root-VN calculates a topology based on the SLA requirements of the SP, spawns a child-VN from the root-VN, and instantiates a new VNRM for the child-VN. The RA of the root-VN also creates new RAs to manage individual virtual resources of the child-VN. Second-level virtual networks can be provisioned from a previously created child-VN in a recursive way as illustrated in Figure 2.6. VNARMS relies on DiffServ for QoS enforcement.

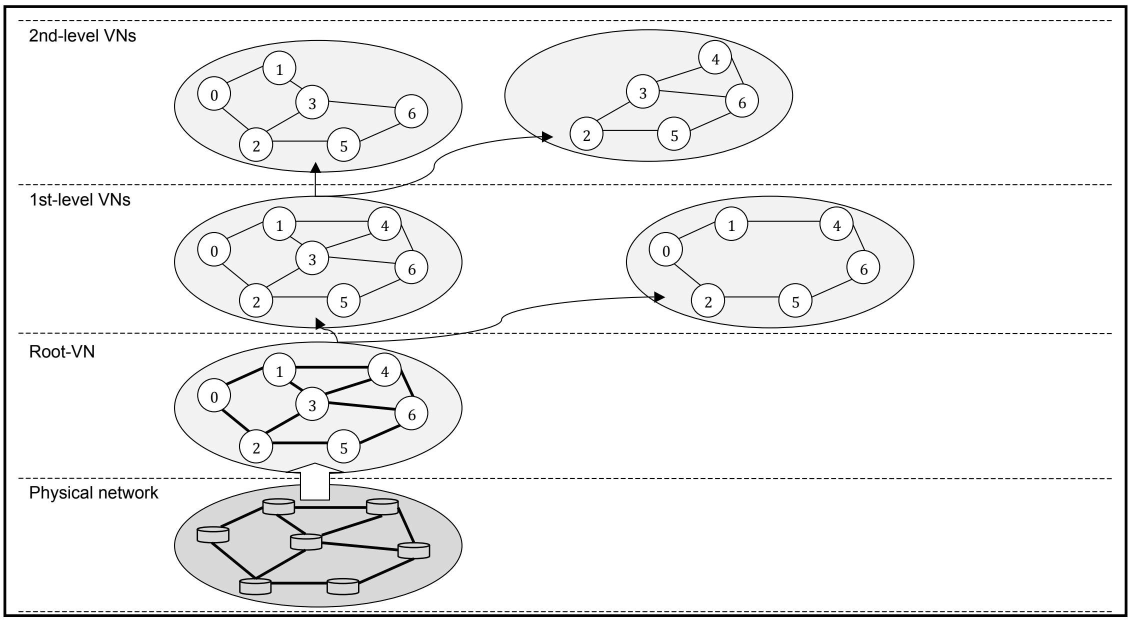


Figure 2.6: VNARMS

2.2.2.7 *OpenFlow/FlowVisor*

OpenFlow (MCKEOWN et al., 2008) is an abstraction layer that enables programming network switches. That is achieved through a flow-based abstraction in which the user/application determines the actions that will be performed by the switch on receiving packets belonging to a specific flow type.

OpenFlow requires a virtualization layer to allow multiple users or applications to share a switch. To this end, the FlowVisor (SHERWOOD et al., 2010) virtualization layer has been introduced. With FlowVisor, a switch can be properly *sliced* and allocated to different users. One of the main issues that FlowVisor has to deal with is managing isolation among multiple slices. FlowVisor achieves isolation through a series of mechanisms. For bandwidth isolation, FlowVisor configures minimum bandwidth queues for each slice sharing a port of a switch. To deal with CPU isolation, FlowVisor limits the number of control messages that a user can send. Other isolation mechanisms include limiting the number of entries in the flow tables for each slice, and the re-writing of control messages originated at a particular slice to prevent conflicts with other slices. OpenFlow-based switches are typically managed by a centralized controller used to create, remove, and modify flow entries.

2.2.2.8 *SNMP for virtual router management*

The Virtual Router MIB (VR-MIB) was proposed by IETF in 2003 (STELZER et al., 2003) as a first attempt to provide an SNMP management interface for virtual routers. Although already abandoned, this MIB module was originally designed for L3VPN management. The VR-MIB contains variables related to the high-level configuration of virtual routers, organized in three main tables: the `vrConfigTable`, responsible for the creation and removal of virtual routers; the `vrStatTable`, which stores statistics of the virtual routers that are hosted in a physical device; and the `vrIfConfigTable` that deals with interface mapping between a virtual router and a physical one. This solution assumes that a single SNMP agent is required to manage all the virtual routers located in a Provider Edge (PE). This is achieved by the use of *SNMP contexts* that determine which virtual router is being accessed, as depicted in Figure 2.7.

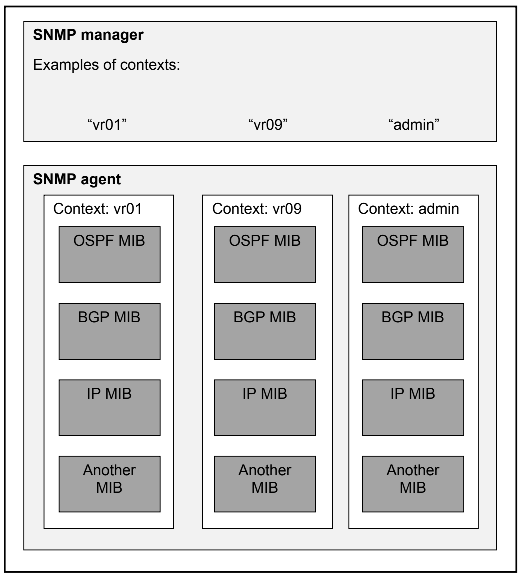


Figure 2.7: SNMP manager and agent using VR-MIB

One drawback of this MIB module is the absence of objects to support proper binding between virtual and physical network interfaces. The binding operation is essential, for example, to limit the physical interfaces to which a virtual one can be mapped to. The operator of the physical router may need to limit the mapping between physical and virtual network interfaces to enforce isolation or to provide performance guarantees.

To overcome the limitations of the Virtual Router MIB (VR-MIB), Daitx *et al.* (DAITX; ESTEVES; GRANVILLE, 2011) propose an extension to the VR-MIB module that allow flexible interface bindings. The extension was evaluated for two virtualization platforms: VMware and XenServer to investigate the virtualization platform influence in the SNMP performance. However, the evaluation found out that SNMP performance largely depends on the virtualization platform that is being used. Given a virtualization platform, additional MIB objects may be required in an SNMP implementation, which also results in different performance levels. Moreover, for large virtualization scenarios, SNMP is not feasible because of the high number of objects that need to be managed.

Another MIB was proposed to the VRRP (*Virtual Router Redundancy Protocol*) protocol (HINDEN, 2004) that uses virtual routers acting as backup routers to reduce the negative impact of failures. The VRRP-MIB (JEWELL, 2000) defines a set of managed objects for the VRRP protocol. The VRRP-MIB is structured in three main groups:

- **Operations**: objects related to configuration and control of virtual routers.
- **Statistics**: objects containing statistics of virtual routers.
- **Notifications**: objects used to enable notifications sent from devices containing virtual routers.

2.2.2.9 V-Mart

V-Mart (ZAHEER; XIAO; BOUTABA, 2010) is an automated framework for service negotiation and contracting in NVEs. V-Mart assumes that large virtual networks are likely to be deployed over several InPs. However, SPs are not aware of possible collaborations between multiple InPs that can ultimately influence the total cost of requested VNs.

Given that multiple InPs can lease resources at varying prices. An SP should be able to choose virtual network resources from different InPs in order to reduce the total cost to deploy a virtual network. V-Mart is based on a two-stage auctioning model and on a virtual network partitioning heuristic. The objective is to minimize costs for SPs and to stimulate a fair competition among multiple InPs.

In V-Mart, the SP sends a Request for Quotation (RFQ) to all interested InPs containing the VN requirements (*e.g.*, location, CPU capacities, bandwidth of virtual links). Then, each InP runs an embedding algorithm and indicates which virtual resources it can host along with their corresponding prices. Next, having the price quotes of all virtual resources from all InP, the SP runs a VN partitioning algorithm to generate a list of VN segments and sends these VN segments to all InPs. Each InP makes a final bid for each VN segment and the SP defines the winner InP for all VN segments. Finally, the SPs contracts all winner InPs to actually build the VN.

2.2.2.10 UCVS

UCVS (CHERKAOUI; HALIMA, 2008) is a framework for resource control in network virtualization. UCVS operates exposing virtual network resources as virtual objects that can be combined to build up entire virtual networks. Virtual objects can belong to different domains and can be mapped to Web services. Heterogeneity at the physical layer is then tackled when physical resources are mapped to Web services. Workflows are used to orchestrate Web services and combine them into more complex virtual networks.

In the UCVS model, a virtual network is called an Articulated Private Network (APN), shown in Figure 2.8, formed by the concatenation of multiple individual services that satisfy SP requirements. UCVS uses a repository called Virtual Network Service Object Registry that contains a list of virtual objects and their corresponding properties.

UCVS provides operations to be performed over the virtual objects, such as sharing and concatenation. Sharing allows a virtual object to be partitioned in smaller objects and published for other users. Concatenation consists in combining two objects in a single one. For example, two objects that represent adjacent virtual links can be combined into a single virtual link.

2.2.2.11 VROOM

Virtual ROuters On the Move (VROOM) (WANG et al., 2008) enables live virtual router migrations between different physical locations. VROOM can act as a network management tool for transparent migration of virtual routers and links.

As can be observed in Figure 2.9, in VROOM, a *hypervisor* is responsible, among other functions, for cloning control and data planes of a specific virtual router in order to enable the migration to another location with minor interruptions. It also supports dynamic interface bindings to help in this migration process. VROOM is used in network management tasks, such as planned maintenance (where routers can be migrated to another location while a physical router is under maintenance) and reduction of energy

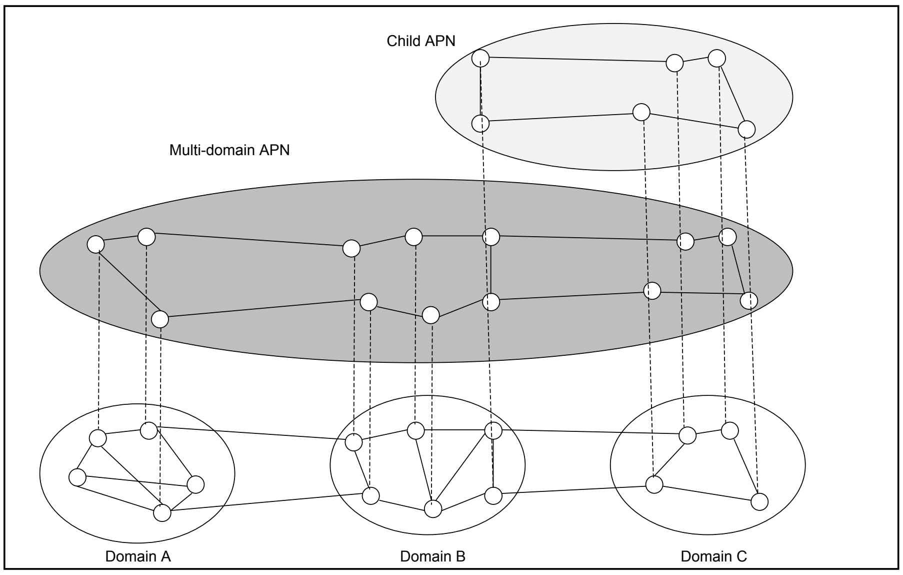


Figure 2.8: UCVS

consumption in periods of low network demand.

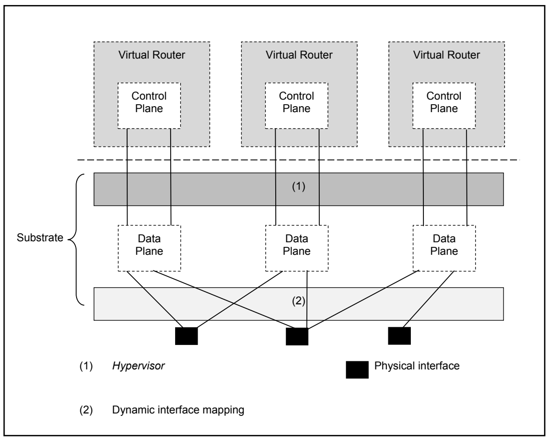


Figure 2.9: VROOM

The main difference between VROOM and the other proposals is that VROOM is used as a network management tool and is not the focus of the management itself.

2.2.2.12 MIBlets

MIBlets (NG et al., 1999) provides selective and isolated views of network resources for VN end-users. Special agents inside a purpose node, called *MIBlet controllers*, partition network resources (*e.g.*, ports, bandwidth) among multiple end-users. Therefore, the management systems of end-users, called CNRMS (Customer Network Resources Management System) have access only to a subset of the physical resources (*i.e.*, the resources allocated to the end-user). MIB controllers are isolated from other MIBlet controllers sharing the same physical node. As a consequence of such isolation, management actions performed by one CNRMS do not affect other CNRMSs sharing the same device.

MIBlets are managed using six well-defined messages: *creation-request*, *creation-response*, *re-configuration-request*, *re-configuration-response*, *termination-request*, *termination-response*. The MIBlet creation-request message specifies the parameters of the request such as a port number and an amount of bandwidth. The MIBlet creation-response message returns a *success indication* if the request is accepted. A MIBlet re-configuration-request message is issued when the CNRMS needs to modify an existing MIBlet. For example, the CNRMS can increase the bandwidth allocated to its MIBlets. The termination-request message is used to release an allocated message.

The communication between CNRMS and MIBlet controllers is based on SNMP. Therefore, the CNRMS can directly access and configure the MIBlet as it is accessing the actual MIB definition of the physical device.

Table 2.1: Comparison of Virtualization Management Proposals

Characteristic	Management target			Management function			Management approach			Management protocol
Project/Proposal	Node	Link	Network	Provisioning	Monitoring	Interfacing	Centralized	Distributed	Autonomic/policies	Protocol
4WARD	✓	✓	✓	✓	✓	✓		✓	✓	XML-RPC
AUTOI	✓	✓	✓	✓	✓	✓		✓	✓	vSPI/vPCI
FEDERICA	✓	✓	✓	✓	✓	✓	✓			SNMP (monitoring)
ProtoGENI	✓	✓	✓	✓	✓	✓	✓			XML-RPC
UCLP		✓	✓	✓		✓		✓		SNMP (monitoring)
VNARMS	✓	✓	✓	✓				✓	✓	N/A
OpenFlow/FlowVisor	✓				✓		✓			OFP
VR-MIB	✓			✓	✓	✓	✓			SNMP
V-Mart	✓	✓	✓	✓				✓		N/A
UCVS	✓	✓	✓	✓		✓		✓		N/A
VROOM	✓	✓	✓	✓				✓		N/A
Miblets	✓	✓		✓	✓	✓		✓		SNMP

2.2.2.13 Comparison

A summary of the discussion is provided in Table 2.1. A checked cell means that the referred aspect (*i.e.*, target, function, or approach) is explicitly tackled by the proposal. An unchecked cell does not mean that the aspect is not considered at all. It just means that it was not possible to find sufficient information to state otherwise. The last column refers to the management protocol used in provisioning or monitoring-related functions. Again, in the case of N/A (not applicable), there was not enough information to define precisely which protocol is used. Comparing the surveyed proposals (see Table 2.1), we found that 4WARD and AUTOI cover most of the criteria we have identified, reflecting the goal of these projects to consider management at the design stage. The other proposals represent significant achievements in the area and emphasize the trend of considering management as a first class requirement in future networks design. Another noteworthy finding is that distributed and, to a lesser extent, autonomic management are considered in many solutions, except perhaps from OpenFlow-based architectures, which reflects the paradigm shift also occurring in traditional network management design, even though some functions (*e.g.*, registry) are still performed by centralized entities.

2.3 Research Opportunities

Research on management of network virtualization is still in its infancy. There are management issues still uncovered and others that need further investigation. In this section, we discuss the requirements we consider important for managing NVEs but not sufficiently explored by current solutions, and which, in our opinion, reflect future trends in managing modern NVEs. Also, considering the recent developments in cloud computing and software-defined networks (SDNs), we dedicate special attention to the issues arising in the management of virtualized cloud computing environments and SDNs.

2.3.1 Federations and SLA negotiations

The federation of virtualized infrastructures from multiple InPs enables access to larger scale infrastructures. This is already happening with virtualized network testbeds allowing researchers to conduct realistic network experiments at scale, which would not have been possible otherwise. ProtoGENI is an example of federation that allows cooperation among multiple organizations. However, guaranteeing predictable performance for participating entities through SLA enforcement has not been properly addressed by current solutions and remains an open issue.

2.3.2 Interplay between InP management and SP management

This refers to the needed cooperation between the management systems of InPs and SPs to avoid/resolve conflicts and ensure overall system stability. Indeed, InPs and SPs often have conflicting management goals whereas the InPs want to maximize the utilization of their infrastructures and hence their revenue and the SPs want predictable performance for their virtual networks.

2.3.3 Standard management protocols and information models

The VR-MIB module (STELZER et al., 2003) described a set of SNMP management variables for the management of physical routers with virtualization support. However, it did not progress in the IETF standardization track, leaving the area with no SNMP-based solution. Other existing management protocols can be used instead. NETCONF, for example, would be more appropriate for configuration aspects, while NetFlow could be expanded for virtual router monitoring. There is a clear lack in this area today, which represent an interesting opportunity for research and standardization.

2.3.4 Management of Virtualized Cloud Computing Environments

Virtualization is a key enabling technology of cloud computing. In order to support a large number of customers (a.k.a. tenants), modern cloud infrastructures require that every resource (*e.g.*, computing, storage, network) is virtualized. Open-source IaaS platforms such as OpenStack and CloudStack represent noteworthy developments in this respect by facilitating the development of private and hybrid clouds supporting multi-tenancy and advanced management capabilities. Nevertheless, several management challenges are still open, some of which are discussed below.

- **Dynamic resource scaling:** this refers to the ability of dynamically modifying a previous resource allocation to satisfy new SP objectives. For example, an SP may need to add more virtual nodes or to increase the bandwidth of a virtual link to accommodate an increasing customer (end-users) base. Cloud management systems must provide this *elasticity* in order to allow rapid adaptation to changes in SPs's demands. Current solutions (*e.g.* Amazon EC2) provide elasticity at the virtual machine level. However, dynamic capacity adjustment of network resources (*e.g.*, bandwidth) requires further investigation.

- **Application-aware resource provisioning:** the main limitation of current resource allocation schemes in clouds is that the characteristics of the applications are commonly ignored (ESTEVES et al., 2013). For example, business-critical applications (*e.g.*, ticket reservation, order processing) may require that virtual (service) nodes are replicated and placed on distinct physical servers. On the other hand, delay-sensitive applications benefit if virtual (service) nodes are placed in edge data centers (*i.e.*, physically close to end users) in order to reduce response time (SAVI, 2011). Adaptive, application-driven resource provisioning can allow multiple tenants and a large diversity of applications to efficiently share cloud infrastructures.
- **Energy management:** energy is a main concern in cloud data centers and accounting for a significant portion of the operational costs of the InP. Achieving energy proportionality in data centers by consolidating virtual resources into a small number of physical devices can alleviate the problem. In this respect, finding a good tradeoff between energy consumption and applications' performance is a promising research direction.
- **Data center network management:** important network issues in data center network management include address configuration, traffic management, and flow scheduling. In modern cloud data centers, the identifier of a resource is decoupled from its physical location, which requires a management infrastructure to efficiently maintain ID/Locator mappings. Also, dealing with different flow patterns typically found in data centers (short vs. long flows), flow scheduling, bandwidth allocation, and leveraging the inherent path diversity of data center networks are important challenges.

2.3.5 Management of SDNs

Software Defined Networking has recently become extremely popular as a means to program network devices and customize their behavior. In SDN, the control and forwarding functions of a networking device are decoupled. This separation of control and data planes and their implementation in software offer flexibility in controlling how network devices forward packets. Commonly, SDN architectures rely on a virtualization layer which abstracts the underlying physical network devices and topology and provides isolation in shared environments. The virtual resources are seamlessly controlled and orchestrated for the efficient delivery of network services. Management in this dynamic environment is of paramount importance. Some of the management issues that need to be addressed include:

- **Management Abstractions:** current SDN solutions require network operators to develop customized management packages using low-level instructions of a network operating system (*e.g.*, NOX), which may be a hurdle for administrators. Providing adequate management information models, interfaces and protocols and advanced monitoring capabilities/tools represent opportunities to facilitate management of SDNs. The OMNI system (MATTOS et al., 2011) is one attempt in this direction.

- **Interoperability and Management API:** SDNs can be deployed using virtualized forwarding resources from different providers using a variety of network operating systems and implementations. The provisioning of services end-to-end and across multiple administrative domains stresses the need for a widely accepted management API.

2.4 Summary

This chapter presented an overview of the state of the art regarding network virtualization and management of network virtualization environments. Network virtualization has recently gained significant importance as a viable platform enabling the development of novel solutions to known structural problems in the Internet. However, the management of virtualized network environments raise a number of challenges yet to be addressed. This chapter surveyed a set of representative research projects related to network virtualization management. The surveyed projects have been analyzed from different perspectives, including their management targets, management functions, and management approach. We found that some management aspects have received or currently receiving more attention than others. For example, resource allocation and monitoring have been extensively addressed in existing projects. Other aspects have been less so such as autonomic/policy-based management, management federations, SLA management, dedicated management information models and protocols, standard management APIs, and cooperative management in a multi-tenant, multi-provider environment. The research issues discussed in this chapter are by no means exhaustive, and will be complemented by others, as the research in this area progresses. In general, we believe that the network virtualization community should take advantage of the developments made by the network management community over the last three decades. In turn, we believe that the network management community should embrace this emerging area and leverage its expertise to develop a management plane for virtualized environments. Virtualized clouds and SDNs are example of such environments calling for novel management solutions. In the next chapter, the application-aware virtual network provisioning problem (AVNP) is formulated and the concept of allocation paradigms, which aims to enable flexible and adaptive VN provisioning to tackle the AVNP problem is introduced.

3 APPLICATION-AWARE ADAPTIVE VIRTUAL NETWORK PROVISIONING BASED ON ALLOCATION PARADIGMS

Provisioning is an essential management function in NVEs (Section 2.2.1.2) and one related open challenge is how to adapt provisioning strategies to reflect changes in InP objectives and to consider SP/application requirements (Section 2.3.4). In this chapter, we propose and detail the concept of allocation paradigms to solve the application-aware adaptive virtual network provisioning problem (AVNP). We start by providing a brief discussion on virtual network provisioning along with some related efforts. Next, we introduce and describe our solution, named allocation paradigms, to tackle the AVNP problem. Then, we define a methodology to quantify the efficiency of allocation paradigms in terms of provisioning quality.

3.1 Virtual Network Provisioning

VN provisioning is one key NVE management task that defines how InPs allocate VNs to SPs. Typically, VN provisioning consists in three main steps: discovery, embedding, and instantiation (CORREIA et al., 2011). In the discovery phase, the InP generates a list of candidate resources to host the virtual network. The list can be based on different criteria, such as geographical location, type of the requested resources (*e.g.*, a virtual router, a slice of an OpenFlow switch, virtual machine), or the capacity of the requested resources in terms of CPU in the case of virtual nodes or bandwidth in the case of virtual links. The embedding phase consists of mapping virtual resources (*i.e.*, virtual routers, virtual switches, virtual machines, virtual links) to the substrate, *i.e.*, the physical network. The InP runs an embedding algorithm to find the best mapping according to a set of objectives defined by the InP operator and/or by the SPs, such as high acceptance ratio and low communication cost. In most cases, discovery and embedding are performed simultaneously. It is also possible to allow the SP operator to explicitly select virtual resources that are already mapped. In this case, there is no embedding algorithm. Examples of management solutions that allow such flexibility include UCLP (BOUTABA; GOLAB; IRAQI, 2004) for virtual links or UCVS (CHERKAOUI; HALIMA, 2008) for entire VNs. However, this user-controlled allocation may not be the optimal allocation and, consequently, result in low utilization and revenue loss. The virtual network embedding (VNE) problem is illustrated in Figure 3.1. Finally, the instantiation phase consists in actually reserving the selected virtual resources on the substrate using a management interface/protocol such as SNMP, XML-RPC, or CLI.

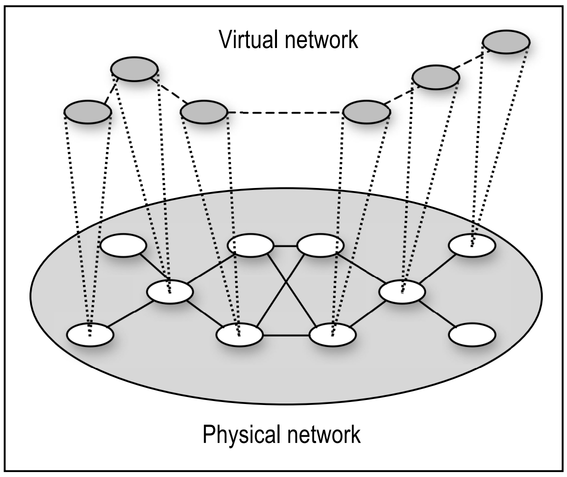


Figure 3.1: Virtual network embedding

3.2 Related Work

Several efforts related to VN provisioning have been proposed in recent years. In this section we highlight some relevant proposals and summarize them in Table 3.1.

Table 3.1: VN provisioning proposals

Characteristic	Management target			Management function			Management approach		
Proposal	Node	Link	Network	Provisioning	Monitoring	Interfacing	Centralized	Distributed	Autonomic/policies
Yu et al. (2008)	✓	✓		✓			✓		
ViNEYard (2012)	✓	✓		✓			✓		
Houidi et al. (2010)	✓	✓		✓	✓			✓	✓
NodeRank (2012)	✓	✓		✓			✓		
Houidi et al. (2011)	✓	✓	✓	✓			✓		
PolyViNE (2010)	✓	✓	✓	✓				✓	✓
VNet (2011)	✓	✓	✓	✓	✓			✓	✓
VINEA (2013)	✓	✓		✓	✓			✓	
SecondNet (2010)	✓	✓		✓			✓		
Oktopus (2011)	✓	✓		✓			✓		

Yu et al. attempts to improve virtual network provisioning by allowing virtual links to be mapped over multiple paths and periodically re-optimizing the mapping of virtual links on the substrate (YU et al., 2008). This feature is called *path splitting*. The motivation behind path splitting is to increase the chances of mapping a virtual link on the substrate even if there is not a corresponding physical link with available bandwidth capacity to host the virtual one. With path splitting, a virtual link can be spread over multiple physical ones that combined offer the requested bandwidth. A direct benefit of such flexible path splitting is an increase in the number of accepted VN requests. On the other hand, as VN requests arrive and depart the substrate the network can become fragmented. In order to improve resource utilization and maximize the acceptance ratio of future requests, virtual links are periodically adjusted by changing their splitting ratio or migrating them to other physical locations.

ViNEYard (CHOWDHURY; RAHMAN; BOUTABA, 2012) is a set of online embedding algorithms for virtual networks that coordinate the node and link mapping phases. The motivation behind this coordinated mapping relies on the fact that ignoring the relation between node and link mapping can lead to poor performance. In ViNEYard, the VNE problem is formulated as a mixed integer program that includes both node and link-related constraints. In order to be solved in polynomial time, the MIP formulation is transformed into a linear program through constraint relaxation. The main objective of ViNEYard is to minimize the cost of embedding a VN request to a physical substrate and, consequently, increase the InP revenue. One limitation of VINEYard is that is does not consider explicitly other types of virtual resources that can have specific requirements, such as ingress and egress bandwidth and storage capacity. Besides, it assumes that the substrate supports path splitting, which is not always the case.

Houidi et al. propose an adaptive framework for virtual network provisioning to maintain virtual network topologies after their initial allocation (HOUIDI et al., 2010). The goal is to allow already provisioned VNs to cope with variations in the substrate (*e.g.*, failures, changes in traffic patterns, modifications of a previously allocated VN), while guaranteeing the enforcement of SLAs. The authors propose a distributed fault-tolerant embedding algorithm following a multi-agent based approach. Autonomous agents are placed at the substrate nodes to offer *situation awareness* to the framework and support the distributed fault-tolerant embedding. These agents are responsible for monitoring, selecting new substrate resources, migration, and re-instantiation of VN. Each agent computes a dissimilarity metric between the non-functional (NF) attributes of a virtual node (*e.g.*, available capacity/bandwidth) and the NF attributes of all physical nodes and chooses the one that provides minimum dissimilarity.

NodeRank is a Markov random-walk model for computing resource ranking of a node and include the notion of topology-awareness in node mapping (CHENG et al., 2012). NodeRank is inspired by the Google's PageRank and sort nodes according to their quality, where a node is considered important if its connections forward to high capacity nodes. Authors propose a two-stage greedy algorithm. In the node mapping stage, virtual and physical nodes are sorted according to their rank. Virtual nodes with the highest rank are mapped to physical nodes with the highest rank. In the link mapping stage, virtual links are mapped to the physical ones using the shortest path algorithm. Since different embeddings result in different costs for the InP, authors use a Particle Swarm Optimization (PSO) to minimize embedding cost. In addition, the proposal does not assume that the substrate support path splitting.

Houidi et al. extends the provisioning problem to the case where VNs request are split across multiple InPs (HOUIDI et al., 2011). In such scenario, VN provisioning is divided in four main phases: resource matching, VN splitting, VN embedding, and resource binding. In the resource matching phase, the VN provider identifies a set of candidate physical resources from different InPs that can host the VN request. The goal of the VN splitting phase is to decide to which InP a virtual node should be sent to, given that there are multiple candidate nodes in different InPs. Splitting is solved in two different ways using heuristic solutions (based on a max-flow/min-cut approach) and linear programming to provide an exact solution. In the embedding phase, virtual nodes and links are mapped simultaneously following a MIP formulation. Authors propose two approaches to process VN requests: sequential and parallel. Resource binding consists in actually reserving the selected resources in the substrate using a variety of virtualization platforms and management tools.

PolyViNE is a policy-based interdomain VN embedding framework that allows VNs to be provisioned across InPs located at different administrative domains in a distributed and decentralized manner (CHOWDHURY; SAMUEL; BOUTABA, 2010). In PolyViNE, individual SPs can enforce their own administrative policies and do not need to expose their internals. Besides, PolyViNE does not assume that an InP has complete knowledge of the substrate network. An SP sends a VN request to multiple InPs. The InPs in their turn will inform possible embeddings with their corresponding prices. Similar to a bidding process, the SP will choose the InP(s) offering the most competitive prices. PolyViNE introduces a distributed communication protocol to exchange information between SPs and InPs and coordinate the embedding. PolyViNE also supports location awareness to avoid that VN requests are forwarded to InP that cannot meet the location requirements defined by the SPs. To support location awareness, PolyViNE relies on a novel addressing scheme, named COST (Continent cOuntry State ciTy), which is based on postal addresses, and on a Location Awareness Protocol (LAP), which is based on Gossip and Publish/Subscribe protocols to decide to which InPs a VN request should be sent to, while enforcing InP internal policies.

SecondNet (GUO et al., 2010) proposes virtual data center (VDC) as the abstraction for resource allocation in multi-tenant cloud environments. The goal of SecondNet is to provide bandwidth guarantees by designing a scalable and deployable VDC embedding algorithm, which results in high utilization of the infrastructure network and supports elasticity when tenants's requirements change. SecondNet's VDC allocation algorithm models the VDC embedding as a bipartite matching, which is reduced to the min-cost flow problem. Physical servers are preconfigured into clusters before starting VDC allocation to reduce the problem size and to take server locality into account. Performance of the proposed allocation algorithm depends on the physical network topology. Network utilization is high for a BCube (GUO et al., 2009) topology, however, it is low for fat-tree (AL-FARES; LOUKISSAS; VAHDAT, 2008) and VL2 (GREENBERG et al., 2009) topologies. Bandwidth reservation between VMs pairs may be inefficient because data transmission between two VMs may vary. This may cause low utilization of network bandwidth.

Oktopus (BALLANI et al., 2011) is data center network architecture that implements two virtual data center abstractions (*i.e.*, virtual cluster and virtual oversubscribed cluster) for controlling the trade-off between the performance guarantees offered to tenants, their costs, and the provider revenue. A virtual cluster provides the illusion of having all VMs connected to a single non-oversubscribed virtual switch. A virtual oversubscribed cluster emulates an oversubscribed two-tier cluster that is a set of virtual clusters interconnected via a virtual root switch. Oktopus uses a greedy algorithm for the resource allocation to the VDC. However, Oktopus has some limitations. It supports only two types of requests. Besides, it can be applied only in tree-like physical topologies.

VN provisioning can be classified according to the criteria presented in Section 2.2.1. With respect to management targets, VN provisioning solutions allows the SP to request virtual nodes and links from the InP. It is also possible that SPs request VNs from multiple InPs located in different administrative domains. This is the case of PolyViNE (CHOWDHURY; SAMUEL; BOUTABA, 2010) and of the solution proposed by Houidi *et al.* (HOUIDI et al., 2011). Concerning management functions, in addition to the actual VN allocation, VN provisioning solutions can rely on monitoring agents to get updated status of the substrate and thus find the best mapping alternatives. Moreover, as previously stated, a management interface is required to allow the InP to interact with the underlying physical resources and complete the instantiation of the VN. However, proposals focused on VN embedding usually do not explicitly define a management interface or protocol for VN instantiation. VN provisioning can also be organized according to the management approach. Most solutions assume that the InP operator has a complete view of the substrate and defines the mapping in a centralized manner. Other proposals such VNet (CORREIA et al., 2011), the solution proposed by Houidi *et al.* (HOUIDI et al., 2010), and VINEA (ESPOSITO, 2013) employ distributed embedding algorithms. Autonomous provisioning is applicable when the InP needs to modify a previous allocated VN in order to optimize resource allocation or handle failures. Policy-based VN provisioning can be used, for example, to enforce internal InP policies in a multi-domain scenario (CHOWDHURY; SAMUEL; BOUTABA, 2010) or to enable different allocation strategies to be used (ESPOSITO, 2013).

3.3 Motivation

Despite the existence of a variety of VN embedding strategies, there are common limitations shared by most approaches. First, current embedding strategies are static. InPs cannot switch from an embedding strategy to another to satisfy a particular objective, such as energy efficiency or load balancing, nor they can support specific application requirements, such as fault-tolerance or exclusivity over resources (ESTEVES et al., 2014). Second, current VN provisioning approaches consider VN embedding as an atomic operation, *i.e.*, the complete mapping of virtual to physical resources is defined prior to VN deployment, which in some cases may not capture the dynamics of the substrate. For example, when a VN expires, it releases resources that could be used by an ongoing VN request because they are better options towards a given objective (*e.g.*, reduce communication cost). In current embedding approaches, if the InP wants to optimize resource allocation and leverage the existence of better mapping alternatives, it has to migrate already deployed VNs to new locations, which increases operational costs.

Given the limited flexibility of current VN embedding strategies, especially because they do not take into account possible changes in InP objectives, the particularities of applications, and the dynamics of the physical substrate, there is a need for more appropriate VN embedding approaches for environments where InPs' objectives change over time and SPs hosting a large diversity of applications. Research has been conducted over the recent years to employ adaptive allocation considering multiple objectives (RAO et al., 2011) (ISLAM et al., 2012) (LI et al., 2011) (FRINCU; CRACIUN, 2011). Such proposals, however, are limited to capacity adjustment of individual resources (*e.g.*, virtual servers), specific metrics (*e.g.*, response time), and reconfiguration of already deployed VNs.

In this thesis we address the problem of provisioning VNs considering multiple (possibly conflicting) InP and SP objectives to define how virtual resources are mapped in the substrate. Adaptive, application-driven resource provisioning allows multiple SPs and a large diversity of applications to efficiently share a virtualized network. To enable such flexible resource allocation, we propose an architecture of a provisioning system for virtualized networks that allows SPs to express high-level requirements for the requested VNs, which ultimately influence how VNs should be allocated in the physical substrate. The proposed provisioning approach is based on the concept of allocation *paradigms*. A paradigm encompasses goals associated with the high-level objectives of the InP and of the SPs. Each goal is realized by allocation actions executed within a window (one per goal), which is defined in real-time according to the current status of the substrate. This approach allows rapid adaptation of the provisioning process to the dynamics of the substrate and to the characteristics of the deployed applications. If an objective is not satisfied or applications of a VN present poor performance, the current allocation paradigm may need to be changed. This approach allows rapid adaptation of the provisioning process to the dynamics of the substrate and to the characteristics of the deployed applications.

3.4 Application-Aware VN Provisioning using Allocation Paradigms

In this section, we begin by defining the basic concepts and the formal problem formulation of our paradigm-based adaptive provisioning approach. Then, we present the architecture of an adaptive provisioning system based on the concept of allocation paradigms. Next, we then describe a methodology to map application high-level goals to allocation paradigms and a policy language used to describe allocation paradigms.

3.4.1 Basic concepts

A paradigm defines how VNs are allocated in the physical substrate. Paradigms are defined by the InP operator and run in the Infrastructure Manager component (Figure 2.2). Each paradigm comprises a set of goals associated with application requirements. A goal is derived in actions, which, in turn, allocate individual VN resources. The main concepts of the paradigm-based provisioning approach are described below.

- **Paradigm:** a paradigm $\mathcal{P}$ represents a group of goals $(G_1, G_2, ..., G_n)$ that are considered in the provisioning of VNs;
- **Goal:** a goal G is a single high-level objective that is defined by the InP or requested by the SP in the provisioning of VNs;
- **Action:** an action A is a single provisioning operation executed to achieve a goal G.
- **Window:** a window W is a set of allocation Actions $(A_1, A_2, ..., A_m)$ executed sequentially according to a goal G. A paradigm $\mathcal{P}$ is thus realized by a set of windows $(W_1, W_2, ..., W_n)$ associated with the goals $(G_1, G_2, ..., G_n)$ of the paradigm;
- **Allocator:** an allocator $Alloc$ executes provisioning of VNs through an allocation window W defined by a goal G. There is one allocator $Alloc$ entity associated with each goal G of a paradigm $\mathcal{P}$.
- **Round:** a round $\mathcal{R}$ is the sequential execution of the actions $(A_1, A_2, ..., A_m)$ of a window W triggered by an allocator $Alloc$.

When provisioning VNs, each goal G is translated into a list of actions $(A_1, A_2, ..., A_m)$ that will be executed sequentially within a window W. Several goals can be combined together in the provisioning of a VN. The choice for specific allocation actions depends on the InP objectives and on the characteristics of the applications to be deployed over the requested VN, on the active paradigm $\mathcal{P}$, and on the current status of the physical substrate (*e.g.*, available servers/links). Unlike current multi-objective VN provisioning proposals, allocation paradigms allow InP operators to modify the VN allocation "on-the-fly" by using different goals for each request. This flexibility is important to make

VN provisioning systems adaptable to a large diversity of VNs, where each one may run different applications. Figure 3.2 depicts the relationship between paradigms, goals, and actions.

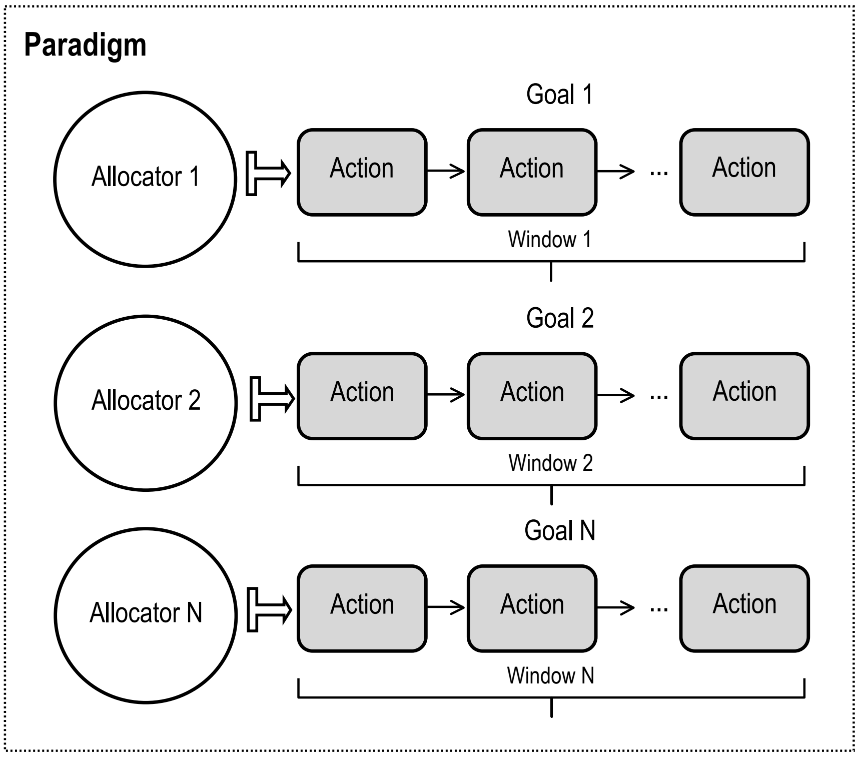
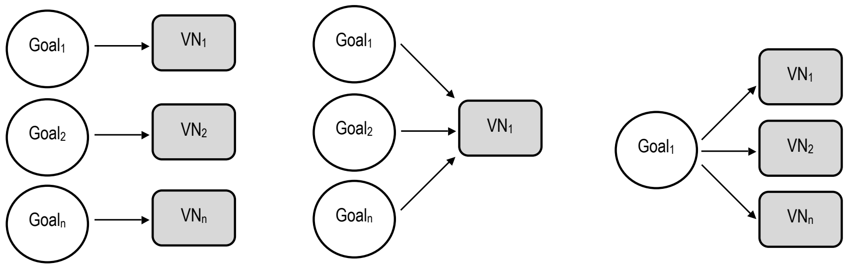


Figure 3.2: Allocation paradigms, goals, and actions

An allocation paradigm can be also seen as a group of provisioning-related policies that are considered in VN provisioning and defines how VNs are allocated in the physical substrate. Each policy is associated with a high-level goal defined by the InP operator or the SP, such as “low latency” or “resiliency”. Therefore, the decomposition of high-level goals to low-level actions is a form of policy refinement (MOFFETT; SLOMAN, 1993) (BANDARA et al., 2004). Allocation paradigms have two main design characteristics that make them suitable to guide resource allocation in NVEs:

Application awareness - Because VNs are used to host applications, the InP needs to define the relationship between VNs and the applications that ultimately will run on them. A possible approach is to simplify the problem by allowing only one application per VN. The disadvantage of such approach is the underutilization of resources when the application is not active. On the other hand, deploying multiple applications over a single VN can improve resource utilization at the cost of more complex allocations. The paradigm model allows both approaches to be used by the InP through three basic procedures illustrated in Figure 3.3. An application can require one or more goals for the VNs, such as “low latency” and/or “reliability”.

The first procedure (Figure 3.3(a)) allows only one goal to be associated with a VN. Different VNs may support different goals. Such procedure is better suited when there is one application per VN or when multiple similar applications share a single VN. In the second procedure (Figure 3.3(b)), all goals of an allocation paradigm are applied simultaneously during the allocation of a VN, resulting in *hybrid* VNs tailored for different goals. It is important to note that, in this procedure, some goals can predominate over others, depending on how the corresponding policies are defined (ESTEVES et al., 2013). Such policy adjustment can be used to handle conflicts between different goals. The third procedure (Figure 3.3(c)) reflects how VN allocation is tackled by current provisioning systems, where one goal is applied to all VNs. In summary, allocation paradigms aim to cope with the diversity of applications inherent in NVEs that require a variety of strategies to coexist under a single provisioning framework.



(a) One goal per VN (b) Multiple goals per VN (c) One goal, multiple VNs

Figure 3.3: VN allocation procedures

Adaptive provisioning - Typical VN embedding schemes map the whole VN on the physical substrate, at once, upon receiving a VN request. Mapping all resources of a VN in an atomic operation is straightforward because the InP has the complete view of the substrate network and the associated capacities. However, some approaches assume that individual resources (*i.e.*, virtual nodes) of the same VN request cannot share a physical one and have to be mapped at distinct locations (CHOWDHURY; RAHMAN; BOUTABA, 2012) (GUO et al., 2010). This limitation may reduce the chances of a successful embedding. In addition, most VN embedding approaches do not properly tackle the case where multiple VN requests arrive simultaneously, which is the typical case in realistic scenarios. Therefore, two or more ongoing VN requests can compete for the same physical resource increasing the chances of failed VN requests.

To overcome the aforementioned problems and allow rapid adaptation of current provisioning approaches to the dynamics of the physical substrate, we argue that a VN request should be mapped *in parts*. To realize such concept we propose the use of allocation *windows* and *rounds*. One allocation window encompasses a fixed number of individual allocation actions defined in real-time by the current allocation paradigm, such as virtual machine creation. The execution of the actions within an allocation window is called a round. Several rounds may be needed in order to complete the full allocation of a VN. In each round, the corresponding window executes the appropriate allocation actions defined by the active paradigm. Figure 3.4 illustrates how a VN would be mapped using the concepts of windows and rounds. The numbers represent the order virtual resources (*i.e.*, nodes and links) are allocated. The order actions are executed is defined by the policies included in the active paradigm.

The benefits of this partial and multi-iteration mapping is threefold. First, it allows multiple virtual resources of the same VN request to be mapped on the same physical asset. Second, windows allow rapid adaptation to changing network conditions. Two consecutive allocation rounds can result in different mappings compared to mapping all resources at once. For example, after the first round, the mapping of virtual machines can be modified dynamically to select a physical server that turned out to be a better mapping option for a given objective (*e.g.*, reduce number of active physical servers) and that was not available at the first round. Finally, a window can run actions from different VN requests making the problem of managing multiple ongoing VN requests more tractable.

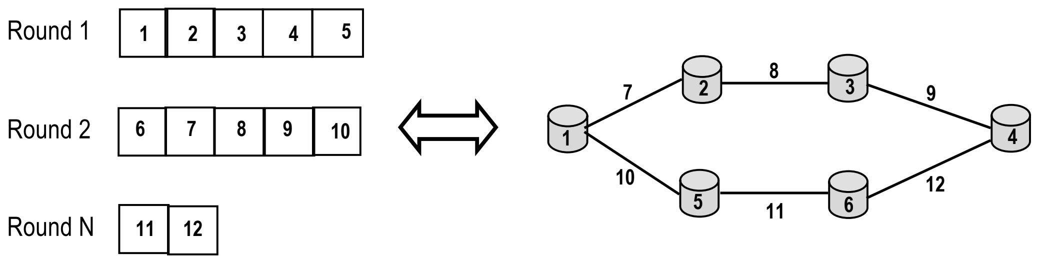


Figure 3.4: Allocation windows and rounds

Possible disadvantages of this paradigm-based allocation may appear in small-sized static scenarios (*i.e.*, VNs having long duration, InP or SP objectives not changing over time) because in such scenarios it is unlikely that allocating VNs in parts will produce a different (and better) result compared to a single round allocation and can potentially increase provisioning times. Besides, it may be not feasible to accommodate conflicting goals in a single paradigm without harming application performance. Partial allocation can also produce incomplete VNs if there are not available resources to complete the request in the subsequent rounds.

Examples -To better illustrate the paradigm-based allocation approach we next provide some examples of paradigms, goals, and actions that can be considered in VN provisioning. This is not an exhaustive list and can be complemented accordingly.

Goal: High server utilization

Description: The high server utilization goal aims to group virtual machines inside the smallest set of physical servers in order to increase server utilization and reduce operational costs. A paradigm using this goal tries to group VMs in a single physical server until it reaches its full capacity.

Goal: Green

Description: The green goal aims to minimize the number of used physical nodes and links in order to reduce energy consumption. To achieve this, virtual nodes and links have to be grouped in a small number of physical resources. It is similar to high server utilization goal described above.

Goal: Load balancing

Description: The load balancing goal aims to spread virtual resources in the substrate. When allocating a virtual resource, this goal will choose a unused physical asset or the least used one, *i.e.*, have the maximum residual capacity. For example, a virtual link will be placed in the physical link having the highest available bandwidth.

Goal: Low latency

Description: The low latency goal tries to allocate resources close to each other in order to minimize the number of hops between two virtual nodes. Therefore, virtual links should be mapped on physical path with small hop number.

Goal: Redundant

Description: The redundant goal creates a copy of each created resource in a different physical location for reliability purposes.

Goal: Protected

Description: When the SP, for security purposes, requires exclusivity over a physical resource or is not willing to share physical resources with potential competitors it may rely on the protected goal to map resources on top of unused nodes and links.

Goal: Cheapest

Description: In scenarios where the price of placing a virtual node on link may vary according to the location (for example, the energy price is different) the cheaper goal may be useful by selecting less expensive physical resources to host virtual resources.

Goal: Low communication cost

Description: When virtual nodes need to exchange large amounts of information it is preferable that they are close to each other in order to reduce not only the latency but the whole communication cost, thus preventing link congestion.

Goal: Random

Description: This objective selects random physical resources to host virtual ones. It may be useful for testing purposes.

Paradigms can be composed of one or more goals. Next we list some possible paradigm combinations that can be useful for InP operators.

Paradigm: Green + low communication cost

Description: This paradigm can be useful to minimize energy consumption and at the same time reduce traffic between physical nodes.

Paradigm: Load balancing + redundant

Description: This paradigm may be used when the InP wants to increase VN acceptance rate and provide backup VNs for *premium* SPs. Mission critical application can also benefit from this paradigm.

Paradigm: Protected + redundant

Description: In addition to secure VNs the SP may require reliable VNs.

Paradigm: Green + cheaper

Description: This paradigm is useful when the InP wants to reduce energy consumption (and consequently its operational costs) and the SP wants to acquire cheap VNs.

Paradigm: Low latency + protected + redundant

Description: This paradigm is useful when the SP requires secure/reliable VNs having low latency among virtual nodes in order to support high performance applications, such as e-science.

Actions can include any provisioning operation in the context of NVEs such as creation/removal of virtual resources (*e.g.*, , machines, routers, links, images, volumes), deployment/undeployment of virtual resources, virtual node migration or copy, virtual link establishment, and configuration of the attributes of virtual nodes and links (GRANVILLE; ESTEVES; WICKBOLDT, 2015).

Paradigm operations - The InP manager may create, remove, modify, or switch allocation paradigms. The active allocation paradigm may need to be modified by adding or removing policies from it, or another paradigm may be activated to allow rapid adaptation of the provisioning service to other types of VN requests or to changes in the physical substrate (*e.g.*, new resources that became available after VN release). The decision to modify or switch to another allocation paradigm depends on the effectiveness of the current active one. The effectiveness of an allocation paradigm can be defined in terms of the performance achieved by the applications running over a VN.

3.4.2 AVNP Problem Formulation

In this subsection we formulate the application-aware virtual network provisioning (AVNP) problem. We model the physical network, the virtual network request, and the allocation paradigm, respectively.

3.4.2.1 Physical Network

We model the physical network as a weighted undirected graph $N^p = (M^p, R^p, L^p, O^p)$, where M^p is the set of physical machines, R^P is the set of physical network elements (*e.g.*, routers and switches), L^p is the set of physical links used to connect physical machines and network elements, and O^p is the set of InP objectives that can be considered during VN provisioning. Each physical machine $m^p \in M^p$ has an associated CPU capacity $c(m^p) \in \mathbb{R}^+$. Each physical link $l^p_{ij} \in L^p$ connecting two physical machines $i, j \in M^p \cup R^p$ has an associated bandwidth $b(l^p_{ij}) \in \mathbb{R}^+$. An objective $o^p \in O^p$ is an overall goal that can be chosen by the InP. Each objective is associated with target index $t(o^p) \in \mathbb{N}$. Possible values that $t(o^p)$ can take are listed in Table 3.2. New objectives can be defined by the InP resulting in additional o^p and $t(o^p)$ values.

3.4.2.2 Virtual Network Request

In our model, a virtual network (VN) request is defined as a weighted undirected graph $N^v = (M^v, L^v, P^v)$, where M^v is the set of virtual machines, L^v is the set of virtual links, and P^v is the set of properties desired for the applications running on the VN. Different from the physical network, a virtual network has no intermediate nodes for routing or switching; virtual links are requested, however, in order to have allocated bandwidth between key virtual machines. Similar to the physical network, each virtual machine $m^v \in M^v$ requests an amount of CPU capacity $c(m^v) \in \mathbb{R}^+$, and each virtual link $l^v_{ij} \in L^v$ connecting two virtual machines $i, j \in M^v$ has a bandwidth requirement denoted by $b(l^v_{ij}) \in \mathbb{R}^+$. A property $p^v \in P^v$ is a non-functional requirement defined by the applications running on the VN. Each property has a corresponding target index $t(p^v) \in \mathbb{N}$ that is associated with a high-level requirement requested for the VN. For now, the values that $t(p^v)$ can take are listed in Table 3.3. These values are used as reference. The model can be easily extended to include as many properties as supported by the InP. In this case, if the InP supports a new VN property (*e.g.*, , low price) a new p^v and a corresponding $t(p^v)$ value have to be included in the model.

Table 3.2: InP Objectives Examples

Target	Goal	Description
0	Green	Virtual machines and links should be mapped on the smallest set of physical assets
1	Load balancing	Virtual machines and links should be mapped in distinct locations and cannot share the same physical resource
2	Low communication cost	Virtual machines with more capacity should be placed close to each other
3	High server utilization	Virtual machines of the same request should be grouped in the same server
4	Random	Picks random physical resources to host virtual ones. Used for testing

3.4.2.3 Allocation paradigm

An allocation paradigm $\mathcal{P}$ is defined by a set of goals $(G_1, G_2, ..., G_n)$ that are considered in VN provisioning. Each goal $G_i \in \mathcal{P}$ reflects an InP objective or a characteristic desired for an application running in the VN, having the same meaning of an objective $o^p \in O^p$ supported by the InP or a property $p^v \in P^v$ defined in a VN request, respectively. An individual goal G_i is realized by a set of allocation actions $(A_1, A_2, ..., A_n)$ executed sequentially within a window W_i. Each window W_i has a size attribute $s(W_i) \in \mathbb{N}^+$ corresponding to the number of actions that are executed in each round. An allocator entity $Alloc$ is responsible to trigger each window W. Multiple allocators can run in parallel to speed up the provisioning process.

The provisioning of a VN is thus a function of: the number of resources (*i.e.*, virtual machines and virtual links) that need to be allocated for the requested VN, the number of allocators deployed, and the maximum size of the window of each allocator, which can be dynamically adjusted in each round. The minimum number of rounds $\mathcal{R}$ required to provision a VN is given by:

$$\mathcal{R} = \left\lceil \frac{\sum m^v + \sum l^v}{\sum_{i=1}^{al} s(W_i)} \right\rceil \tag{3.1}$$

where ($\sum m^v + \sum l^v$) is the total number of virtual resources (*i.e.*, machines and links) that need to be instantiated per VN, ($\sum_{i=1}^{al} s(W_i)$) is the maximum number of allocation actions allowed per round, and al is the number of allocators deployed.

Table 3.3: VN Properties Examples

Target	Property	Description
0	Reliability	Replicas of allocated resources should be placed in different locations
1	Security	A virtual machine should not share the same physical machine of another SP
2	Best-effort	Virtual machines can be placed at any location
3	Cheap	cheap physical machines should be selected to host virtual ones
4	Low latency	Virtual links should be mapped on physical paths with small hop number

The size of allocation windows can vary according to the current provisioning status of the requested VNs. If a VN is already deployed and no changes are expected in the short run, the size of the allocation window for that VN is zero. On the other hand, if the VN provisioning has just started or modifications on a previously allocated VN are scheduled, then the size of the window is greater than zero. The size of an allocation window can also be adjusted to prioritize one goal over the others. The higher the priority of a goal, the larger the size of its corresponding allocation window because more actions of the goal are executed in a single round. There is a clear tradeoff between the size of the allocation windows and the provisioning time. A large paradigm window requires fewer rounds to allocate a whole VN, but it is unlikely to take advantage of a better allocation option that becomes available. On the other hand, a small paradigm window is more adaptable to dynamic environments at the price of higher overhead, which can result in larger provisioning times.

3.4.3 Conceptual Architecture of a Paradigm-Based Provisioning System

A conceptual architecture of the paradigm-based provisioning system that is run by the InP operator in the Infrastructure Manager (Figure 2.2) is depicted in Figure 3.5. The system is structured in four main layers: *Access Layer*, *Operator Layer*, *Provisioning*

Layer, and *Infrastructure Layer*. The access layer is the entry point for SP operators to request VNs from the InP. In the operator layer, InP operators can define paradigms, goals, and associated actions. The provisioning layer implements the core logic of the system and comprises allocation and monitoring services. The infrastructure layer consists of a set of *resource agents* that are responsible for device-level resource management. The resource agents also collect device status data that is forwarded to the upper layers.

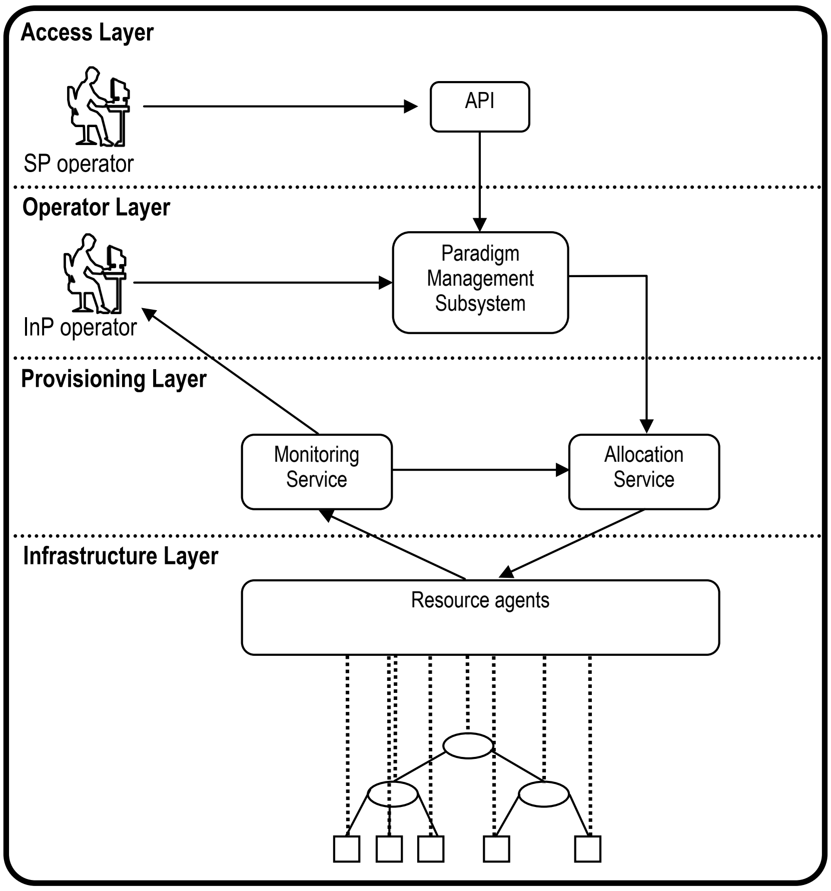


Figure 3.5: Architecture of the paradigm-based provisioning system

The SP operator requests virtual resources from the physical infrastructure to build VNs using an API supported by the InP. The InP operator defines paradigms using a *Paradigm Management Subsystem (PMS)*. The PMS allows InP operators to create new paradigms, goals, define the set of allocation actions available to an objective, and determine the current active paradigm. The *Allocation Service (AS)* is responsible to deploy the active paradigm and execute allocation actions. The AS supports two operation modes: *system-driven* or *human-driven*. In the system-driven mode, the AS automatically determines the set of allocation actions to be executed during the next paradigm window. In contrast, the human-driven mode requires the InP operator to explicitly select the actions to be included in the window of the next round. The *Monitoring Service (MS)* provides updated status of the applications running on the physical substrate to the InP operator and to the AS component. The monitored data collected by the MS is used in AS system-driven mode to determine the next set of allocation actions. Similarly, the InP operator can use the information provided by the MS to manually decide upon the actions to be included in the subsequent round.

In the system-driven AS operation mode, the actions of a paradigm window are selected automatically by the system. The decision on which actions will be included in the next window is based on a combination of the requirements of both SPs and InPs and on the current status of the substrate. Policies must be initially defined by the InP operator to allow the system to automatically reconfigure the allocation process for each objective. For example, if a goal is defined to support applications requiring low response times, the policy can be, for example, "create next virtual machine in the least overloaded physical server" or "create next virtual machine in the same physical machine of the previous one" (to avoid network bottlenecks). Guided by this high-level policy, the AS contacts the MS to retrieve updated information about the residual capacities of the physical machines. The AS then selects the physical servers with the highest residual capacity.

In the human-driven AS operation mode, the InP operator is responsible for manually selecting the actions to be included in a paradigm window. The operator can define both the number and the nature of actions of a paradigm window. The human-driven mode is useful to assist InP operators in new situations that are not covered by the policies defined for the system-driven mode. For example, the InP operator may want to evaluate a new paradigm in order to define the best set of actions to be modified or included in a policy of the system-driven mode. The human-driven mode is thus useful in fine adjustments of policies of the system-driven mode and to handle exceptional situations that can arise in allocating virtual networks.

3.4.4 Mapping application goals to paradigms

The choice of a paradigm by the InP operator is influenced by the requirements of the customers of the physical infrastructure (*i.e.*, SPs). However, in virtualized networks, InP operator has no means to know in advance which applications will run in his/her infrastructure. SPs usually request a number of virtual resources (*e.g*, VMs, virtual routers) with specific capacities (*e.g.*, high memory, large disk space) and do not specify particular performance objectives. Therefore, specific requirements of the application(s) running in a VN are not considered in resource allocation. For example, a VN that hosts services needing high reliability can be duplicated, while virtual resources running applications requiring low response times can be placed close to one another.

The SP must express the requirements of the applications that will ultimately run in their VN to allow InP operators to change between allocation paradigms and choose the best one. To enable such application-oriented adaptive allocation we argue that the interface between SPs and InP operators should be extended to allow the specification of high-level goals that need to be fulfilled by the VN. The high-level goals defined by the SP will be mapped to the paradigm goals and actions, which, in turn, actually allocate resources. The API component of the access layer should provide a means to allow the SP operator to define VN properties. For example, the API can offer a fixed-set of properties that can be selected by the SP operator for his/her VNs.

In our solution, the SP can choose among a set of predefined high-level properties those that better fit the application(s) he/she wants to deploy in a VN. The SP can specifically select, for example, "reliability" and/or "security", among all available properties, to build a VN able to support applications requiring high availability and security, respectively. With this information, the InP operator selects the most appropriate allocation paradigm to satisfy the goals defined by the SP. If there is no paradigm suitable to handle a specific set of goals, the InP operator will have to define a policy to allow the system to automatically define the best allocation actions.

3.4.5 Paradigm Policy Specification

As a part of our paradigm-based provisioning approach, we have developed a policy language to allow InP operators to specify the relationship between paradigms, goals, and allocation actions. Although there are many policies available (IETF-POLICY, 2012) designed for a variety of purposes, none of them define adequate constructs to allow InP operators expressing allocation paradigms. Our paradigm policies are used by the PMS to automatically define the allocation actions in the system-driven mode. The main constructs of our paradigm-policy language are described below and an example of paradigm policy is given in Figure 3.6.

- **goal:** an objective g defines a customized allocation goal for a paradigm p. Each goal g has a corresponding *window* attribute w specifying the identifier of the allocation window hosting the actions $(a_1, a_2, ..., a_n)$ associated with the goal;
- **action:** each action a is defined by an identifier, a list of conditions $(c_1, c_2, ..., c_n)$ that trigger the action, and an operation that actually implements the action, such as *SelectServer* or *MoveToServer* that are used to place or migrate a VM to a given physical machine, respectively;
- **window:** a window w defines a group of actions that are executed sequentially. A window w has a *size* attribute s that defines the number of actions that are executed before the next scheduling. A window also defines the *order* on which the actions are verified in each turn.

The general format of a paradigm policy is depicted below.

```
goal <name> := <list-of-actions>
                   <window>

action <name> := <conditions>
                 <operation>

window := <size> <order>
```

An action is triggered when a set of associated *conditions* is satisfied. The action is then realized by a low-level *operation* (*e.g.*, create a VM) supported by the substrate. The *window* construct defines the size of allocation windows and the order that the actions of a policy are evaluated.

In addition to the main constructs, the paradigm policy language can use a set of auxiliary functions in the main constructs. The auxiliary functions provide updated information about the physical infrastructure and underway requests. Such functions can also refer to individual allocation actions (*e.g.*, virtual machine creation, virtual machine migration). A non-exhaustive list of auxiliary functions that can be used in policy language is given in Table 3.4. The functions provided here are used for a proof of concept. They can, however, be extended to include other operations. In practice, the implementation of auxiliary functions like the ones of Table 3.4 is dependent on the API provided by the underlying virtualization platform.

In the paradigm policy example of Figure 3.6 there are three defined objectives: *High-ServerUtilization*, *LoadBalancing*, and *LowCommunicationCost*. The HighServerUtilization objective tries to map all virtual machines to a reduced number of physical machines.

Table 3.4: Auxiliary Functions Examples

Function	**Description**
Num_Allocated_VMs(VNR)	returns the number of already allocated virtual machines of a given VN request
VMs_ToAllocate(VNR)	returns the number of remaining virtual machines of a given VN request that were not yet allocated
UnusedServers(VNR)	returns the number of physical machines that do not host a VM of a given VN request
UnusedLinks(VNR)	returns the number of physical links that do not host a virtual link of a given VN request
SelectServer(V, P)	maps a virtual machine V to a physical machine P if P has enough capacity
MoveToServer(V, S, D)	migrates a virtual machine V from the physical machine S to the physical machine D
SelectPath(VL, PL)	maps a virtual link VL to a physical link PL if PL has enough bandwidth
random(RES)	returns a random physical resource (*i.e.*, machine or link) from the subset RES
lowest_residual_capacity(type)	returns the resource (*i.e.*, machine or link) with the lowest residual capacity in terms of a given resource (*e.g.*, CPU, memory, disk, bandwidth) or a combination of the available resources
highest_residual_capacity(type)	returns the resource (*i.e.*, machine or link) with the highest residual capacity in terms of CPU, memory, disk, or bandwidth
all()	returns all available resources of the physical substrate
used_servers(VNR)	returns all physical machines used to host virtual machines of a given VN request
used_links(VNR)	returns all physical links used to host virtual links of a given VN request
closest_server(machine)	returns the closest physical $machine$ to a given one in terms of number of hops
shortest_path(S, D)	returns the shortest physical path for a virtual link starting and terminating at physical machines S and D, respectively

If there is not enough capacity to map all virtual machines of a VN request in a single physical one, the AS component finds the physical machine with the lowest residual capacity. In contrast, the LoadBalancing objective spreads the virtual resources (*i.e.*, virtual machines and virtual links) among all available physical machines and links by selecting the machines and links with highest residual (*i.e.*, more available) capacity at each round. The LowCommunicationCost objective is very similar to the HighServerUtilization objective, except that LowCommunicationCost tries to keep virtual machines of the same VN close to one another, even if they are not sharing the same physical server.

Figure 3.7 illustrates another example of paradigm composed of two objectives. The *Protected* objective ensures that a VM of a request is placed alone in a physical machine or shares a physical machine with other VMs of the same request. The *Redundant* objective forces the creation of an additional VM in a distinct location for each VM of a VN request.

```
HSU_LB_LCC {
 goal HighServerUtilization {
       action HSU-CreateVM-first {
             conditions  = {Num_Allocated_VMs = 0, VMs_toAllocate > 0}
             operation = {SelectServer(VM, random(all))}
       }
       action HSU-CreateVM-others {
             conditions  = {Num_Allocated_VMs > 0, VMs_toAllocate > 0}
             operation = {SelectServer(VM, lowest_residual_capacity(machine))}
       }
       action HSU-MigrateVM {
             conditions  = {VMs_toAllocate = 0}
             operation = {MoveToServer(VM, S, lowest_residual_capacity(machine))}
       }
       window HSU {
             size = 5
             order = {HSU-CreateVM-first, HSU-CreateVM-others, HSU-MigrateVM}
       }
 }
 goal LoadBalancing {
       action LB-CreateVM-first (HSU-CreateVM-first)
       action LB-CreateVM-others {
             conditions  = {Num_Allocated_VMs > 0, VMs_toAllocate > 0}
             operation = {SelectServer(VM, highest_residual_capacity(machine))}
       }
       action LB-CreateVL-first {
             conditions  = {Num_Allocated_VLs = 0, VLs_toAllocate > 0}
             operation = {SelectPath(VL, shortest_path(S,D)}
       }
       action LB-CreateVL-others {
             conditions  = {Num_Allocated_VLs > 0, VLs_toAllocate > 0}
             operation = {SelectPath(VL, highest_residual_capacity(link))}
       }
      window LB {
             size = 3
             order = {LB-CreateVM-first, LB-CreateVM-others, LB-CreateVL-first,  LB
             CreateVL-others}
      }
 }
 goal LowCommunicationCost {
       action LCC-CreateVM-first (HSU-CreateVM-first)
       action LCC-CreateVM-others {
             conditions  = {Num_Allocated_VMs > 0, VMs_toAllocate > 0}
             operation = {SelectServer(VM, closest_server(first))}
       }
       action LCC-MigrateVM {
             conditions  = {VMs_toAllocate = 0}
             operation = {MoveToServer(VM, S, closest_server(last_used)}
       }
       window LCC {
             size = 4
             order = {LCC-CreateVM-first, LCC-CreateVM-others, LCC-MigrateVM}
       }
 }
}
```

Figure 3.6: Example of paradigm policy - High server utilization + load balancing + low communication cost

The paradigm policy language is flexible to allow InP operators to describe customized objectives and actions tailored for a variety of VN requests and associated applications. Embedding algorithms and heuristics can be translated into objectives and actions through appropriate auxiliary functions. For example, if the InP operator wants to deploy a paradigm based on an algorithm that aims at minimizing energy consumption, he/she can use or implement an auxiliary function that prioritizes nodes and links with high utilization, *i.e.*, , high residual capacity. New functions can be implemented using basic primitives provided by underlying virtualization platforms. Paradigm policies are used by the AS to automatically schedule actions during the provisioning of a VN.

```
Protected_redundant {
 goal Protected {
       action Protected-CreateVM-new {
             conditions  = {VMs_toAllocate > 0}
             operation = {SelectServer(VM, empty(all))}
       }
       action Protected-CreateVM-shared {
             conditions  = {VMs_toAllocate > 0}
             operation = {SelectServer(VM, used_servers(request))}
       }
       window Protected {
             size = 2
             order = {Protected-CreateVM-new, Protected-CreateVM-shared}
       }
 }
 goal Redundant {
       action Redundant-CreateVM {
             conditions  = {VMs_toAllocate > 0}
             operation = {SelectServer(VM, empty(all))}
             operation = {SelectServer(VM, unused_servers(request))}
       }
       window Redundant {
             size = 3
             order = {Redundant-CreateVM}
       }
 }
}
```

Figure 3.7: Another example of paradigm policy - Protected + redundant

3.5 Determining the Efficiency of Allocation Paradigms

Evaluating the efficiency of an allocation paradigm and determining the associated conditions for paradigm switching is a critical task. That is highly dependent on the performance achieved by the applications running on top of a VN, which requires proper feedback from SPs. Current research focuses on static provisioning approaches that do not support changes in InPs' and SPs' objectives. In this section, we address the problem of evaluating the effectiveness of allocation paradigms in terms of application performance. We propose a VN computation model that considers all the applications running in a VN and whose output is used to guide paradigm switching. The model is based on multiple linear regression.

3.5.1 Applications

In our model we consider three basic types of applications: Mail, Web 2.0, and E-commerce. Such applications represent typical workloads of virtualized environments (VMMARK, 2012a) (SPECVIRT, 2012). The Mail application is a typical mail server running on a virtual machine to provide communication for the employees of an organization. The Web 2.0 application simulates a social network and is structured in two tiers (Web and database) running on separate virtual machines. The E-commerce application reflects a multi-tiered system composed of four virtual machines (three Web servers and one database server).

The performance of each application is defined by a particular metric. We follow the same definition adopted in measurements using the VMmark benchmark (VMMARK, 2012a). For each application, VMmark defines a reference value for the associated metric. Table 3.5 summarizes the characteristics of the applications considered in our model.

3.5.2 VN Scoring Methodology

The VN computation model is based on the concepts of *tiles* and *scores*, typically found in benchmarking systems (VMMARK, 2012a) (SPECVIRT, 2012). A tile is a

Table 3.5: Application performance metrics

Application	VMs	Metric	Reference value
Mail	1	Actions/minute	330.25 actions/minute
Web 2.0	2	Operations/minute	4641.43 operations/minute
E-commerce-A	4	Transactions/minute	2199.18 Transactions/minute
E-commerce-B	4	Transactions/minute	1518.55 Transactions/minute
E-commerce-C	4	Transactions/minute	1058.05 Transactions/minute

fixed-size group of virtual machines running multiple applications. In our case, a tile is composed by seven VMs belonging to individual instances of the Mail (1 VM), Web 2.0 (2 VMs), and E-commerce (4 VMs) applications, respectively. The score is a numerical value attributed to the VN reflecting the combined performance of all tiles (and applications). Our score calculation is adapted from the VMMark benchmarking system (VMMARK, 2012a) (VMMARK, 2012b), used to measure the performance of applications running in virtualized environments. The score metric $\mathcal{S}$ for a VN is calculated as follows:

$$\mathcal{S} = \sum_{i=1}^{\tau} \mathcal{T}_i \tag{3.2}$$

where $\mathcal{T}_i$ is the performance of the tile i and τ is the total number of tiles a VN supports. The total score of a VN is thus the sum of the performance of all its tiles. The performance of a individual tile $\mathcal{T}$ is defined by:

$$\mathcal{T} = (\prod_{j=1}^{n} \frac{App_j}{Ref_j})^{\frac{1}{n}} \tag{3.3}$$

where App_j refers to the performance achieved by the jth application in terms of the metrics defined in Table 3.5, Ref_j is the reference value for the application App_j, and n is the total number of applications of the tile. The $\mathcal{T}$ value is thus the geometric mean of the normalized performance of all applications of a tile.

3.5.3 VN Performance Prediction Model

In order to evaluate the efficiency of an allocation paradigm in terms of application performance, the InP operator needs to monitor the performance of the applications running on a VN. However, such reactive evaluation may result in excessive monitoring traffic in large NVE scenarios and in performance degradation of short-lived applications. Therefore, predicting the performance of the applications to be deployed over a VN and evaluating allocation paradigms *in advance* can improve overall VN performance. As a first step towards the definition of a VN performance model we analyze the relation between the number of tiles and the total score of a VN by analyzing data submitted to the VMmark Web site. By the time we collected this data, the number of tests sent to VMmark was around 60. Figure 3.8 depicts a scattered plot showing the VN score as a function of the number of tiles.

From Figure 3.8, it is possible to observe that the overall score of a VN grows linearly with the number of tiles. We thus propose a simple linear regression model to predict the performance of a VN given the number of tiles used. Using the R statistical package (R, 2013) we found that the predicted score $\mathcal{S}'$ of a VN can be defined by:

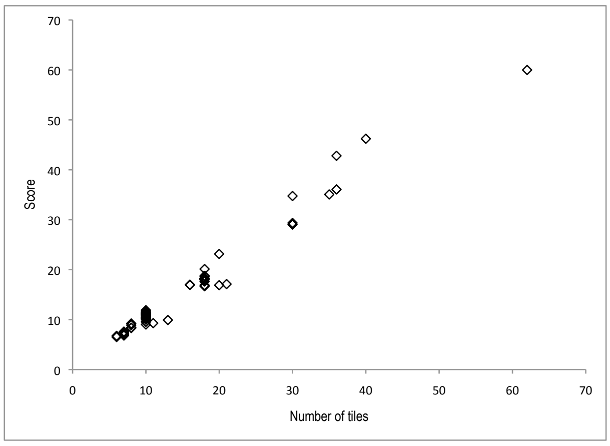


Figure 3.8: Score vs. number of tiles

$$\mathcal{S}' = 0.1573 + \tau \times 1.0206 \tag{3.4}$$

where τ is the number of tiles. The adjusted R-squared is 0.973.

3.5.4 Efficiency of an Allocation Paradigm

The efficiency of an allocation paradigm is influenced by two main factors: the number of rounds required to complete the provisioning of a VN and the predicted score of the allocated VNs. The number of rounds is directly related to the provisioning time of the VN. The score, in turn, reflects the quality of the allocation paradigm because VNs with high score indicate that hosted applications experience good performance and the VN is unlikely to change in the short run. The quality of an allocation paradigm $\mathcal{Q}$ is given by:

$$\mathcal{Q} = \frac{S'}{\mathcal{U}} \tag{3.5}$$

where $\mathcal{U}$ is the performance of the reference system obtained when applying Equations 3.2 and 3.3 to the reference values of Table 3.5.

The ultimate goal of an allocation paradigm is to reduce the number of necessary rounds to allocate a VN, which impacts VN deployment time, and avoid excessive paradigm changes, which is related to the stability of the provisioning system. A paradigm change can occur when the score of the provisioned VNs are below a threshold defined by the InP operator. Therefore, the efficiency of an allocation paradigm is defined as:

$$\mathcal{E} = \frac{1}{\alpha\mathcal{R} + \beta(1 - \mathcal{Q})} \tag{3.6}$$

where $\mathcal{R}$ is the number of rounds as defined in Equation 3.1, $\mathcal{Q}$ is the paradigm quality, α and β are adjusting factors to weigh the influence of $\mathcal{R}$ and $\mathcal{Q}$ in the paradigm efficiency $\mathcal{E}$.

The main objective of our paradigm-based provisioning framework is to:

$$\text{minimize} \sum \frac{1}{\mathcal{E}} \tag{3.7}$$

subject to:

$$0 < \sum_{W \in \mathcal{P}} s(W) \leq \sum m^v + \sum l^v \tag{3.8}$$

$$\forall m^v \in M^v, \forall l^v \in L^v$$

$$\tau > 0 \tag{3.9}$$

The objective is to minimize the 'inneficiency' (*i.e.*, maximize the efficiency $\mathcal{E}$) of an allocation paradigm $\mathcal{P}$. Constraint 3.8 assures that the number of rounds is bounded by the sum of the required virtual resources of a VN. Constraint 3.9 guarantees that at least one tile is allocated.

3.5.5 Resource Allocation Algorithm

We propose an algorithm to guide resource allocation in virtualized networks using the concept of paradigms. The algorithm is triggered upon receiving of a VN request. The algorithm checks each goal of the paradigm and the corresponding policy set (lines 10-12). For each policy, the algorithm verifies its set of conditions following the order predefined in the policy (lines 16 and 17). If a condition is met (line 18), the associated allocation action is added to the next allocation window (line 19) until the window reaches its maximum size defined in the policy. The algorithm then executes the allocation windows for each goal of the paradigm (lines 22-25).

In order to avoid that a request waits indefinitely for resources that may not be available in the subsequent rounds we define a parameter called *maxrounds*, which is the maximum number of waiting rounds for an ongoing request. If such number is reached then the request must be "rolled-back" by releasing previously allocated resources. After each execution, the algorithm calculates the efficiency of the active paradigm. The number of used tiles is obtained from the VN request (line 29) and the predicted score $\mathcal{S}'$ is calculated (line 30). The quality $\mathcal{Q}$ is also calculated (line 31). If $\mathcal{Q}$ and is below a certain threshold defined by the InP operator, then the active paradigm is updated by adding, removing goals, or switching to another paradigm (lines 32-34).

For example, using a Paradigm Management Subsystem (PMS) (ESTEVES et al., 2013) the InP operator initially defines that the allocation paradigm is composed by the Green goal only. Since the paradigm is composed by one goal only, there will be only one policy in LP. However, it is possible to have as many policies as there are goals in the allocation paradigm. The policy related to the Green goal has a condition regarding machine allocation that states that if there are machines to be allocated for a given request, the correspondent action is to select the machine with lowest residual capacity (CPU, memory, disk) satisfying the request to host the virtual one (the LB goal policy on the other hand chooses the machine with highest residual capacity instead). If such condition is met, the action (select the machine with lowest residual capacity) is included in the allocation window to be executed until there are no action to be included or the window reaches its maximum size s(W).

Algorithm 1 Paradigm-Based Allocation Algorithm

```
1: $W$: window of the paradigm $\mathcal{P}$
2: $LR \leftarrow M^v \cup L^v$
3: $NLR \leftarrow$ size of $LR$
4: $maxrounds \leftarrow$ maximum number of rounds
5: $nrounds \leftarrow$ round number
6: $nrounds \leftarrow 0$
7: while $NLR > 0$ and $nrounds <= maxrounds$ do
8:     run active paradigm:
9:     $W \leftarrow$ empty
10:    for all $G \in \mathcal{P}$ do
11:        schedule actions:
12:        $LP \leftarrow$ policy set of the paradigm $\mathcal{P}$
13:        for each $p \in LP$ do
14:            $C \leftarrow$ conditions of the policy $p$
15:            $A \leftarrow$ actions of the policy $p$
16:            repeat
17:                check next condition $c \in C$
18:            until $c =$ true
19:            Add action $a \in A$ triggered by $c$ to $W$
20:        end for
21:        run allocation window:
22:        for $i \leftarrow 1$ to $s(W)$ do
23:            Execute action $W(i)$
24:            $NLR \leftarrow NLR - 1$
25:        end for
26:    end for
27:    $nrounds \leftarrow nrounds + 1$
28:    calculate the efficiency of the active paradigm:
29:    determine the number of tiles $m$
30:    calculate the predicted performance $S'$
31:    calculate the quality $\mathcal{Q}$ of the current paradigm
32:    calculate the efficiency $\mathcal{E}$ of the current paradigm
33:    if $\mathcal{E} < threshold$ then
34:        Update current paradigm $\mathcal{P}$
35:    end if
36: end while
```

The actions of an allocation window are executed sequentially until the window is empty, which completes a round. Next, the system calculates the efficiency of the allocation paradigm by comparing the predicted performance of the paradigm quality with a threshold that was manually defined by the InP operator. If the efficiency is below this threshold then the system notifies the InP operator, which, in turn, using the PMS, has the option to change the current allocation paradigm by adding, removing goals, modyfing the policies, or switching to another paradigm.

3.6 Summary

In this chapter, we have proposed a provisioning framework based on the concept of allocation paradigms that enables adaptive allocation in NVEs. Our proposal allows SPs to express high-level requirements for their VNs, which are used by InP operators to define how a VN is allocated on the physical substrate. Furthermore, multiple InP objectives can be considered when allocating a single VN, enabling flexibility and allowing a large number of diverse applications to coexist in an NVE. In addition, we have proposed a methodology to evaluate the efficiency of allocation paradigms. We have defined a quality metric for allocation paradigms that summarizes the predicted performance of applications running in a VN. The paradigm quality can be used by the InP operator to decide on the changing of replacement of a current deployed allocation paradigm. In the next chapter, we conduct an evaluation of our proposed paradigm-based provisioning approach.

4 EVALUATION OF THE PARADIGM-BASED VN PROVISIONING APPROACH

In this chapter we evaluate our paradigm-based provisioning framework. The goal of this evaluation is to quantify the benefits of our approach in terms of a set of metrics reflecting different aspects of VN provisioning. The metrics we have defined in our evaluation include: acceptance ratio, provisioning cost, resource (CPU and link) utilization, paradigm quality, number of rounds, and paradigm efficiency. We have adopted a simulation-based approach for evaluating allocation paradigms. We begin by describing the simulation scenario. Next, we define and explain the metrics we used to conduct the evaluation. Finally, we discuss the results achieved so far.

4.1 Simulation scenario

We have adapted the discrete-event simulator used by Chowdhury *et al.* (CHOWDHURY; RAHMAN; BOUTABA, 2012) (CHOWDHURY, 2011) to implement the core logic of the paradigm-based VN allocation. The ViNE-Yard simulator was adapted for convenience purposes. It has classes to represent both the physical substrate and virtual requests, and comes with a workload generator. New classes to represent paradigms, objectives, and actions were created and the mapping methods were replaced by the ones defined in the paradigm policies. We have also extended the simulator to include our paradigm efficiency calculation and to allow paradigm changes on the fly. The scenario we consider in the simulations is of a virtualized data center network (BARI et al., 2012). We have chosen virtualized data center networks as the evaluation scenario because such networks form the core of modern cloud computing environments and are typically shared by a variety of applications with diverse requirements.

The physical substrate is represented by a Clos-based topology (GREENBERG et al., 2009), which is a topology typically used in data center networks, as shown in Figure 4.1. The physical topology is composed of 24 servers, 22 switches, and 72 links. Bandwidth capacity of links varies according to the type of switch used: 1000 Mbps for ToR (*Top-of-Rack*) switches, and 10000 Mbps for aggregate and intermediate switches (RABBANI et al., 2013). Server CPU and storage capacities are uniformly distributed between 50 and 100 units each. The unit for CPU is GFLOPS. The unit for storage is Gigabytes.

Similar to Chowdhury *et al.* (CHOWDHURY; RAHMAN; BOUTABA, 2012), VN requests arrive in a Poisson process with an average rate of 4 VNs per 100 time units and an average duration of 1000 time units, exponentially distributed. The number of virtual machines of each VN request is randomly defined by a uniform distribution between 2 and 10. CPU requirements of virtual machines are uniformly distributed between 0 and

20 GFLOPS, and the bandwidth requirements of virtual links are uniformly distributed between 0 and 50 Mbps.

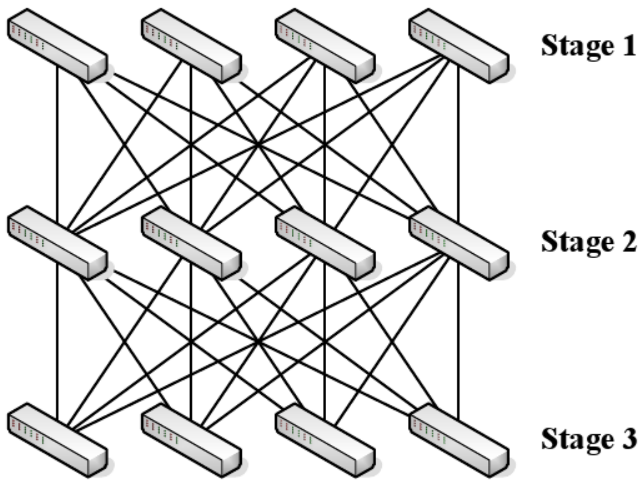


Figure 4.1: Clos topology

Our evaluation is divided in two main parts. In the first part, the goal is to quantify the benefits of our proposed paradigm-based adaptive provisioning approach in terms of acceptance ratio, provisioning cost, and resource (CPU and link) utilization when multiple objectives are considered in VN provisioning. In the second part, we measure the efficiency of allocation paradigms in terms of application performance and number of required rounds to complete VN provisioning.

4.2 Evaluation - paradigm goals comparison

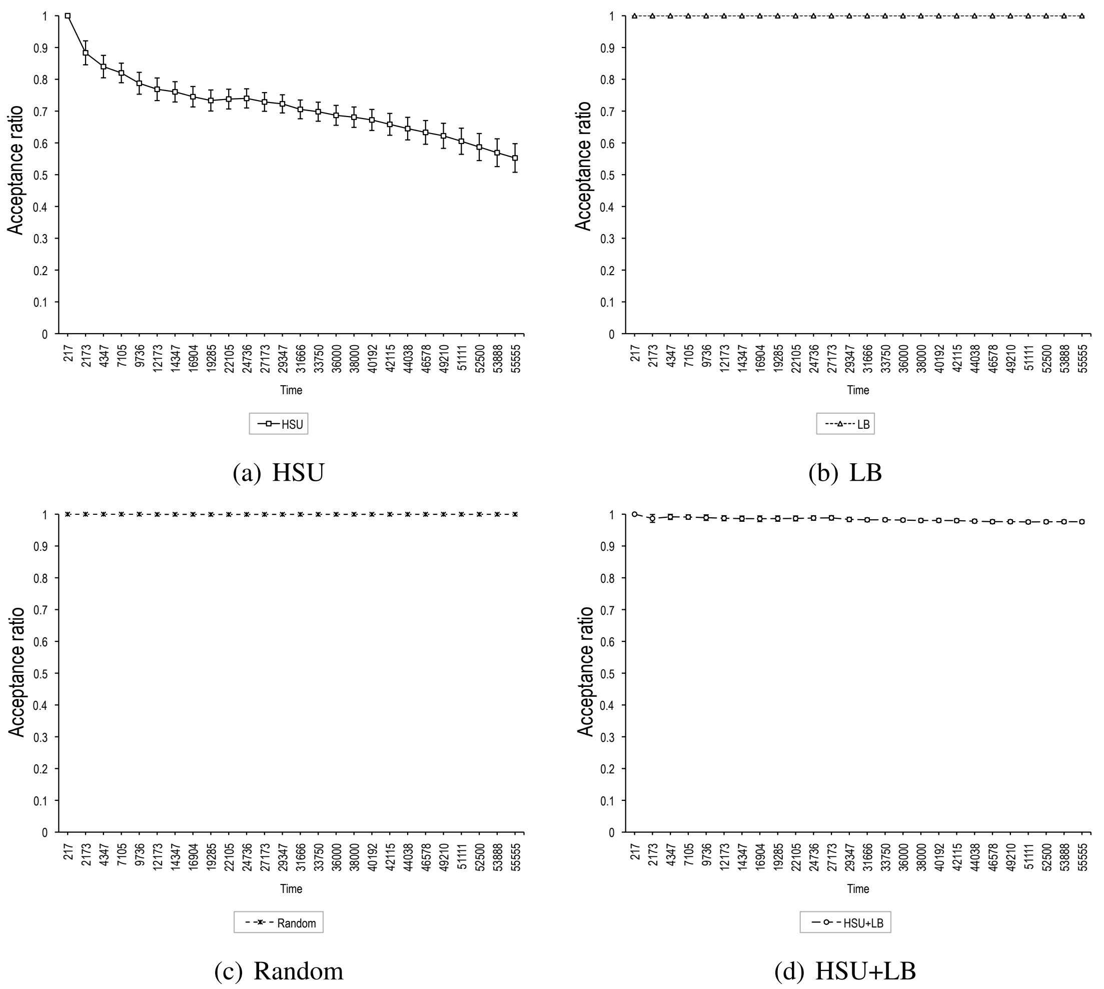
In the first part of the evaluation we have defined three main goals to compose allocation paradigms: *HighServerUtilization (HSU)* and *LoadBalacing (LB)*, detailed in Section 3.4.5, and a *Random* goal, which tries to map all virtual machines randomly. The window size is 1 for all paradigms, which means that one action of each goal of a paradigm is executed in each round.

4.2.1 Metrics - paradigm goals comparison

We have defined four metrics in the first evaluation. *Acceptance ratio* is the number of VN requests that are accepted over the total number of requests. *Provisioning cost* is defined in terms of allocated resources and calculated using the model of Chowdhury *et al.* (CHOWDHURY; RAHMAN; BOUTABA, 2012) where the cost is defined as $\sum_{l^v_{ij} \in L^v} \sum_{l^p_{ij} \in L^p} b(l^v_{ij}) + \sum_{m^v \in M^v} c(m^v)$. *CPU utilization* and *link utilization* reflect the average usage of individual servers and links over their total capacity. For each experiment, we evaluate 3 paradigms composed of one single goal (HSU, LB, and Random) and a paradigm combining the HSU and LB goals. Figures 4.2 to 4.5 summarize the obtained results. Each experiment was repeated 30 times with a confidence level of 95%.

4.2.2 Results - paradigm goals comparison

The average acceptance ratio over time is shown in Figure 4.2. The *LB* goal achieves very high acceptance ratio indicating that distributing virtual machines over the substrate is better than concentrating requests in a limited subset of machines. Such behavior can be explained by the fact that LB always try to find the resources with highest residual capacity, which increases the chances of a successful mapping. Randomizing resource allocation also results in high acceptance ratio, comparable to LB. According to the literature, randomization can lead to good results in some scenarios, especially when the number of machines is high (CHOWDHURY; RAHMAN; BOUTABA, 2012) (MITZENMACHER; RICHA; SITARAMAN, 2000). On the other hand, HSU has the worst performance among all paradigms because HSU, in contrast to LB, looks for the machines with the lowest residual capacity in order to reduce the number of used resources (*i.e.*, minimize resource fragmentation). The side effect is that such machines run out of capacity very quickly, reducing the mapping possibilities. However, when HSU and LB goals are combined under the same paradigm, the acceptance ratio improves considerably, achieving similar performance to LB and Random goals.

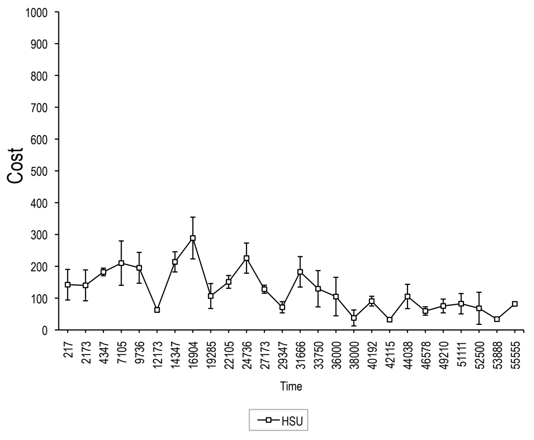


(a) HSU (b) LB

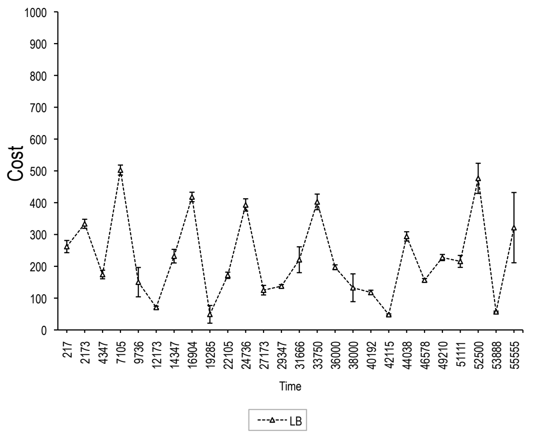
(c) Random (d) HSU+LB

Figure 4.2: Acceptance ratio over time

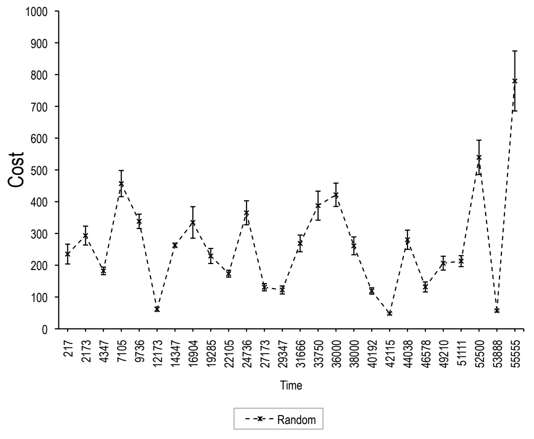
The average cost of provisioning a VN in terms of allocated resources is depicted in Figure 4.3. The cost is a function of the number of allocated virtual resources defined as(CHOWDHURY; RAHMAN; BOUTABA, 2012). Here, LB and Random goals result in higher provisioning costs compared to HSU. The reason is twofold. First, since LB and Random goals accept more requests, the number of allocated resources is also higher. Second, LB and Random goals use more resources when provisioning a single VN (because virtual resources tend to be spread) and more links are allocated. In HSU, in turn, because multiple virtual machines of the same VN are mapped to a smaller subset of available servers (some can even share the same server), the number of used links is minimized, reducing the overall cost of the VN. Again, when HSU and LB are combined, the provisioning cost approximates to the ones achieved by LB and Random.



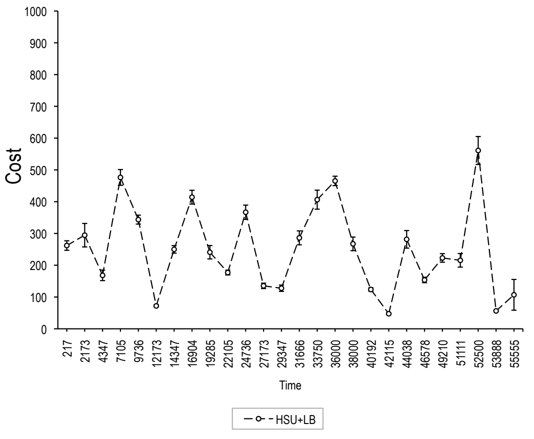
(a) HSU



(b) LB



(c) Random

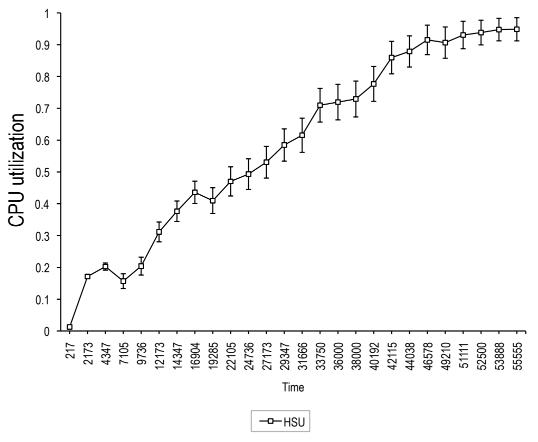


(d) HSU+LB

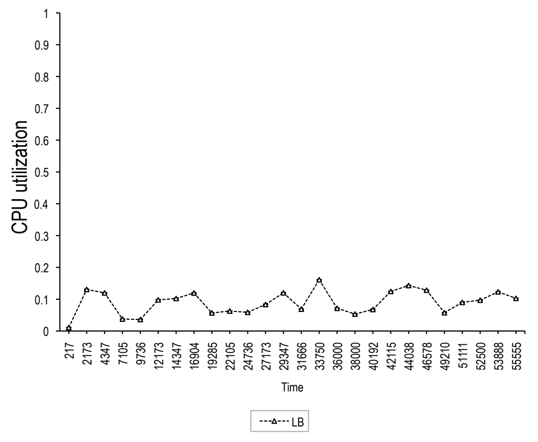
Figure 4.3: Provisioning cost over time

Figure 4.4 depicts the average CPU utilization over time achieved by the four paradigms. HSU achieves the highest CPU utilization, since it attempts to increase server utilization mapping virtual machines on the servers with the lowest residual capacity. LB and Random, in turn, go on the opposite way by choosing resources with high residual capacity and low utilization. It is possible to observe that LB has a big influence in the combined (HSU+LB) paradigm compared to HSU because CPU utilization values are closer to the ones achieved by LB than to the ones obtained when HSU is applied isolated.

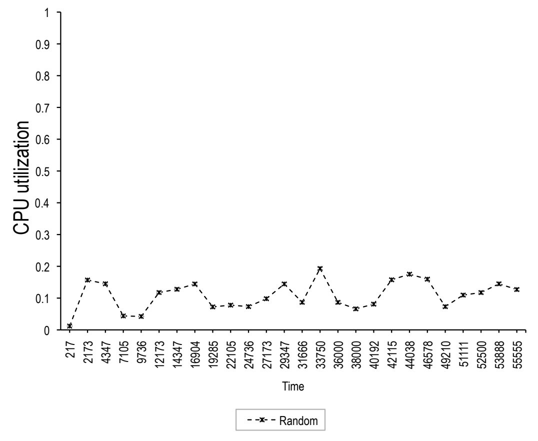
Finally, the average link utilization over time is illustrated in Figure 4.5. As stated before, HSU tends to use less links when provisioning VNs, which explains in part why HSU has the lowest link utilization. The other reason is the fact the HSU rejects more requests (Figure 4.2). By using less links, VNs provisioned with HSU are "cheaper", as discussed in the cost comparison. The Random objective has slightly superior performance compared to LB and HSU+LB. The similarity between link utilization achieved by both LB and HSU+LB confirms again that LB dominates over HSU, even in the combined paradigm. LB has the major influence even in the combined paradigm because the number of physical machines (24) is more than twice the number of virtual machines of a request (10). Therefore, LB will likely find a suitable machine to host a VM for all requests. In a more restrictive scenario, *e.g.*, few physical machines and many VMs to be requested, there is no guarantee that LB would still dominate over HSU.



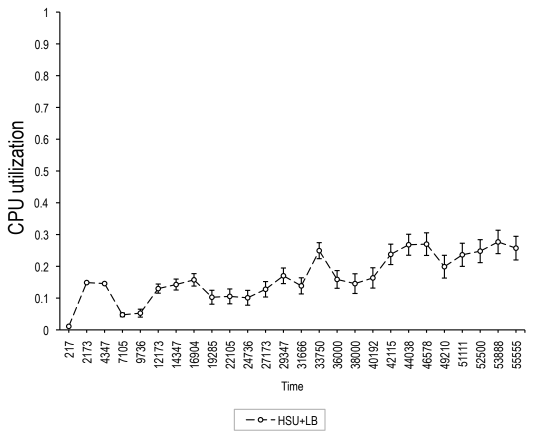
(a) HSU



(b) LB

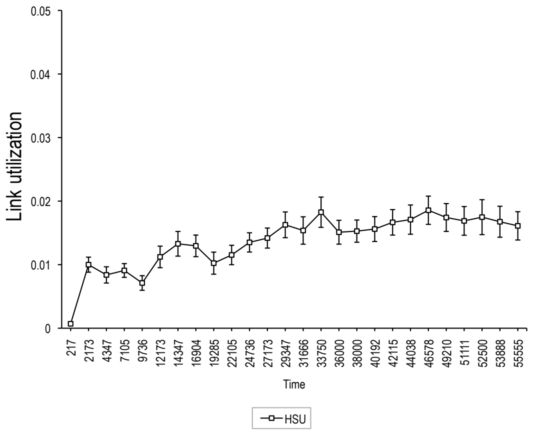


(c) Random

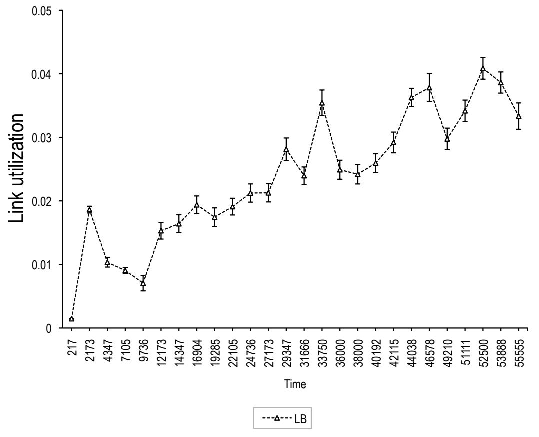


(d) HSU+LB

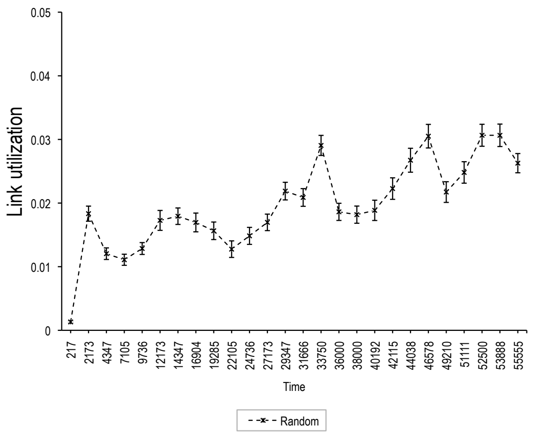
Figure 4.4: CPU utilization over time



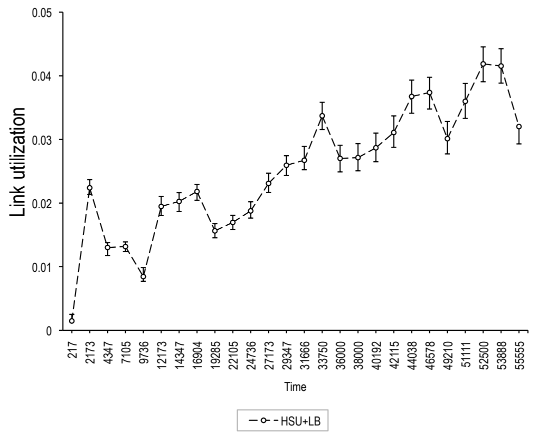
(a) HSU



(b) LB



(c) Random



(d) HSU+LB

Figure 4.5: Link utilization over time

4.3 Evaluation - paradigm efficiency analysis

In this section we evaluate the efficiency of an allocation paradigm in terms of application performance and investigate the benefits of paradigm switching. To determine the efficiency of a paradigm we use the VN performance prediction model proposed in Section 3.5. In order to obtain the efficiency of allocation paradigms, we have extended the simulator described in Section 4.1 to include our paradigm efficiency calculation and to allow paradigm switching on-the-fly.

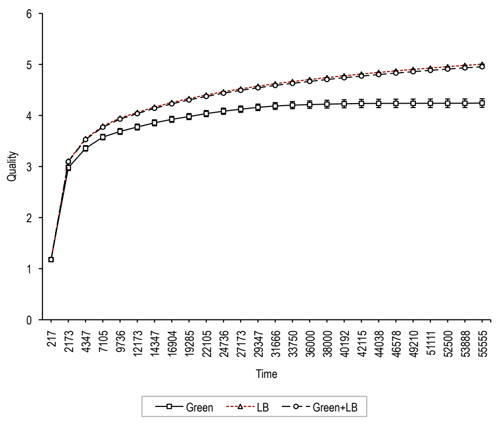
4.3.1 Metrics - paradigm efficiency analysis

We defined three additional metrics in this second evaluation. *Paradigm quality* reflects the quality of an allocation paradigm in terms of the score predicted for the provisioned VNs. In order to have higher precision and facilitate the interpretation of the results we plot paradigm quality as a harmonic series (HAVIL; DYSON, 2003), where each point is calculated by $\sum_{i=1}^{nreq} \frac{Q_i}{i}$, where $nreq$ is the number of the current VN request. *Number of rounds* is the number of allocation rounds necessary to complete the provisioning of a VN as defined in Section 1.1. *Paradigm efficiency* combines both previous metrics according to Equation 3.6. α and β have the same weight (1) indicating that the number of rounds and the paradigm quality have the same importance.

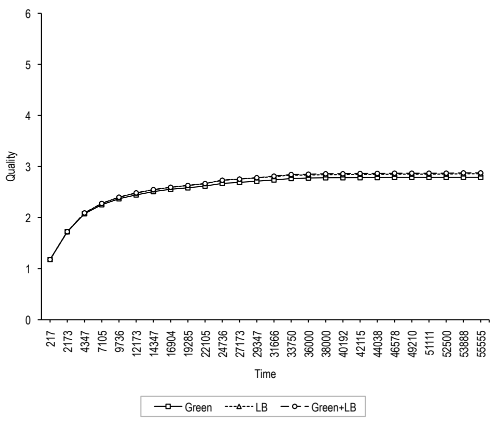
For each experiment, we evaluate two allocation paradigms composed of a single goal: Green and Load Balancing (LB) defined in Table 3.3 and a combination of the two. We also vary the size of the allocation window from 1 to 3. We also investigate the impact of switching from one paradigm (Green) to another (LB) during the simulation. In the paradigm switching evaluation, actions of one goal are scheduled and executes at each round. In the next round, actions of the other goal are included in the window and the process continues until the provisioning is complete. The maximum number of rounds for each request is defined as the number of requested resources over the size of the allocation window (Equation 3.1). The number of tiles of each request is calculated by dividing the number of request machines (m^v) by 7, which is the size of one tile.

4.3.2 Results - paradigm efficiency analysis

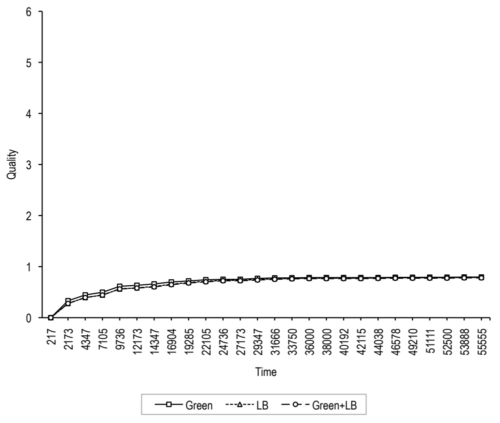
The quality of the paradigms composed by the Green, LB, and the combination of Green and LB goals is depicted in Figure 4.6. It is possible to observe that the paradigm quality is better for small-sized paradigm windows and gets worse when the paradigm window size is large. This happens because mapping a high number of resources in a single round reduces the *adaptability* of VN provisioning because the effect of VN arrivals and departures will only be perceived in the next round, which will happen when the actions of the current one are completed. As a consequence, the chances of taking advantage of better mapping alternatives are smaller for paradigms having large windows.



(a) s(W) = 1



(b) s(W) = 2

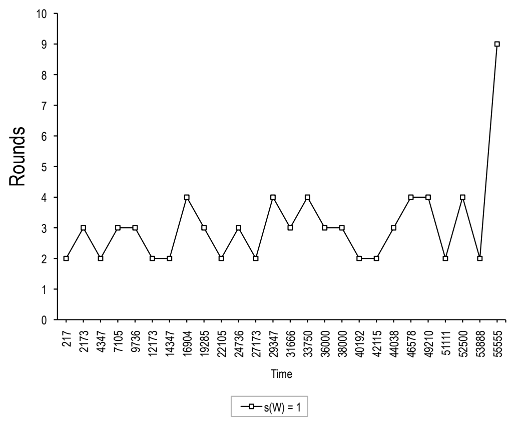


(c) s(W) = 3

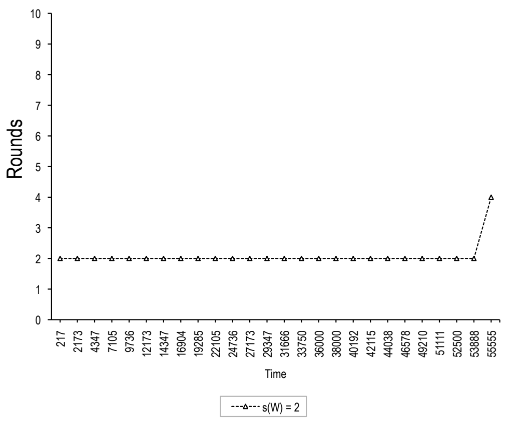
Figure 4.6: Paradigm quality

When s(W) = 1 the Green goal presents quality slightly inferior than LB and the combination of the two. This happens because LB always searches for the physical machine with the highest residual capacity. Since the initial configuration (*i.e.*, capacity) of physical machines is the same, LB selects a different machine in each turn, thus increasing the number of allocated tiles and, consequently, the paradigm quality. The quality of the combined paradigm approaches the values obtained by the LB goal isolated. The explanation for such behavior is that the size of the requests are relatively small (2 to 10 virtual machines) and since LB goal tends to select a different physical machine for each virtual one, the impact of such spreading is more evident for small VN requests. The performance of the goals tend to converge when the window size increases.

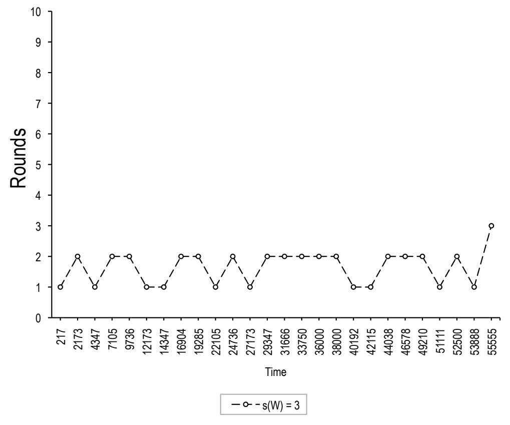
Figure 4.7 shows the average number of rounds required to complete the allocation of a VN with varying window sizes for the Green goal. For the sake of simplicity we use the paradigm approach to allocate the virtual machines, while the virtual links are mapped in a single step using the shortest path first (SPF) algorithm. When s(W) = 1 the number of rounds to complete the allocation is the exact number of virtual machines of the VN request. For s(W) = 2, the number of rounds is static because most of the virtual machines were allocated in only two rounds. As expected, when s(W) = 3 most VN requests are completed in two rounds and some (the smallest ones) in just one round. The number of rounds influences the time needed to allocate a VN. VNs allocated in few rounds (*i.e.*, large paradigm windows) are rapidly available to SPs while VN requests taking many rounds to be deployed have larger provisioning times.



(a) s(W) = 1



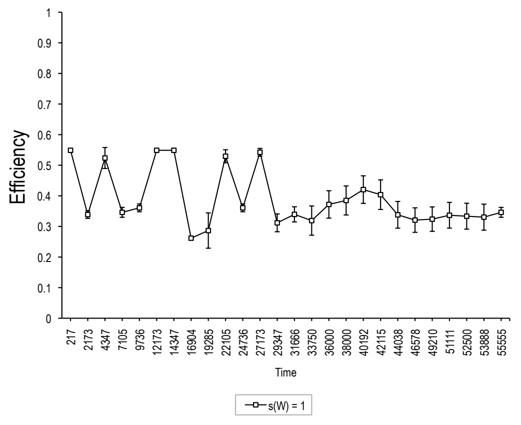
(b) s(W) = 2



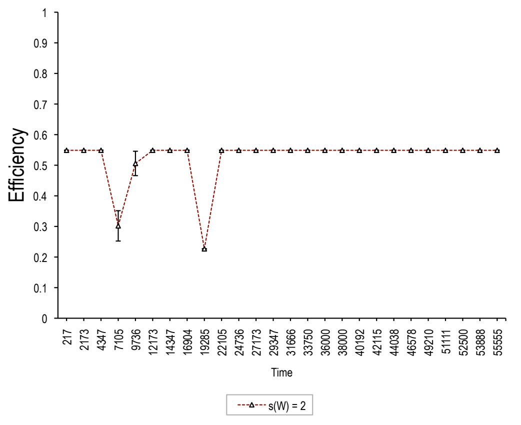
(c) s(W) = 3

Figure 4.7: Number of rounds - Green goal

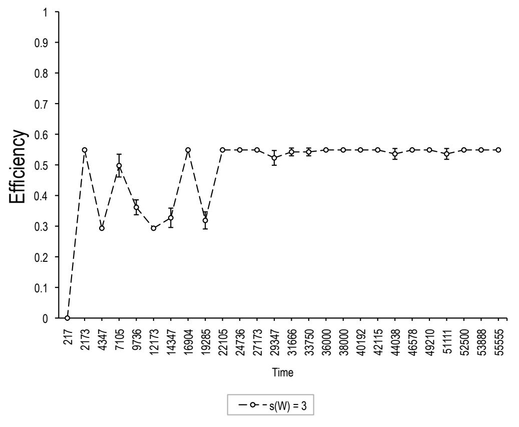
Figure 4.8 shows the efficiency of the Green goal, which is a function of both number of rounds and paradigm quality. When s(W) = 1, the efficiency presents a high variation. This can be explained by the fact that the number of rounds needed to allocate VNs also varies more for s(W) = 1 (Figure 4.7(a)). When s(W) = 2, the number of rounds is the same (2) most of the simulation (Figure 4.7(b)). This results in a stable efficiency of approximately 0.55 most of the time, except for two VN requests that did not achieve good quality. The efficiency presents a variation in the first part of the simulation for s(W) = 3 because the quality is the lowest in this case (Figure 4.6(c)). The efficiency becomes stable for all window sizes after some point of the simulation because the quality also stabilizes as shown in Figure 4.6.



(a) s(W) = 1



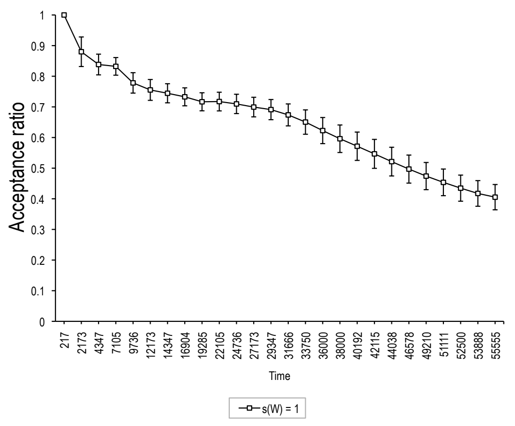
(b) s(W) = 2



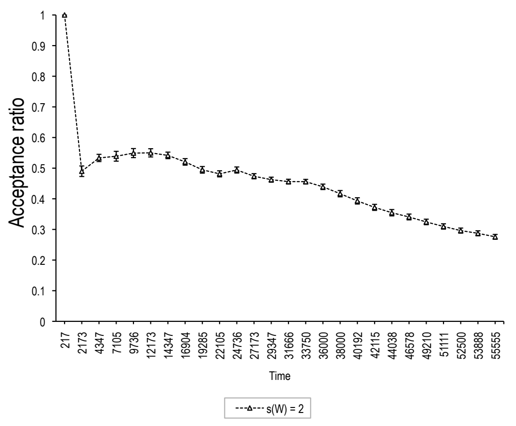
(c) s(W) = 3

Figure 4.8: Paradigm efficiency - Green goal

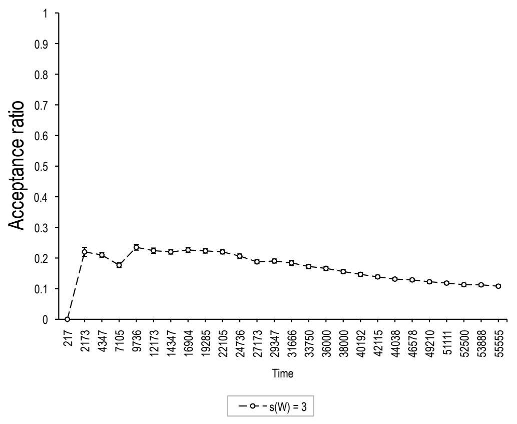
The acceptance ratio for the Green goal is illustrated in Figure 4.9. For s(W) = 1, acceptance ratio stays above 40% for the whole simulation time. When s(W) = 2, the acceptance ratio rapidly decreases after 2000 time units and stays between 55% and 27% for the rest of the simulation. The situation is even worse for s(W) = 3, when acceptance ratio is no higher than 25%, indicating that similar to paradigm quality, there is a clear association between the size of the window and the acceptance ratio of VN requests. VN requests allocated using paradigms with small windows are more adaptable and less unlikely to be rejected, while larger windows result in a higher number of rejected requests because the Green paradigm will tend to choose the same node in a single round and the capacity of the chosen node will run out fast.



(a) s(W) = 1



(b) s(W) = 2



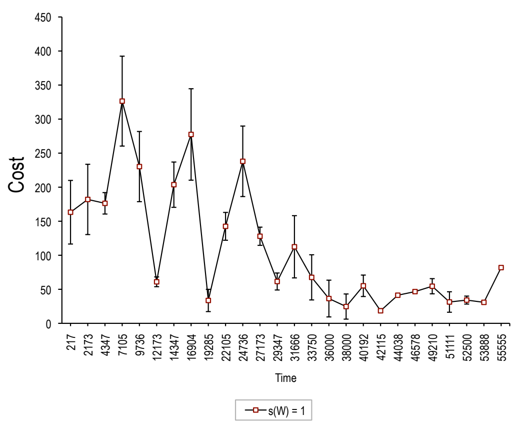
(c) s(W) = 3

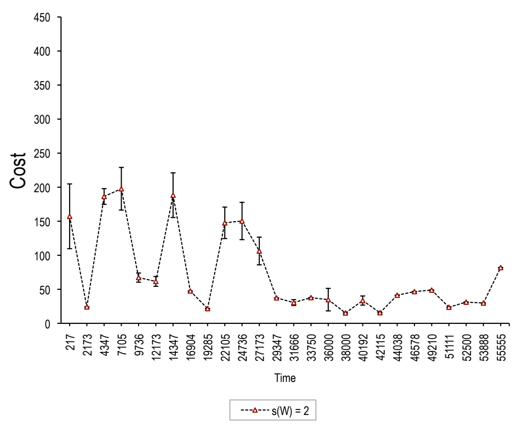
Figure 4.9: Acceptance ratio - Green goal

Figure 4.10 presents the average cost of provisioning a VN. During most of the simulation (time < 36000), the cost of allocated VNs is higher for s(W) = 1 compared to s(W) = 2, which, in turn, results in higher costs compared to s(W) = 3. This can be explained in part by the fact that the acceptance ratio is higher for s(W) = 1, which means that more resources are allocated and the overall cost for the InP increases. However, after 36000 time units, the cost of allocated VNs converges regardless the size of the paradigm window. This indicates that the size of the paradigm windows does not have a major influence in the overall performance of provisioning in the long term for the Green goal.

The number of rounds used by the LB goal to provision VNs is the same of the Green goal, which can be easily explained by the fact that the number of rounds depends only on the size of the paradigm window s(W) and not on the goals employed. s(W) defines the number of allocation actions that will be executed independent of which goal they belong to. The efficiency of the LB goal is illustrated in Figure 4.11.

The LB efficiency presents a higher variability when s(W) = 1 if compared to the Green goal. This happens because the quality of LB goal is higher compared to Green. Even if the efficiency suffers a momentary reduction due to an increase in the number of rounds, LB is able to produce good quality VNs, thus rapidly increasing the overall efficiency. When s(W) = 2 or s(W) = 3 the LB efficiency approximates the values achieved by Green because the quality of the goals tend to converge as the paradigm window gets larger (Figure 4.6).


(a) s(W) = 1


(b) s(W) = 2

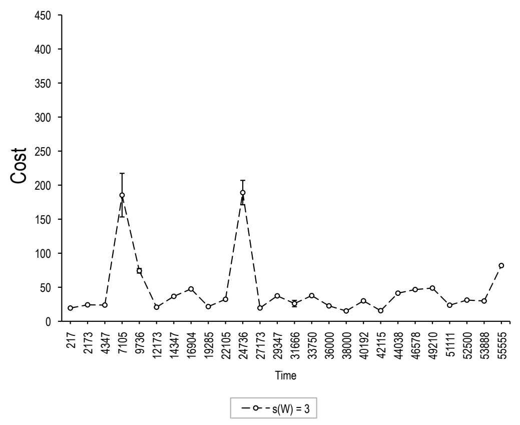
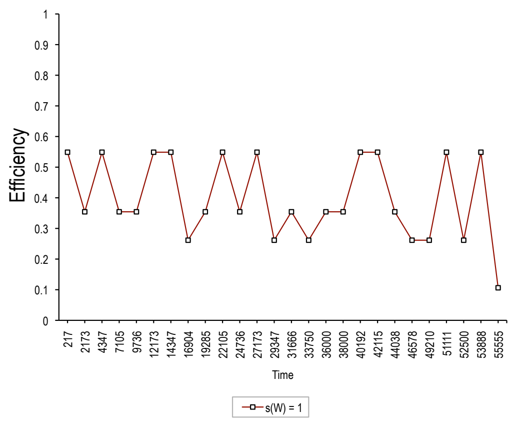

(c) s(W) = 3

Figure 4.10: Provisioning cost - Green goal

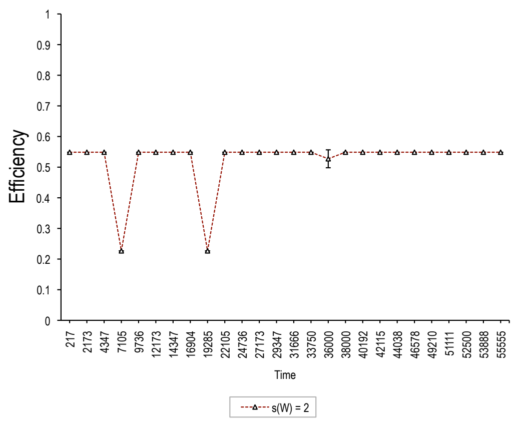
Acceptance ratio for the LB goal is shown in Figure 4.12. s(W) = 1 provides the best results where all VNs are successfully allocated. This can be explained by the fact that LB tends to select a different machine with high residual capacity in each round, increasing the chances of accepting the VN request. For s(W) = 2, the acceptance ratio stays between 55% and 30%, similar to the results of the Green goal, indicating that performance of allocation paradigms tend to converge when the window size increases. The same holds when s(W) = 3.

The cost of VN provisioning for the LB goal is presented in Figure 4.13. Unlike Green goal, the cost of VNs provisioned with the LB goal when s(W) = 1 is higher than s(W) = 2 and s(W) = 3 for most of the simulation. This reflects the fact that more VNs are allocated when s(W) = 1 and, consequently, the total InP cost increases because cost is defined in terms of allocated resources (*i.e.*, virtual machines and links) per VN request. Similar to acceptance ratio, the average cost of allocated VNs for s(W) = 2 and s(W) = 3 converge after 36000 time units and is comparable to the cost obtained by the Green goal.

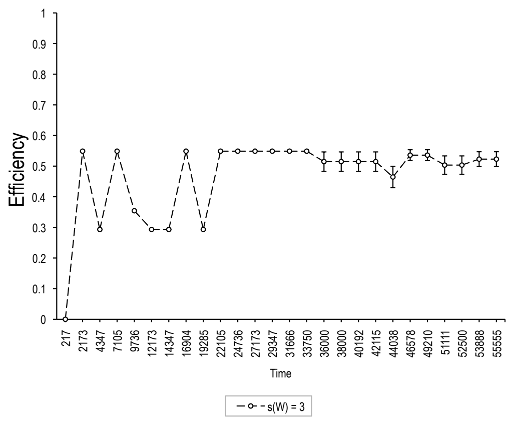
Figure 4.14 shows the paradigm quality when s(W) = 1 and the current allocation paradigm is switched during simulation time. In our experiments, the used thresholds to switch between paradigms were defined manually (half of the simulation time). However, our approach is still valid even if we consider more sophisticated techniques to estimate these thresholds. The quality of the allocation in the first half of the simulation stays between 1.2 and 4.2, which reflects the performance of the Green goal (see Figure 4.6).



(a) s(W) = 1



(b) s(W) = 2



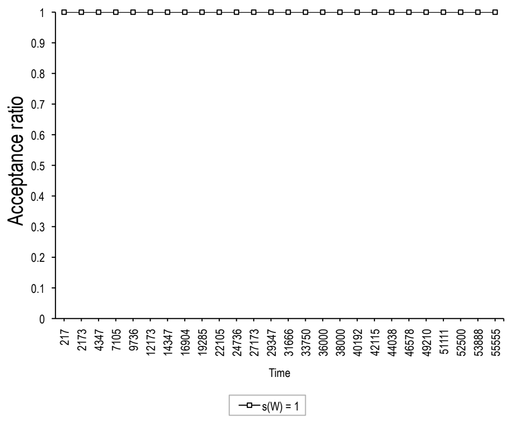
(c) s(W) = 3

Figure 4.11: Paradigm efficiency - Load balancing goal

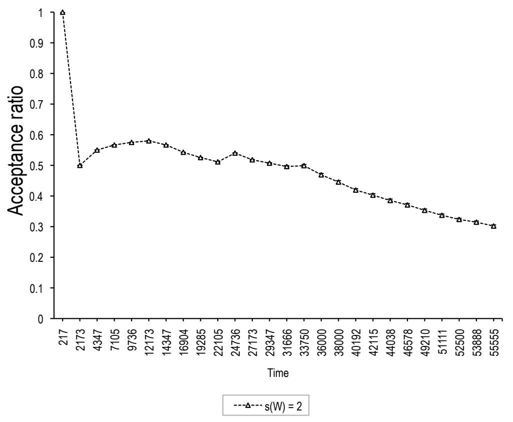
After that, the quality increases and stays between 4.5 and 4.7, which are close to the values obtained by the LB goal, which shows that changing between goals can improve paradigm quality.

The paradigm efficiency when paradigm switch occurs is illustrated in Figure 4.15. When s(W) = 1, the paradigm efficiency is exactly the efficiency obtained by the Green goal in the first part of the simulation. After the paradigm switches from Green to LB, the efficiency resembles the values achieved by the LB goal. The same behavior can be observed for s(W) = 2 and s(W) = 3.

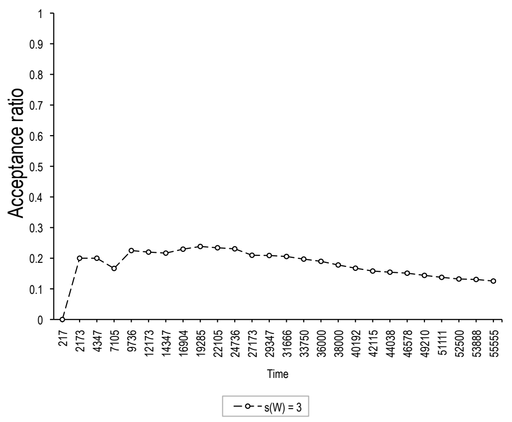
The acceptance ratio for paradigm switching is shown in Figure 4.16. The impact of paradigm switching is more evident for s(W) = 1. After paradigm switching, the acceptance ratio starts to increase because LB offers very high acceptance ratio (Figure 4.12). The provisioning cost is depicted in Figure 4.17. Again, for s(W) = 1 the cost does not reduce as in Green goal (see Figure 4.10) after paradigm switching. Instead, the average provisioning cost stays higher than s(W) = 2 and s(W) = 3, confirming the impact of changing between goals during VN provisioning on the InP revenue.



(a) s(W) = 1

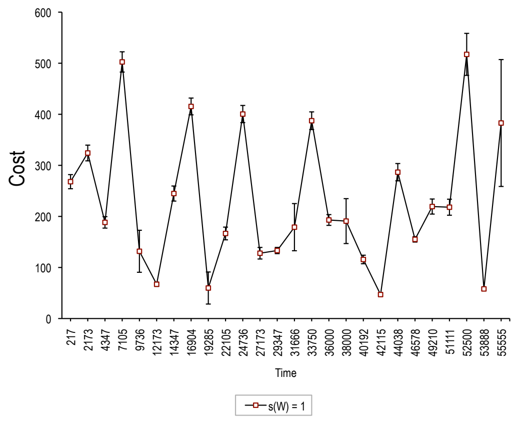


(b) s(W) = 2

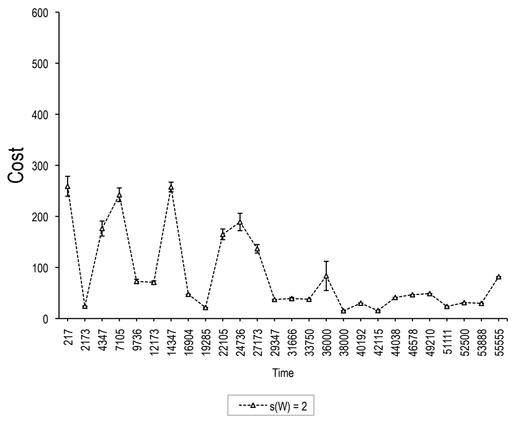


(c) s(W) = 3

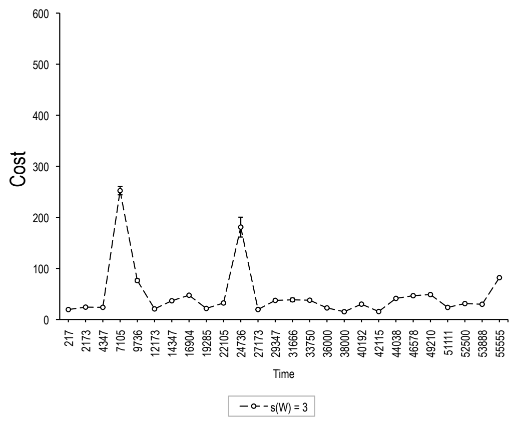
Figure 4.12: Acceptance ratio - Load balancing goal



(a) s(W) = 1

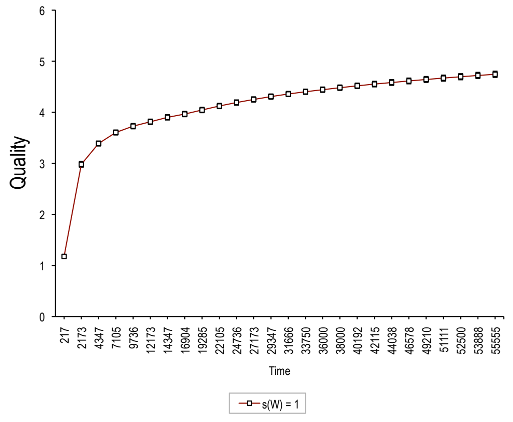


(b) s(W) = 2

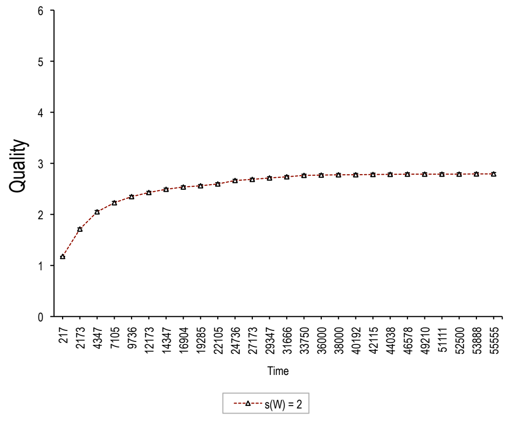


(c) s(W) = 3

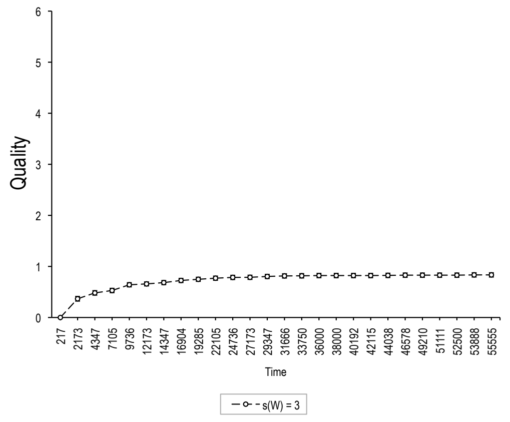
Figure 4.13: Provisioning cost - Load balancing goal



(a) s(W) = 1

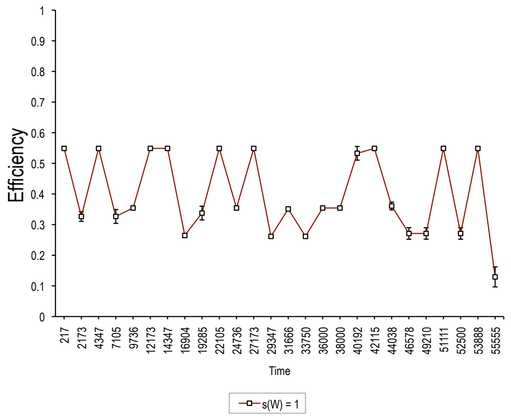


(b) s(W) = 2

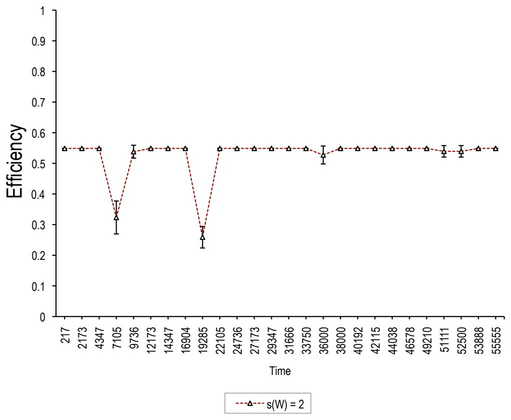


(c) s(W) = 3

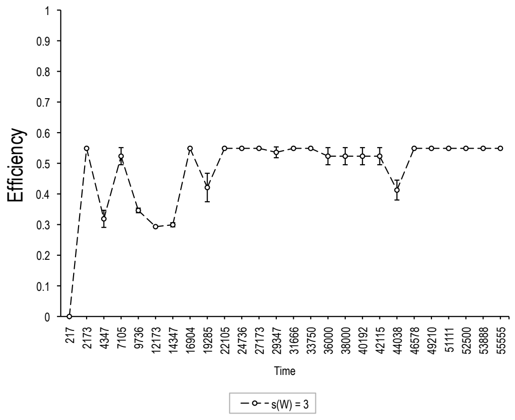
Figure 4.14: Paradigm quality - Paradigm switching



(a) s(W) = 1

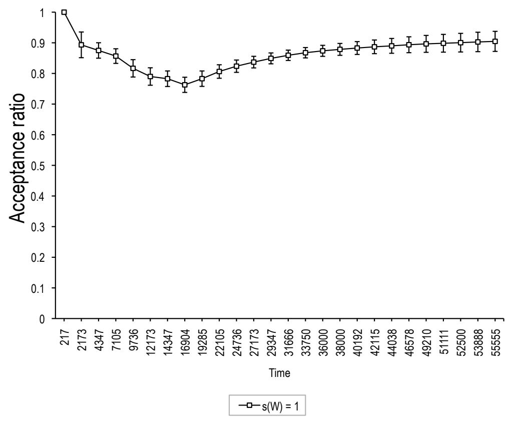


(b) s(W) = 2

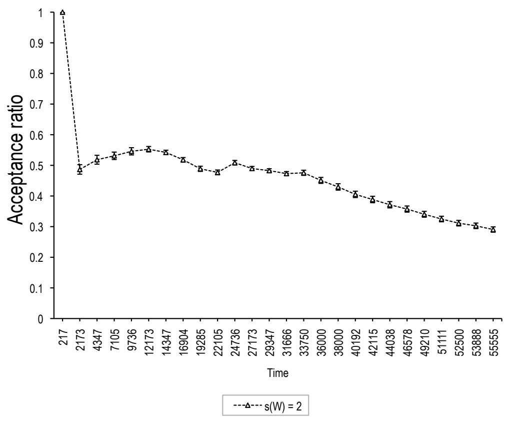


(c) s(W) = 3

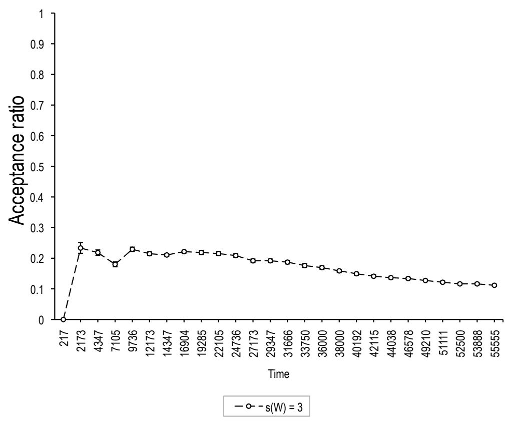
Figure 4.15: Paradigm efficiency - Paradigm switching



(a) s(W) = 1

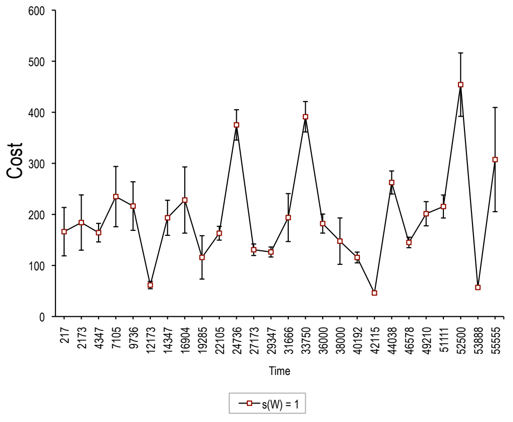


(b) s(W) = 2

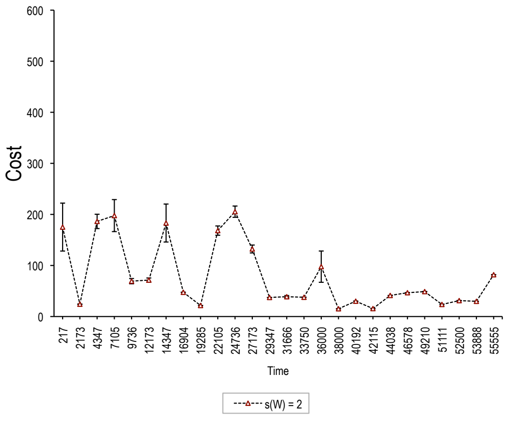


(c) s(W) = 3

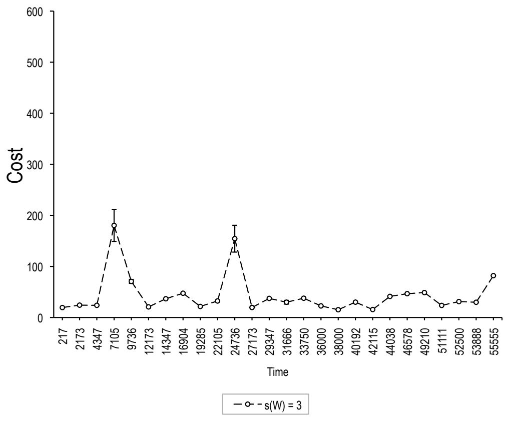
Figure 4.16: Acceptance ratio - Paradigm switching



(a) s(W) = 1



(b) s(W) = 2



(c) s(W) = 3

Figure 4.17: Provisioning cost - Paradigm switching

4.4 Summary

In this chapter, we have evaluated our paradigm-based provisioning framework. The evaluation was divided in two parts. The first one compares different goals in terms of acceptance ratio, provisioning cost, and resource utilization. The second part evaluates the efficiency of allocation paradigms (and associated goals) analyzing different paradigm configurations (*i.e.*, window size) and including new metrics such as paradigm quality, number of rounds, and paradigm efficiency.

Regarding paradigm goals comparison, it is possible to conclude that HSU goal is better suited for bandwidth-limited scenarios and useful to allocate VNs hosting network-sensitive applications. HSU uses less links, reducing the occurrence of bottlenecks in the network. On the other hand, LB is appropriate for best-effort scenarios and for VNs that need high availability (*i.e.*, VNs hosting critical applications) because resources are spread over the network. LB also results in increased revenue for InPs, since more VNs are allocated when LB is employed (isolated or combined). When HSU and LB are used together, provisioned VNs share characteristics of both goals, although LB has a larger influence on the achieved performance.

When analyzing paradigm efficiency, the LB goal presents better performance in terms of paradigm quality when compared to the Green goal. This happens because LB tries to spread VN requests over a high number of physical resources, thus increasing the acceptance ratio, which reflects in the quality of the provisioned VNs. However, this comes at the cost of more expensive VNs because more resources are allocated by LB compared to Green. When s(W) = 1 the number of rounds required to provision VN is higher, which means that VNs will take longer to be fully allocated. Paradigm quality can also be helpful for InP operators to decide on changing between paradigms and improve VN provisioning.

When we increase the window size to 2 we observe that the quality of the provisioned VNs decreases. This reflects in the reduction in the acceptance ratio of the provisioned VNs. The explanation for lower quality of the allocated VNs when the window size is higher relies on the fact that simultaneous mapping of multiple resources increases the chances of failed requests. On the other hand, the number of rounds is smaller and cost also reduces, which results in cheaper VNs that are rapidly deployed. This behavior is more evident when we increase the window size to 3.

Paradigm efficiency combines both paradigm quality and number of rounds. When s(W) = 1 the paradigm efficiency presents high variability. This happens because the number of rounds also changes more often in this case. For s(W) = 2, the number of rounds is stable most of time, which reflects in the paradigm efficiency. The quality reduction observed for s(W) = 3 also impacts the overall paradigm efficiency, indicating that paradigm quality and number of rounds have to be considered together.

5 CONCLUSIONS

The investigation carried out during this thesis demonstrated the feasibility of the our proposed application-aware virtual network provisioning approach. Specifically, our proposal is based on the hypothesis that **"Limitations in virtual network provisioning can be reduced through the use of *allocation paradigms*"**. Based on the hypothesis, we proposed a provisioning framework based on the concept of allocation paradigms that enables adaptive allocation in virtualized networks. In addition to allowing InPs to consider different goals in VN provisioning, our proposal empowers SPs to express high-level requirements for their VNs, which are used by InP operators to define how a VN is allocated on the physical substrate. Furthermore, multiple objectives can be considered when allocating a single VN, enabling multi-tenancy and allowing a large number of diverse applications to coexist in an NVE.

Our paradigm-based provisioning represents a novel way of looking at the VN provisioning problem. Instead of dealing with complex analytical VN provisioning models that may be a hurdle for many InP operators, allocation paradigms can be easily defined and do not require advanced expertise in order to be deployed. Besides, an important characteristic of our proposal is its *adaptability*, that is, the capacity of capturing the dynamics (*i.e.*, VN arrivals and departures) of the NVE and respond rapidly to them. This is achieved by splitting a VN request in several parts that are allocated independently. This *mapping in parts* represents another break with current VN provisioning approaches that consider VN mapping as a single, atomic operation.

Another noteworthy feature of our approach is to give the opportunity to the SPs to have an active role in VN provisioning. SPs can specify desired properties for their VNs that are considered by InPs in VN provisioning, which usually is not possible in most provisioning solutions deployed today. The InP is also able to change the current paradigm on-the-fly to reflect a new objective or to better support the requirements of its customers.

In order to evaluate the efficiency of allocation paradigms, we formulated a VN computation model, based on multiple linear regression, which considers all the applications running in a VN and whose output can be used by InP operators to decide whether to change a current allocation paradigm. The model resulted in the definition of a paradigm quality metric that reflects the aggregated performance of the applications running in a VN. In addition to paradigm quality, the number of rounds necessary to complete the provisioning of a VN has a significant impact on the overall efficiency of the paradigm and has to be considered accordingly.

The evaluation aimed at analyzing two different dimensions of our paradigm-based provisioning approach. In the first evaluation, we demonstrated the feasibility of our approach to accommodate multiple goals and compared the performance of different paradigms. When multiple goals are combined together, some may prevail over the others depending on factors such as the paradigm window size and the amount of available physical resources. In the second evaluation, we analyzed the efficiency of different paradigms considering the combined performance of applications running in a VN, *i.e.*, the quality, and verified the impact of changing between paradigms during VN provisioning.

5.1 Answers to the Research Questions

The motivation behind the definition of the research questions was to define the main points to be analyzed during the investigation of the hypothesis and to establish the roadmap to achieve the contributions of this thesis. Therefore, the description below summarizes and highlights answers to each research question.

Research question I. *How allocation paradigms can improve virtual network provisioning?*

Answer: Allocation paradigms improve virtual network provisioning by adding flexibility to resource allocation. Different allocation strategies based on application requirements and on InP objectives can be designed using paradigms. Besides, allocation paradigms allow rapid adaptation to the changes that can occur in the substrate and in the allocated VNs through the use of varied-size windows. Furthermore, allocation paradigms allow SPs to participate actively in the provisioning of their VNs by indicating high-level characteristics for VNs that may influence the paradigm selection.

Research question II. *What methods could be employed to calculate the efficiency of an allocation paradigm*

Answer: The efficiency of allocation paradigms can be calculated by models that capture and quantify the performance of applications running in the VNs. By using a simple model, we have defined a new metric, called paradigm quality, that reflects the combined performance of all applications deployed in the VNs allocated using a specific paradigm. The number of rounds taken to complete the provisioning of a VN also has a high influence on the overall performance of VN provisioning because it is related to VN provisioning time. The InP can adjust these two factors (*i.e.*, paradigm quality and number of rounds) according to its priorities. Acceptance ratio can also be a good indicator of the performance of a paradigm from the InP perspective.

Research question III. *What are the possible disadvantages of providing a high level of flexibility to VN provisioning?*

Answer: Although adding flexibility to VN provisioning has several benefits, there are some situations where such advantages may not be present and there are scenarios where the use of allocation paradigms can lead to inefficient VN allocation. For example, small NVEs where VNs are active for long periods and similar VN requests will probably not benefit of splitting a VN request in several parts because the resulting mapping will most likely be the same as if the VN was allocated in a single step at the cost of more rounds, and, consequently, taking longer.

The InP operator is free to choose any goals to compose a paradigm and he/she may end up choose conflicting goals (*e.g.*, green and load balancing) that undermine the performance of each other. The InP operator can handle such situations, for example, by adjusting the paradigm window sizes. Also, an ongoing VN request that cannot complete because there are not available resources in an allocation round will become a partial allocated VN, potentially increasing the fragmentation of the substrate.

5.2 Contributions

The contributions of this thesis can be divided into: conceptual and specific ones. Conceptual contributions could be identified from the investigations of the literature and the experiences gathered during the development of thesis. In contrast, specific contributions are associated with individual solutions developed in this thesis. Both contributions are listed as follows.

- Conceptual contributions are:
  - Identification of management requirements of NVEs and definition of a conceptual NVE management model. The model provides an holistic view of how management is tackled in modern NVEs and helped us to place our proposal in the NVE management spectrum.
  - This thesis presents a survey on NVE management. Prominent proposals related to NVE management are reviewed and the research challenges in this context are discussed. By analyzing the open challenges, it was possible to identify and explore a new opportunity to tackle the VN provisioning problem in this thesis.
- Specific contributions are:
  - Introduction of a new and flexible provisioning framework based on the concept of allocation paradigms that allows multiple InP and SP objectives to be considered during VN provisioning. Allocation paradigms also allow rapid adaptation of provisioning actions to the dynamic nature of NVEs by mapping VNs in parts.
  - Specification of a policy language to allow InP operators to create allocation paradigms and associate paradigms with goals and allocation actions
  - Definition of a VN computation model to quantify the efficiency of allocation paradigms. The efficiency of an allocation paradigm can be defined in terms the performance of the applications running in the deployed VNs and the number of rounds necessary to complete the allocation of a VN. Two new metrics were defined to this end: paradigm quality and paradigm efficiency.

The concept of allocation paradigms is a novel approach for provisioning virtual networks. Although such approach may add some level of complexity to resource allocation, InPs can offer value-added services to their customers and increase their revenue by using such flexible allocation. The advantage of VN virtualization is not reduced because InPs do not need to know which applications will run on the VNs.

5.3 Future work

The investigation developed in this thesis leads to the identification of further opportunities for research. These opportunities are described in this section as future work.

Analysis of the cost of using allocation paradigm vs. resource migration - a further analysis comparing the performance of our paradigm-based provisioning approach with the current model of reconfiguring VNs, typically using migrations, is important to verify when the cost of migration justifies the need for the paradigm concept.

Concurrency and fault tolerance - Concurrency and fault tolerance issues constitute good research opportunities. Proper strategies to handle the situation where two VN requests attempt to use the same resource at the same time, rollback mechanisms to release partially allocated VNs that cannot be completed due to absence of resources, and appropriate methods to allow allocation paradigms to react in case of failures require further investigation.

Goal refinement and handling of conflicts - Allocation paradigms can be seen as a group high-level policies that are translated to low-level actions. In network and systems management, transforming high-level policies to low-level ones is commonly referred to goal or policy refinement (MOFFETT; SLOMAN, 1993) (BANDARA et al., 2004). Therefore, policy refinement techniques developed for other contexts can be used to improve the process of mapping InP and SP goals to paradigms. In addition to that, solutions aimed at detecting and solving policy conflicts (LUPU; SLOMAN, 1999) (MOFFETT; SLOMAN, 1994) can be used to handle the cases where conflicting goals coexist in the same paradigm.

Application performance models - In this thesis we proposed a simple linear regression model to quantify the performance of the applications that are hosted in a VN. Other models (*e.g.*, queueing, autoregressive) to reflect the performance of a higher number of applications can improve the paradigm efficiency analysis.

The list of future work presented above represents the major opportunities of research that can be directly derived from the work presented in this thesis. Nevertheless, other opportunities that require further investigation include extending allocation paradigms to handle inter-domain VN provisioning, developing monitoring approaches to collect application performance data, and using additional constructs from existing policy languages to improve paradigm definition.

REFERENCES

4WARD. **The FP7 4WARD Project**. Available at http://www.4ward-project.eu/. Accessed in January 2013.

AL-FARES, M.; LOUKISSAS, A.; VAHDAT, A. A Scalable, Commodity Data Center Network Architecture. In: ACM SIGCOMM. **Anais...** [S.l.: s.n.], 2008.

ANDERSON, T. et al. Overcoming the Internet Impasse through Virtualization. **Computer**, [S.l.], v.38, n.4, p.34–41, April 2005.

AUTOI. **autoi - Autonomic Internet**. Available at http://ist-autoi.eu. Accessed in January 2013.

BALLANI, H. et al. Towards Predictable Datacenter Networks. In: ACM SIGCOMM. **Anais...** [S.l.: s.n.], 2011.

BANDARA, A. et al. A Goal-based Approach to Policy Refinement. In: IEEE INTERNATIONAL SYMPOSIUM ON POLICIES FOR DISTRIBUTED SYSTEMS AND NETWORKS. **Anais...** [S.l.: s.n.], 2004. p.229–239.

BARI, M. et al. Data Center Network Virtualization: a survey. **IEEE Communications Surveys and Tutorials**, [S.l.], v.15, n.2, p.909–928, 2012.

BOUTABA, R.; GOLAB, W.; IRAQI, Y. Lightpaths on Demand: a web-services-based management system. **IEEE Communications Magazine**, [S.l.], v.42, n.7, p.101–107, 2004.

CARAPINHA, J.; JIMÉNEZ, J. Network Virtualization: a view from the bottom. In: ACM SIGCOMM WORKSHOP ON VIRTUALIZED INFRASTRUCTURE SYSTEMS AND ARCHITECTURES. **Anais...** [S.l.: s.n.], 2009. p.73–80.

CHENG, X. et al. Virtual Network Embedding Through Topology Awareness and Optimization. **Computer Networks**, [S.l.], v.56, n.6, p.1797 – 1813, 2012.

CHERKAOUI, O.; HALIMA, E. Network Virtualization under User Control. **International Journal of Network Management**, [S.l.], v.18, n.2, p.147–158, Mar. 2008.

CHOWDHURY, M. **ViNE-Yard**. [S.l.: s.n.], 2011.

CHOWDHURY, M.; RAHMAN, M.; BOUTABA, R. ViNEYard - Virtual Network Embedding Algorithms With Coordinated Node and Link Mapping. **IEEE/ACM Transactions on Networking**, [S.l.], v.20, n.1, p.206 –219, feb. 2012.

CHOWDHURY, M.; SAMUEL, F.; BOUTABA, R. PolyViNE: policy-based virtual network embedding across multiple domains. In: ACM SIGCOMM WORKSHOP ON VIRTUALIZED INFRASTRUCTURE SYSTEMS AND ARCHITECTURES. **Anais...** [S.l.: s.n.], 2010. p.49–56.

CHOWDHURY, N.; BOUTABA, R. Network Virtualization: state of the art and research challenges. **IEEE Communications Magazine**, [S.l.], v.47, n.7, p.20–26, July 2009.

CLAYMAN, S.; GALIS, A.; MAMATAS, L. Monitoring Virtual Networks with Lattice. In: IEEE/IFIP NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM WORKSHOPS. **Anais...** [S.l.: s.n.], 2010. p.239–246.

CLEMM, A. **Network Management Fundamentals**. [S.l.]: Cisco Press, 2006.

COMER, D. E. **Network systems design using network processors**: intel 2xxx version. [S.l.]: Prentice-Hall, 2005.

CORREIA, L. et al. **Architecture and design for the future internet**: 4ward project. [S.l.]: Springer, 2011.

DAITX, F.; ESTEVES, R. P.; GRANVILLE, L. Z. On the Use of SNMP as a Management Interface for Virtual Networks. In: IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT. **Anais...** [S.l.: s.n.], 2011. p.177–184.

ESPOSITO, F. **A Policy-Based Architecture for Virtual Network Embedding**. 2013. Tese (Doutorado em Ciência da Computação) — Graduate School of Arts and Sciences, Boston University.

ESTEVES, R. P. et al. Paradigm-Based Adaptive Provisioning in Virtualized Data Centers. In: IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT. **Anais...** [S.l.: s.n.], 2013. p.169–176.

ESTEVES, R. P. et al. Evaluating Allocation Paradigms for Multi-objective Adaptive Provisioning in Virtualized Networks. In: IEEE/IFIP NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM. **Anais...** [S.l.: s.n.], 2014. p.1–9.

ESTEVES, R. P.; GRANVILLE, L. Z.; BOUTABA, R. On the Management of Virtual Networks. **IEEE Communications Magazine**, [S.l.], v.51, n.7, p.2–10, July 2013.

FARINACCI, D. et al. **RFC 2784 - Generic Routing Encapsulation (GRE)**. [S.l.]: Internet Engineering Task Force, 2000.

FRINCU, M.; CRACIUN, C. Multi-objective Meta-heuristics for Scheduling Applications with High Availability Requirements and Cost Constraints in Multi-Cloud Environments. In: IEEE INTERNATIONAL CONFERENCE ON UTILITY AND CLOUD COMPUTING. **Anais...** [S.l.: s.n.], 2011. p.267 –274.

GALIS, A. et al. Management Architecture and Systems for Future Internet Networks. **Towards the Future Internet–A European Research Perspective**, [S.l.], p.112–122, 2009.

GENI. **Global Environment for Network Innovations (GENI)**. Available at http://www.geni.net. Accessed in January 2013.

GILADI, R. **Network Processors: architecture, programming, and implementation**. [S.l.]: Morgan Kaufmann, 2008.

GRANVILLE, L. Z.; ESTEVES, R. P.; WICKBOLDT, J. A. Virtualization in the cloud. In: **Cloud services, networking and management**. [S.l.: s.n.], 2015. p.1–28.

GREENBERG, A. et al. VL2 - A Scalable and Flexible Data Center Network. In: ACM SIGCOMM. **Anais...** [S.l.: s.n.], 2009.

GUO, C. et al. BCube: a high performance, server-centric network architecture for modular data centers. In: ACM SIGCOMM. **Anais...** [S.l.: s.n.], 2009.

GUO, C. et al. SecondNet - A Data Center Network Virtualization Architecture with Bandwidth Guarantees. In: INTERNATIONAL CONFERENCE ON EMERGING NETWORKING EXPERIMENTS AND TECHNOLOGIES. **Anais...** [S.l.: s.n.], 2010.

HAVIL, J.; DYSON, F. The Harmonic Series. **Gamma: Exploring Eulerâs Constant**, [S.l.], p.21–25, 2003.

HEPPEL, A. **An Introduction to Network Processors**. Available at http://www.roke.co.uk/download/white_papers/network_processors_introduction.pdf. Accessed in January 2013.

HINDEN, R. **Virtual Router Redundancy Protocol (VRRP)**. [S.l.: s.n.], 2004.

HOUIDI, I. et al. Adaptive Virtual Network Provisioning. In: ACM SIGCOMM WORKSHOP ON VIRTUALIZED INFRASTRUCTURE SYSTEMS AND ARCHITECTURES. **Anais...** [S.l.: s.n.], 2010. p.41–48.

HOUIDI, I. et al. Virtual Network Provisioning Across Multiple Substrate Networks. **Computer Networks**, [S.l.], v.55, n.4, p.1011–1023, Mar. 2011.

IEEE. **Virtual Bridged Local Area Networks**. Available at http://standards.ieee.org/getieee802/download/802.1Q-2005.pdf. Accessed in January 2013.

IETF-POLICY. **Policy Languages Review - Policy Languages Interest Group**. [S.l.: s.n.], 2012.

IFIP. **Network and Service Management Taxonomy**. 2011.

ISLAM, S. et al. Empirical Prediction Models for Adaptive Resource Provisioning in the Cloud. **Future Generation Computer Systems**, [S.l.], v.28, n.1, p.155–162, 2012.

JEWELL, B. **Definitions of managed objects for the virtual router redundancy protocol**. [S.l.: s.n.], 2000.

JUNIPERNETWORKS. **The Juniper M Series Multiservice Edge Routers for IP/MPLS.** [S.l.: s.n.], 2009.

KHAN, A. et al. Network Virtualization: a hypervisor for the internet? **IEEE Communications Magazine**, [S.l.], v.50, n.1, p.136–143, January 2012.

KIM, M.-S.; LEON-GARCIA, A. Autonomic Network Resource Management Using Virtual Network Concept. In: MANAGING NEXT GENERATION NETWORKS AND SERVICES. **Anais...** [S.l.: s.n.], 2007. p.254–264. (LNCS, v.4773).

LI, J. et al. CloudOpt - Multi-goal Optimization of Application Deployments Across a Cloud. In: INTERNATIONAL CONFERENCE ON NETWORK AND SERVICE MANAGEMENT. **Anais...** [S.l.: s.n.], 2011. p.1 –9.

LUPU, E.; SLOMAN, M. Conflicts in Policy-based Distributed Systems Management. **IEEE Transactions on Software Engineering**, [S.l.], v.25, n.6, p.852–869, Nov 1999.

MATTOS, D. et al. OMNI: openflow management infrastructure. In: INTERNATIONAL CONFERENCE ON THE NETWORK OF THE FUTURE. **Anais...** [S.l.: s.n.], 2011. p.52–56.

MCKEOWN, N. et al. OpenFlow: enabling innovation in campus networks. **ACM Computer Communication Review**, [S.l.], v.38, n.2, p.69–74, 2008.

MITZENMACHER, M.; RICHA, A. W.; SITARAMAN, R. The Power of Two Random Choices: a survey of techniques and results. In: HANDBOOK OF RANDOMIZED COMPUTING. **Anais...** Kluwer, 2000. p.255–312.

MOFFETT, J. D.; SLOMAN, M. S. Policy conflict analysis in distributed system management. **Journal of Organizational Computing**, [S.l.], v.4, n.1, p.1–22, 1994.

MOFFETT, J.; SLOMAN, M. Policy Hierarchies for Distributed Systems Management. **IEEE Journal on Selected Areas in Communications**, [S.l.], v.11, n.9, p.1404–1414, Dec 1993.

NetFPGA. **The NetFPGA Project.** [S.l.: s.n.], 2009.

NG, W. et al. MIBlets - A Practical Approach to Virtual Network Management. In: IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT. **Anais...** [S.l.: s.n.], 1999. p.201 –215.

OPENVSWITCH. **Open vSwitch**: an open virtual switch. 2010.

PROTOGENI. **ProtoGENI**. Available at http://www.protogeni.net. Accessed in January 2013.

QEMU. **QEMU: open source processor emulator**. Available at http://wiki.qemu.org/Main$_{p}age.AccessedinJanuary$2013.

QUAGGA. **Quagga software routing suite**. [S.l.: s.n.], 2010.

R. **The R Project for Statistical Computing.** . [S.l.: s.n.], 2013.

RABBANI, M. G. et al. On Tackling Virtual Data Center Embedding Problem. In: IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT, Ghent, Belgium. **Anais...** [S.l.: s.n.], 2013.

RAO, J. et al. DynaQoS - Model-free Self-Tuning Fuzzy Control of Virtualized Resources for QoS Provisioning. In: INTERNATIONAL WORKSHOP ON QUALITY OF SERVICE. **Anais...** [S.l.: s.n.], 2011. p.31:1–31:9.

SAVI. **Smart Applications on Virtual Infrastructure**. [S.l.: s.n.], 2011.

SHAH, N. **Understanding Network Processors**. 2001. Dissertação (Mestrado em Ciência da Computação) — Department of Electrical Engineering and Computer Science, University of California, Berkeley.

SHERWOOD, R. et al. Can the Production Network Be the Testbed? In: USENIX SYMPOSIUM ON OPERATING SYSTEMS DESIGN AND IMPLEMENTATION. **Anais...** [S.l.: s.n.], 2010. p.1–6.

SPECVIRT. **Standard Performance Evaluation Corporation (SPECvirt_sc2010)**. [S.l.: s.n.], 2012.

STELZER, E. et al. **Virtual router management information base using SMIv2**. [S.l.: s.n.], 2003.

SZEGEDI, P. et al. With Evolution for Revolution: managing federica for future internet research. **IEEE Communications Magazine**, [S.l.], v.47, n.7, p.34–39, July 2009.

TURNER, J. **Towards a Diversified Internet**. Available at http://www.arl.wustl.edu/netv/contrib/diversifiedInternet.pdf. Accessed in January 2013.

VMMARK. **VMmark Benchmark**. [S.l.: s.n.], 2012.

VMMARK. **VMware VMmark Benchmarking Guide**. [S.l.: s.n.], 2012.

VMWARE. **VMWare**. Available at http://www.vmware.com. Accessed in January 2013.

VYATTA. **Vyatta open networking**. [S.l.: s.n.], 2010.

WANG, Y. et al. Virtual Routers on the Move - Live Router Migration as a Network-Management Primitive. **SIGCOMM Computer Communication Review**, [S.l.], v.38, p.231–242, Aug. 2008.

XEN. **The Xen Project**. Available at http://www.xenproject.org. Accessed in January 2013.

XORP. **XORP**: extensible open router platform. 2012.

YU, M. et al. Rethinking Virtual Network Embedding - Substrate Support for Path Splitting and Migration. **ACM Computer Communication Review**, [S.l.], v.38, n.2, p.17–29, April 2008.

ZAHEER, F.-E.; XIAO, J.; BOUTABA, R. Multi-Provider Service Negotiation and Contracting in Network Virtualization. In: IEEE/IFIP NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM. **Anais...** [S.l.: s.n.], 2010. p.471–478.

APPENDIX A - SCIENTIFIC PRODUCTION

A.1 Published Papers

1. RAFAEL PEREIRA ESTEVES, Lisandro Zambenedetti Granville, Mohamed Faten Zhani, Raouf Boutaba. **Evaluating Allocation Paradigms for Multi-Objective Adaptive Provisioning in Virtualized Networks**. 14th IFIP/IEEE Network Operations and Management Symposium (NOMS 2014), 5-9 May 2014, Krakow, Poland, pp. 1-9.

   - *Status.* Published.

2. RAFAEL PEREIRA ESTEVES, Lisandro Zambenedetti Granville, Raouf Boutaba. **On the Management of Virtual Networks**. IEEE Communications Magazine, v. 51, n. 7, pp. 80-88, July 2013. ISSN: 0163-6804.

   - *Status.* Published.

3. RAFAEL PEREIRA ESTEVES, Lisandro Zambenedetti Granville, Hadi Bannazadeh, Raouf Boutaba. **Paradigm-Based Adaptive Provisioning in Virtualized Data Centers**. 13th IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), 27-31 May 2013, Ghent, Belgium, pp. 169-176.

   - *Status.* Published.

4. Fábio Fabian Daitx, RAFAEL PEREIRA ESTEVES, Lisandro Zambenedetti Granville. **On the use of SNMP as a Management Interface for Virtual Networks**. 12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011), 23-27 May 2011, Dublin, Ireland, pp. 177-184.

   - *Status.* Published.

A.2 Other collaborations

1. Runxin Wang, RAFAEL PEREIRA ESTEVES, Lei Shi, Juliano Araujo Wickboldt, Brendan Jennings, Lisandro Zambenedetti Granville. **Network-aware Placement of Virtual Machine Ensembles using Effective Bandwidth Estimation**. 10th International Conference on Network and Service Management (CNSM 2014), 17-21 November 2014, Rio de Janeiro, Brazil.

   - *Status.* Published.

2. Juliano Araujo Wickboldt, RAFAEL PEREIRA ESTEVES, Márcio Barbosa de Carvalho, Lisandro Zambenedetti Granville. **Resource Management in IaaS Cloud Platforms made Flexible through Programmability**. Elsevier Computer Networks (COMNET), Special Issue on Communications and Networking in the Cloud, Volume 68 (2014), pp. 54-70, ISSN 1389-1286.

   - *Status.* Published.

3. Márcio Barbosa de Carvalho, RAFAEL PEREIRA ESTEVES, Guilherme da Cunha Rodrigues, Clarissa Cassales Marquezan, Lisandro Zambenedetti Granville, Liane Margarida Rockenbach Tarouco. **Efficient Configuration of Monitoring Slices for Cloud Platform Administrators**. 19th IEEE Symposium on Computers and Communications (ISCC 2014), 23-26 June 2014, Madeira, Portugal, pp. 1-7.

   - *Status.* Published.

4. Md. Faizul Bari, Raouf Boutaba, RAFAEL PEREIRA ESTEVES, Lisandro Zambenedetti Granville, Maxim Podlesny, Md. Golam Rabbani, Qi Zhang, Mohamed Faten Zhani. **Data Center Network Virtualization: A Survey**. IEEE Communications Surveys and Tutorials, v. 15, n. 2, pp. 909-928, Second Quarter 2013. ISSN: 1553-877X.

   - *Status.* Published.

5. Márcio Barbosa de Carvalho, RAFAEL PEREIRA ESTEVES, Guilherme da Cunha Rodrigues, Lisandro Zambenedetti Granville, Liane Margarida Rockenbach Tarouco. **A Cloud Monitoring Framework for Self-Configured Monitoring Slices Based on Multiple Tools (short paper)**. 9th International Conference on Network and Service Management (CNSM 2013), 14-18 October 2013, Zurich, Switzerland, pp. 180-184.

   - *Status.* Published.

6. Md. Golam Rabbani, RAFAEL PEREIRA ESTEVES, Maxim Podlesny, Gwendal Simon, Lisandro Zambenedetti Granville, Raouf Boutaba. **On Tackling Virtual Data Center Embedding Problem**. 13th IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), 27-31 May 2013, Ghent, Belgium, pp. 177-184.

   - *Status.* Published.

APPENDIX B - CAPÍTULO EM PORTUGUÊS

A investigação conduzida durante esta tese demonstrou a viabilidade da nossa proposta de aprovisionamento de redes virtuais orientado à aplicação. Especificamente, nossa proposta é baseada na hipótese de que **"Limitações no aprovisionamento de redes virtuais podem ser reduzidas através do uso de *paradigmas de alocação*"**. Com base nessa hipótese, foi proposto um arcabouço de aprovisionamento baseado no conceito de paradigmas de alocação que permite a alocação adaptativa de recursos em ambientes de rede virtualizados. Além de permitir que provedores de infraestrutura considerem diferentes objetivos durante o aprovisionamento de redes virtuais, a proposta permite que provedores de serviço expressem objetivos de alto-nível para as suas respectivas redes virtuais, que por sua vez são considerados pelos provedores de infraestrutura para definir como redes virtuais são mapeadas no substrato físico. Múltiplos objetivos podem ser considerados na alocação de uma única rede virtual, o que permite que diferentes provedores de serviço e um grande número de aplicações compartilhem o mesmo ambiente de virtualização de rede.

O aprovisionamento baseado em paradigmas de alocação representa uma forma inédita de se olhar para o problema de alocação de redes virtuais. Ao invés de ter que lidar com modelos analíticos complexos, o que pode se mostrar uma dificuldade para administradores de rede, paradigmas de alocação podem ser facilmente criados e não exigem conhecimentos avançados por parte dos administradores para serem utilizados. Ademais, uma importante característica da nossa proposta é a sua *adaptabilidade*, isto é, a capacidade de se capturar a dinâmica (*i.e.*, chegadas e partidas de redes virtuais) do substrato e reagir rapidamente às mesmas. Isso é conseguido pelo particionamento de redes virtuais em partes menores que são alocadas de forma independente. Este mapeamento *em partes* representa outra ruptura com as abordagens tradicionais de aprovisionamento, que realizam o mapeamento como uma operação única e indivisível.

Outra característica notável da nossa solução é a possibilidade de que os provedores de serviço tenham papel ativo no aprovisionamento de redes virtuais. Os provedores de serviço podem especificar propriedades desejadas para suas redes virtuais. Essas propriedades são consideradas pelos provedores de infraestrutura na alocação da rede virtual. Essas característica não é encontrada na grande maioria das soluções de aprovisionamento atuais. Os provedores de infraestrutura podem também mudar o paradigma atual de forma dinâmica para refletir um novo objetivo ou para atender melhor os requisitos dos provedores de serviço.

Para avaliar a eficiência de paradigmas de alocação, foi formulado um modelo de computação de redes virtuais, baseado em regressão linear múltipla, que considera todas as aplicações que rodam em uma rede virtual e cuja saída pode ser utilizada por administradores de rede para decidir quando mudar um paradigma de alocação. O modelo proposto resultou na definição de uma métrica chamada qualidade do paradigma, que reflete o desempenho combinado das aplicações que rodam em uma rede virtual. Além da qualidade do paradigma, o número de rodadas necessárias para completar a alocação de redes virtuais apresenta um impacto significativo na eficiência do paradigma e também deve ser considerado na análise.

A avaliação analisou duas dimensões diferentes do arcabouço proposto. A primeira avaliação demonstrou a viabilidade da proposta em acomodar mútliplos objetivos e comparou o desempenho de diferentes paradigmas. Quando vários objetivos coexistem no mesmo paradigma, é possível que alguns se sobressaiam sobre os demais, dependendo de fatores como o tamanho da janela do paradigma e a quantidade de recursos disponíveis. A segunda avaliação verificou a eficiência de diferentes paradigmas de alocação considerando o desempenho agregado das aplicações que rodam nas redes virtuais e analisou o impacto de se trocar paradigmas durante o aprovisionamento de redes virtuais.

O conceito de paradigmas de alocação representa uma abordagem nova de alocação de redes virtuais. Embora esta abordagem possa adicionar complexidade na alocação de recursos, ela permite que provedores de infraestrutura podem oferecer serviços diferenciados para seus clientes e, assim, aumentar sua receita utilizando uma abordagem flexível de alocação de redes virtuais. As vantagens da virtualização de redes não são perdidas uma vez que os provedores de infrestrutura não precisam saber detalhes sobre que aplicações serão de fato hospedadas nas redes virtuais.