Universidade de Aveiro
Departamento de Electrónica, Telecomunicações e Informática
2014

Márcio Daniel Tavares de Melo

Network Virtualisation from an Operator Perspective

Márcio Daniel
Tavares de Melo

Network Virtualisation from an Operator Perspective

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Doutor em Engenharia Eletrotécnica, realizada sob a orientação científica da Doutora Susana Sargento, Professora Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro e do Mestre Jorge Carapinha, Consultor Sénior Tecnológico na PT Inovação S.A.

Este trabalho foi realizado com o apoio de uma bolsa de Doutoramento em Empresa, com referência SFRH/BDE/33751/2009, financiada ao abrigo do programa POPH - QREN, - Formação Avançada, comparticipada pelo Fundo Social Europeu (FSE) e por fundos do Ministério da Ciência, Tecnologia e Ensino Superior (MCTES) através da Fundação para a Ciência e Tecnologia (FCT) e cofinanciada pela PT Inovação S.A.

FCT Fundação para a Ciência e a Tecnologia
MINISTÉRIO DA EDUCAÇÃO E CIÊNCIA
POPH PROGRAMA OPERACIONAL POTENCIAL HUMANO
QREN QUADRO DE REFERÊNCIA ESTRATÉGICO NACIONAL PORTUGAL 2007.2013
UNIÃO EUROPEIA Fundo Social Europeu
PT INOVAÇÃO

Dedico este trabalho à minha esposa pelo incansável apoio.

o júri / the jury

presidente / president

Profª. Doutora Nilza Maria Vilhena Nunes da Costa
Professora Catedrática do Departamento de Didáctica e Tecnologia Educativa da Universidade de Aveiro

vogais / examiners committee

Prof. Doutor Andreas Timm-Giel
Full Professor School of Electrical Engineering, Computer Science and Mathematics at Hamburg University of Technology

Prof. Doutor Alexandre Júlio Teixeira Santos
Professor Associado com Agregação no Departamento de Informática, Escola de Engenharia, da Universidade do Minho

Profª. Doutora Marília Pascoal Curado
Professora Auxiliar do Departamento de Engenharia Informática da Universidade de Coimbra

Profª. Doutora Susana Isabel Barreto de Miranda Sargento
Professora Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Doutor Francisco Manuel Marques Fontes
Consultor Sénior Tecnológico na PT Inovação S.A.

agradecimentos / acknowledgments

Agradeço a Deus por me ter dado o existir e me ter concedido a sabedoria.
À professora Susana Sargento por me ter dado esta oportunidade e por me ter guiado a levar a bom porto este Doutoramento através sua determinação e força de vontade. Ao Engenheiro Jorge Carapinha por ter partilhado comigo o seu vasto conhecimento no domínio das redes de operador e pelas inúmeras discussões sobre a temática da virtualização de rede. A vós um muito obrigado!
Aos meus colegas André Barbosa, João Monteiro, Christopher Viana, Lucas Guardalben, Alberto Gomes, Ricardo Silva, Hélder Alves e Tiago Moreira por me terem apoiado durante o Doutoramento.
Ao João Nogueira, Romeu Monteiro, Bruno Sendas e Rafael Gomes pelo seu contributo para este Doutoramento.
Ao Engenheiro Vítor Mirones, à Engenheira Telma Mota, ao Engenheiro Pedro Neves, ao Engenheiro Nuno Carapeto, ao Engenheiro Ricardo Azevedo e ao Engenheiro Filipe Cabral Pinto da PT Inovação pela partilha de conhecimento e de experiências de vida.
Ao Professor Andreas Timm-Giel por me ter concedido a oportunidade de alargar o meu leque de conhecimentos e por me ter guiado durante o tempo em que estive na Universidade Tecnológica de Hamburgo. Ao Professor Ulrich Killat por ter partilhado a sua enorme sapiência no domínio da otimização linear e pelos princípios científicos transmitidos.
Ao Kishore Angrishi, à Yunqi Luo, ao Ming Li, ao Jonas, ao Chunlei, à Nga, ao Luis Torres, ao Frank, ao Thomas, ao Prof. Kreft e ao Tiago Silva, pelo vosso caloroso bem-vindo e amizade.
Ao Instituto de Telecomunicações pelas excelentes condições de trabalho, pelos excelentes colegas de trabalho, e ainda pelos excelentes profissionais de secretariado e de suporte.
Ao Engenheiro Alcino Lavrador e ao Engenheiro Luís Miguel pelo suporte financeiro dado através da PT Inovação, S.A. Quero igualmente agradecer à Doutora Gabriela Moura, à Vera Santos e à Anabela Moreira pelo suporte dado.
Quero ainda agradecer à Fundação para Ciência e Tecnologia por me ter apoiado financeiramente durante todo o Doutoramento.
Aos meus pais por me terem transmitido os valores da vida. Ao meu irmão pelas inúmeras brincadeiras de infância, que contribuíram para a pessoa que sou hoje. Aos meus familiares por me terem apoiado. À minha esposa, agradeço por existir e por me apoiar.
Por fim, quero agradecer a todos aqueles cujos nomes ora por esquecimento ora por desconhecimento não foram mencionados. A todos vós um caloroso obrigado!

Palavras-chave

Internet do Futuro, Formulação Matemática, Migração de Redes Virtuais, NP-complexo, Problema de Mapeamento, Programação Linear Inteira, Redes Virtuais, Solução Ótima, Virtualização de Rede

Resumo

A virtualização de rede é vista como uma abordagem promissora para ultrapassar o "Impasse da Internet" e permitir inovação na Internet, possibilitando assim uma migração fácil para novas abordagens de redes, bem como a coexistência de arquiteturas de redes complementares numa infraestrutura compartilhada e em ambiente comercial. Recentemente tem crescido de forma bastante significativa o interesse pela virtualização de rede por parte dos operadores e dos grandes fabricantes, desde que os potenciais benefícios da virtualização se tornaram claros, tanto de ponto de vista económico como operacional. No início, o conceito foi versado pelo meio académico, onde foram realizadas provas de conceito de pequena escala, e em que a virtualização de rede foi considerada como forma de investigação de novos protocolos. Esta Tese de Doutoramento tem como objetivo geral dotar uma rede de operador de um conjunto de mecanismos e algoritmos capazes de gerir e controlar redes virtuais. Para este fim, é proposta uma $framework$ que visa alocar, monitorizar e controlar recursos virtuais de uma forma centralizada e eficiente. De forma a analisar o desempenho da $framework$, procedeu-se à sua implementação e avaliação numa rede de pequena dimensão. De forma a permitir que se possa efetuar uma alocação eficiente, em tempo real, e a pedido, de redes virtuais numa rede física, é proposta uma heurística para efetuar o mapeamento na rede física. Para que o operador de rede possa rentabilizar ao máximo a sua infraestrutura de rede, é ainda proposta uma formulação matemática que, através de programação linear, visa maximizar o número de redes alocadas na infraestrutura de rede. Dado que o consumo energético de uma infraestrutura de rede começa a ter significância nos custos de operação, é importante que se faça a alocação das redes virtuais no menor número de recursos físicos e também em recursos físicos ativos. Para endereçar este desafio é proposta uma formulação matemática que visa minimizar o consumo energético da rede física sem afetar a eficiência da alocação de redes virtuais. Para minimizar a fragmentação da infraestrutura de rede e ao mesmo tempo aumentar as receitas do operador, é também estendida a formulação inicial para contemplar a re-otimização de redes virtuais previamente mapeadas, fazendo com que o operador tenha um melhor aproveitamento da sua infraestrutura física. Será ainda necessário endereçar a migração de redes virtuais, quer por motivos de balanceamento de carga, quer por motivos de falha iminente de recursos físicos, sem afetar o bom funcionamento da rede virtual. Para este fim, é proposto um método baseado em técnicas de clonagem, para efetuar a migração de redes virtuais entre recursos da infraestrutura física de forma transparente e sem impacto para a rede virtual. De forma a avaliar a resiliência das redes virtuais a falhas na rede física, e ao mesmo tempo obter a solução ótima de migração de redes virtuais em caso de falha iminente dos recursos físicos, a formulação matemática é estendida para minimizar o número de nós migrados em simultâneo com a realocação de ligações virtuais. Em comparação com as nossas propostas de otimização verificou-se que as heurísticas existentes para mapeamento de redes virtuais têm um desempenho muito baixo. Verificou-se ainda que é possível efetuar a redução do consumo energético sem a penalização da alocação eficiente. Com a re-otimização das redes virtuais mostrou-se que é possível obter mais recursos livres, assim como obter uma melhor distribuição dos recursos. Por último, demonstrou-se que as redes virtuais são bastante resilientes a falhas na rede física.

Keywords Future Internet, Integer Linear Programming, Mapping Problem, Mathematical Formulation, NP-Hard, Optimal Solution, Virtual Network Migration, Virtual Networks, Network Virtualisation

Abstract Network virtualisation is seen as a promising approach to overcome the so-called "Internet impasse" and bring innovation back into the Internet, by allowing easier migration towards novel networking approaches as well as the coexistence of complementary network architectures on a shared infrastructure in a commercial context. Recently, the interest from the operators and mainstream industry in network virtualisation has grown quite significantly, as the potential benefits of virtualisation became clearer, both from an economical and an operational point of view. In the beginning, the concept has been mainly a research topic and has been materialized in small-scale testbeds and research network environments. This PhD Thesis aims to provide the network operator with a set of mechanisms and algorithms capable of managing and controlling virtual networks. To this end, we propose a framework that aims to allocate, monitor and control virtual resources in a centralized and efficient manner. In order to analyse the performance of the framework, we performed the implementation and evaluation on a small-scale testbed. To enable the operator to make an efficient allocation, in real-time, and on-demand, of virtual networks onto the substrate network, it is proposed a heuristic algorithm to perform the virtual network mapping. For the network operator to obtain the highest profit of the physical network, it is also proposed a mathematical formulation that aims to maximize the number of allocated virtual networks onto the physical network. Since the power consumption of the physical network is very significant in the operating costs, it is important to make the allocation of virtual networks in fewer physical resources and onto physical resources already active. To address this challenge, we propose a mathematical formulation that aims to minimize the energy consumption of the physical network without affecting the efficiency of the allocation of virtual networks. To minimize fragmentation of the physical network while increasing the revenue of the operator, it is extended the initial formulation to contemplate the re-optimization of previously mapped virtual networks, so that the operator has a better use of its physical infrastructure. It is also necessary to address the migration of virtual networks, either for reasons of load balancing or for reasons of imminent failure of physical resources, without affecting the proper functioning of the virtual network. To this end, we propose a method based on cloning techniques to perform the migration of virtual networks across the physical infrastructure, transparently, and without affecting the virtual network. In order to assess the resilience of virtual networks to physical network failures, while obtaining the optimal solution for the migration of virtual networks in case of imminent failure of physical resources, the mathematical formulation is extended to minimize the number of nodes migrated and the relocation of virtual links. In comparison with our optimization proposals, we found out that existing heuristics for mapping virtual networks have a poor performance. We also found that it is possible to minimize the energy consumption without penalizing the efficient allocation. By applying the re-optimization on the virtual networks, it has been shown that it is possible to obtain more free resources as well as having the physical resources better balanced. Finally, it was shown that virtual networks are quite resilient to failures on the physical network.

Contents

List of Figures

List of Tables

List of Acronyms

AN Active Networks

ATM Asynchronous Transfer Mode

BCM Bandwidth Consumption Minimization

CABO Concurrent Architectures are Better than One

CAPEX CAPital EXPenditure

CLI Command Line Interface

C-NLM Coordinated - Node-Link Mapping

CONEX CONgestion EXposure

CPU Central Processing Unit

CSPF Constrained Shortest Path First

DNS Domain Name System

D-ViNE Deterministic Node Mapping with Slipttable Link Mapping using Multi-Commodity Flow Constraint

D-ViNE-LB Deterministic Node Mapping with Splittable Link Mapping using Multi-Commodity Flow Constraint and Load Balancing based

D-ViNE-SP Deterministic Node Mapping with Shortest Path based Link Mapping

EA-VNE-NLF Energy Aware - Virtual Network Embedding - Node-Link Formulation

ECM Energy Consumption Minimization

ETSI European Telecommunications Standards Institute

XORP Extensible Open Router Platform

FEDERICA Federated E-infrastructure Dedicated to European Researchers Innovating in Computing network Architectures

FN Future Network

GENI Global Environment for Network Innovations

GEYSERS Generalized Architecture for Dynamic Infrastructure Services

GLPK Gnu Linear Programming Kit

G-MCF Greedy Node Mapping with Splittable Link Mapping using Multi-Commodity Flow Constraint

GRE Generic Routing Encapsulation

G-SP Greedy Node Mapping with Shortest Path based Link Mapping

GT-ITM Georgia Tech Internetwork Topology Models

HDD Hard Disk Drive

ILP Integer Linear Programming

InP Infrastructure Provider

IP Internet Protocol

IPTV Internet Protocol Television

IRTF Internet Research Task Force

ISG Industry Specification Group

ISP Internet Service Provider

IT Information Technology

ITU-T International Telegraph Union - Telecom

KVM Kernel-based Virtual Machine

LAN Local Area Network

MB Mega Byte

MCFP Multi-Commodity Flow Problem

MIP Mixed Integer Programming

MIT Massachusetts Institute of Technology

MPLS Multi Protocol Label Switching

MP-TCP Multi-Path Transmission Control Protocol

NAT Network Address Translation

NFV Network Functions Virtualisation

NM-BCM Node Migration and Bandwidth Consumption Minimization

NP-hard Non-deterministic Polynomial-time hard

NREN National Research and Education Networks

NV Network Virtualisation

NVC Network Virtualisation Controller

NVE Network Virtualisation Entity

NVGRE Network Virtualisation using Generic Routing Encapsulation

NVSS Network Virtualisation System Suite

ON Overlay Network

OPEX OPerational EXpenditure

ORBIT Open-Access Research Testbed for Next-Generation Wireless Networks

OS Operating System

OSPF Open Shortest Path First

P2P Peer-to-Peer

PE Provider Edge

PoP Point-of-Presence

PSO Particle Swarm Optimization

QoS Quality of Service

RAM Random Access Memory

RAMC Resource Allocation, Monitoring and Controlling

RON Resilient Overlay Network

RSVP Resource ReSerVation Protocol

R-ViNE Randomized Node Mapping with Splittable Link Mapping using Multi-Commodity Flow Constraint

SAIL Scalable & Adaptive Internet soLutions

SCP Session Control Protocol

SDN Software Defined Networking

SDP Shortest Distance Path

SN Substrate Network

SSH Secure Shell (Unix program)

SVNE Survivable Virtual Network Embedding

TCP Transmission Control Protocol

TNI Tenant Network Identifier

UDP User Datagram Protocol

UPS Uninterruptible Power Supply

VL Virtual Link

VLAN Virtual Local Area Network

VM Virtual Machine

VN Virtual Network

VNE Virtual Network Embedding

VNE-ESPH Virtual Network Embedding - Enhanced Shortest-Path Heuristic

VNE-NLF Virtual Network Embedding Node-Link Formulation

VNM Virtual Network Migration

VNO Virtual Network Operator

VNP Virtual Network Provider

VNR Virtual Network Request

VNRE Virtual Network Re-Embedding

VNRE-NLF Virtual Network Re-Embedding Node-Link Formulation

VNRG Virtual Networks Research Group

VoIP Voice Over IP

VPLS Virtual Private LAN Services

VPN Virtual Private Network

VPWS Virtual Private Wire Services

VR Virtual Router

VROOM Virtual ROuters On the Move

VXLAN Virtual Extensible LAN

WSDP Weighted Shortest Distance Path

XML Extensible Markup Language

Chapter 1 - Introduction

"The greatest enemy of knowledge is not ignorance, it is the illusion of knowledge."

—Stephen Hawking

As the first chapter in the Thesis, the introduction sets the motivation for pursuing a PhD on network virtualisation. It presents the hypothesis, along with goals that guided the work evolution (section 1.2), followed by the contributions that resulted from the concepts explored (section 1.3). Finally, it presents the overall structure of the document (section 1.4).

1.1 Scope & Motivation

New networking applications are emerging, which are introducing increasingly higher requirements on the networks, such as more bandwidth, quality of service, support for a large number of end-devices, peer-to-peer connectivity, a high degree of user mobility, well-integrated security features, and so forth. Although individual mechanisms and protocols have been or are being developed to enable many of these features, the current Internet architecture is rather restrictive when it comes to supporting networks that are flexible, extensible, composable, and re-usable.

In this context, there is a significant trend on the concept of "networks of networks", which considers that the Internet of the Future will be a group of networks that share the same infrastructure [PACR03, Fel07]. These networks may have a common architecture, but they will be used for different purposes, and with different application requirements.

On the other hand, it is clear now the importance of reducing the energy consumption with respect to the carbon footprint. The consumption of energy due to Telecom and Broadcasting has been frenetically increased over the years. In Japan the consumption of energy due to Telecom and Broadcasting achieved astonishing proportions [AN08]. Several procedures can be taken for reducing the energy consumption such as putting nodes that are not being used into sleep mode or by using "green" protocols. In [NPI+08a] it is discussed the impact on network protocols by putting network interfaces and components into sleep for saving energy, and in [GS03a] it is presented and evaluated two forms of power management schemes that reduce the energy consumption of networks.

To this effect, it has been proposed to use virtualisation as the basis for a highly flexible architecture for the Future Internet in order to circumvent the restrictions of the current Internet and provide a platform enabling the introduction of novel, and possibly revolutionary, networking capabilities [EFI14, FIN14, 4WA09], and also to reduce the energy consumption [LFDM14].

The basic concept of virtualisation itself is well known and already implemented in the Information Technology (IT) domain [BDF+03]. It is used to improve resource efficiency, to reduce management complexity and deployment times, to isolate failure domains, and to provide real-world test environments. The application of the idea to networking (hardware) environments and the advantages of virtualisation can be used to optimize network resources usage.

The virtualisation of networks will trigger the development of green protocols and will facilitate the load distribution among the network based on the usage of the network or even in data previsions, allowing to save energy by switching the unnecessary resources into idle stages. It will be possible to put the virtual components into sleep or even in hibernated modes.

Moreover, it allows to create, control, and manage coherent networks spanning multiple infrastructure providers offering more powerful control for virtual network operators; operators will be able to share the same infrastructure for different purposes and without affecting each other.

Network virtualisation has the potential to facilitate a variety of new scenarios and business use cases, where providers may want to occupy different parts of the value chain and develop their own business strategies [4WA09]. This concept can allow the co-existence and operation of different networks, which would use different protocols, for different service requirements on the same infrastructure [CJ09].

The main advantages of network virtualisation realized by the industry goes beyond the one of future Internet scenarios, mainly to operators and service providers on a short/medium

term, are as follows [Cis09b, Jun09]:

- Reduction of costs: by using a single virtualised infrastructure to run multiple services, CAPital EXPenditure (CAPEX) and OPerational EXpenditure (OPEX) can be reduced, compared to the typical scenario where different types of service (e.g. voice, data, broadcast) are run in separate networks.

- Increase of revenues: by sharing infrastructure, the network operator achieves a better utilization of the network resources and optimizes profitability.

- Flexible network planning: the swift and easy establishment of virtual networks can be used as a safeguard against unpredictability of the service demand.

- Security and isolation: virtualisation can provide real isolation of network resources, with benefits in terms of fault isolation, security, and performance guarantees.

- Flexibility and programmability: virtual networks can be tuned to fulfil specific service and application requirements (e.g. security, performance, dependability), thus a "one-size-fits-all" approach is no longer required.

According to the existing work in this area, there is already some architectural work that initiates the concept of virtual networks, and defines mechanisms for exchanging information between virtual networks, for example, Plutarch [CHM+03]. However, it is necessary to improve and re-think the features of interaction between networks to enable the coexistence of networks with different architectural principles.

There are also several tools to enable the virtualisation of machines, working mainly in virtualisation of operating systems, XEN [BDF+03], Denali [WSG02], VMware [DBR02], Connectix [con03], Kernel-based Virtual Machine (KVM) [kvm14]. However, the support for multi-protocol routers to enable and consolidate multiple level 3 technologies still requires more research, given that it is very complex and difficult to manage the isolation between different routing domains. At the level of virtualisation of networks, an example of a practical application of the concept of virtualisation is the Virtual Private Network (VPN) [RR06]. However, the utilization of VPNs for enabling the deployment of new architectures and protocols is restrictive, since only the virtual links are virtualized.

In the past, other examples have appeared that take into practice some of the concepts of network virtualisation, but on the sense of running a set of parallel experiments in the same experimental platform. Examples of this process are the experimental platforms PlanetLab [Pla14], EmuLab [WLS+02a] and Panlab [pan14]. Additionally, there are initiatives like Global Environment for Network Innovations (GENI) [PAB+06, PW06] that aim to support real traffic. Moreover there are several proposed approaches for the support of virtualisation, as VINI [BFH+06], virtual testbed [APST05a], meta-network [TT05], Concurrent Architectures are Better than One (CABO) [FGR07a, PJ06, TWEF03], all these approaches are used for experimental networks. Recently, there are initiatives like Software Defined Networking (SDN) [Fou14] and Network Functions Virtualisation (NFV) [ETS] that aim to increase the network programmability through the decoupling of control functions from the forwarding functions. SDN is a new approach to designing, building and managing networks. This architecture decouples the control plane from forwarding plane to enable the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services. NFV offers a new way to design, deploy and manage networking services. NFV decouples the network functions, such as Network Address Translation (NAT), firewall, intrusion detection, Domain Name System (DNS), caching, etc., from proprietary hardware appliances, so they can be run in software.

Although there are already several proposals on how to perform the virtualisation of machines and network resources, there is not a clear vision of how this process can be handled, particularly in the view of the operator: i) there is not a clear approach on how an operator can manage and efficiently provision virtual network resources; ii) and on how to create and manage multiple virtual networks; iii) as well as on how to re-configure virtual resources and seamlessly move them across the infrastructure without causing any disruption on the services running on each virtual network, either due to load balancing policies or even for energy consumption purposes.

1.2 Objectives

The purpose of this PhD Thesis is to corroborate the following hypothesis *"Can network virtualisation be the basis for a highly flexible architecture for the Future Internet to circumvent the restrictions of the current Internet model, and to provide a platform enabling the introduction of novel, and possibly revolutionary, networking capabilities?"*

The general objective of this PhD Thesis is to evaluate the potential of network virtualisation from an operator's perspective, with the short-term goal of optimizing service delivery and rollout, and on a longer term as an enabler of technology integration and migration.

To accomplish this aim, it will define a framework for virtual resource control, allocation, and monitoring, which allows an operator to support virtual network technologies, with different capabilities and requirements, in the same physical infrastructure and network platform.

Additionally, it will define, evaluate and implement mechanisms to efficiently embed different virtual networks on the same physical infrastructure in a dynamic and optimised form, which will allow operators to increase their revenue by leasing their infrastructure, and eventually to reduce provisioning costs by applying a heuristic algorithm.

Moreover, the energy consumption of the physical network is a big concern to network operators, either due to environmental policies imposed by the local governments or due to energy costs. Therefore it is required that the existent embedding mechanisms, not only aim to efficiently embed VNs, but also to reduce the energy consumption of the physical network.

In addition, virtual networks are provisioned on-demand, for a fixed amount of time, and not only its initial requirements can change during the VN lifetime (e.g. bandwidth, Central Processing Unit (CPU)), but also the VN topology. Therefore, mechanisms need to be defined, evaluated and implemented to reduce the physical network fragmentation and at the same time to increase the revenue of the operator.

Finally, the virtual network assignment can change during the VN lifetime, i.e. either due to physical resource failure or to load balancing policies. To support these events, mechanisms are required to perform VN migration seamlessly, and in an optimal way (e.g. number of migrated nodes, bandwidth re-allocation).

1.3 Scientific Contributions

The scientific work performed during this PhD has been published in several international conferences and peer-reviewed journals, and inclusively in technical reports of research projects where the author was involved. The scientific contributions made throughout this PhD can be grouped into three parts:

i The first part is devoted to the topic of virtual resource allocation and control. A framework to handle these requirements is proposed, and a proof of concept is performed in a

small-scale testbed in [MSC09, Mel10];

ii The second part corresponds to the migration and reconfiguration of virtual network resources. To this extent, it is proposed a method to move virtual resources seamlessly across the physical network without affecting the virtual networks in [MCS13a, MSC14], and an Integer Linear Programming (ILP) formulation to obtain the optimal solution per VN re-embedding in [MSC14];

iii The last part incorporates the virtual network embedding problem and the optimal embedding solution. To address this research challenge, it is proposed an enhancement to an existing heuristic, and it is also proposed an ILP formulation to obtain the optimal solution per VN embedding in [MCS$^+$12, MSK$^+$13]. Extensions of this formulation to perform VN re-optimization and energy consumption minimization are proposed in [MCS$^+$13b, MSK$^+$14], respectively.

The scientific work performed during this PhD has also positively contributed to the development of several Msc Thesis:

- *"Demonstrador de Criação de Redes Virtuais no Âmbito do Operador"* [Nog10];

- *"Criação e Reconfiguração de Redes Virtuais na Perspetiva do Operador"* [Mon11];

- *"Integração da Cloud com Rede na Perspectiva de Operador"* [Par12];

- *"Demonstrador de uma rede com tecnologia OpenFlow"* [Gom13].

For convenience, we summarize the contributions in Table 1.1, organized by year and type, so the output can be better understood.

Table 1.1: List of scientific contributions made throughout the PhD.

Title	**Type**	**Venue**	**Year**
Network virtualisation from an Operator Perspective [MSC09]	Conference	*Conf. sobre Redes de Computadores (CRC)*	2009
Virtual Network Mapping - An Optimization Problem [MCS$^+$12]	Conference	*Springer Mobile Networks and Management (MONAMI)*	2011
A Re-optimization Approach for Virtual Network Embedding [MCS$^+$13b]	Conference	*Springer Mobile Networks and Management (MONAMI)*	2012
Network Virtualisation: A step closer for seamless resource mobility [MCS13a]	Workshop	*IEEE Integrated Network Management (IM)*	2013
Optimal Virtual Network Embedding: Node-Link Formulation [MSK$^+$13]	Journal	*IEEE Transactions on Network and Service Management (TNSM)*	2013
Optimal Virtual Network Embedding: Energy Aware Formulation [MSK$^+$14]	Journal	*Submitted to IEEE Transactions on Network and Service Management (TNSM)*	2014
Optimal Virtual Network Migration: A Physical Network Resilience Evaluation [MSC14]	Journal	*Submitted to IEEE/ACM Transactions on Networking*	2014

1.4 Structure

The main body of work presented in this Thesis is structured around the concept of network virtualisation and open research challenges, followed by the application on the operator's network. We present a structure consisting of an introduction and conclusion, a related work chapter, and two central chapters incorporating the most novel contributions. The two initial chapters, chapter 1 and chapter 2, frame the content and topics presented in the Thesis.

The Introduction presents the motivation, setting the stage for the hypothesis and goals of the Thesis. It also highlights the contributions stemming from the presented proposals.

Chapter 2 presents an overview on network virtualisation concepts and terminology, where existing network virtualisation technologies and new business models that may advent from the appliance of network virtualisation are discussed. Moreover, it examines the research challenges and reviews the related work. Existing Future Internet research projects and standardisation groups are also presented.

Chapter 3 starts with a description of the mechanisms and algorithms required to control virtual network resources. It is followed by a framework proposal to address these requirements: resource allocation, monitoring and controlling. A small-scale testbed is presented that is used as a proof of concept, and an evaluation is performed on the framework. Moreover, it provides a description on the different triggers used for VN migration, and it presents the architecture of the VN clone migration method. Finally, the evaluation results, which concern the VN migration, are depicted and discussed.

In Chapter 4 a description of the virtual network embedding problem is given. Then, it discusses an enhancement to a heuristic algorithm. Furthermore is proposed a ILP formulation, i.e. Virtual Network Embedding Node-Link Formulation (VNE-NLF), to solve the Virtual Network Embedding (VNE) as an optimization problem. Moreover, different proposed cost functions for resource allocation are discussed. Extensions on this formulation are proposed to address re-optimization of VNs previously embedded, energy consumption, and the optimization of Virtual Network Migration (VNM). The major evaluation results concerning the VNE-NLF and the proposed extensions are depicted and analysed.

As the final chapter, chapter 5 summarizes the findings of the Thesis in the context of network virtualisation, and presents future impacts of the proposed solutions and concepts through a discussion of some of the insights gathered throughout the research process detailed in this document.

Chapter 2 - Network Virtualisation: Related Work

"We cannot solve our problems with the same thinking we used when we created them."

—Albert Einstein

This chapter presents an overview of virtualisation concepts and related work of network virtualisation. The first section will start with a general description of the concepts and terminology. In Section 2.2, an overview of the network virtualisation background is given. Section 2.3 presents network virtualisation business models. Moreover, research challenges in network virtualisation will be pointed out in Sections 2.4 and 2.5, with special focus on the Virtual Network Embedding (VNE) problem. Furthermore in Section 2.6, Future Internet projects will be discussed. In Section 2.7 research groups and standardization organizations will be referred. Finally, Section 2.8 summarizes the chapter.

2.1 Concepts & Terminology

The virtualisation concept, as it is, is not new in the literature and it is broadly used in several fields. In the sixties, IBM introduced the Virtual Machine concept [Gol74], where it was described how to apply virtualisation into computers to have a set of simulators/emulators with the same physical (virtual) hardware; this resulted in the virtual machines. One common example of the use of virtualisation are the logical partitions on the disk drive.

2.1.1 Network Virtualisation

Network virtualisation is an approach whereby several network instances can co-exist on a common physical network infrastructure. The type of network virtualisation needed is not to be confused with current technologies such as VPNs, which merely provide traffic isolation. Full administrative control as well as potentially full customization of the Virtual Networks (VNs) is also required to realise the vision of using network virtualisation as the basis for a Future Internet. This includes e.g. the possibility of non Internet Protocol (IP) networks running alongside the current Internet within one future virtualised network infrastructure. Each of these virtual networks can be built according to different design criteria and operated custom-tailored to a specific network service.

2.1.2 Virtual Link

In several forms (for example, by means of technologies such as Asynchronous Transfer Mode (ATM) and Frame Relay and, more recently, Ethernet-based technologies), link virtualisation has been deployed in large scale operator environments for a long time. The basic purpose of link virtualisation is to divide, share and isolate the resources of physical links. A virtual link is an abstract entity that allows to represent the functionality of a traditional physical link (i.e. bit transport between connected endpoints), but is not based on physical resources. In the context of network virtualisation, virtual links provide the ability to flexibly connect virtual nodes with certain guarantees such as isolation, dedicated resources or Quality of Service (QoS). Link virtualisation may have different issues, depending on the different types of physical links. There is a wide range of available options for wired link virtualisation, from Ethernet Virtual Local Area Networks (VLANs) (IEEE 802.1Q [IEE11], IEEE 802.1ad [IEE06]), to optical circuit switching. For each technology, a specific virtual link identifier is used to provide a separation between different virtual links. Since many of these technologies are expected to be deployed in core networks, an important feature is the capability to aggregate multiple virtual links into a single pipe, rather than handling a potentially large number of individual virtual links. This feature is crucial to evaluate scalability of these technologies and whether or not they can be deployed in large scale scenarios.

Two standards have been proposed to overcome the limited number of VLANs that the IEEE 802.1Q [IEE11] specification imposes, which are inadequate for complex virtualised environments, and make it difficult to stretch network segments over the long distances required for dispersed data centres: Virtual Extensible LAN (VXLAN) [SKD+13] standard extended the VLAN address space by adding a 24-bit segment ID; Network Virtualisation using Generic Routing Encapsulation (NVGRE) [SPG+13] standard includes identifying a 24-bit Tenant Network Identifier (TNI) to address problems associated with the multi-tenant network, and using a Generic Routing Encapsulation (GRE).

2.1.3 Virtual Node

Virtualisation of the nodes that constitute a physical network is a fundamental issue to network virtualisation, where router virtualisation is the most notable aspect. Modern routers are built on very powerful hardware and software that allows resources to be "sliced" amongst many virtual networks passing the node. A modern router is a complicated network element with a range of functions. It operates conceptually in two operational planes, the forwarding and the control plane. The forwarding plane functionality is to actually forward traffic from ingress to an egress interface. The control plane decides on the route a packet is forwarded to, QoS issues, and other aspects. Traditionally, routers operate on Layer 3 packets, but modern devices extend their operation across lower or higher layers. The so called switching routers are now common using Multi Protocol Label Switching (MPLS) technology to create fast switched paths through the network instead of the traditional hop-by-hop routing approach.

2.2 Existing Network Virtualisation Technologies

2.2.1 Asynchronous Transfer Mode

ATM networks implement only one aspect of network virtualisation, namely, virtual links [Vet95]. ATM virtual paths and virtual channels are packet switched paths which provide isolation by resource reservation on the ATM nodes alongside of the paths. However, ATM does not support dynamically configured virtual relay nodes inside the network, which is essential for network virtualisation in order to allow dynamic and manageable network architectures. In addition, although ATM is a robust technology for individual islands within a global network, it suffers from scalability issues.

2.2.2 Multi Protocol Label Switching

MPLS is a technology for virtual call setup and resource reservation over wide area networks [RVC+01]. It can support multiple types of protocols by encapsulation. The objective and functions of MPLS are very similar to those of ATM; however, MPLS addresses the scalability problem of ATM by using label swapping instead of cell switching in ATM. MPLS also uses a combination of IP routing algorithms (for path routing, instead of datagram routing in IP) and Resource ReSerVation Protocol (RSVP) for reservation of resources alongside MPLS paths. MPLS addresses QoS issues to some extent within the scope of individual autonomous systems; MPLS and network virtualisation can be combined to address the end-to-end QoS problem in an improved global scale, beyond what can be achieved by MPLS alone.

2.2.3 Virtual Private Network

The networking community embraced the concept of virtualisation in the late 1990s, network-based IP/MPLS/VPNs[1] [RR06] materialized the concept of building multiple virtual IP networks over a common large scale network infrastructure. Later on, the concept was extended to layer 2 technologies, such as Ethernet, through services like Virtual Private Wire Services (VPWS) and Virtual Private LAN Services (VPLS). However, all these incarnations of the concept were tightly bound to a specific protocol, either layer 3 (e.g. IP) or layer 2 (e.g. Ethernet). On the other hand, at the node level, virtualisation was elusive, in the sense that

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

it was basically just a separation of addressing spaces, but not a real separation of network resources.

2.2.4 Overlay Networks

Overlay Networks (ONs) are usually a collection of software routers deployed at the edges of the Internet in order to allow different forwarding mechanisms other than those of the Internet. The common denominator of the overlay networks is their operation on the top of the Internet routing substrate [ABKM01]. They are often considered as application layer networks with custom datagram structures that are usually encapsulated inside IP, Transmission Control Protocol (TCP), or User Datagram Protocol (UDP) packets. The Massachusetts Institute of Technology (MIT) Resilient Overlay Networks (RONs) project [ABKM01] is a prime example of overlay networks of this type. These networks can help to implement custom routing algorithms that may be suitable for multicasting or broadcasting over the Internet [ST02, CRSZ01], for instance.

A common example of the application of ONs are the VPNs (not to be confused with the IP/MPLS/VPNs[2]). An overlay VPN allows building private networks on top of the Internet [Con06]. The private networks can deploy different networking and upper layer protocols. The whole concept is based on building an overlay network by the use of protocol tunnelling; packets of the virtual networks are encapsulated inside the packets of the underlying public network at the ingress gateway, and de-capsulated again at the egress gateway. The VPN tunnels can also be used to encapsulate layer-2 frames inside IP datagram's in order to enable geographically dispersed nodes to form a local area network over the Internet. VPNs are particularly powerful tools to improve security of corporate networks against adversaries.

Unlike the proposed framework, the overlay networks allow agility only at the edges of the current Internet. In addition, link virtualisation is not integrated into the architecture of the overlay networks.

2.2.5 Active Networks

Active network architecture is composed of execution environments (similar to a Unix shell that can execute active packets), a node operating system capable of supporting one or more execution environments [HKM+98]. It also consists of active hardware, capable of routing or switching, as well as capable of executing code within active packets. This differs from the traditional network architecture which seeks robustness and stability by attempting to remove complexity, and by the ability to change its fundamental operation from underlying network components. Network processors are one mean of implementing active networking concepts.

2.2.6 Software Defined Networking

SDN is an emerging architecture that is dynamic, manageable, cost-effective, and adaptable, making it ideal for the high-bandwidth, dynamic nature of today's applications. In the SDN architecture, the control and data planes are decoupled, network intelligence and state are logically centralized, and the underlying network infrastructure is abstracted from the applications [Fou14]. As a result, enterprises and carriers gain unprecedented programmability, automation, and network control, enabling them to build highly scalable, flexible networks that readily adapt to changing business needs.

[2]The term overlay VPN is used throughout this Thesis to avoid misunderstanding between the two types of VPNs.

The OpenFlow [MAB+08] protocol is a foundational element for building SDN solutions. OpenFlow is added as a feature to commercial Ethernet switches, routers and wireless access points – and provides a standardised hook to allow researchers to run experiments, without requiring vendors to expose the internal workings of their network devices. OpenFlow is used for applications such as virtual machine mobility, high-security networks and next generation IP based mobile networks. The OpenFlow-based SDN is being rolled out in a variety of networking devices and software.

2.3 Business Models & Roles

Players in the network virtualisation model (Figure 2.1) are different from those in the traditional networking model. The main distinction is the presence of three different roles: Infrastructure Provider (InP), Virtual Network Provider (VNP), and the Virtual Network Operator (VNO), in contrast with the conventional model, characterized by a single role: Internet Service Provider (ISP) in the conventional model [APST05a, FGR07a, APST05b, FGR07a, ZZRR08]. From a commercial point of view, this decoupling amortizes high fixed cost of maintaining a physical presence by sharing capital and operational expenditure across multiple infrastructure providers, as well as one infrastructure provider for several service providers. It should be noted that business roles do not necessarily map one-to-one to distinct business entities (i.e., any business entity can assume multiple roles). From a business model point of view, a very significant impact of network virtualisation is the ability to cleanly decouple infrastructure from services, which has been pursued for a long time but never really accomplished. This potential separation of infrastructure and services paves the way for the creation of new business models and roles. Network virtualisation can be deployed in a number of very different scenarios and business models but, in general, it is based on three distinct roles, as defined in the 4WARD project [4WA09], and represented in Figure 2.1.

It should be noted that this model does not preclude the possibility of more than one role being played by a single entity. In a vertically integrated scenario, the three roles would be typically played by the same operator. Yet, even in this case, a functional separation of roles based on the model should make sense.

2.3.1 Infrastructure Provider

The Infrastructure Provider (InP) deploys and runs the network physical resources, and partitions them into isolated virtual resources using some virtualisation technology. These resources are typically offered to virtual network operators and not to end users (but the customer of the InP might as well be a corporation using the virtual network for its internal use, rather than to build commercial end user services). The InP has visibility into what resources are leased to each VN, but not into the protocols running inside.

2.3.2 Virtual Network Provider

The virtual network provider (VNP) is responsible to find and compose the adequate set of virtual resources from one or more infrastructure providers, in order to fulfil the virtual network operator request. The VNP leases slices of the virtualised infrastructure from one or more InPs and puts them together. In reality, what the VNP provides is not a network, but just an empty container where the virtual network operator builds the protocols that will make the VN to come alive. The role of the VNP is particularly important in scenarios where multiple InP domains are involved, but may be redundant in the case where a VN is limited to a single network infrastructure domain.

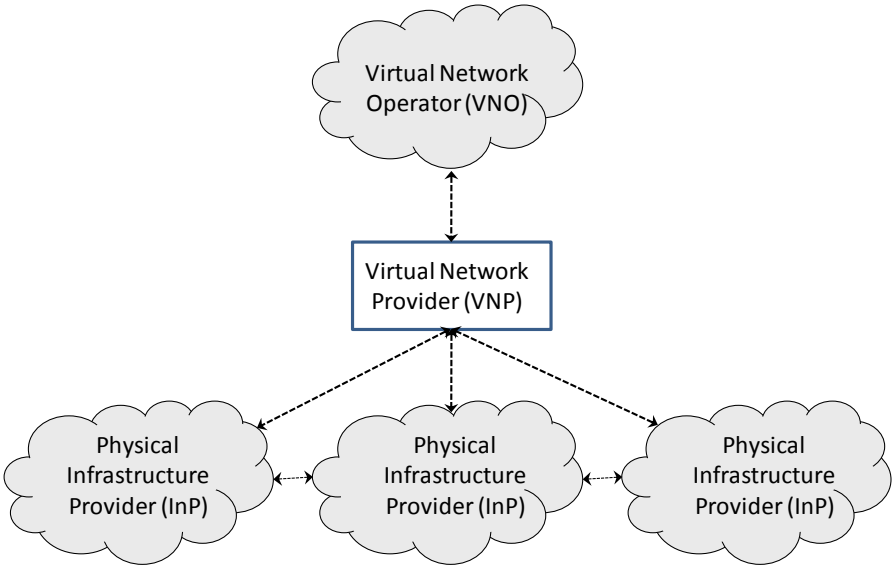


Figure 2.1: Network virtualisation Business Roles.

2.3.3 Virtual Network Operator

In each isolated network partition, the VNO is, in principle, free to deploy any protocol stack and network architecture, independently of the underlying network infrastructure technology. The VNO operates, maintains, controls and manages the virtual network. From a functional viewpoint, the role of the VNO should be indistinguishable from that of any operator running a native network infrastructure. Ideally, the fact that resources are virtual, rather than physical, should not imply any major impact from an operational point of view. VNOs have a unified view of the network, regardless of the multiple infrastructure domains on which it is built.

2.3.4 The VNP-InP Interface

The VNP-InP interface is a key aspect of the network virtualisation architecture. Through this interface, the VNP is able to request the establishment, modification or removal of virtual networks (supposedly, as a result of a corresponding request from the VNO). In its turn, the InP is supposed to acknowledge the VNP requests and notify any relevant event (e.g. a network error). The split of responsibilities and the information flow between the VNP and the InP are therefore of the utmost importance. Ultimately, this will depend on the information flowing through the VNP/InP interface in both directions, as illustrated in Figure 2.2. In principle, one of two basic approaches could be taken:

- The InP announces the resources which are available to be leased by VNPs, i.e. the internal structure of the InP infrastructure (or a virtual representation thereof) and the current state of resources. The InP is supposed to publish this information in some way, e.g. by means of a specific notice-board, such as proposed in [RR06]. It is up to the VNP to pick one or multiple InPs amongst all possible candidates, that would be able to provide the required resources, while fulfilling any applicable constraints (e.g. performance guarantees, cost). Since the relationship between the VNP and the InP is quite straightforward, the complexity

of the VNP-InP interface is quite low in this case. This approach is appropriate for research testbeds, or whenever there is a trust relationship between the InP and all potential VNPs; in a commercial environment this is not likely to be the case, except perhaps in a vertically integrated scenario, in which VNP and InP have a common business affiliation.

- The InP exposes a minimal set of resources, namely the Point-of-Presences (PoPs), and hides the internal structure and the state of resources. Since the VNP does not know in advance whether the InP is able to fulfil its request, the virtual network characteristics have to be provided to the InP and a negotiation has to be carried out through the VNP/InP interface prior to VN establishment phase, when the resources are actually reserved. In turn, the InP decides whether the request can be accommodated in the physical resources and, if so, it maps virtual nodes into substrate nodes and finds the substrate path between every pair of directly connected virtual nodes. This is likely to be the approach followed in a commercial environment, where a relationship of trust between VNPs and InPs is not expected.

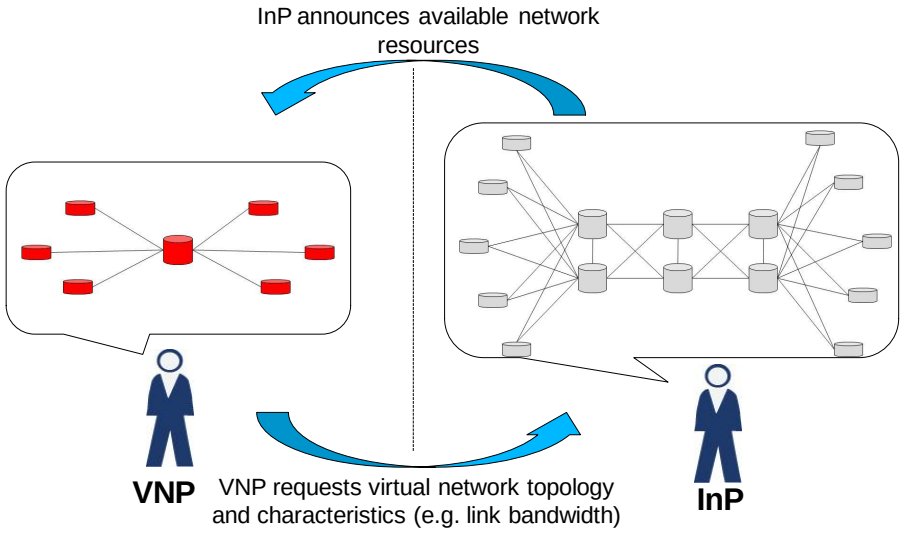


Figure 2.2: VNP-InP information flow.

2.4 Virtual Network Embedding Problem

Although there is a large interest on virtualized networks both from the research community and network operators, several challenges still prevent them from being deployed in real environments [CB09]. One of the major obstacles lies in the efficient embedding[3] of a Virtual Network (VN) onto a physical network[4], commonly known by the research academia as the VNE problem. Since this process requires the simultaneous optimization of virtual nodes and links placement, it is complex in nature, both in formulation and computationally. Several works, such as [ZA06, LK09, LT06, YYRC08, CRB09, FBCB10, NMCS11b, BHFM12,

[3]The terms embedding, mapping and assignment are used interchangeably throughout this Thesis.
[4]The terms physical and substrate are used interchangeably throughout this Thesis.

CRB12], have already proposed solutions to this problem, mostly based on heuristic approaches; however, they either do not provide the optimal solution for each VN mapping, or eventually, simplify the domain problem by not considering one or more parameters of the VNE: e.g., finite and heterogeneous physical resources, heterogeneous physical and virtual network topologies, online problem.

The VNE can be described as a graph assignment problem. It involves the mapping of Virtual Network Requests (VNRs) onto a physical network; this problem not only touches the mapping of virtual links onto physical links as addressed on VPN scenarios, but also the mapping of virtual nodes onto physical nodes. This simultaneous node and link mapping can be formulated as an un-splittable flow problem [ZA06], known to be $\mathcal{NP}$-hard.

To provide the reader with a better understanding on the full extent of the VNE domain problem, we first present in Sub-section 2.4.1 the domain characteristics. Next, we present the related work on VNE algorithms, organized according to the following criteria: resource allocation (Sub-section 2.4.2); approaches that aim to reduce the energy consumption of the physical network per VNE (Sub-section 2.4.3); approaches on virtual network resilience (Sub-section 2.4.4); other research directions not endorsed on this PhD Thesis (Sub-section 2.4.5).

2.4.1 VNE Characteristics

2.4.1.1 Online VNE Problem

Researchers have investigated various methodologies for handling VN requests as they arrive (online VNE, e.g., [YYRC08, CRB09]). On the other hand, for the offline version of the VNE problem, the assumption is made that all VN requests are known a *priori* (e.g., [ZA06, LT06]). In the online case, resource mapping results in sub-optimal performance for the substrate network with regard to load balancing/utilization and VN request acceptance ratio, compared to the offline case, where optimal mapping can facilitate balanced allocation of physical resources among the VN's requests. To provide a trade-off among the two VNE problem instances, hybrid approaches have been also considered, where incoming VN requests during a time window are simultaneously processed (e.g., [YYRC08, CRB12]).

2.4.1.2 Limited Physical Network Resources

Admission control is often ignored by assuming infinite capacity on substrate nodes and links, in order to deal with the complexity of the VNE problem (e.g.,[ZA06]). However, this results in service degradation for existing VNs on the physical substrate, due to bottlenecks created by resource overbooking. Finite substrate resources present a more realistic approach where some VN requests must be rejected or postponed to avoid violating resource guarantees for existing VNs (e.g., [YYRC08, CRB09]).

2.4.1.3 Heterogeneous Resources

The virtual resources may be heterogeneous in nature (e.g., routers, servers). Most frameworks follow a simplistic approach where only a specific resource (e.g., CPU) is requested for one type of node (server), whereas link capacity is examined as the single requirement related to a link (e.g., [YYRC08]). More realistic approaches are needed (e.g., [NMCS11b]), supporting nodes and links with a diverse set of parameters functional (e.g., node type, hardware type, virtualisation environment, location, etc.) and non-functional attributes (e.g., available capacity, QoS requirements).

2.4.1.4 Diverse Network Topologies

VN requests may include a diverse set of topologies. The VNO may require that the VN topology be tailored to the specific needs of the application to be delivered, for example, a hub-and-spoke geographically distributed VN topology to access to a centralized application server. Hub and spoke, tree, mesh, and others are quite common application topologies. In that sense, it is important to support arbitrary topologies.

2.4.1.5 Centralized vs. Distributed Approach

Centralized approaches have been devised mainly to solve the VNE problem, where a central entity (i.e., a broker) is responsible for receiving VN requests and assigning these sets of interconnected virtual resources to a set of physical resources that may belong to one or more InPs. To surmount scalability limitations, distributed algorithms have been suggested (e.g., [HLZ08]). However, these algorithms do not perform as well as existing centralized approaches (e.g., [YYRC08]).

2.4.1.6 Single-Domain vs. Multi-Domain

The central entity may not have access to full state information due to the multi-domain nature of the virtualized environment. Therefore approaches oriented for multi-domain have been suggested (e.g., [HLZ08, CSB10]) to coordinate the overall VNE.

2.4.1.7 Fixed vs. Dynamic Mapping

Static resource mapping approaches (e.g., [YYRC08, CRB09]), where during the lifetime of the request no change is allowed in resource assignment, may lead to inefficient resource utilization. Dynamic reconfiguration of VNs (links or links/nodes) via resource migration allows the resource allocation to be altered adaptively on the basis of current demand and performance (e.g.,[ZA06, FA06]). Migration of virtual resources improves resource utilization, increases energy efficiency and acceptance ratio of VN requests, and enhances network resilience in case of node/link failures, thus leading towards a more flexible and reliable network virtualisation environment.

2.4.1.8 Concise vs. Redundant Resources

A failure of a single substrate entity will affect all virtual entities that are mapped upon it. Therefore, in environments where fault-sensitive applications are deployed inside the virtual networks, it can be advisable to set-up backup resources that can be used as fall-back resources in case the corresponding primary resources fail. To do that, the embedding result itself can be resilient regarding node and/or link failures. The redundant resources can be reserved either per VN assignment (e.g., [RAB10]) or shared among several VNs to decrease provisioning costs (e.g., [SYL+10]), or eventually the pool of redundant resources can be geographically distributed to increase resilience to catastrophic disasters (e.g., [YQA+10]).

2.4.1.9 Efficient Resource Allocation vs. Energy-Aware Allocation

It is clear now the importance of reducing the energy consumption with respect to the carbon footprint. In [NPI+08b] it is discussed the impact on network protocols by putting network interfaces and components into sleep for saving energy, and in [GS03b] it is presented and evaluated two forms of power management schemes that reduce the energy consumption

of networks. To this extent, efficient resource allocation is required to be energy-aware to reduce energy costs.

2.4.1.10 Embedding Stages

Several of the proposed approaches decompose the problem into the node assignment phase and the link assignment phase, to reduce the overall complexity of resource assignment. The corresponding problems are solved sequentially. Initially, a greedy heuristic is employed for virtual to physical node assignment. Next, an appropriate link mapping technique is applied — either a shortest path algorithm in case of un-splittable flows (e.g., [ZA06]) or using multi-commodity flow algorithms (e.g., [YYRC08]) when flow bifurcation is supported. However, lack of correlation between the two phases restricts the solution space and can result in poor performance. Recent approaches tend to solve the two problems by providing some type of coordination among the two phases by taking link mapping constraints into account during the node mapping phase (e.g., [CRB09]).

2.4.1.11 Embedding Efficiency vs. Complexity

The existing approaches to handle the VNE problem can be further grouped onto three different categories according to two important evaluation metrics: i) the quality of the VNE solution - how good is the embedding solution compared to others, or even, how far is the obtained solution from the optimal embedding; ii) how much time do they require to perform the VNE:

i heuristics - obtain reasonable enough embedding solutions in a short amount of time (e.g. few tens of milliseconds [NMCS11b]);

ii meta-heuristics - obtain good enough embedding solutions in a medium amount of time (few hundreds of milliseconds [CRB09];

iii exact solution- obtain the optimal solution according to the objective function considered using linear programming solvers in a longer amount of time (e.g. ranging from few thousands of seconds with commercial solvers [cpl12] to many thousands of seconds with open-source solvers [Mak12]).

Noteworthy, the VNE solving time strongly depends both on the virtual network request size and on the physical network size (the examples given before are for small virtual network request sizes ranging from 2 to 10 virtual nodes, and physical network size of 50 nodes); the solving time curve strongly depends on the VN embedding method used. While the solving time is a direct measurable metric, the same does not apply to measure how good a VNE solution is. To this extent different auxiliary performance metrics have been suggested by the scientific community:

i VN Request Acceptance Ratio (or Rejection Ratio) - Represents the ratio between the sum of VN requests accepted (or rejected) over the sum of all VN requests.

ii VN Provisioning Cost - Represents the overall sum of physical resources, e.g. processing capacity and bandwidth, weighted by their cost that were allocated to accommodate the VN.

iii Substrate Revenue - This represents the sum of all types of virtual resources, e.g. processing capacity and bandwidth, weighted by their revenue that was demanded by the VN request.

iv Revenue over Provisioning Cost Ratio - Represents the ratio of the generated revenue over the VN provisioning cost. Also some authors have considered the opposite, i.e. Provisioning Cost over Revenue Ratio.

2.4.2 Resource Allocation

2.4.2.1 Heuristics

Zhu and Ammar [ZA06] tackle the offline VNE problem; they proposed two algorithms, one to embed the VN requests that aim to maintain the load of the physical links and of the physical nodes balanced per embedding, and another algorithm that aims to reduce the cost of VN re-assignment, i.e. it considers that the VN assignment is allowed to change during the VN lifetime. Both algorithms use a two step-approach to embed the VNs: in the first step a greedy algorithm is applied to map virtual nodes onto substrate nodes with more available resources; in the second step the shortest-distance path algorithm [MS97] is used to embed the virtual links. Admission control is not supported, since it is assumed that the substrate network resources are unlimited which is not realistic in on-demand provisioning scenarios. Fan and Ammar [FA06] proposed heuristic methods for constructing different flavours of reconfiguration policies for overlay networks. The proposed methods ignore node requirements, making them not suitable for the VNE problem.

Lu and Turner [LT06] also consider an offline VNE problem. They consider a set of premises about the virtual topology, i.e. the backbone nodes are star-connected and the access-nodes connect to a single backbone node. Based on these premises, an iterative algorithm is run, with different steps for core and access mapping. However, the algorithm can only work for specific topologies.

Ricci *et al.* [RAL03] proposed an assignment algorithm with the bandwidth constraint to be used in Emulab testbed [WLS+02b] that considers the online VNE problem. The assignment algorithm is based on the simulated annealing meta-heuristic [KJV83]. The proposed algorithm performed better than a genetic algorithm [GH88] used as a performance indicator. The node constraint in Emulab is provided as the exclusive use of nodes. Therefore, different virtual networks cannot share a substrate node.

Yu *et al.* [YYRC08] proposed a two-step mapping algorithm which considers finite resources in the physical network and supports path splitting (i.e. virtual link composed by different paths). In the first-step, it embeds the virtual nodes using a greedy node mapping algorithm (i.e. map the virtual nodes onto substrate nodes that have more resources available), and in the second it embeds the virtual links either using the k-shortest path algorithm [Epp98], or by solving the Multi-Commodity Flow Problem (MFP) [EIS75], if path splitting is used. To re-optimize the utilization of the substrate network, Yu *et al.* also proposed an algorithm that performs virtual link migration (i.e. to change the underlying mapping of virtual links previously assigned) periodically. However, this level of freedom can lead to a level of fragmentation that is unfeasible to manage on large scale networks. To improve the performance of [YYRC08], Liu *et al.* [LHCL11] proposed a new greedy algorithm for the embedding of the virtual nodes based on the "proximity principle". Instead of only considering the available capacity of the substrate nodes, the algorithm also considers the physical path between the former selected substrate node. To increase the utilization on the substrate nodes, Zhou *et al.* [ZLJ+10] proposed a VNE scheme with two-stage node mapping that considers node migration. This scheme improved the performance of [YYRC08]. To decrease the VN request rejection ratio, due to the shortage of bandwidth resources on some substrate links that are in heavy load state compared with others, Lü *et al.* [LHkW+11] proposed an adaptive VNE embedding algorithm based on the status feedback of the sub-

strate link. Their algorithm proved to decrease the VN request rejection ratio compared to [YYRC08]. Cheng *et al.* [CSZ+11] improved the performance of two existing embedding algorithms [ZA06, YYRC08] by applying Markov Random Walk model to rank a network node based on its resource and topological attributes.

A distributed algorithm was studied in [HLZ08]. It considers that the virtual topologies can be decomposed in hub-and-spoke clusters and each cluster can be mapped independently, therefore reducing the complexity of the full VN mapping. This proposal has lower performance when compared with centralized approaches.

In [LK09] a backtracking method based on sub-graph isomorphism was proposed; it considers the on-line version of the mapping problem, where the VN requests are not known in advance, and proposes a single stage approach where nodes and links are mapped simultaneously, taking constraints into consideration at each step of the mapping. When a bad mapping decision is detected, a backtrack to the previous valid mapping decision is made, avoiding a costly re-map.

A formal approach is taken by Chowdhury *et al.* [CRB09] to solve the online VN mapping problem, using a Mixed Integer Programming (MIP) formulation. A two step approach is applied to embed VNs onto the substrate. In the first step, the VN request is mapped onto "meta-nodes" and "meta-links" by solving MIP formulation that has been relaxed (i.e. the binary constraint on the node assignment variable has been relaxed to a real number), and in the second step the meta-nodes are mapped onto substrate nodes either using randomized rounding or deterministic rounding, and the virtual links are assigned to physical paths by solving the Multi-Commodity Flow Problem (MCFP). Compared to the previous state of the art heuristics, i.e. [ZA06, YYRC08], the formulation proposed by Chowdhury *et al.* provides a better coordination of the two phases, since an *augmented substrate graph construction* is used. Chowdhury *et al.* [CRB12] extended their preliminary results [CRB09] and included a generalized window-based VN embedding to evaluate the effect of look ahead on the mapping of VNs.

To better improve the coordination between the mapping phases, Gao *et al.* [GYA+10] added constraints after the MIP relaxation, that consider greedy and progressively mapping of virtual nodes.

Butt *et al.* [FBCB10] proposed a topology aware heuristic for VN mapping, and also suggested algorithms to avoid bottlenecks on the physical infrastructure, where they consider virtual node reallocation and link reassignment for this purpose.

In Razzaq and Rathore [RR10] approach, virtual nodes are mapped as close as possible to each other, thus ensuring that the paths found for the virtual link mapping are the shortest in length. Based on the premise of reducing the number of bottleneck nodes and leave more available nodes for future VN requests, Razzaq *et al.* [RSH11] devised an algorithm that preserves the resources of those nodes for future use.

Nogueira *et al.* [NMCS11b] proposed a heuristic that takes into account the heterogeneity of the VNs and also of the physical infrastructure. Chen *et al.* [CLW12] followed a different approach to reduce the physical network fragmentation and at the same time to increase the revenue of the Substrate Network (SN). They proposed an iterative algorithm, i.e. "border matching", that is inspired in the theory of *power law* and $effective\ diameter\ law$ [FFF99]. They consider that most nodes have a low degree of connectivity (i.e. few physical links). To reduce the network fragmentation, the virtual nodes with a low degree of connectivity are firstly embedded onto the physical nodes with a low degree of connectivity, leaving the remaining virtual nodes to be embedded onto the physical nodes with higher degree of connectivity.

Botero *et al.* [BHFM12] proposed an algorithm to solve the VN mapping problem, which

also considers the CPU demand of the "hidden hops" (i.e. intermediate physical nodes). They argue that each substrate node that belongs to a physical path demands a small amount of CPU to accommodate a virtual link. To this extent, their algorithm also considers the CPU demand of the intermediate nodes. Li *et al.* [LWD+14] applied the top-k dominating model to rank the nodes, aiming to balance different resources for evaluating the resources of the network nodes, and consider the hops of the substrate paths in the node mapping stage, which can reduce the hops of the substrate paths and lead to high utilization of the substrate resources.

2.4.2.2 Meta-Heuristics

To improve the performance of existing VNE algorithms mostly based on heuristics, several approaches based on meta-heuristics, e.g. ant colony, simulated annealing, Particle Swarm Optimization (PSO), shuffled frog leaping have been suggested.

Fajjari *et al.* [FAPZ11a] proposed an algorithm to perform the VNE based on the Ant Colony meta-heuristic [SH00]. Wenzhi *et al.* [WSYX11] proposed a meta-heuristic algorithm based on the shuffled frog leaping algorithm for solving the VNE problem.
Zhang *et al.* [ZQGL11] proposed a flexible VNE algorithm with guaranteed load balancing based on the meta-heuristic simulated annealing [KJV83]. Cheng *et al.* [CSZ+12] extended his preliminary work [CSZ+11] on node-ranking and proposed two algorithms to perform the VNE based on the meta-heuristic PSO [Ken10]. Zhang *et al* [ZCS+13] extended Cheng work [CSZ+12] to take the location constraints on the virtual nodes into consideration.

2.4.2.3 Exact Solution

Inführ and Günther [IR11] introduced the VNE problem with delay, routing and location constraints, and to solve VNE they applied a multi-commodity flow integer linear program. However, neither an evaluation of the proposed formulation nor a comparison with existing methods was performed.

Alkmim *et al.* [ABSdF11] proposed a mathematical formulation that aims to: i) map virtual routers and virtual links; ii) minimize the bandwidth consumption; and iii) minimize the time required to instantiate a virtual router. In contrast to this work, we also aim to optimize link load, CPU load distribution, energy consumption and VN migration. To reduce the VNE solving time, a set of approximate algorithms based on randomized rounding and iterative randomized rounding techniques were proposed in [ABdF13].

To minimize the VN request rejection ratio, Tran *et al.* [TCTG12] proposed a reactive reconfiguration mechanism mathematically formulated as an ILP problem. This algorithm re-embeds virtual networks previously assigned to make room for VN requests rejected. To minimize the VN reconfiguration cost (e.g. migration of virtual nodes), Tran *et al.* [TTG13] extended their initial formulation to maximize the net gain of the VN reconfiguration.

2.4.2.4 Remarks

The VNE problem has received increasing attention by the research community in the last few years. A concise survey on VNE methods for resource allocation is presented in [HPN09] and a detailed set of parameters on the VNE problem was given in [SHT12]. An extended survey on resource discovery and allocation in network virtualisation is given by Belbekkouche *et al.* in [BHK12]. The authors claim that the resource allocation problem has been addressed using mainly heuristic solutions. Consequently, exact resolution needs to be explored

in the future by proposing new formulations that can resolve some instances of the problem in reasonable time. Table 2.1 provides a concise view on the existing Virtual Network Embedding approaches.

Although all these algorithms provide a solution for the VN mapping problem, an optimal solution for the embedding task and its efficiency is not provided. Also, some of them fail to solve the assignment problem as a simultaneous optimization of the virtual node and link placement, which leads to non-optimal solutions. To this end, we propose and evaluate different objective functions in Chapter 4, that aim at simultaneously optimizing the load on the nodes, the load on links and the bandwidth consumption. A comparison with existing heuristics [ZA06, YYRC08, CRB09] is also provided.

2.4.3 Energy-Aware Resource Allocation

Nowadays, network operators are required to pay more attention to the power consumption of their networks, either due to environmental policies imposed by the local governments or due to energy costs [AN08]. In fact, the power consumption of the data plane in idle mode is 90% of the one in full mode [CSB+08, LGL+11], while the power consumption of the control plane, i.e. CPU, is about 70% [BH09] of the one in idle mode.

Recent research works focused on the energy consumption aspects. Zhang *et al.* [SZC+12] proposed an efficient energy-aware algorithm using a consolidation technique to reduce the energy consumption. A comparison with the optimal solution is not provided.

Botero *et al.* [BHD+12] extended their preliminary work and proposed a MIP with the aim of providing optimal energy efficient embeddings. This formulation, despite providing the optimal solution for energy embeddings, it does not consider the resource allocation objective, and therefore, it penalizes the VN acceptance ratio.

Rodriguez *et al.* [RABdF12] proposed a sequential execution of two ILPs: in the first step it maps the virtual networks onto the substrate network with the aim of saving energy; and the second determines the path in the substrate used to transfer the images. However, this approach is mainly oriented for energy saving, which can jeopardize the VN request acceptance ratio.

Recently, Fischer *et al.* presented in [FBTB+13] an updated survey on VNE algorithms, and proposed a classification scheme of the current approaches. The authors pointed out that energy-efficient algorithms or security constraints have been up to now neglected by the scientific community. Table 2.2 provides a concise view on the existing Virtual Network Embedding approaches for Energy-Aware.

Although all these algorithms provide a solution for the VN mapping problem, an optimal solution for energy consumption and resource allocation is not provided.

2.4.4 Virtual Network Resilience

To handle physical network failures, Rahman *et al.* [RAB10] incorporate single substrate link failures on the VNE problem and proposed a heuristic to solve the problem. To handle resiliency protection against network failures during the process of online VN services provision, Chen *et al.* [CLW+10] proposed an efficient resource allocation approach to balance the trade-off between service resource consumptions and service resiliency.

Cai *et al.* [CLX+10] proposed an algorithm to address network changes in response to network growth, node failures or node joining/leaving.

To reduce the physical network fragmentation, i.e. due to the VNs dynamic lifecycle, Fajjari *et al.* [FAPZ11b] proposed a virtual network reconfiguration algorithm, that relocate the VN star topology, instead of the whole VN topology [ZA06], with the aim of minimizing

Table 2.1: State of the Art on Virtual Network Embedding - Efficient Resource Allocation Algorithms.

Reference	**Method**	**Major Contributions**
Zhu and Ammar [ZA06]	Heuristic	Provides a balanced link and node load in the SN.
Fan and Ammar [FA06]	Heuristic	Reconfiguration policies for overlay networks.
Lu and Turner [LT06]	Heuristic	Embedding in specific backbone-star VN topologies.
Lischka and Karl [LK09]	Heuristic	Backtracking algorithm based on a subgraph isomorphism search method.
Yu *et al.* [YYRC08]	Heuristic	Added path splitting and considered finite resources.
Chowdhury *et al.* [CRB09, CRB12]	Heuristic	Added better coordination between the two mapping phases. Added generalized window-based VN embedding.
Gao *et al.* [GYA+10]	Heuristic	Added constraints after the MIP relaxation, and also greedy and progressively mapping of virtual nodes.
Butt *et al.* [FBCB10]	Heuristic	Added topology aware heuristic for VN mapping.
Houidi *et al.* [HLZ08]	Heuristics	Distributed VNE algorithm.
Zhou *et al.* [ZLJ+10]	Heuristic	Multiple physical nodes to host a virtual node.
Razzaq *et al.* [RR10, RSH11]	Heuristic	Nodes are mapped as close as possible. Preserves the resources of nodes loaded for future use.
Nogueira *et al.* [NMCS11b]	Heuristic	Add heterogeneity of the VNs and also of the SN.
Lü *et al.* [LHkW+11]	Heuristic	Adaptive VNE embedding algorithm based on the status feedback of the substrate links.
Liu *et al.* [LHCL11]	Heuristic	Algorithm based on the proximity principle.
Cheng *et al.* [CSZ+11]	Heuristic	Added Markov Random Walk model to rank a network node.
Botero *et al.* [BHFM12]	Heuristic	CPU demand of the hidden hops.
Chen *et al.* [CLW12]	Heuristic	Iterative algorithm following the "border matching" approach.
Li *et al.* [LWD+14]	Heuristic	Top-k dominating model to rank the nodes.
Ricci *et al.* [RAL03]	Meta-heuristic	Algorithm based on the simulated annealing.
Fajjari *et al.* [FAPZ11a]	Meta-heuristic	Algorithm based on the meta-heuristic Ant Colony.
Wenzhi *et al.* [WSYX11]	Meta-heuristic	Algorithm based on the meta-heuristic Shuffled Frog Leaping.
Zhang *et al.* [ZQGL11]	Meta-heuristic	Flexible algorithm with guaranteed load balancing based on the meta-heuristic simulated annealing.
Cheng *et al.* [CSZ+12]	Meta-heuristic	Algorithm based on the meta-heuristic PSO.
Zhang et al. [ZCS+13]	Meta-heuristic	Added location constraints on the virtual nodes into consideration.
Inführ and Günther [IR11]	Exact Solution	Introduced the VNE problem with delay, routing and location constraints.
Alkmim *et al.* [ABSdF11, ABdF13]	Exact Solution	Added formulation to minimize the time required to instantiate a virtual router. Added set of approximative algorithms to reduce the solving time.
Tran *et al.* [TCTG12, TTG13]	Exact Solution	Proposed a reactive reconfiguration mechanism using ILP. Maximized the net gain of the VN reconfiguration.

Table 2.2: State of the Art on Virtual Network Embedding - Energy Aware Algorithms.

Reference	**Method**	**Major Contributions**
Zhang *et al.* [SZC+12]	Heuristic	Energy-aware algorithm using a consolidation technique.
Botero *et al.* [BHD+12]	Exact Solution	Added Energy-aware MIP formulation.
Rodriguez *et al.* [RABdF12]	Exact Solution	Added Energy-aware ILP formulation.

the cost of reconfiguration. Zhang and Qiu [ZQ11] proposes an algorithm that identifies virtual nodes and virtual links mapped in a non optimal way, and migrates them to better locations to save SN resources.

Yeow *et al.* [YWK11] added node and link redundancy for reliability, and use a multi-commodity flow problem formulation to solve the VNE. To increase the resilience to link failures, they also consider path-splitting.

To solve the Survivable Virtual Network Embedding (SVNE) from a regional failure scenario, Yu *et al.* [YQA+10] consider failure dependent protection approach, whereby there is a backup solution associated with each regional failure scenario. To this extent, they propose two heuristic algorithms with the objective of minimizing the redundant resources/cost. To endorse the SVNE problem, Sun *et al.* [SYL+10] proposed two relaxation-based algorithms: Lagrangian relaxation and decomposition. To handle network resource failures, Houidi *et al.* [HLZ+10] proposed an adaptive embedding algorithm that is based on an existing distributed algorithm previously proposed in [HLZ08], to deal with three resource failure scenarios: virtual node, substrate node and link failures.

To increase the resource efficiency of backup resources, i.e. reuse them by other VNs to make room for accepting more incoming VN requests, Guo *et al.* [GWMT11] proposed two shared backup network provision schemes for virtual network embedding: shared on-demand approach and shared proactive approach. Herker *et al.* [HKA13] presented a survey on the survivable virtual network embedding problem and different approaches to solve this problem. The authors point out the lack of research in the survivability of VN embedding issue in a multi-domain environment, and they also propose to extend existing approaches to handle multiple link and node failures at the same time. Table 2.2 provides a concise view on the existing Virtual Network Re-embedding approaches.

Although all these algorithms provide a solution for the VN mapping problem, an optimal solution for Virtual Network Re-Embedding (VNRE) taking into account the migration costs is not provided. Also, some of the proposals fail to solve the assignment problem as a simultaneous optimization of the virtual node and link placement, which leads to non-optimal resource re-allocation solutions. Our approach, the Virtual Network Re-Embedding Node-Link Formulation (VNRE-NLF), applies a node-link formulation to solve the VNRE problem in a single step using the multi-commodity flow constraint. This approach provides the optimal solution for the objective problem considered, both in terms of node migration and bandwidth re-allocation, for a given set of weights in the objective function.

2.4.5 Other VNE Research Directions

Other VNE research directions have been proposed by the scientific community. This sub-section is an attempt to frame these research directions.

2.4.5.1 Resource Description and Discovery

Houidi *et al.* [HLZB09] proposed a resource description and clustering schema for Virtual Network resource discovery. However, the meta-information of the virtual resource is

Table 2.3: State of the Art on Virtual Network Resilience Algorithms.

Reference	Method	Major Contributions
Ammar and Zhu [ZA06]	Heuristic	Added VN reconfiguration algorithm to reduce the physical network fragmentation.
Rahman *et al.* [RAB10]	Heuristic	Incorporate single substrate link failures on the VNE problem.
Cai *et al.* [CLX+10]	Heuristic	Algorithm to address network changes in response to network growth, node failures or node joining/leaving.
Fajjari *et al.* [FAPZ11b]	Heuristic	Relocate the VN star topology, instead of the whole VN topology.
Zhang and Qiu [ZQ11]	Heuristic	Identifies virtual nodes and virtual links that were initially mapped in a non optimal.
Yeow *et al.* [YWK11]	Heuristic	Added node and link redundancy for reliability.
Yu *et al.* [YQA+10]	Heuristic	Regional failure scenario.
Sun *et al.* [SYL+10]	Heuristic	Relaxation-based algorithms.
Houidi *et al.* [HLZ+10]	Heuristic	Distributed adaptive embedding algorithm.
Guo *et al.* [GWMT11]	Heuristic	Proposed two shared backup network provision schemes.

registered randomly to the repositories in this method. Moreover, the searching algorithm needs to search all the repositories blindly to find out the virtual resource candidates matching the VN user's query. To improve the resource discovery efficiency, Lv *et al.* [LWH+10] proposes the design of a virtual resource organization and discovery framework.

2.4.5.2 Hierarchical Single domain

Based on a graph matching method [Bun00], Ghazar and Samaan [GS11] proposed a hierarchical substrate management framework that performs concurrent searches for more than one efficient VN mapping solution from which the best solution is selected.

2.4.5.3 Multi-Domain

To endorse the multi-domain scenario, i.e. multiple infrastructure providers, Houidi *et al.* [HLAZ11] proposed an exact embedding algorithm that enables optimal and simultaneous node and link mapping across multiple substrate networks. The goal is to increase the acceptance ratio of VN requests while decreasing the provisioning cost for InPs. Chowdhury *et al.* [CSB10] proposed a policy-based inter-domain VN embedding framework that embeds end-to-end VNs in a decentralized manner.

2.4.5.4 Resource Pricing

To address the multi-provider service negotiation and contracting in network virtualisation, Zaheer *et al.* [ZXB10] proposed an open market model and enabling framework for automated service negotiation and contracting in network virtualisation.

2.4.5.5 Security

VN requests can contain security requirements, e.g. VN to securely inter-connect different bank offices; to this extent Fischer and De Meer [FDM11] added a set of security constraints to the VNE problem.

2.4.5.6 Cloud Computing

Papagianni *et al.* [PLP+13] proposed an algorithm to endorse the VNE on cloud computing scenarios. Leivadeas *et al.* [LPP12] added socio performance metrics.

2.4.5.7 Self-organizing

Instead of focusing on the initial VNE phase, Marquezan *et al.* [MGNB10] proposed a self-organizing model to address the management of substrate resources during the lifetime of a VN.

2.4.5.8 Wireless Physical Links

To tackle VNE challenges in wireless multi-hop networks, e.g. link interference, Yun *et al.* [YY11] proposed a preliminary algorithm. Yang *et al.* [YLZ+12] proposed a karnaugh-map-like online embedding algorithm to endorse the VNE in wireless networks.

2.5 Virtual Network Migration

Another major obstacle lies in the VN migration problem. It is of utmost importance to provide mechanisms for the InPs to freely move VNs across the physical network to either reduce the fragmentation of the physical network, or to offload some physical resources, or even to move away VNs from physical resources that are on imminent fail over.

One of the major features of Network Virtualisation is the possibility to move virtual resources, i.e. Virtual Router (VR), on-demand and seamlessly from one physical host to another without losing network connectivity. The ability to migrate virtual resources gives infrastructure providers a great amount of flexibility and enables them to optimise their resource utilisation.

The Virtual Machine (VM) migration was initially proposed for data-centres as a way to move the CPU load from one physical server to another. This feature is not only important for load balancing purposes, but it can also be applicable for planned maintenance operations, i.e. moving away all the VMs from one physical server that needs to be rebooted or shut down for maintenance to different physical servers.

In order to reduce the downtime due to the VM migration process, Clark *et al.* [CFH+05] proposed the live migration of VMs, within the same Local Area Network (LAN), which allows a VM to be migrated while still running. This not only reduces significantly the downtime provoked by the VM migration, but it also makes the migration process seamless to the end-users or to the running applications. Ma *et al.* [MLL10] proposed some improvements on the live VM migration process, which both reduces the overall migration time and the total data transmitted in the order of 30%.

The VM migration approach is also applicable and important to the networking area, where it can be used for networking maintenance operations or for networking service deployment. For instance, it can be used to move critical (or non-critical) VRs from physical hosts that need some kind of maintenance operation, e.g. firmware update or hardware module replacement or upgrade, without disrupting the routing protocols. This is much preferable and human error safe than manually configuring routing protocol metrics to move away the networking traffic from that physical router.

With that in mind, Wang *et al.* [WvdMR07] proposed Virtual ROuters On the Move (VROOM) as a primitive for networking management tasks, which makes it possible to move virtual routers freely without changing the IP-layer topology. Wang *et al.* [WKB+08]

extended their prior research work on VR migration and proposed to decouple the data plane from the control plane of the VRs, which results in no performance impact on the data traffic when a hardware data plane is used, and very low impact when a software date plane is used.

The VR migration feature was also considered and evaluated on Internet Protocol Television (IPTV) scenarios by Marquezan *et al.* [MNG+09]. Pisa *et al.* [PFC+10] proposed a new migration model for XEN, using also data and control plane separation, which outperforms the XEN standard migration model.

Lo *et al.* [LAZ12] used the virtual router migration [WKB+08] as a primitive to perform the virtual network migration, i.e. the migration of an entire VN, and also proposed three algorithms to address the VN migration scheduling problem and to minimize the total migration cost.

Although the separation of the data plane from the control plane seems to be a very effective approach on the VR migration, since it reduces the downtime of the VN, it is also a limiting factor on the conception of new network architectures and on the deployment of new network protocols.

We argue that not only the VN migration process should be independent of the networking protocols that are running on the virtual network, but also no assumption should be taken on the router architecture itself. Therefore, we consider each VR as a black-box and propose the VR Cloning as an alternative to the current VR live migration process [WvdMR07].

2.6 Future Internet Research Projects

Several worldwide research initiatives have investigated and deployed new architectures, and also transition mechanisms to mitigate the current Internet limitations. This section describes the existing research projects on Future Internet.

2.6.1 Federated E-infrastructure Dedicated to European Researchers Innovating in Computing network Architectures

The Federated E-infrastructure Dedicated to European Researchers Innovating in Computing network Architectures (FEDERICA) project created a European wide "technology agnostic" infrastructure made of Gigabit circuits, transmission equipment and computing nodes capable of virtualisation to host experimental activities on new Internet architectures and protocols [FED14]. The FEDERICA network was based on the Research & Education multi-gigabit networks footprint. Circuits are terminated in Points of Presence (PoPs) of National Research and Education Networks (NREN) and GÉANT, hosting FEDERICA nodes capable of virtualising hosts e.g. open source routers and end nodes. Virtual slices of FEDERICA's infrastructure may be allocated to network researchers for testing even with disruptive experiments within a large production substrate.

2.6.2 Global Environment for Network Innovations

GENI is a virtual laboratory for exploring future internet at scale, creates major opportunities to understand, innovate and transform global networks and their interactions with society [gen14]. Dynamic and adaptive, GENI opens up new areas of research at the frontiers of network science and engineering, and increases the opportunity for significant socio-economic impact.

2.6.3 Trilogy

The Trilogy project [TRI14] focused on the control functions of the Internet, in particular Reachability and Resource Control. Two new protocols have resulted from this research project: Multi-Path Transmission Control Protocol (MP-TCP) protocol that enables a TCP connection to be spread over multiple paths; CONgestion EXposure (CONEX) that enables all network elements to have full visibility of end-to-end congestion, information that is currently only available at the endpoints.

2.6.4 4WARD

The concept of Network Virtualisation as an overall vision of virtualising complete networks that can realize independent architectures and coexist with the current Internet was the focus of the 4WARD project [4WA14].

In 4WARD approach, it is combined on one hand innovations needed to improve the operation of any single network architecture, and on the other hand multiple different and specialised network architectures that are made to work together in an overall framework. A number of integrated feasibility tests and prototyping activities were carried out during the evaluation phase of the project. Additionally, new concepts and algorithms for provisioning, embedding and management were defined and evaluated throughput the project.

2.6.5 Open-Access Research Testbed for Next-Generation Wireless Networks

Open-Access Research Testbed for Next-Generation Wireless Networks (ORBIT) [ORB14] is a two-tier laboratory emulator/field trial network testbed designed to achieve reproducibility of experimentation, while also supporting evaluation of protocols and applications in real-world settings. The laboratory-based wireless network emulator uses a novel approach involving a large two-dimensional grid of 400 802.11 radio nodes which can be dynamically interconnected into specified topologies with reproducible wireless channel models.

The ORBIT testbed is available for remote or on-site access by other research groups nationally. Additional research partners and testbed equipment/software contributors are actively sought from both industry and academia.

2.6.6 GEYSERS

Generalized Architecture for Dynamic Infrastructure Services (GEYSERS) vision is to qualify optical infrastructure providers and network operators with a new architecture, to enhance their traditional business operations [gey14]. Optical network infrastructure providers will compose logical infrastructures and rent them out to network operators; network operators will run cost-efficient, dynamic and mission-specific networks by means of integrated control and management techniques.

To address these points, the main technical focus of GEYSERS has been the definition and implementation of an architecture and an optical network solution, capable of provisioning "Optical Network and IT resources" for end-to-end service delivery. In this context, one of the most important issues that the GEYSERS consortium has concentrated on was to develop software solutions to make the optical network more flexible and programmable, through virtualisation, in order to optimally serve the needs of highly dynamic computing applications and their variable workloads.

2.6.7 Scalable & Adaptive Internet soLutions

Scalable & Adaptive Internet soLutions (SAIL) aimed to both design technologies for the Networks of the Future and develop techniques to transition from today's networks to such future concepts [SAI14]. Cloud networking in the context of SAIL is about integrating the on-demand provisioning of network resources, through virtualisation, to data centre resources. The concept of a flash network slice, a network resource that can be provisioned and dimensioned on a time scale comparable to existing compute and storage resources, was proposed within the project to overcome the limitations of VPNs with respect to elasticity and dynamicity.

2.7 Standardisation and Research Groups

With the prominence interest of Industry on Network Virtualisation, standardisation entities have defined research groups with interest of identifying potential needs for standardization.

2.7.1 Internet Research Task Force

The Internet Research Task Force (IRTF) created the Virtual Networks Research Group (VNRG) to identify architectural challenges resulting from Virtual Networks, addressing network management of Virtual Networks, and exploring emerging technological and implementation issues [IRT14]. However, due to lack of energy and participation the VNRG has decided to close on February of 2012.

2.7.2 European Telecommunications Standards Institute

To accelerate progress, a new network operator-led Industry Specification Group (ISG) was setup under the auspices of European Telecommunications Standards Institute (ETSI), to define the requirements and architecture for the virtualisation of network functions and to address the technical challenges. The aim of the NFV is to offer a new way to design, deploy and manage networking services. It is designed to consolidate and deliver the networking components needed to support a fully virtualised infrastructure – including virtual servers, storage and even other networks. It utilizes standard IT virtualisation technologies that run on high-volume service, switch and storage hardware to virtualise network functions. It is applicable to any data plane processing or control plane function in both wired and wireless network infrastructures. ETSI has published the first five specifications on NFV [ETS13a, ETS13b, ETS13c, ETS13d, ETS13e].

2.7.3 International Telegraph Union - Telecom

The ITU-T Focus Group on Future Networks (FNs) was set up to collect and identify visions of future networks, based on new technologies, assess the interactions between future networks and new services, familiarize ITU-T and standardization communities with emerging attributes of future networks, and encourage collaboration between ITU-T and FN communities [ITU14, MEN+13]. ITU-T has developed and published during 2009–2012 four recommendations: Y.3001 [ITU11], Y.3011 [ITU12a], Y.3021 [ITU12b], and Y.3031 [ITU12c].

2.8 Summary

This chapter presented a state-of-the-art of network virtualisation with special emphasis on mechanisms for resource control and allocation, and existing methods to perform virtual network migration. Virtual network embedding algorithms and mathematical formulation were also addressed. The concepts and terminology were described. Future Internet research projects were pointed out, and the standardisation organizations responsible were mentioned.

Chapter 3 - Network Virtualisation: Building Blocks

"The four building blocks of the universe are fire, water, gravel and vinyl."

—Dave Barry

Prior to provision virtual networks on demand, an Infrastructure Provider (InP) must firstly have mechanisms to discover, control, and allocate virtual resources individually. This chapter aims at providing a contribution in this direction. It presents an architecture for automatic virtual network creation and the corresponding approach for control of virtual network resources, which has been proposed in [MSC09, 4WA10a, Mel10]. It comprises the main building blocks of the network and their functionalities, and the communication required to provide the virtual network creation. Furthermore, this Chapter includes a description of a method that was proposed in [4WA10b, MCS13a, MSC14] to perform seamless VN migration. This method enables the operator to freely move the virtual resources across the physical network and without affecting the VN itself, either due to load balancing policies or even maintenance reasons.

This Chapter starts with a description on the mechanisms and algorithms required to control virtual network resources. In section 3.2, a resource allocation, monitoring and controlling framework will be proposed and evaluated to address these requirements. Moreover, in section 3.3 is provided a description on the VN migration process, while in section 3.3.3 is presented and evaluated the architecture of the VN clone migration method. Section 3.4 provides a summary of the chapter.

For further details on the framework proposed and also on the virtual network clone method please see the Appendix sections A, and F, respectively.

3.1 Controlling Virtual Network Resources

This section presents an architecture proposed for automatic virtual network creation and the corresponding approach for control of virtual network resources. It comprises the main building blocks of the network and their functionalities, and the communication required to provide the virtual network creation.

3.1.1 Building Blocks

Prior to the creation of a new virtual network, the InP should find the adequate physical resources, taking into account the current state of the network and the level of occupancy of the network resources, at that moment, in the case of an "on-the-fly" reservation, or at the requested future time, in the case of an advance reservation. In practice, the mapping of virtual nodes into physical nodes is often constrained by physical location, in which case the selection of the physical node to associate with a specific virtual node is fixed, or limited to a small set of choices. This is the case of edge nodes, or PoPs, which for economical reasons are supposed to be physically close to customers or end-users. Typically, at least one virtual node should be located in each geographic area (e.g. city, region) where the service is to be deployed. By contrast, for other types of virtual nodes, physical location is not relevant from the VNP point of view – this is usually the case with core nodes, with no direct connection to end users. The mapping of virtual nodes and links into physical nodes and links should follow a set of constraints and optimization criteria to be defined by the InP (e.g. minimum cost, resource load balance, segregation of resources according to the service type), and can be materialized in a complex algorithm. Physical resource control and virtual resource embedding include three basic components (Figure 3.1):

- VN admission control: the InP verifies whether there are available resources to fulfil the virtual network request made by the VNP, and decides whether it can be accepted or not. VN admission control does not necessarily find an optimal solution for a VN yet – this is supposed to be the role of resource mapping, as described below – it only verifies that, at least, one solution can be found.

- Resource mapping: the InP identifies the set of possible substrate nodes and links to host the requested virtual network and selects the optimal solution. The VN embedding is a Non-deterministic Polynomial-time hard (NP-hard) problem and a trade-off had to be chosen between computation time and embedding optimization.

- Re-optimization: the network state keeps changing as new VNs are setup and others are torn down, or as a result of node or link failure conditions. This often leads to inefficient utilization of resources, in which some parts of the network infrastructure (either link resources or node resources) become excessively loaded, while others are under-utilized. Therefore, the capability to re-optimize the allocation of virtual resources across the substrate network without traffic disruption, either on a periodic basis or triggered by a specific event (e.g. when a specific resource availability threshold has been reached) is a key VN requirement. In addition, the resource management process typically makes use of two auxiliary components:

  . Discovery: this function is responsible for discovering network resources and providing them available for the admission control and mapping functions.

  . Monitoring: this function collects real-time information from nodes and links, and signals any significant deviation from the expected network behaviour.

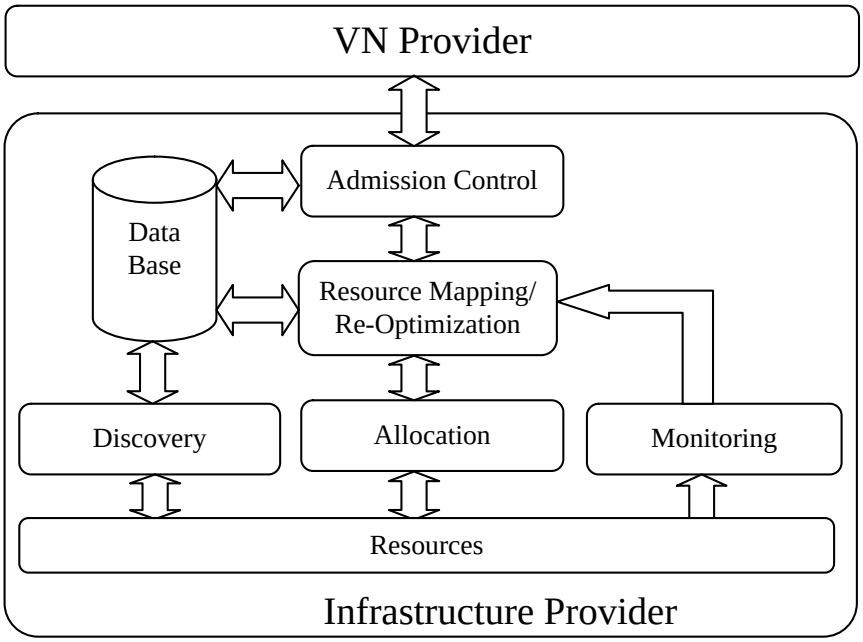


Figure 3.1: Infrastructure Provider (InP) Building Blocks.

3.1.2 VN Setup Negotiation Process

As explained before, it is likely that in most cases the VNP has limited knowledge of the physical resources provided by the InP. On the other hand, there will be multiple candidate InPs to provide the required network resources in many cases. Therefore, the VNP must be able to inquire a set of candidate InPs and, based on their responses, select one or more that will actually provide the network resources simultaneously and cooperatively. This requires the VNP/InP negotiation to be divided in two stages, as depicted in Figure 3.2:

- Query: the VNP inquires the InP about the availability of resources to build a specific VN. The InP is expected to provide a yes/no reply, possibly with additional information, e.g. cost, QoS parameters.

- Commit: the VNP requests the reservation of network resources and the InP enforces the corresponding resource reservation, after establishing the mapping between virtual and physical resources. It should be noted that virtual networks can be created "on-the-fly", i.e. just before the resource is required, or in advance, i.e. at some future point in time. In either case, a time may be optionally specified for resources to be released; otherwise, the VN will only be torn down through explicit signalling.

From the InP point of view, a relevant issue is how to map the blocks represented in Figure 3.1 into these two phases. The right hand side of Figure 3.2 suggests a possible approach, but this will be further discussed in the next section. The VNP is expected to build the virtual network topology and define resource capacity requirements, namely link bandwidth and node computational capability. As discussed previously, other characteristics such as geographical location of the edge nodes will be needed in most cases. The information provided by the VNP to the InP must contain a model to describe the virtual network topology (e.g. graph),

with a set a virtual nodes and virtual links and including the applicable constraints (e.g. link bandwidth, node computational capacity, physical location). Each virtual node and virtual link must be characterized by a number of parameters. A tentative list of parameters to characterise virtual networks, nodes, and links is shown in Table 3.1. The list of parameters present was inspired in the work of Carapinha and Javier [CJ09].

Table 3.1: Virtual Network Characteristics.

Network virtualisation components	**Parameters**
Virtual network	Virtual network ID Start time of the service End time of the service
Virtual node	Node ID Physical location Physical node ID CPU Memory Storage
Virtual link	Link ID End points Physical Path Bandwidth Delay

3.1.3 Signalling and Control

As explained earlier the creation of a VN involves two phases, query and commit. Figure 3.2 depicts the VN creation process: the left hand side represents the message flow between the VNO and the VNP when a new VN is requested. In this example, the VNP contacts two candidate InPs to accommodate that VN, InP X and InP Y, and then decides to opt for InP X, based on some criterion, e.g. InP X provides the requested resources at lower cost than InP Y. Then, the process continues with the commit phase, in which the resources are actually reserved.

3.1.3.1 Query Phase

The process is started when the VNO sends a VN Request to the VNP, including the VN topology and its constraints, node constrains (i.e. physical location, CPU) and link constrains (i.e. bandwidth, delay). The VNP is then in charge of assigning a VN ID and an ID for each virtual resource. Then, a $VN\_Query.request$ message is sent to one or more InPs. This message must contain the VN ID, the nodes/links IDs, the VN topology and its specifications, according to Table 3.1. At a first stage, the InP will perform a VN Admission Control, checking that every requested virtual node and virtual link can be accommodated by at least one substrate node and substrate link, respectively. If one or more virtual nodes and/or virtual links cannot be accommodated due to the lack of resources in the substrate (i.e. insufficient bandwidth or computational resources), the process should be stopped here and the $VN\_Query.response$ is sent to the VNP, indicating that the request cannot be fulfilled (optionally, indicating the reason of the failure). Otherwise, if every virtual resource can be accommodated by the substrate, then a $VN\_Query.response$ message with a positive reply, including the VN ID, should be sent to the VNP. It should be noted that the VN Admission Control is not expected to find the optimal solution for the virtual-to-physical resource mapping problem yet, but only whether at least one solution exists. This is understandable, since

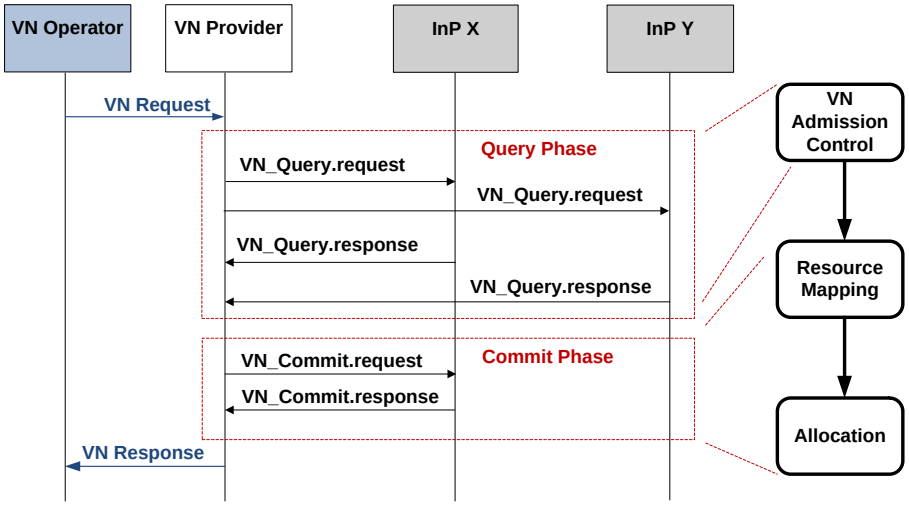


Figure 3.2: VN Creation Sequence Chart and Flow Diagram.

the resource mapping is the most complex step of this process, and in many cases, the query will not be followed by a corresponding commit. However, this may not be the case in all circumstances, and the InP may decide to go further than just performing admission control. So, optionally, at a second step, the InP may perform a pre-reservation by mapping the VN into the network infrastructure, making use of an optimization algorithm, knowing a priori that every requested virtual resource can be accommodated. The choice of the first available or optimal solution may be based on different criteria, such as: preferring substrate nodes with more resources, selecting substrate links with more available bandwidth, link aggregation (i.e. virtual link that maps into 2 substrate links) and link segmentation (i.e. virtual link spanning through multiple substrate links). After obtaining the best solution, the InP must perform a reservation (at this stage, at logical level only) of the concerned substrate resources. This reservation will be cancelled if a specific timeout expires without any effective reservation being made by a corresponding $VN\_Commit.request$ from the VNP, and the reserved resources will be released again. Optionally, this timeout should be included in the $VN\_Query.response$ message.

3.1.3.2 Commit Phase

If the VNP receives one or more positive responses from the candidate InPs in the query phase, the process will typically continue with the selection of the InP (if more than one candidate InP answered positively), followed by a $VN\_Commit.request$, with the corresponding VN ID. After receiving the $VN\_Commit.request$, the InP verifies whether a pre-reservation exists for the given VN ID and if it is still valid. If so, it proceeds with the allocation of the virtual resources. After allocating each virtual resource, it sends a $VN\_Commit.response$, including the VN ID and the ID of each virtual resource. If the VN ID is not valid or the pre-reservation has expired, or if for some reason it cannot allocate any particular virtual resource, the InP should send a $VN\_Commit.response$ indicating the reason of the negative response. If a pre-reservation has not been performed beforehand, then the complete process has to be executed. A potential issue in this case is that, because no resources have been reserved, it may be the case that when the commit request arrives, the resources are no longer

available. Thus, from the point of view of the InP, there is a trade-off between increasing the complexity of the query phase and improving the reliability of the whole process.

3.2 Resource Allocation, Monitoring and Controlling

In this section we present and describe the RAMC framework, which was implemented in C++ and provides the build up, management and control of virtual networks using a small-scale network virtualisation testbed. In the envisioned network virtualisation environment, the infrastructure provider is responsible for managing and controlling physical network resources. Virtual networks are established as a result of a VN Provider explicit request, or through the network management console. Whenever a request to establish or modify a virtual network is received, the network resource controller, based on specific resource utilisation policies, should decide whether or not the request can be accepted and, if it can, how needs to map the virtual resources into physical resources.

3.2.1 Architecture

The RAMC is composed of 2 software modules: the Agent module, and the Manager module; their hierarchical decomposition can be analysed on Figure 3.3. The Agent module is designed to run on every substrate node, in order to act upon it and periodically gather data from it. The Agents receive and send requests to the Manager, which is a centralized entity in charge of aggregating all Agents' knowledge and sending them commands. Additionally, the Manager is built-in with a Command Line Interface (CLI), which is the user's front-end, and provides him with ability to use virtual network creation, discovery, and monitoring functionalities.

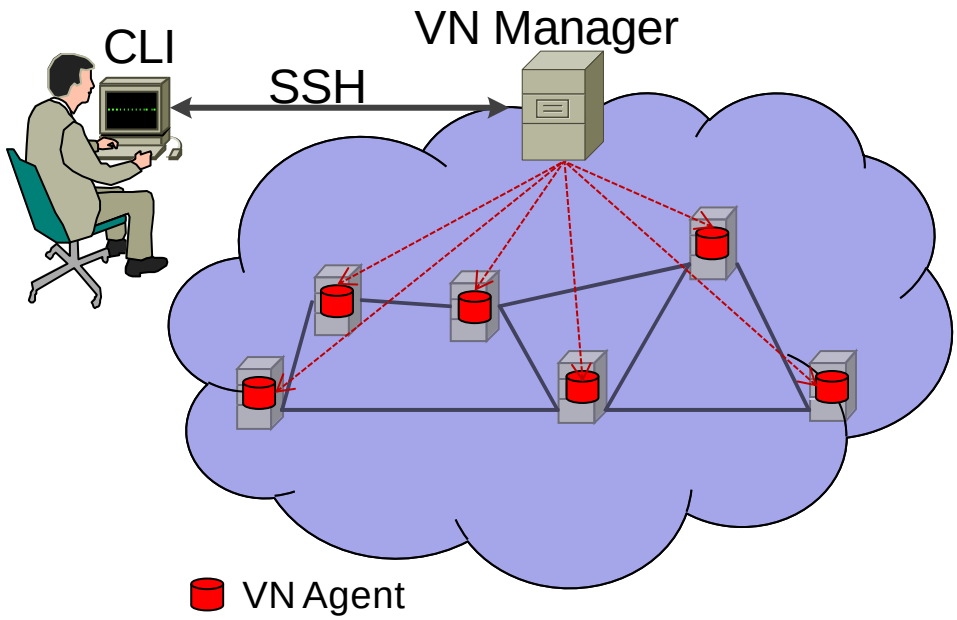


Figure 3.3: Resource Allocation, Monitoring and Controlling (RAMC) Architecture

3.2.2 Built-in Capabilities

3.2.2.1 Virtual Network Discovery

Physical and virtual network discovery provides a fast and easy way of having a global view of the virtual networks running on top of a given physical infrastructure, and it is also fundamental for the process of embedding new virtual networks, since the embedding process requires an accurate and up-to-date view of the substrate and currently running virtual networks. The experimental virtualisation platform works with a centralized discovery mechanism. The Manager requests information from the agents, e.g. virtual nodes allocated within the physical node, virtual links identifiers (i.e. VLAN tag), to build the overall topology.

3.2.2.2 Virtual Network Creation

The Manager module gives the possibility for the user to create a new virtual network based on a specification in a Extensible Markup Language (XML) file. The user may specify the resources CPU capabilities, RAM amount, location, number of interfaces and also perform network addressing configurations. The final step in creating a new virtual network is to map it in the physical infrastructure. In the next chapter 4, two distinct approaches are proposed to provide the VN embedding solution: a heuristic algorithm that combines a greedy node mapping algorithm with a Constrained Shortest Path First (CSPF) algorithm; and an ILP formulation to obtain the optimal solution.

3.2.2.3 Substrate and Virtual Network Monitoring

A dynamic resource monitoring feature is required to have an accurate view of the virtual and physical networks, and quickly react to failures or configuration problems. The implemented monitoring functions periodically update the resources' information; therefore, it is possible to identify diverse situations, such as failures and high resource usage, which may require immediate action. Every Agent periodically checks its local resources' configuration and status, and reports back to the Manager if any change occurs. Several parameters are monitored: CPU load, RAM, HDD usage, interface and link status, interface bridge attachment and configuration, number of running virtual machines and their state.

3.2.3 Testbed Description

To demonstrate the network virtualisation concept and to evaluate the RAMC framework, a small-scale testbed was deployed. The experimental testbed presented in Figure 3.4 is composed of 6 physical nodes, whose CPU and RAM characteristics are described in Table 3.2. The nodes are interconnected according to Figure 3.4, with a total of 8 Ethernet links, ranging from 100Mbps to 1Gbps. The Operating System (OS) virtualisation is based on Xen hypervisor version 3.1 [XEN14]. Both physical and virtual machines run Fedora 8 i386 (Linux 2.6.21.7) OS. Virtual routers are based on the Quagga Routing Suite [qua12].

Table 3.2: Testbed specification.

Node	**Susan**	**Lynette**	**Gabrielle**	**Bree**	**Eddie**	**Mary**
CPU	PentiumD	PentiumD	Core 2 Duo	Xeon E3110	Xeon X3220	Xeon X3330
RAM	6GB	6GB	4GB	6GB	6GB	6GB

On the top of the substrate network, 3 different VNs were created on-demand and on-the-fly. The instantiation of the virtual nodes is performed using XEN hypervisor [XEN14].

Figure 3.4: Network Virtualisation Testbed Photo.

To make this process faster and easier, it was used clone techniques, where it has been pre-configured one or more (default) virtual nodes to be replicated. On this testbed, each virtual node needs to have its own filesystem, where we can refer the creation, or to be more accurate, the cloning of the new filesystem. This is the process that takes longer time: in average it can take up to 20 seconds per virtual node depending on the substrate node characteristics. To enable the virtual links, VLANs [VLA14] were configured in each substrate link.

3.2.4 Evaluation Results

To make an evaluation of the RAMC framework, two experiments were designed: embedding and instantiation of virtual networks. The performance metrics considered are the following: time to embed a virtual network request and time to instantiate a virtual network, with the increase of the number of existing virtual networks. The virtual networks embedded and created were the same, and always a replica of the underlying physical network, and served the purpose of being a reference throughout the tests. The virtual nodes were configured with 1 CPU, 64MB of RAM, 1GB of Hard Disk Drive (HDD), and 1Mbps links. The Manager was running on a separate physical machine directly connected to the testbed (Intel Core 2 Duo P8600; 4GB of RAM; 100Mbps link). During the tests, all virtual nodes were idle and so were the physical nodes and links. The maximum amount of created virtual networks was 40, which corresponds to 40 virtual nodes in each physical node. The results presented on the following sections always assume a 95% confidence interval.

3.2.4.1 Virtual Network Embedding

The considered virtual network embedding time encompasses the elapsed time between receiving a VN request and performing the assignment of the VN. The purpose of this test is to evaluate whether the overall time to perform the VN embedding depends or not on the current physical network status, e.g. number of existing VNs, and not to evaluate the performance of the embedding algorithm. Either the algorithm description or its evaluation

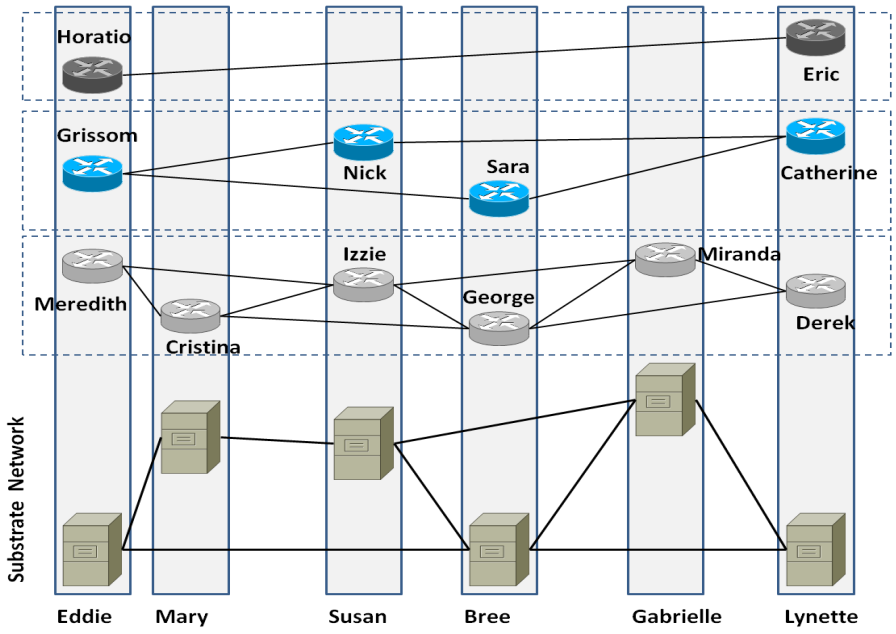


Figure 3.5: Network Virtualisation Testbed

will be provided in Chapter 4 and Appendix B, respectively.

In order to assess the mapping times, 40 virtual networks were created, one at the time. The time required for the Manager to process the received unmapped XML and return a mapped one was measured. The tests were repeated 3 times. The time required to perform the mapping is shown to increase with the number of existing virtual networks (Figure 3.6). Since the mapping procedure only depends on the virtual network to be embedded and on the physical network, it would be expected that the mapping times remained constant. However, this was not the case. In order to understand the increase in the required mapping time, one must take into consideration that, when performing the mapping, the Manager needs to refresh its internal database to reflect the current status of the physical network, therefore needs to access each existing virtual network's information. Thus, for each additional virtual network, the Manager will require more time to determine each physical resource load. This increment in needed time is revealed in the obtained results, which shows a linear scaling with the number of existing virtual networks. Regarding the absolute mapping times, they remained in the order of low tens of millisecond, which is very good and can be considered real-time.

3.2.4.2 Virtual Network Creation

In order to evaluate the time required to create a virtual network on the available testbed and its scaling with the amount of previously existing virtual networks, several creation tests were performed. The amount of previously existing virtual networks was varied between 0, i.e. without virtual networks, and 39. For each considered point, a virtual network was created and deleted 10 times, and the time required for creation was recorded. The considered creation time encompasses the time required for the Manager to split the mapped XML and send the different command messages to the Agents, as well as the time required for the Agents to report back with updated information about the created resources and links. The Manager was in charge of measuring these creation times. The obtained results,

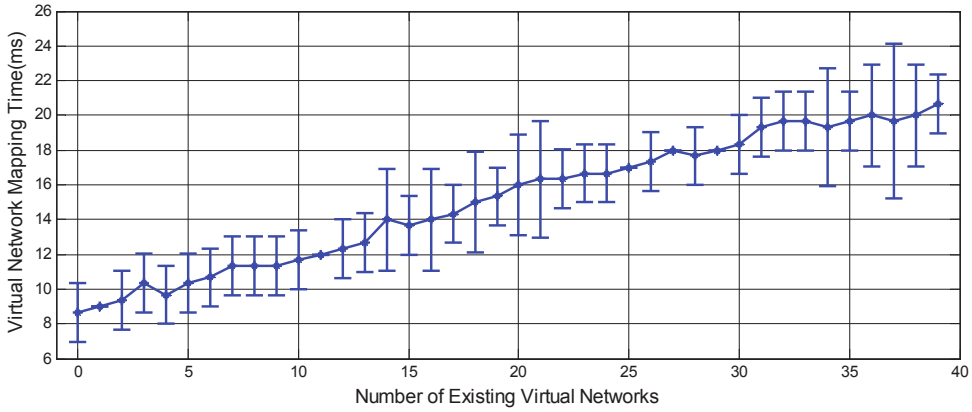


Figure 3.6: Virtual Network Mapping Time as a Function on the number of existing VNs.

shown in Figure 3.7, follow a linear trend with the increase in the amount of existing virtual networks. It is worth noting that the total creation time, encompassing both node creation and subsequent topology discovery, only depends on the slowest physical node (Table 3.2), from the ones chosen to have a virtual node embedded. The demonstrated increase in the VN creation time is due to the increase in time required to gather resource information, e.g. virtual nodes, and virtual links.

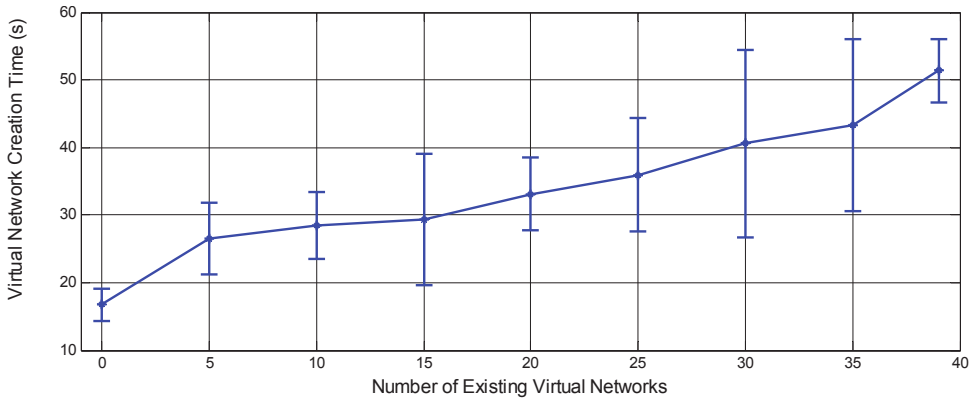


Figure 3.7: Virtual Network Creation Time as a Function of the existing VNs.

3.3 VN Migration

The VN migration process can be started from a predicted event in the sense that it can be scheduled in time, i.e. planned maintenance, or it can be triggered from a non-predicted event, i.e. VN security attack or hardware failure. In this section, we start with a description of the several factors that can lead to a VN migration process, and we finalize the section by explaining the different actions required in our approach for VN Cloning.

3.3.1 Triggers for VN Migration

The VN migration process can be the result of a planned event or of a non-planned event, and the time to perform all the VN migration operations can be considered critical or non-critical depending on the event that leads to the migration process.

3.3.1.1 Equipment/Facilities Maintenance

The migration process can be triggered due to maintenance reasons within the physical hosts or external to the physical hosts: within the physical hosts due to the replacement or upgrade of hardware modules, firmware updates or patches where the physical host needs to be rebooted or shutdown and disassembled; external to the physical hosts with the need to temporary or permanent change the location of the physical host, or Uninterruptible Power Supply (UPS) upgrade or power grid maintenance.

3.3.1.2 Network Performance

The VN migration can also be started due to networking performance reasons. If we consider the fact that VNs are dynamically provisioned and constantly being created and removed, the consumption of network resources is likely to become unbalanced due to the dynamics of the VNs arrivals and departure events. In order to optimize the physical resources and at the same time to distribute the network load equally per all physical hosts, or even to alleviate the load at some physical hosts that are reaching critical levels, a re-optimization process on the existing VNs should be performed. This usually involves the re-mapping of existing VNs; therefore, the VN migration process would take place in order to move the VRs to physical hosts that are less loaded.

3.3.1.3 End-user Requirement/Service Level Agreement

It may also be required to move a VR from one PoP to another PoP, which is much closer to the end-user. It may also be required to reduce the overall round trip delay of a VN throughout the migration of parts, i.e. VRs or Virtual Link (VL), of the VN or even the entire VN to physical places and hosts which provide smaller round trip delays.

3.3.1.4 Energy Saving

The migration of VNs can also be triggered due to power consumption reasons, either to move VNs closer to greener power grids, or to move VNs to physical hosts that consume less energy, or even to concentrate VNs on the minimum number of physical hosts as possible, without interfering on the service levels reliability, in order to reduce the number of active physical hosts. This is not only due to diurnal traffic patterns, where the traffic is much smaller in the night and therefore the number of pieces of network equipment required to operate in the night is smaller, but it is also due to the fact that a physical router in idle mode consumes 90% of the consumption in full mode [CSB+08].

3.3.1.5 Security Protection

The migration of a VN can be even required as a way to provide higher security protection, or as a way to move VNs from physical hosts which are currently under attack "moving target defense" [AS11]. A VN can evade detection or attack by changing its location in the physical network.

3.3.1.6 Fault Management

The VN migration process can also be triggered due to fault management decisions. If we consider a reactive fault management, the VN migration is triggered when the hardware fails. If we consider a proactive fault management, the VN migration process is started when the hardware that is prone to fail is flagged.

3.3.1.7 Service Deployment

In order to deploy new services on production networks, VNs running on trial scenarios can be phased migrated to physical hosts located within these networks, or only just parts of the VNs need to be moved away.

3.3.1.8 End-User Mobility

The VN migration process can be even required due to mobility reasons. An end-user can be moving away from his home network to a foreign network and, in order to avoid the connection break during the handover process, e.g. change the IP address and or the access technology of the end-user, the VR migration process can be used as a complementary mechanism (or even in some situations in the absence of any mobility process, as an alternative) to mobile IP implementations [Per02] and IEEE 802.21 - Media Independent Handover Services [GWC+09]. In this way, the IP addresses of the end-users are preserved and the TCP sessions are maintained during the transition of one network to another.

In the Table 3.3 we summarize the types of triggers considered, and we also express them in terms of event periodicity, i.e. when they are expected to occur, event duration, i.e. the expected time that the event will consume, and priority, i.e. this can be used as a classification of the event in terms of urgency to be taken.

Table 3.3: VN migration types of trigger events, event duration and event priority.

Event Type	Event Periodicity	Event Duration	Event Priority
Physical Maintenance	Monthly/Annually	Hours/Days	Medium/Low
Network Performance	Daily/Weekly	N/A	High/Medium
Energy Saving	Daily	Hours	Medium/Low
Security Protection	N/A	Minutes/Hours	Urgent/High
Fault Management	N/A	Minutes/Hours	Urgent/High
Service Deployment	Weekly/Monthly	Hours/Days	Medium/Low
End-user Mobility	Hourly/Daily	Minutes/Hours	Urgent/High

As a result of the previously described triggers, we can foresee three different cases where VN migration is required:

i In the first situation the host is considered to be removed from the physical network topology. Therefore, all VNs using that specific host need to be reallocated to other physical hosts.

ii In the second situation the host still exists, but one or more physical links had been removed from the physical network topology, e.g. hardware failure. Likely some VNs need to be reallocated to other physical hosts. Although it does not strictly mean that we need also to migrate VRs, it may be even possible that only VLs migration can fix the problem.

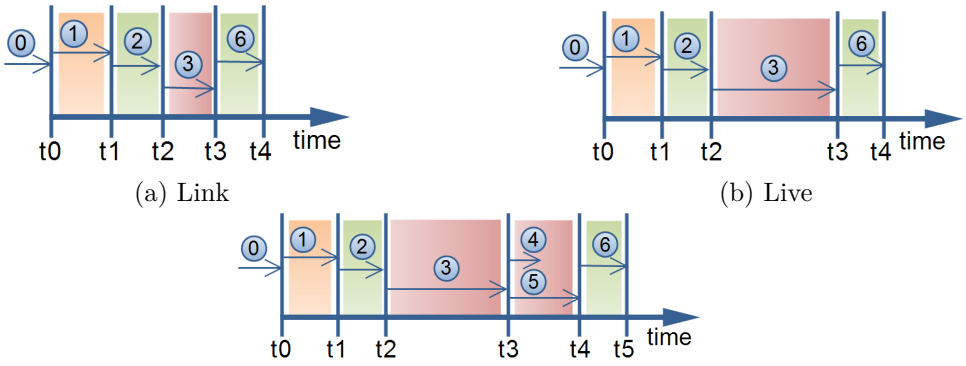
iii In the third scenario the host exists on the physical network, but it is only capable of doing switching operations [nex12]. This implies only the migration of the VRs; the migration of the VLs is not required.

3.3.2 VN Clone Migration Procedure

As a result of the three cases, one of two things can take place in the migration process: migration of virtual links, or the migration of virtual links and virtual routers. Each migration process encompasses a set of different operations. In this sub-section, we describe the

VN migration process with VL migration, and with VL plus VR migration. Two different approaches are evaluated for the VR migration: the VR live migration and the VR clone migration (our proposed approach).

Figure 3.8 represents the VN migration timeline; it is intended to reflect the amount of time that each action takes when comparing with others, to reveal its level of criticality to the overall VN migration process, and also to identify the common actions, which are represented in the timeline by numbers, taken in the different VN migration processes (i.e. link, live and clone migration).



(a) Link

(b) Live

(c) Clone

Figure 3.8: VN Migration Timeline

In the Figure 3.8, we can observe that the VN link migration takes less time to be fully performed when compared to the live or the clone migration. This is due to the migration of the VR itself, since it requires moving the VR content (e.g. memory RAM) from one physical host to another, while in the case of the virtual link migration no data needs to be transferred across physical nodes.

The different operations which compose the clone migration process are described here:

0. *VN Migration Trigger* - The VN migration process starts after receiving a VN migration trigger (i.e. which can be one of the triggers presented on Table 3.3), and it corresponds to step 0.

1. *Compute VN Mapping* - The time that this action takes to be performed can be considered as critical or non-critical to the overall migration time, and it will strictly depend on the event which causes the VN migration (see Table 3.3), and that involves the computation of a new VN mapping (i.e. VN re-embedding).

   The VN mapping takes as inputs the VN nodes and links that need to be migrated, and the output will be the new location of those nodes and links[1]. The VN mapping can be performed either using a heuristic approach [NMCS11b] which performs the VN embedding in a faster and efficient way, or using mixed integer linear programming approach [MCS+12], which takes relatively more time to perform the VN embedding, but achieves the optimal solution.

   The "available" time to perform the VN (re-)embedding will be dictated by the event type that leads to the VN migration process, i.e. a critical event (e.g. fault management)

[1]It may also be possible that it is less costly (i.e. shorter virtual links=>less provisioned bandwidth) to move not only the nodes and links that were initially considered to be migrated, but also other nodes and links.

requires that the VN embedding should be performed as fast as possible, and a non-critical event (e.g. network performance) requires that the VN (re-)embedding should be as good as possible (i.e. optimal bond).

2. *VLs Setup* - This operation comprises the setup of new virtual links, e.g. setup of VLAN interfaces and virtual bridges, and it is considered as non-critical, since it is performed beforehand and in the time-frame of the order of milliseconds (or even nanoseconds with optical switches).

3. *Clone/Move/Restore* - This is the most critical and time consuming task to the overall VN migration process, and it can be divided into three sub-operations:

   a) *Clone VR* - The cloning of the VR involves saving the current state, i.e. memory RAM, to the physical host hard-disk or even to a RAM-disk[2].
   This can be considered as critical or non-critical action, if the VR is put into suspend while being cloned in the first case, or if the VR is still running while being cloned in the latter. The time required to perform this process is given by formula (3.1), where VR_{memory} is the memory RAM size of the VR.

   b) *Move VR Clone* - This part encompasses the transfer of the VR clone to the new physical host and it is, in principle, the most time consuming task of the three sub-operations, and of the overall VN clone migration process. Despite contributing to the overall VN migration execution time, it does not affect the VN downtime, since the VN is still running while the VR clone is being relocated. The time required to perform this task is given by formula (3.2), where $BW_{reserved}$ is the bandwidth which is effectively reserved by the operator to this kind of operations, and BW_{free} is the bandwidth that is not provisioned (or available) on the physical path between the physical hosts and at that time period. In theory, the reserved bandwidth can be equal to zero, although operators do tend to reserve bandwidth for this kind of operations. Note that, if this phase takes too much time to be performed, the VR clone can easily become outdated.

   c) *Restore VR Clone* - The restore of the VR clone is performed after it is allocated on the new host, and it does not influence the VN downtime, since the VR clone is not yet connected to the VN. This operation is performed in the time-frame of high hundreds of milliseconds.

4. *Add Virtual Bridge Int. & Destroy Original VR* - This is a critical action and encompasses adding a new virtual interface (e.g. VLAN) to the virtual bridge that will be used to connect the VR clone to the VN. It also includes the shutdown of the original VR, which is performed in order to avoid duplicated VRs operating on the virtual network. The VN transition, from the old VLs and VRs to the new ones, is signalled by the execution of these two operations. The VR shutdown (destroy) is executed in the time-frame of low hundreds of milliseconds.

5. *Remove VLs* - The removal of old virtual links is a non-critical action and is performed in the same time-frame as the setup of virtual links (i.e. milliseconds). This phase does not count to the execution time, though it is part of the VN migration process.

The VN downtime due to the migration process can be obtained using formula (3.3), where the $VN_{downtime}$ is mostly given by the VR cloning operation (i.e. t_{clone}). With formula (3.4), we can obtain the VN migration execution time, where the *action* 3*b* contributes the most.

[2]A RAM drive (RAM disk) is a block of RAM that is treated as if the memory were a disk drive. In our implementation we have used a ram-disk due to performance reasons

$$t_{clone} = \frac{VR_{memory\_size}}{RamDisk_{write\_speed}} \tag{3.1}$$

$$t_{move} = \frac{VR_{memory\_size}}{BW_{reserved} + BW_{free}} \tag{3.2}$$

$$t_{downtime} \cong t_{clone} \tag{3.3}$$

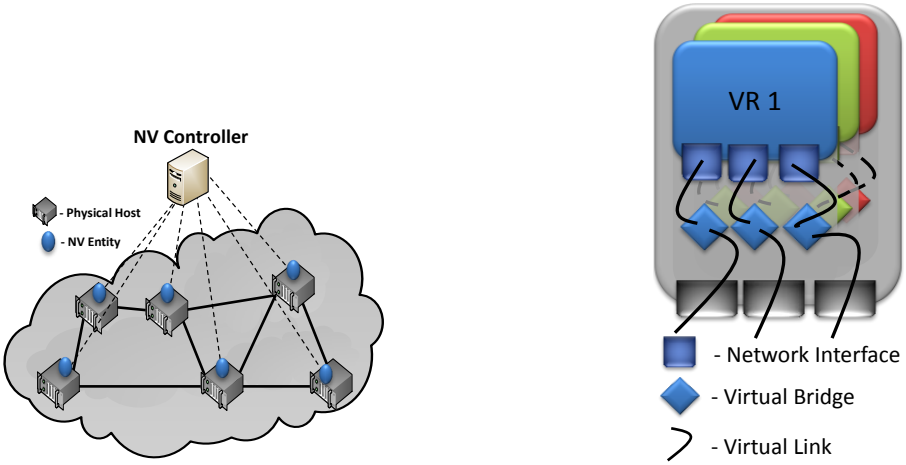
$$t_{execution} = t_4 - t_0 \tag{3.4}$$

3.3.3 VN Clone Migration Architecture

In this section we describe the overall architecture, which was considered to support the VN clone migration, and also the building blocks of the VR. For further details on the clone migration method, please see the appendix section F.

3.3.3.1 VN Clone Migration Architecture

Figure 3.9a presents the Network Virtualisation (NV) architecture which was proposed to support VR migration, comprising the Network Virtualisation Controller (NVC) and the Network Virtualisation Entity (NVE). The NVC is responsible for coordinating the VN migration process, and is also responsible for performing the VN mapping, choosing the new location of the VLs and VRs. Each NVE is responsible for enforcing the NVC commands, e.g. $CloneVR$. The list of possible commands performed by the NVC is shown in Table 3.4.



(a) Network Virtualisation Architecture. (b) Virtual Router Building Blocks.

Figure 3.9: VN Clone Migration Architecture.

All communications between the NVC and the NVEs are secured and performed using the Secure Shell (Unix program) (SSH) protocol, and the secure copy protocol is also used to move the VR clone.

3.3.3.2 Virtual Router Implementation

The VR architecture considered is shown in the Figure 3.9b. It is composed by the VR instance, (virtual) network interfaces and virtual bridges. The virtual bridges are used either to interconnect network interfaces within the physical host, or to interconnect virtual network interfaces of VRs running inside of the host. The Linux Bridge Utils tool [bri12] is used in

Table 3.4: NV Controller - List of Commands.

Command Name	Command Description
AddVInt *IntId*	Add virtual Interface on the Physical Host
DelVInt *IntId*	Remove a Virtual Interface on the Physical Host
AddVBridge *BrId*	Add a Virtual Bridge on the Physical Host
DelVBridge *BrId*	Remove Virtual Bridge on the Physical Host
AddVBridgeInt *BrId IntId*	Add a (virtual) Interface to the Virtual Bridge
DelVBridgeInt *BrId IntId*	Remove a (virtual) Interface on the Virtual Bridge
CloneVR *VRId*	Clone Virtual Router
MoveVR *VRId Dst*	Move the VR Clone to the new physical host
RestoreVR *VRId*	Restore Virtual Router on a physical host
DestroyVR *VRId*	Destroy Virtual Router on a physical host

order to setup virtual bridges, and the Linux VLAN implementation [VLA14] is used to setup the virtual links.

3.3.4 Evaluation Results

To analyze the different VN migration methods, the traffic generator D-ITG [AGE+04] is used to evaluate the impact on the traffic carried by the virtual network due to the VN migration process. The total experiment time is set to 100 seconds, where the traffic is continuously generated before, during and after the VR migration event. It is considered UDP traffic with a packet size of 1000 Bytes, and either with a bitrate of 10Mbps (i.e. 1250 packets/s) or of 20Mbps (i.e. 2500 packets/s) to evaluate the influence of the traffic bitrate on the VN migration process. The memory RAM of the VRs, i.e. the size of the routing tables, is set from 64MB to 256MB with intervals of 32MB. During each experiment, it is measured the number of packets sent and the number of packets lost. The execution time of each Linux command is also measured: bridge setup, VLAN setup, bridge interface setup, VR cloning, VR move, VR clone restore, VR destroy, VR live migration (only for the live migration process). For each experiment, 10 trials are performed and confidence intervals of 95% are used for every plot.

3.3.4.1 VN Downtime

Figure 3.10 shows the VN downtime as a function of the VR memory size. According to the figure, we can observe that the VN downtime exhibits two distinct behaviours: either it does not depend on the VR memory or it does strongly depend on it. In the live migration and the clone migration without VR suspend, the VN downtime does not strongly depend on the VR memory size and it is almost constant for all the memory sizes evaluated.

The live migration has a VN downtime (or percentage of dropped packets) of 400 milliseconds (0.4%), and the clone migration has no downtime when using UDP traffic at 10Mbps. This behaviour is expected for both methods, where the downtime experienced on the VN

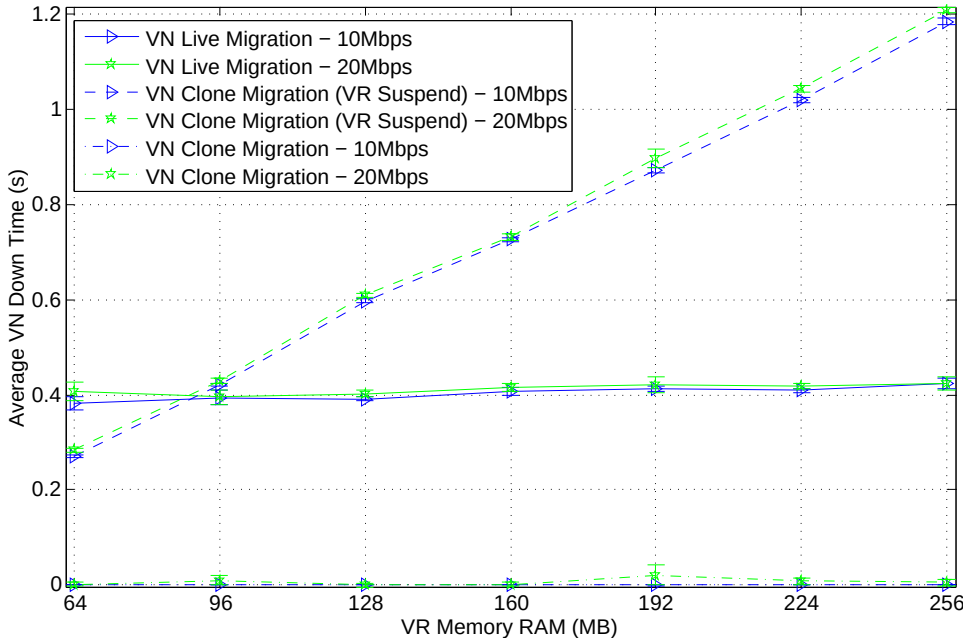


Figure 3.10: Virtual Network downtime (or Percentage of Dropped Packets) as a Function of the VR Memory RAM.

live migration is mainly due to the XEN live migration procedure [CFH+05], which is an iterative process based on memory copy of dirty pages. In the VN clone approach, the VR memory RAM is copied at once, while the VR is still running, and at memory RAM write speeds.

In the case of the VN clone migration with VR suspend, the VN downtime does vary with the VR memory size and increases linearly with it. This is in fact due to the cloning phase of the VR, where the VR is put into suspend mode while its memory RAM is being copied. The VN clone migration without VR suspend outperforms the VN live migration approach and achieves no VN downtimes.

The traffic carried out by the VN does not significantly affect the $VN_{downtime}$, although a slightly higher percentage of dropped packets is registered for the UDP traffic with higher bitrate (i.e. 20Mbps).

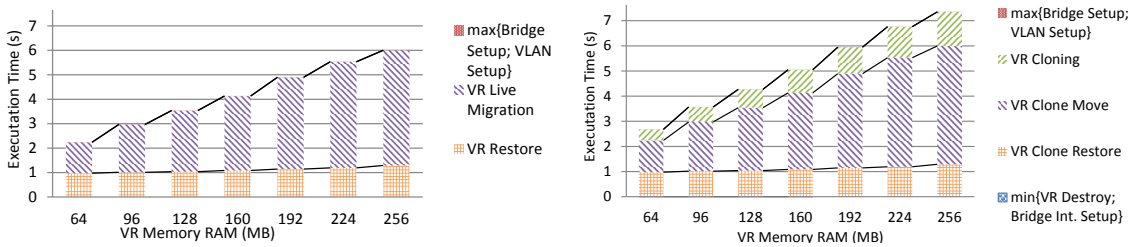
3.3.4.2 VN Migration Execution Time

The VN migration execution time is illustrated in Figure 3.11. On the left side (Figure 3.11a), it is represented the execution time of the live migration process, and on the right side (Figure 3.11b), it is represented the execution time of the clone migration process. The VN execution time grows linearly for both migration methods with the VR memory size. The live migration process is the one that takes less time to be fully performed, since it is performed internally by the XEN hypervisor.

In the clone migration process, the most time consuming operation is the VR Clone move (i.e. the relocation of the VR clone to the new physical host). This operation is influenced both by the VR memory size and by the available bandwidth at the physical path, between the physical host source and the physical host destination (*Mary-Susan-Bree*), which in our experiment is of 1Gbps (see formula 3.2). The second most time consuming operation is the VR cloning operation, which is also dependent on the VR memory size.

Both the bridging of virtual interfaces and the VLAN setup are the less time consuming operations, and take up to 4 and 8 milliseconds, respectively, to be fully performed. The total execution time of the clone migration process, with a VR of 64 MB memory RAM, is 2.75 seconds, while the total execution time of the live migration process is 2.36 seconds.

Notice that the total execution time of the clone migration process can be reduced, if it is also performed internally by the hypervisor. Moreover, this time can be further reduced if the VR clone relocation takes place in parallel with the VR cloning phase.



(a) Live (b) Clone

Figure 3.11: VN Migration Execution Time as Function of the VR Memory Size

3.4 Summary

This chapter provided a new architecture for Network Virtualisation: it promotes the deployment of new protocols and enables the emergence of new players in the telecommunications market. As can be seen from the testbed description, the base testbed is running with intervention of the RAMC framework. This chapter presented the VN clone migration as an alternative to the live migration approach. The VN clone migration performs cloning of the VR and transfers the VR clone to the new physical host. This approach requires no restrictions on the virtual network itself or on the networking protocols running inside of the virtual network. The results show that the proposed approach achieves no VN downtime, and it takes just a few seconds to be fully performed.

Chapter 4 will address the resource mapping and resource re-optimization modules of the RAMC framework.

Chapter 4 - Network Virtualisation: VN Embedding Problem

"If I were again beginning my studies, I would follow the advice of Plato and start with mathematics."

—Galileo Galilei

It is of the Infrastructure Provider (InP) best interest to have the lowest provisioning cost solution per VN, while increasing the overall revenue income. This chapter is aligned with this requirement and it proposes an Integer Linear Programming (ILP) formulation, the Virtual Network Embedding Node-Link Formulation (VNE-NLF), that has been proposed in [MCS+12] to solve the VN embedding problem as an optimisation problem, and therefore, to provide the optimal solution per VN embedding. This approach has the following requirements: i) to minimize the resource allocation cost; ii) and to maximize the overall revenue. An enhancement to an existing heuristic [NMCS11b] has also been proposed in [MCS+12]. Extensions to this formulation have been proposed in [MCS+13b] to address the re-optimization problem (i.e. re-embedding of previously mapped VNs), and additionally a comparison with existing state of the art heuristics have been performed in [MSK+13]. Moreover, in [MSK+14] a formulation to address the energy consumption problem (i.e. minimization of the energy consumption per VN embedding) has been proposed. Additionally, a formulation to address the optimal VN migration has been proposed in [MSC14]. This chapter is an overview of the previous articles.

This chapter starts with a description of the virtual network embedding problem. Section 4.2 proposes an enhancement to a heuristic algorithm [NMCS11b]. Section 4.3 describes the VNE-NLF and the applied constraints, while Section 4.4.1 discusses the different proposed cost functions for resource allocation. Moreover, Sections 4.5, 4.6, and 4.7 present the proposed extensions for re-optimization, energy-aware, and VN migration, respectively. Section 4.8 depicts the major results and analyses the performance of the VNE-NLF with different cost functions, and compares it with six existing heuristics. An evaluation on the proposed extensions (i.e. re-optimization, energy-aware, and VN migration) is also presented within this section. Finally, section 4.9 summarizes the chapter.

For further details on the formulation proposed and also on the different extensions please see the appendix sections B, C, D, E, and F, respectively.

4.1 Problem Description

In this section, we introduce the virtual network embedding problem. In addition, the VN embedding notations used throughout the chapter are presented, and the virtual network embedding system is explained. Finally, the mapping goals are introduced to support the mathematical formulation.

4.1.1 Network Description

We use superscript to distinguish the physical network from the virtual network, where p and v correspond to physical and virtual, respectively.

4.1.1.1 Physical network

A physical network can be described as a weighted undirected graph $G^p = \{N^p, L^p, C^p, B^p, D^p, \text{Dis}^p\}$ composed by a set of physical nodes, N^p, and a set of physical links, L^p. Each physical node i is characterized by its processing capacity, C_i^p, commonly referred to as the CPU, by its physical location, which can be defined by x and y coordinates. Power state - active if the node is power-up, inactive if the node is power-off; and role - hosting node if it accommodates virtual nodes, forwarding node if its physical links accommodate virtual links but the node itself does not accommodate virtual nodes.

The physical distance between nodes, Dis^p, can be obtained using equation (4.1). With respect to the physical links, we consider that each link ij has a given bandwidth, B_{ij}^p, and a given link delay, D_{ij}^p, and we also assume that each link is an undirected link. The bottom-right side of Figure 4.1 illustrates a physical network topology example composed of 6 physical nodes and 8 physical links; the corresponding capacities of the nodes and the links are presented on top of the elements. The physical nodes power state is represented using a colour scheme: gray for inactive and blue for active.

$$\text{Dis}_{ij}^p = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \tag{4.1}$$

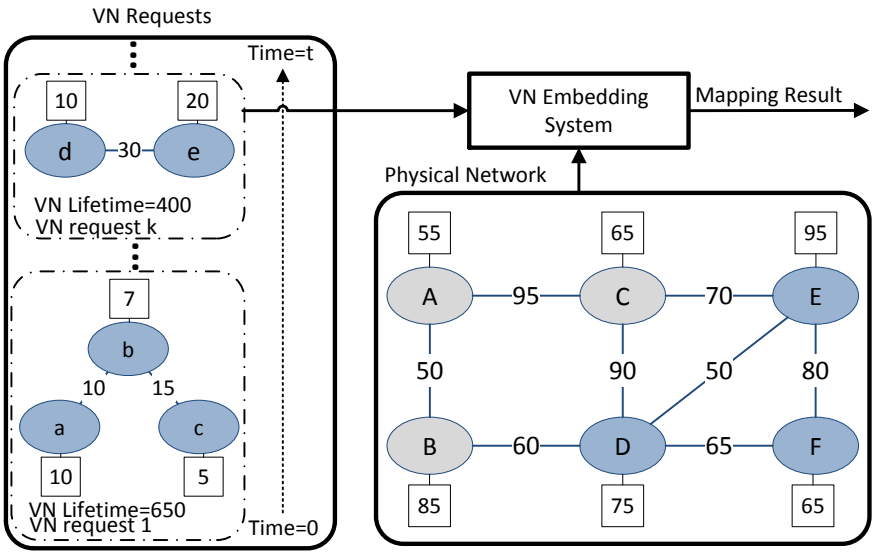


Figure 4.1: VN Embedding System - Topology Example

4.1.1.2 Virtual Network Request

VN request can be described as a weighted undirected graph $G^v = \{N^v, L^v, C^v, B^v, D^v, \text{Dis}^v\}$ composed by a set of virtual nodes, N^v, and a set virtual links, L^v. Each virtual node m is characterized by the amount of required CPU, C^v_m, and the virtual links mn are logical connections between virtual nodes and characterized by the amount of dedicated bandwidth, B^v_{mn}, and by the maximum link delay permitted, D^v_{mn}. We also assume that each virtual link is an undirected link. The maximum virtual distance between virtual nodes, Dis^v, can be used to limit the physical distance between virtual nodes. The left part of Figure 4.1 represents the example of two virtual network requests, VN request 1 on the bottom-left and VN request k on the top-left. Each VN request has a given lifetime that is, in principle, independent from each other, and each lifetime could have different time scales, since it is strongly dependent on the purpose of the virtual network request itself. If we consider a VN request for a live rock concert, the time scale will be hours, but if we consider a VN for a culinary workshop of one week, the time scale will be days.

4.1.1.3 VN Assignment Notations

First, we start with the convention used for the index notation: N^p represent the set of nodes that belong to the physical network; L^p represent the set of links that belong to the physical network; and L^p_i represents a subset of links ij that are directly connected to the node i. The same type of notation is used to represents the VN using the letters m and n in the virtual network. The notations used throughout this chapter for the VN assignment problem are presented in Table 4.1. The Table is divided into three parts: the static parameters of the physical network, the dynamic parameters of the physical network, and the virtual network requests with the demanded capacities.

Table 4.1: VN Assignment Problem Notation.

G^p	Physical Network
N^p	Set of Physical Nodes
$i,\ j$	Physical Nodes
ij	Physical Link
L^p	Set of Physical Links
L^p_i	Set of Physical Links directly connected to Physical Node i
$C^p_i(t_0)$	Available CPU at time t_0 on Physical Node i
Dis^p_{ij}	Distance Between Physical Nodes ij
$B^p_{ij}(t_0)$	Available Bandwidth at time t_0 on Physical Link ij
$C^p_i(t)$	Available CPU at time t on Physical Node i
$P_i(t)$	Power Consumption at time t on Physical Node i
$B^p_{ij}(t)$	Available Bandwidth at time t on Physical Link ij
$G^v(k)$	Virtual Network Request k
$N^v(k)$	Set of Virtual Nodes of VN Request k
$L^v(k)$	Set of Virtual Links of VN Request k
$L^v_m(k)$	Set of Virtual Links directly connected to Virtual node m of VN Request k
m, n	Virtual Nodes
mn	Virtual Link
$C^v_m(k)$	CPU of Virtual Node m of VN Request k
$\text{Dis}^v_{mn}(k)$	Maximum Distance Between Virtual Nodes mn of VN Request k
$B^v_{mn}(k)$	Bandwidth of Virtual Link mn of VN Request k
$D^v_{mn}(k)$	Delay of Virtual Link mn of VN Request k

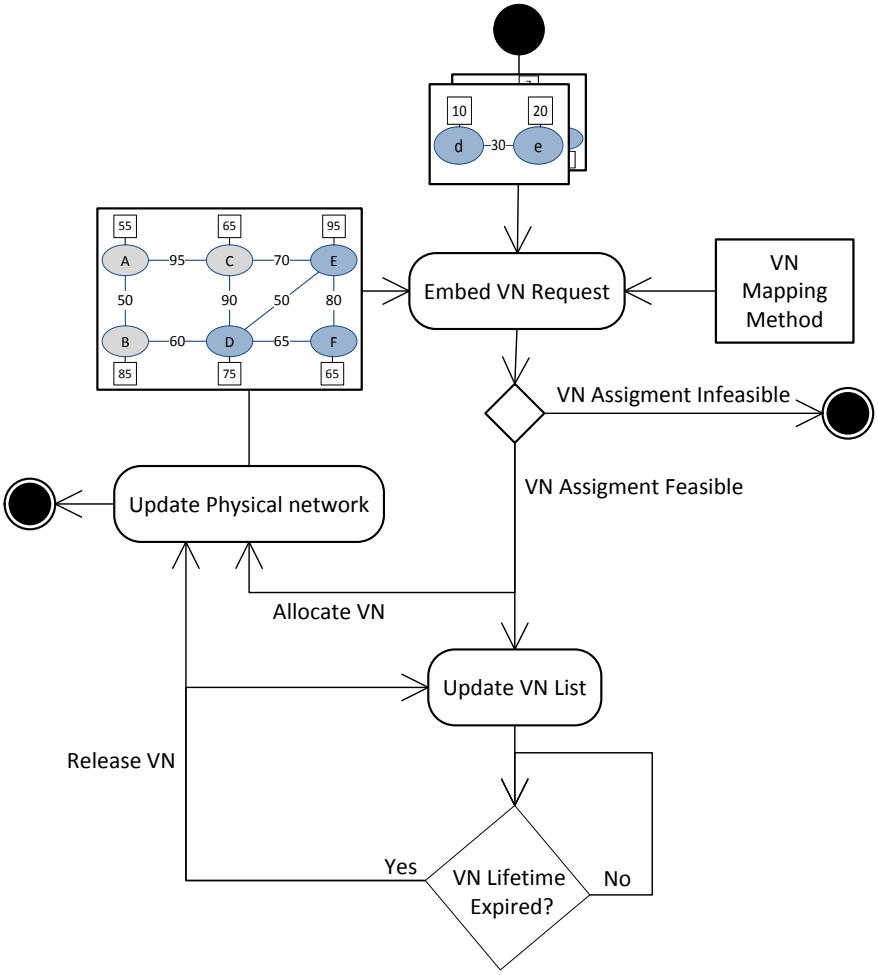


Figure 4.2: VN Request Life Cycle - Activity Diagram

4.1.2 Unfilled Physical Network Resources

The remaining capacity of each physical node at a specific time t is given by the difference between the total processing capacity and the capacity consumed by all virtual nodes allocated on that physical node, and is presented in equation (4.2), where U represents the set of all virtual nodes allocated on that precise physical node and at time t. We assume that the physical network is empty at time t_0.

$$\forall i \in N^p : C_i^p(t) = C_i^p(t_0) - \sum_{\forall u \in U} C_u^v(t) \tag{4.2}$$

In parallel, the available bandwidth of each physical link at a specific time t is given by the difference between the total bandwidth and the bandwidth consumed by all virtual link segments allocated on that physical link, and is presented in equation (4.3), where w represents the set of all virtual link segments allocated on that specific physical link and at time t.

A virtual link can be composed by one or more physical links, the physical path. We consider that each virtual link has a single physical path, and we do not consider link aggregation (i.e. virtual link composed by different physical paths).

One physical link can accommodate one or more virtual link segments belonging to dif-

ferent virtual links.

$$\forall ij \in L_i^p : B_{ij}^p(t) = B_{ij}^p(t_0) - \sum_{\forall w \in W} B_w^v(t) \tag{4.3}$$

4.1.3 VN Request Embedding Process

The VN request embedding process can be divided into two components: the component that ensures the mapping of the virtual nodes, and the one that handles the mapping of the virtual links.

4.1.3.1 Virtual Node Mapping

Each virtual node needs to be mapped onto one physical node. This relation is given by the mapping function $\mathcal{M}[m \in N^v(k)] = i$, where virtual node m is mapped onto exactly one physical node i. Each physical node candidate needs to have, at least, the same amount of available CPU as required by the virtual node, which is represented in equation (4.4).

$$\forall i, \forall \mathcal{M}[m \in N^v(k)] = i : C_m^v(k) \leq C_i^p(t) \tag{4.4}$$

4.1.3.2 Virtual Link Mapping

Each virtual link can be mapped onto one or more physical links (i.e. physical path). This relation is given by the mapping function $\mathcal{M}[L_{mn}^v]$, where the virtual link mn is mapped onto one physical path. Each physical link candidate belonging to the physical path needs to have, at least, the same amount of bandwidth available as required by the virtual link which is presented in equation (4.5).

$$\forall ij \subseteq \mathcal{M}[mn \in L^v(k)] : B_{mn}^v(k) \leq B_{ij}^p(t) \tag{4.5}$$

4.1.4 VN Request Life Cycle

The embedding process begins upon a new VN arrival request, which is depicted in Figure 4.2. A VN mapping method is used to embed the VN; it takes as inputs the current status of the physical network (e.g. available CPU capacity, existing bandwidth, and physical node power state: active or inactive) and the VN request itself. If the result of the mapping process is a viable solution, the mapping is considered to be feasible; if not, it is considered to be unfeasible and the VN embedding process stops.

4.1.5 Mapping Metrics

In order to assess the performance of an embedding method and at the same time to evaluate the energy impact, different metrics were defined.

4.1.5.1 VN Request Acceptance Ratio

The VN request acceptance ratio, $\mathcal{A}(G^v)$, is given by equation (4.6) and defines the overall performance of an embedding method: the sum of all VN requests accepted, k', over the sum of all VN requests, k.

$$\mathcal{A}(G^v) = \frac{k'}{k} \tag{4.6}$$

4.1.5.2 Revenue

The revenue, $\mathcal{R}(G^v)$, is given by equation (4.7) and represents the sum of all requested CPU capacity plus the sum of the total requested bandwidth per VN request accepted. The weight parameter α can be used to express the current value of the different types of resources.

$$\mathcal{R}(G^v) = \alpha \sum_{m} C_m^v + (1-\alpha) \sum_{mn} B_{mn}^v \tag{4.7}$$

4.1.5.3 Provisioning Cost

The provisioning cost, $\mathcal{C}(G^v)$, is given by equation (4.8) and represents the sum of all allocated CPU capacity plus the sum of the total consumed bandwidth per VN request accepted. The weight parameter β can be used to express the current cost of the different types of resources.

$$\mathcal{C}(G^v) = \beta \sum_{i} C_i^p + (1-\beta) \sum_{ij} B_{ij}^p \tag{4.8}$$

4.1.5.4 Embedding Factor

The embedding factor, $\mathcal{E}(G^v)$, is given by equation (4.9) and represents the ratio between the amount of virtual resources that were requested and the amount of physical resources that were effectively provisioned per VN request accepted, i.e. the efficiency on embedding, where α can be used to weight the revenue and the cost of each type of resource, respectively.

$$\mathcal{E}(G^v) = \frac{\alpha \sum_m C_m^v + (1-\alpha) \sum_{mn} B_{mn}^v}{\beta \sum_i C_i^p + (1-\beta) \sum_{ij} B_{ij}^p} \tag{4.9}$$

4.1.5.5 Physical Network Resilience Factor

The physical network resilience factor, $R(t)$, is given by equation (4.6) and defines the overall resilience of the physical network to migration events, i.e. the ratio of successfully re-embedded sets of VNs, k', over the sum of sets of VNs, k.

$$R(t) = \frac{k'}{k} \tag{4.10}$$

4.1.5.6 Physical Node Power State

The physical node power state is given by equation (4.11). The value of $u_i(t)$ is set to 1 if physical node i hosts one or more virtual links. With this equation, we can take into account the situation where a physical node is being used as a forwarding node only.

$$u_i(t) = \begin{cases} 1, \text{ if physical link } ij \in L_i^p \text{ hosts a virtual link} \\ 0 \text{ otherwise} \end{cases} \tag{4.11}$$

4.1.5.7 Power Consumption

The power consumption, P_i, of physical node i is given by equation (4.12), and it is obtained by summing the power required to maintain the node active, which is denoted in the equation as the baseline power, P_b, with the load power, P_l, currently allocated. The baseline power is given by equation (4.13), and the load power is obtained with equation

(4.14). Both equations are multiplied by the parameter θ, that is used to weight the cost of each power consumption source given in equation (4.15).

In equation (4.12), we assume that the baseline power already incorporates the power required to operate the data plane. This assumption is relevant for the case where a physical node is powered up to perform only the role of forwarding, i.e. its physical links host one or more virtual links, but no virtual nodes are being hosted in it. Therefore, the power consumption of this specific physical node will be equal to P_b only. According to [LGL+11], it is plausible to assume that the power required to power up all physical links of a given physical node is much smaller than the power required to power up the physical node itself [BH09]. Therefore, we are not considering the power required to power up each physical link individually in the equation.

$$P_i(t) = P_{bi} + P_{li} \tag{4.12}$$

$$P_{bi}(t) = \theta_b \times u_i(t) \tag{4.13}$$

$$P_{li}(t) = \theta_l \times [C_i^p(t_0) - C_i^p(t)] \tag{4.14}$$

$$\theta_b + \theta_l = 1 \tag{4.15}$$

4.1.5.8 Physical Network Energy Consumption

The energy consumption of the physical network, $E^p(t)$, is given by equation (4.16) and it represents the amount of energy that will be required to operate the physical network. This is obtained by summing the amount of energy required to power up (i.e. P_b) the physical nodes and the amount of energy required to host (i.e. P_l) the virtual nodes.

$$E^p(t) = \theta_b \sum_{i \in N^p} u_i(t) + \theta_l \sum_{i \in N^p} C_i^p(t_0) - C_i^p(t) \tag{4.16}$$

4.1.5.9 VN Energy Consumption

The energy consumption of a VN, $E^v(t_k)$, is given by equation (4.17) and it represents the amount of energy that will be required to host a new VN request. This is obtained by summing the amount of energy required to power up (i.e. P_b) new physical nodes at time t_k, and the amount of energy required to host new virtual nodes (i.e. P_l), where ϵ is an arbitrarily small value.

$$E^v(t_k) = \theta_b \sum_{i \in N^p} (u_i(t_k) - u_i(t_k - \epsilon)) + \theta_l \sum_{m \in N^v(t_k)} C_m^v(t_k) \tag{4.17}$$

4.2 Heuristic Algorithm

In this section we propose a heuristic for the VNE problem, based on the one from [NMCS11a].

4.2.1 Baseline Heuristic

A pseudo-code description of the baseline mapping algorithm proposed in [NMCS11a] is shown in algorithm 1.

Let us define $k_j = 0...(L_{V_j} - 1)$ and $i = 0...(L_S - 1)$, where k_j is the link number of a given virtual link belonging to the j^{th} VN, L_{V_j} is the number of virtual links in the same VN, i is the link number of a given physical link, and L_S is the number of links of the Substrate Network. The virtual link stress (S_{LV_j}) of the link k_j belonging to the j^{th} VN is equal to its allocated bandwidth: $S_{LV_j}(k_j) = BW(k_j)$.

After all virtual links stresses are determined, the physical link stresses are calculated: $S_{LS}(i)$ is the link stress of the i^{th} physical link and is defined in equation (4.18), where N_V is the number of existing VNs.

$$S_{LS}(i) = \sum_{j}^{N_V} \sum_{k}^{L_{V_j}} ((S_{LV_j}(k_j) | k_j \supseteq i)) \tag{4.18}$$

Afterwards, it proceeds with the determination of Node Stress (S_N), which is a combination of the currently available Substrate Node resources and weights active Virtual Machines, free RAM (Free RAM) amount in Mega Byte (MB), number of CPUs (N.CPU), CPU frequency in MHz (CPU Frequation) and current CPU Load, which varies between 0 and *N.CPU* (number of physical CPU Cores). The Λ function, defined in 4.19, determines whether or not a virtual node n_j, belonging to the j^{th} VN, is active and running on the i^{th} physical node.

$$\Lambda(n_j, i) = \begin{cases} 1 & \text{if } n_j \supseteq i \wedge n_j \text{ is active} \\ 0 & \text{otherwise} \end{cases} \tag{4.19}$$

The Node Stress of the i^{th} physical node is given in equation (4.20), where δ is a small constant to avoid dividing by 0, and N_{V_j} is the number of virtual nodes on the j^{th} VN.

$$S_{N_i} = \frac{\sum_{j}^{N_V} \sum_{n}^{N_{V_j}} \Lambda(n_j, i)}{\delta + \text{Free RAM} \cdot \text{CPU Freq} \cdot (\text{N.CPU - Load})} \tag{4.20}$$

The next step is the determination of node candidates. For each virtual node, a set of physical candidates is determined based on eliminative constraints: location, number of CPUs, CPU frequency, free RAM amount, and available HDD space.

After determining the candidates for each virtual node, a sorting algorithm is run that orders the virtual nodes by their number of candidates so that virtual nodes with fewer candidates will be mapped first.

The algorithm terminates with the final node mapping and path selection. For each possible candidate v, a CSPF algorithm to all other candidates (u) of the virtual neighbour nodes is calculated using the previously calculated Link Stresses as weights, and the path cost is stored ($D(v, u)$). The node potential is then determined using the equation (4.21), where V_C is a set containing the candidates of the neighbour virtual nodes.

$$\pi(v) = \sum_{u \in V_C} D(v, u) \cdot S_{N_v} \tag{4.21}$$

The algorithm terminates successfully when all the requested virtual nodes are properly mapped, each one on a different physical node, chosen from within its candidate set, and the best-constrained paths for each virtual link are determined.

Algorithm 1: Baseline Mapping Algorithm Pseudo-Code

input : *Substrate* (Substrate Network) , $V_{Request}$ (Requested VN)
output: V_{Map} (Mapped Virtual Network)
foreach *Link i in Substrate.Links* **do**
  $S_{LS}(i) = \sum_j^{N_V} \sum_k^{L_{V_j}} ((S_{LV_j}(k_j)|k_j \supseteq i))$;
end
foreach *Node i in Substrate.Nodes* **do**
  $S_{N_i} = \frac{\sum_j^{N_V} \sum_n^{N_{V_j}} \Lambda(n_j,i)}{\delta + \text{Free RAM}\cdot\text{CPU Freq}\cdot(\text{N.CPU - Load})}$;
end
foreach *Node n in* $V_{Request}$*.Nodes* **do**
  foreach *Node i in Substrate.Nodes* **do**
    if *MeetsConstraints(n, i)* **then**
      n.Candidates.Add(i) ;
    end
  end
end
SortVirtualNodes($V_{Request}$) ;
foreach *Node n in* $V_{Request}$*.Nodes* **do**
  foreach *Link k connected to n* **do**
    ConnectedVNode=GetLinkDestination(k) ;
    foreach *SourceCandidate v in n.Candidates* **do**
      $\pi(v) = 0$;
      foreach *DestCandidate u in ConnectedVNode.Candidates* **do**
        D(v,u)= Cost(CSFP_Dijkstra(v,u)) ;
      end
      $\pi(v) = \sum_{u \in V_C} D(v,u) \cdot S_{N_v}$;
    end
  end
  n.Map = $v : \pi(v) = min(\pi)$;
end
foreach *Node n in* $V_{Request}$*.Nodes* **do**
  V_{Map}*.Nodes* $\cup$ *n* ;
  foreach *Link k connected to n* **do**
    ConnVNode=GetLinkDestination(k) ;
    V_{Map}*.Links* $\cup$ CSFP_Dijkstra(n.Map,ConnVNode.Map) ;
  end
end

4.2.2 Virtual Network Embedding - Enhanced Shortest-Path Heuristic

A pseudo-code description of the Virtual Network Embedding - Enhanced Shortest-Path Heuristic (VNE-ESPH) algorithm is shown in algorithm 2. For further details on the VNE-ESPH algorithm, please see the appendix section B.

Two phases are considered in the baseline algorithm to embed the VNs: in the first phase, the virtual nodes are mapped onto physical nodes using a greedy algorithm; in the second phase, the virtual links are embedded onto physical links using the CSPF Dijkstra algorithm. The greedy node mapping algorithm is composed by two stages:

- Pre-candidate filtering - Physical node candidates that do not satisfy one or more constraints (e.g. CPU, memory, HDD, and location) are excluded.

- Candidate sorting - The virtual node candidates are sorted according to the node potential. First, the physical link stress and physical node stress are determined using equation (4.18) and equation (4.20), respectively. Secondly, the physical node potential is determined using equation (4.21).

With respect to the baseline algorithm (1), this one (algorithm 2) contains several changes. First, we used a different equation to determine the node stress, S_N, which is reflected at

Algorithm 2: Virtual Network Embedding - Enhanced Shortest-Path Heuristic (VNE-ESPH) Pseudo-Code

input : *Substrate* (Substrate Network) , $V_{Request}$ (Requested VN)
output: V_{Map} (Mapped Virtual Network)

```
1  foreach Link i in Substrate.Links do
2      foreach VN j in Substrate.VNs do
3          foreach Link k in j.Links do
4              if Link k_j ⊇ Link i then
5                  S_LS(i) += S_LV_j(k_j) ;
6              end
7          end
8      end
9  end
10 foreach Link i in Substrate.Links do
11     S_LS(i) = Σ_j^{N_V} Σ_k^{L_Vj} ((S_LV_j(k_j)|k_j ⊇ i)) ;
12 end
13 foreach Node i in Substrate.Nodes do
14     S_Ni = CPU_Freq × [(M^Pused / M^Ptotal)^2 + (C^Pused / C^Ptotal)^2];
15     π(v) = 0 ;
16 end
17 foreach Node n in V_Request.Nodes do
18     foreach Node i in Substrate.Nodes do
19         if MeetsConstraints(n, i) then
20             n.Candidates.Add(i) ;
21         end
22     end
23 end
24 foreach Node n in V_Request.Nodes do
25     foreach Link k connected to n do
26         ConnectedVNode=GetLinkDestination(k) ;
27         foreach SourceCandidate v in n.Candidates do
28             foreach DestCandidate u in ConnectedVNode.Candidates do
29                 D(v,u)= Cost(CSFP_Dijkstra(v,u));
30                 if u.Map then
31                     D(v,u)=β × D(v,u);
32                 end
33             end
34             if π(v) then
35                 π(v) = mean[π(v), min(∀u ∈ V_C : D(v,u)] ;
36             end
37             else
38                 π(v) = min[∀u ∈ V_C : D(v,u)] ;
39             end
40         end
41         π(v) = π(v) × S_Nv;
42     end
43     n.Map = v : π(v) = min(π) ;
44 end
45 foreach Node n in V_Request.Nodes do
46     V_Map.Nodes ∪ n ;
47     foreach Link k connected to n do
48         ConnVNode=GetLinkDestination(k) ;
49         V_Map.Links ∪ CSFP_Dijkstra(n.Map,ConnVNode.Map) ;
50         foreach Link i in Substrate.Links do
51             if V_Map.Links then
52                 S_LS(i)+ = S_LV_n(k) ;
53             end
54         end
55     end
56 end
```

line 14. The former equation represented in equation (4.20) tends to balance the number of virtual nodes per physical node, to favour nodes with lower CPU clock frequency and to

reduce the combination of consumed Random Access Memory (RAM) and CPU. However, we could have physical nodes with different capacities and also virtual nodes with different requirements, which do not cope well with the objective of distributing the virtual nodes per physical nodes uniformly. Moreover, physical nodes could be highly loaded at the CPU and mostly free at the RAM or the opposite, which, for the equation is totally transparent as long as the combination of the two has the lower value. The equation proposed for the node stress, presented in equation (4.22), tends to balance the use of both RAM and CPU, and to favour nodes with higher clock CPU frequency.

Lines 30 to 32 are used to tune the $link - path$ cost, $D(u, v)$ according to neighbours. We have set the value of β to 0.01, which reduces the $link - path$ cost to virtual neighbours that have been already assigned. Lines 34 to 39 are the replacement of line 32 in algorithm (1) used to calculate the node potential i.e., π. Here, the node potential is the average of the minimum $link - path$ cost to all the possible candidates to virtual neighbours, multiplied by the node stress, which is represented in line 41. Lines 50 to 54 are used to update in runtime the link stress, S_{LS}, of the physical links that have been already assigned to virtual links.

$$S_{N_i} = \text{CPU\_Freq} \times [(\frac{M^{P_{used}}}{M^{P_{total}}})^2 + (\frac{C^{P_{used}}}{C^{P_{total}}})^2] \tag{4.22}$$

4.3 Mathematical Formulation

This section describes the mathematical formulation developed to solve the online VN embedding problem with the defined constraints. For further details on the mathematical formulation, please see the appendix section B.

An Integer Linear Programming (ILP) approach is used to solve the online VN embedding problem; we propose a node-link formulation, and two assignment variables are applied during the embedding process. The index notation used here is the same as in sub-section 4.1.1.3.

4.3.1 Assignment Variables

The binary variable x is used in the mapping of the virtual nodes and is defined in equation (4.23), where $x_i^m \rightarrow N^V \times N^P$ matrix. With respect to the virtual links, the binary variable y is used and it is represented in equation (4.24), where $y_{ij}^{mn} \rightarrow (L^V)^2 \times (L^P)^2$ matrix.

4.3.1.1 Virtual Node Assignment

$$x_i^m = \begin{cases} 1, \text{virtual node } m \text{ is allocated at physical node } i \\ 0, else \end{cases} \tag{4.23}$$

4.3.1.2 Virtual Link Assignment

$$y_{ij}^{mn} = \begin{cases} 1, \text{virtual link } mn \text{ uses physical link } ij \\ 0, else \end{cases} \tag{4.24}$$

4.3.2 Constraints

To assure the correct mapping of the virtual nodes and of the virtual links, and also to obey to the conservation law on the capacities of the physical nodes and physical links, a set of constraints is defined.

4.3.2.1 Assignment of virtual nodes to physical nodes

Equation (4.25) ensures that each virtual node is assigned, and that it is assigned to just one physical node.

$$\forall m : \sum_{i} x_i^m = 1 \tag{4.25}$$

4.3.2.2 One virtual node per physical node

Equation (4.26) guarantees that each physical node can accommodate in the maximum one virtual node per VN request, although each physical node can accommodate other virtual nodes from different VNs. This constraint is used to ensure that each virtual node is assigned to a different physical node per VN embedding, and can be suitable in application scenarios where it is required to have physical node diversity for redundancy reasons.

$$\forall i : \sum_{m} x_i^m \leq 1 \tag{4.26}$$

4.3.2.3 CPU conservation

Equation (4.27) assures that the available CPU capacity of each physical node is not exceeded.

$$\forall i : \sum_{m} x_i^m \cdot C_m^v \leq C_i^p \tag{4.27}$$

4.3.2.4 Virtual Node distance

Equation (4.28) assures that the maximum distance between virtual nodes, D_{mn}^v, is not violated. The maximum distance between virtual nodes is a parameter of the VN embedding problem. The effect of this parameter on the VN embedding will be studied on a separate section (see sub-section 4.8.4).

This parameter is given in distance units and can be used to express the maximum radius between virtual nodes (in the simulated scenario the location of the physical nodes is set in a grid). The distance between physical nodes, i.e. Dis_{ij}^p, is obtained using equation (4.1), and K represents a large constant which is used only in situations where the virtual node n is not mapped at the physical node i, i.e. $x_i^n = 0$.

$$\forall m, n \in L_m^v, m < n, \forall i : \sum_{j} \text{Dis}_{ij}^p \cdot x_j^m \leq \text{Dis}_{mn}^v \cdot x_i^n + (1 - x_i^n) \cdot K \tag{4.28}$$

4.3.2.5 Assignment of virtual links to physical links - multi-commodity flow conservation with node-link formulation

To simultaneously optimize the mapping of virtual links and virtual nodes, the multi-commodity flow constraint [EIS75] is applied with a node-link formulation [PM04]; moreover, the notion of direct flows on the virtual links is used, which is represented in Eq. (4.29), where L_m^v represents all the virtual links that are directly connected with the virtual node m, and L_i^p represent all the physical links that are directly connected with the physical node i.

$$\forall mn \in L_m^v, m < n, \forall i : \sum_{ij \in L_i^p} (y_{ij}^{mn} - y_{ji}^{mn}) = x_i^m - x_i^n \tag{4.29}$$

4.3.2.6 Bandwidth conservation

To ensure that the available bandwidth at each physical link is not surpassed, Equation (4.30) is defined.

$$\forall ij \in L_i^p, i < j : \sum_{mn \in L_m^v, m<n} B_{mn}^v (y_{ij}^{mn} + y_{ji}^{mn}) \leq B_{ij}^p \tag{4.30}$$

4.3.2.7 Link delay limit

The virtual link delay, D_{mn}^v, is a parameter of the VN embedding problem, and is equal to the sum of the delay of all physical links that compose the virtual link. To ensure that the constraint on the link delay is not violated we apply equation (4.31).

$$\forall mn \in L_m^v, m < n, \forall i : \sum_{ij \in L_i^p, i<j} D_{ij}^p (y_{ij}^{mn} + y_{ji}^{mn}) \leq D_{mn}^v \tag{4.31}$$

4.4 Objective Functions - Resource Allocation

One of the major challenges when formulating an ILP model for VN assignment resides in the definition of the objective function: the allocation of resources need to be optimized in order to support the efficiency of the corresponding VN process.

Moreover, the correct specification of the VN mapping constraints (see section 4.3) is also a challenge of this approach. In this section, we describe the main goals that need to be achieved when formulating an objective function for virtual network embedding; three different objective functions are proposed to achieve these goals.

4.4.1 Objective Goals

A primary goal for the embedding algorithm is to minimize resource consumption in order to have resources available for forthcoming VN embedding requests. Minimization of resource consumption is only possible for the bandwidth consumption depending on the number of links involved in an embedding process. The processing power has just to be installed exactly in the amount required by the VN request on some physical nodes.

Resource minimization consequently means that the VNs should exhibit minimal hop counts on their paths. This in turn means that almost every physical node should be available to host a virtual node. As long as the resources required by VNs are small compared to physical capacities of nodes and links, this availability is guaranteed with high probability by a load balancing strategy, which results in some spare capacity for each physical node or link.

Therefore, the dominating aspects in the formulation of an objective function for the ILP problem are the minimization of bandwidth consumption and load balancing.

4.4.2 Load Balancing plus ϵ Shortest Path

The objective function Load Balancing plus ϵ Shortest Path (LB+ϵSP) is proposed in equation (4.32), and it achieves two goals: the primary goal is to minimize the maximum load per physical resources; in the case of different mapping solutions with the same maximum utilization, the second part of the objective function is activated which will opt for the solution

which consumes the lowest bandwidth. $\mathcal{L}^{C^p}_{max}$ represents the overall maximum node load; $\mathcal{L}^{B^p}_{max}$ represents the overall maximum link load. The parameters $C^p_i(0)$, $B^p_{ij}(0)$, $C^v_m(k)$, $B^v_{mn}(k)$ were defined in Table 4.1; the parameter ϵ represents a small constant, which should be small enough to not affect the first objective; and the parameters α and β are used to weight the load cost of each type of resources.

$$
\begin{aligned}
&minimize\ \alpha \cdot \mathcal{L}^{C^p}_{max} + \beta \cdot \mathcal{L}^{B^p}_{max} + \epsilon \cdot \sum_{mn} y^{mn}_{ij} \cdot B^v_{mn}(t), \\
&\forall i \in N^p : \frac{C^p_i(t) + \sum_m x^m_i \cdot C^v_m(k)}{C^p_i(0)} \leq \mathcal{L}^{C^p}_{max} \\
&\forall ij \in L^p : \frac{B^p_{ij}(t) + \sum_{mn} y^{mn}_{ij} \cdot B^v_{mn}(k)}{B^p_{ij}(0)} \leq \mathcal{L}^{B^p}_{max}
\end{aligned} \tag{4.32}
$$

4.4.3 Shortest Distance Path

The previous objective function (4.32) works well in situations where there are abundant resources in the physical network. Then, bandwidth consumption is of no concern and load balancing is beneficial because it gives a high degree of flexibility in the resource allocation process.

Nevertheless, in situations where the physical resources are scarce, it is desirable to reduce the number of physical links consumed to the minimum possible.

Therefore, the objective function Shortest Distance Path (SDP), proposed in equation (4.33), aims to minimize the number of physical links consumed due to the VN embedding, while it prefers physical links with more available bandwidth, and at the same time chooses physical nodes with more available CPU power, thereby supporting the load balancing aspect. The parameters α and β are used to weight the cost of each type of resource. (Note that the first term in equation (4.33) would result in a constant, if $C^p_i(t)$ was missing in the denominator.)

$$
minimize\ \alpha \left(\sum_m \sum_i \frac{x^m_i}{C^p_i(t)} \right) + \beta \left(\sum_{mn} \sum_{ij} \frac{y^{mn}_{ij}}{B^p_{ij}(t)} \right) \tag{4.33}
$$

4.4.4 Weighted Shortest Distance Path

The objective function Weighted Shortest Distance Path (WSDP), proposed in equation (4.34), is similar to equation (4.33), although here the demanded capacity by the VN is included in the objective function. This has the effect that high demands are allocated to nodes or links with a large amount of free capacity.

$$
minimize\ \alpha \left(\sum_m C^v_m(k) \left[\sum_i \frac{x^m_i}{C^p_i(t)} \right] \right) + \beta \left(\sum_{mn} B^v_{mn}(k) \left[\sum_{ij} \frac{y^{mn}_{ij}}{B^p_{ij}(t)} \right] \right) \tag{4.34}
$$

4.5 Re-Optimization Extension

In order to support the re-optimization process, equation (4.35) is proposed and differs from the initial formulation (see equation 4.26), since it takes into consideration all the VNs that are currently assigned, and not only one VN request. Equation (4.35) is also used to

guarantee that each physical node accommodates, in maximum, one virtual node per VN[2], where k is used to represent all VNs running on that specific physical node. However, each physical node can accommodate, in principle, more virtual nodes from other VNs (i.e. VN_k). For further details on the re-optimization, please see the appendix section C.

$$\forall k, \forall i : \sum_{m \in VN_k} x_i^m \leq 1 \tag{4.35}$$

4.6 Energy Aware - Extension

Another important objective is related with the energy consumption of the physical network, and how it can be minimized by concentrating the load on the minimum amount of physical nodes. For further details on the proposed extension for energy-aware, please see the appendix section E.

4.6.1 Energy Consumption Minimization

One can realize from the previous objective functions that they are agnostic to energy consumption aspects, i.e. the power-up of new nodes.

The objective function Energy Consumption Minimization (ECM), which is proposed in equation (4.36), is energy consumption oriented and fulfills three objectives: i) to minimize the power-up of new physical nodes including the forwarding nodes per VN embedding - this is achieved using the parameter $P_{ij}(t)$ in the first term that represents the power state of the physical nodes i, j immediately prior to the embedding, and it is used to penalize the allocation of virtual links on physical links attached to inactive physical nodes; ii) to minimize the number of physical links required per VN embedding - this is achieved by summing the decision variable y_{ij}^{mn}, which is used to represent the mapping of virtual links; iii) to minimize the load power - this is obtained by using the second term that considers the current CPU allocation $[C_i^p(t_0) - C_i^p(t_k)]$ on physical node i, plus the CPU demand $[C_m^v(t_k)]$ of virtual node m over the total CPU capacity $[C_i^p(t_k)]$. This gives the CPU ratio which is then multiplied by P_l. To not jeopardize the second objective when physical nodes i and j are active, we consider $P_{ij}(t)$ to be equal to P_l. The parameters α and β are used to weight the cost of each term of the objective function.

$$\begin{aligned} & minimize\ \alpha \left(\sum_{mn \in L^v, n<m} \sum_{ij \in L^p} y_{ij}^{mn} \times P_{ij}(t) \right) \\ & + \beta \left(\sum_{m \in N^v} P_l \sum_{i \in N^p} x_i^m \frac{C_i^p(t_0) - C_i^p(t_k) + C_m^v(t_k)}{C_i^p(t_0)} \right), \text{where} \\ & P_{ij}(t) = \begin{cases} P_l, \text{ if physical node } i \text{ and } j \text{ are active} \\ P_b, \text{ if physical node } i \text{ and } j \text{ have different states} \\ 2P_b, \text{ if physical nodes } i \text{ and } j \text{ are not active} \end{cases} \end{aligned} \tag{4.36}$$

4.6.2 Bandwidth Consumption Minimization

In situations where the bandwidth resource is scarce or more expensive when compared to the CPU, it is preferable to obtain the minimum bandwidth allocation. The objective function Bandwidth Consumption Minimization (BCM), which is proposed in equation (4.37), fulfils this objective: bandwidth allocation minimization per VN embedding request.

[2]This assumption is also taken by other authors, i.e., [ZA06, YYRC08, CRB09].

$$minimize \sum_{mn \in L^v, n<m} B^v_{mn}(t_k) \sum_{ij \in L^p} y^{mn}_{ij} \tag{4.37}$$

4.7 Virtual Network Migration Extension

The VN re-embedding problem requires the re-mapping of virtual nodes and virtual links, i.e. decision variables. On the nodes we can minimize the virtual nodes that need to be migrated, while on the virtual links we can minimize the overall bandwidth allocation. In this sub-section, we propose one objective function to address the VN re-embedding problem from a cost migration standpoint: i) virtual node migration minimization; ii) bandwidth allocation optimization. For further details on the virtual migration extension, please see the appendix section F.

4.7.1 Node Migration and Bandwidth Consumption Minimization

The objective function Node Migration and Bandwidth Consumption Minimization (NM-BCM), proposed in equation (4.38), aims to minimize the overall number of virtual nodes migrated, which is achieved by using the first term of the equation. This objective function also aims to minimize the overall bandwidth consumption in the second term. The parameter α is used to weight the cost of each virtual node migration, and M is a set of physical nodes that require intervention.

$$minimize \sum_{k \in K} \sum_{m \in N^v_k} \sum_{i \in N^p} x^m_i \times X^m_i + \tag{4.38}$$
$$\sum_{k \in K} \sum_{mn \in L^v_k, n<m} B^v_{mn} \sum_{ij \in L^p} y^{mn}_{ij}, \text{where}$$
$$X^m_i = \begin{cases} 0, \text{ if virtual node } m \text{ is allocated} \\ \text{at physical node } i \text{ or } i \subseteq M \\ \alpha, \text{ otherwise} \end{cases}$$

4.8 Evaluation Results

In this section, we describe the simulation scenario, the evaluation metrics, and depict our major results. We compare the VNE-NLF model in its several versions with six state of the art methods. Evaluation results on the extensions are also provided.

4.8.1 Baseline Heuristics

This sub-section presents a description on the baseline heuristic algorithms that have been used as a performance comparison with the proposed mathematical formulation, i.e. VNE-NLF. For further details on the baseline heuristics, please see the appendix section D.

4.8.1.1 Greedy Node Mapping with Shortest Path based Link Mapping (G-SP)

The G-SP [ZA06] is a two-step mapping algorithm: in the first-step, a greedy algorithm is applied to map virtual nodes onto substrate nodes with more available resources; in the second step, the shortest-distance path algorithm [MS97] is used to embed the virtual links.

4.8.1.2 Greedy Node Mapping with Splittable Link Mapping using Multi-Commodity Flow Constraint (G-MCF)

The Greedy Node Mapping with Splittable Link Mapping using Multi-Commodity Flow Constraint (G-MCF) [YYRC08] is also a two-step mapping algorithm: in the first-step, it embeds the virtual nodes using a greedy node mapping algorithm (i.e. map the virtual nodes onto substrate nodes that have more resources available); in the second it embeds the virtual links by solving the Multi-Commodity Flow Problem (MFP) [EIS75].

4.8.1.3 Coordinated - Node-Link Mapping (C-NLM)

The C-NLM [CRB09] is still a two-step algorithm but with better node-link mapping coordination: in the first step, the VN request is mapped onto "meta-nodes" and "meta-links"; in the second step, the meta-nodes are mapped onto substrate nodes using randomized rounding (R-ViNE), or deterministic rounding (D-ViNE), and the virtual links are assigned to physical paths by applying the shortest-path algorithm, or by solving the MCFP. The several versions of this heuristic are the following: Randomized Node Mapping with Splittable Link Mapping using Multi-Commodity Flow Constraint (R-ViNE)[CRB09], Deterministic Node Mapping with Slipttable Link Mapping using Multi-
Commodity Flow Constraint (D-ViNE)[CRB09], Deterministic Node Mapping with Shortest Path based Link Mapping (D-ViNE-SP)[CRB09] and Deterministic Node Mapping with Splittable Link Mapping using Multi-
Commodity Flow Constraint and Load Balancing based (D-ViNE-LB)[CRB09].

4.8.2 Simulation Parameters

To evaluate the VNE-NLF model, we have implemented a discrete event simulator in Matlab®, with the proposed formulation using different objective functions.

The physical network topology is created using the Georgia Tech Internetwork Topology Models (GT-ITM) tool [ZCB96], the number of physical nodes is set to 50, which is representative of a medium scale infrastructure provider, and the link probability between two physical nodes is set to 0.5. The node CPU capacity and the link bandwidth are real numbers uniformly distributed between 50 and 100. The VNs requests are also representative of either small or medium scale virtual networks, and are created using the same topology generation method; the number of virtual nodes is not fixed, but follows a uniform distribution, from 2 to 10 virtual nodes per VN topology; the virtual link probability is set to 0.5. The CPU capacity of the virtual nodes and the bandwidth of the virtual links are also real numbers uniformly distributed between 0 and 20, and between 0 and 50, respectively[2]. The considered values for the bandwidth and for the CPU are normalized, since the objective function aims at simultaneously optimizing the allocation of both types of resources.

We assume that VN requests arrive according to a Poisson process, and that each VN has an associated lifetime measured in time units with an average of $1/\mu = 1000$, following an exponential distribution. The same assumption was also taken by the authors of [CRB09]. The average number of VN requests per time unit, i.e., value of λ, is started with 3 VN requests per 100 time units, and increases by 1 VN request, up to 10 requests. This can give an insight into two opposite case scenarios, with a very high and very low acceptance ratio. For each value of λ, 10 trials are performed. A new set of VN requests and a new physical network topology are generated for each trial. All simulations are set to run up to 50000

[2]These values were also considered by the authors of [YYRC08, ZA06, CRB09]

time units to mitigate the transient phase effect [Jai91] and to obtain the steady-state. A confidence interval of 95% is used for all results presented below.

The evaluated embedding methods are G-SP[ZA06], G-MCF[YYRC08], R-ViNE[CRB09], D-ViNE[CRB09], D-ViNE-SP[CRB09], D-ViNE-LB[CRB09], and the proposed linear programming formulation, i.e. VNE-NLF, with 3 different cost functions which were described in the previous section. All these methods are briefly summarized in Table 4.2.

The state of the art methods are simulated using an existing implementation [vin12]; to solve the mixed integer programming on the methods G-MCF, R-ViNE, D-ViNE, D-ViNE-LB, and D-ViNE-SP, the Gnu Linear Programming Kit (GLPK) [glp12] solver version 4.20 is used.

All the simulations for the different embedding methods were performed using an Intel® Xeon® CPU X3220@2.4GHz, and the time consumed per VN request embedding was registered.

Table 4.2: Compared VN Embedding Methods.

Notation	Method Description
G-SP [ZA06]	Greedy Node Mapping with Shortest Path Based Link Mapping.
G-MCF [YYRC08]	Greedy Node Mapping with Splittable Link Mapping using MCF.
R-ViNE [CRB09]	Randomized Node Mapping with Splittable Link Mapping using MCF.
D-ViNE [CRB09]	Deterministic Node Mapping with Splittable Link Mapping using MCF.
D-ViNE-SP [CRB09]	Deterministic Node Mapping with Shortest Path Based Link Mapping.
D-ViNE-LB [CRB09]	Deterministic Node Mapping with Splittable Link Mapping using MCF, where $\alpha_{uv} = \beta_{uv} = 1, \forall u, v, w \in N^S$.
VNE-NLF-LB+ϵSP	VN Embedding with node-link Formulation using overall Load Balancing; in the case of having more than one solution, it uses Shortest Path, where $\epsilon = 1.0 \times 10^{-11}$.
VNE-NLF-SDP	VN Embedding with node-link Formulation using overall Short Distance Path.
VNE-NLF-WSDP	VN Embedding with node-link Formulation using overall Weighted Short Distance Path.

The CPLEX®[cpl12] version 12.2 was used to solve the linear programming problem of the VNE-NLF; a time limit of 600 seconds is defined for each VN mapping, although most of the VNs are embedded in hundreds of milliseconds; and the CPLEX® was set to use only one CPU core for comparison purposes with the remaining methods. The evaluation metrics are the ones defined in section 4.1.5.

4.8.3 Impact of the Number of VN Requests

This sub-section presents the evaluation results as a function of the VN request rate, for all the previously described metrics. To increase the readability of all figures, we have considered different x values for different strategies, e.g.: 3.4, 4.4, 5.4 for G-SP; 3.3, 4.3, 5.3 for G-MCF; 3.2, 4.2, 5.2 for R-ViNE.

Before comparing the different embedding methods and algorithms, we should group them into four different categories according to the nature of the method itself, i.e. heuristic, heuristic combined with mixed integer programming, and linear programming:

i Heuristic - the VN embedding problem is solved using a simple algorithm; this method performs the VN embedding very $fast$ and a possibly sub-optimal embedding solution is

obtained. The method G-SP [ZA06] fits into this category;

ii Heuristic combined with Mixed Integer Programming (MIP) - the VN embedding problem is solved in two steps: in the first step a mathematical algorithm is used to map virtual nodes on physical nodes, and in the second step the MIP is performed to embed the virtual links. The method G-MCF [YYRC08] fits into this category.

iii Heuristic combined with Mixed Integer Programming (MIP) and a better coordination between mapping phases is added - the same principle is applied, as in the above category, to solve the VN embedding problem, although a better coordination between the mapping phases is achieved using an augmented "*substrategraphconstruction*" [CRB09]. The methods R-ViNE, D-ViNE, D-ViNE-SP, D-ViNE-LB [CRB09] fit into this category;

iv Integer Linear Programming (ILP) - the VN embedding problem is solved using integer linear programming. This method obtains an optimal solution for a given cost function combining resource consumption minimization with a load balancing strategy. The method VNE-NLF and its different objective functions fit into this category.

4.8.3.1 VN Request Acceptance Ratio

One of the main aspects of the performance of each embedding method is the VN request acceptance ratio, which is shown in Figure 4.3 and is given by equation (4.6). As can be observed, all methods show a linear behaviour with the variation on the VN requests, where the acceptance ratio decays linearly with the number of VN requests, and the slope is approximately the same for all methods. This decay represents the fact that there are no infinite physical resources.

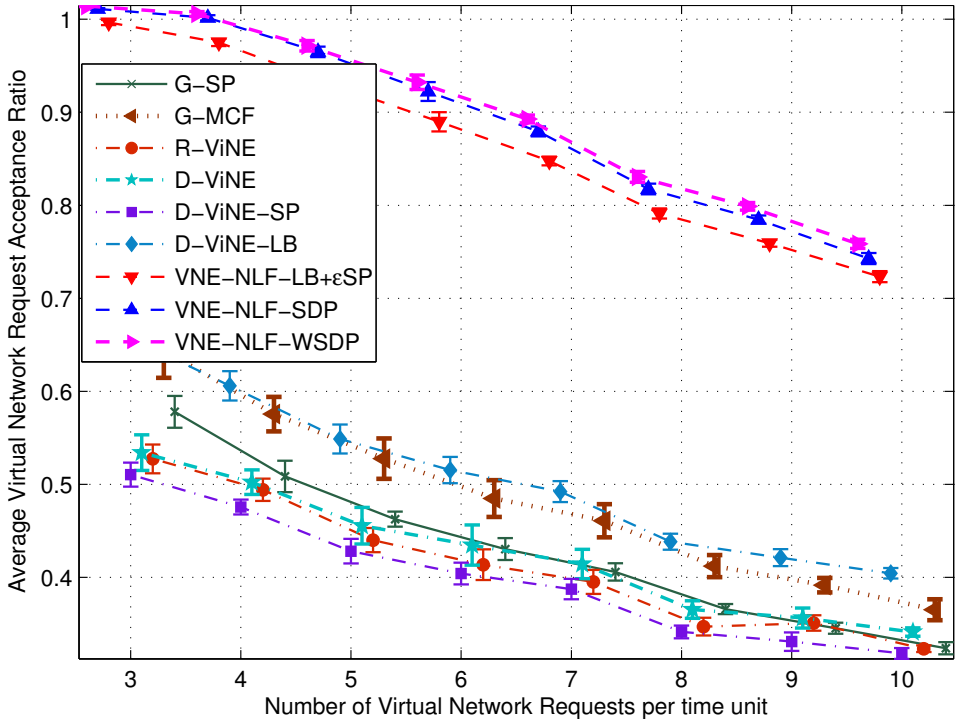


Figure 4.3: Average VN Acceptance ratio as a function of VN Request rate.

The method VNE-NLF, with its different objective functions, achieves the highest performance, and it clearly outperforms the other approaches. This is expected since integer linear programming is applied to solve the VN embedding problem, and the optimal solution, according to the objective function considered, is obtained per VN embedding.

The reason for these results, not only resides in the usage of an integer linear programming approach, but also in the utilization of the node-link formulation by the VNE-NLF, which considers the universe of all possible embedding solutions, instead of a few solutions. If we take, for example, the first case with only 3 VN requests per 100 time units, the VNE-NLF is able to accept nearly all requests, while the remaining methods are able to accept only 70% of the requests. The embedding method that has the lowest acceptance ratio is the D-ViNE-SP, and the method with the highest VN request acceptance ratio is the VNE-NLF-WSDP.

It is expected that the VNE-NLF method will perform better in all cases. For instance, if the embedding problem is feasible, i.e., possible solutions exist, the VNE-NLF will find out the optimal solution according to the cost function. Using a heuristic approach or even a combined approach, this is not always the case: frequently only a feasible solution will be presented.

4.8.3.2 Node Utilization

The average node utilization as a function of the number of VN requests is depicted in Figure 4.4. With a small number of VN requests, i.e., 3 VN requests, the node utilization does not go beyond 20% and 35% for the heuristic group (i.e. groups *i*, *ii*, and *iii*) and for the VNE-NLF, respectively. The VNE-NLF group is consuming more resources of the physical nodes than the heuristics, which is expected according to the acceptance ratio. When the number of VN requests is increased, the node utilization also increases, since we are trying to accommodate more VNs on the infra-structure, but with the same amount of available

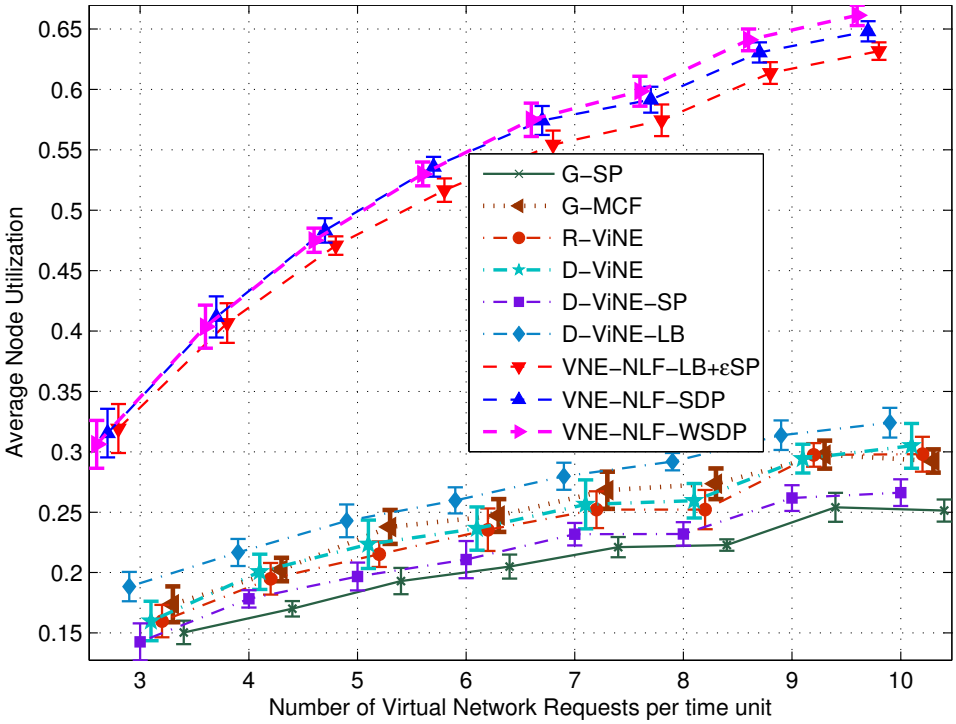


Figure 4.4: Average Node Utilization as a function of VN Request rate.

physical resources.

An efficient embedding method in situations of high VN demand would be able to load the nodes to their full capacity. The important aspect to retain here is how much node resources can be loaded and what kind of embedding methods tends to saturate them firstly.

The node utilization shows a dependency on the VN acceptance ratio, as it can be perceived from Figure 4.3 and Figure 4.4. To provide a better understanding on this issue, we plot the acceptance ratio metric times the node utilization, which is shown in Figure 4.5. We observe that the methods that make use of heuristics, e.g. G-SP, or heuristics combined with MIP, e.g. G-MCF and D-ViNE-LB, show the same behaviour for all the VN requests considered, i.e. the VN acceptance metric multiplied by the node utilization metric is nearly constant. The same does not apply to the VNE-NLF, since it increases per VN request considered, until 6 VN requests per 100 time units, and beyond the 6 VN requests per 100 time units, it shows the same behaviour as its counterparts. This means that, although the VN request rate is increasing, the VNE-NLF approach is able to keep with this increase until the VN embedding problem moves from an optimization problem (i.e. there are sufficient physical resources for the demand), to a feasibility problem (i.e. there are no sufficient resources for the demand).

4.8.3.3 Link Utilization

The physical link utilization metric is plotted in Figure 4.6. Here we do not have the same regular behaviour according to the number of VN requests for all the methods, as shown before for the node utilization. Also, there is no consensus in terms of clearly identifying which group causes the highest utilization on the physical links due to the embedding process. Nevertheless, we can clearly state that, on average, either the G-MCF or D-ViNE-LB shows

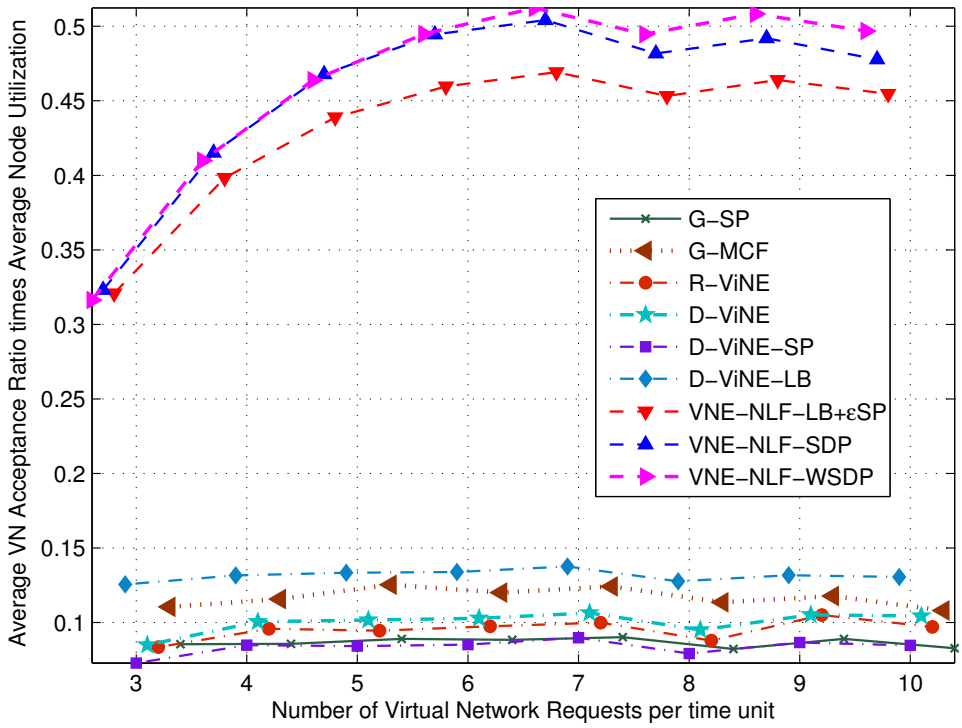


Figure 4.5: Average VN Acceptance Ratio times Average Node Utilization as a function of VN Request rate.

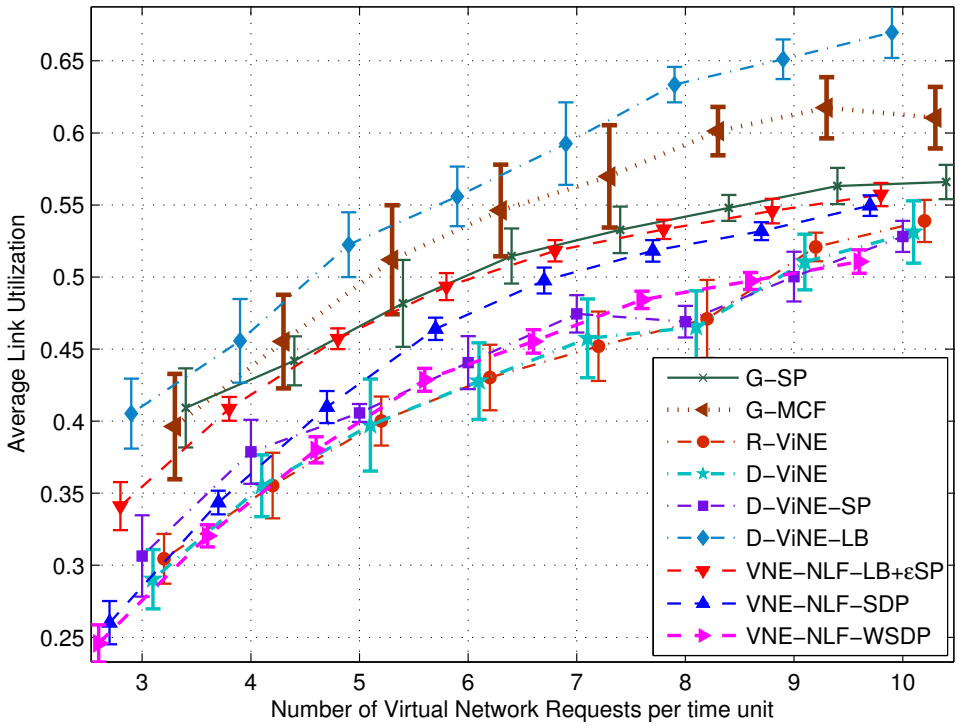


Figure 4.6: Average Link Utilization as a function of VN Request rate.

the highest utilization on the links; in the extreme case scenario (i.e. with 10 VN requests) they have an average link utilization of 60% and 67%, respectively. With respect to the lowest link utilization, we observe that the embedding methods R-ViNE, D-ViNE, D-ViNE-SP, and VNE-NLF-WSDP are the ones that tend to consume less bandwidth, reaching values of 50% of average link utilization, for the same considered situation.

Having in mind that one virtual link could be mapped in several ways, it is reasonable to observe different behaviours according to the strategy of the method. If the strategy is to save bandwidth, i.e. SP, the embedding will consume the least bandwidth possible per VN mapping; if the strategy is load balancing on the links, i.e. LB, it will tend to balance the utilization among all links in order to distribute the total load.

The link utilization also shows a dependency on the VN acceptance ratio, as can be observed in Figure 4.3 and Figure 4.6. To provide a better understanding, we plotted the acceptance ratio metric times the link utilization in Figure 4.7. In contrast to the node utilization, the dependency factor on the link utilization shows a more complex behaviour: it still increases significantly until reaching 6 VN requests for the case of the VNE-NLF group, although it starts to decrease after 7 VN requests, in a not so expressive way. For the other methods the dependency on the VN request rate is less pronounced.

4.8.3.4 Embedding Factor

Figure 4.8 shows the embedding factor as a function of the VN request rate, where the weight parameters, α, and β of equation (4.9) are set to 0.5. The embedding factor slightly decreases with the number of VN requests, except for the case of the VNE-NLF-LB+ϵSP. The behaviour obtained when using the VNE-NLF-LB+ϵSP is as expected, since it performs an overall load balancing of the physical nodes and links, choosing the solution that consumes the least bandwidth. Therefore, in the situation of only a few VN requests, the method will tend to allocate more resources than required due to the nature of the load balancing; with

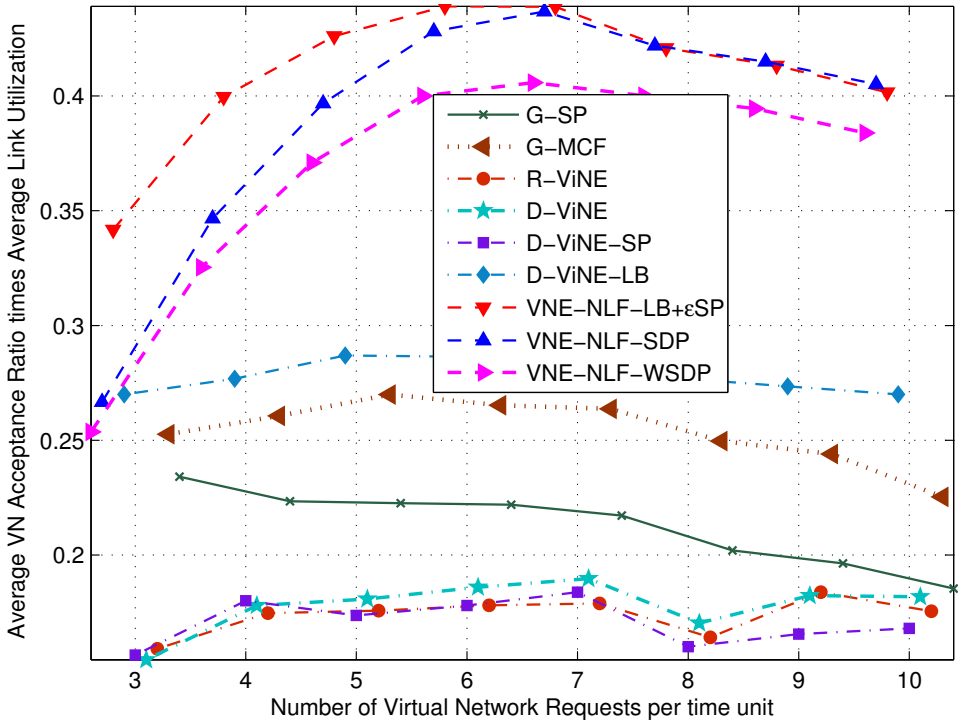


Figure 4.7: Average VN Request Acceptance Ratio times Average Link Utilization as a function of VN Request rate.

a higher VN request rate, this situation tends to disappear once the available resources are scarcer. Therefore, the embedding factor will increase with the number of VN requests. We can also state that the efficiency of the heuristic group, in general, is very low, lower than 50%. With respect to the VNE-NLF group, it has a good efficiency, being in most of the cases higher than 85%; the efficiency of the VNE-NLF-WSDP is closer to 100% which means that, on average, this method provisions the same amount of resources as requested per VN.

4.8.3.5 VN Embedding Time

An important aspect of all the VN embedding methods is the time that they require to embed, on average, a VN request and how it varies with respect to the different loads on the physical infrastructure, i.e. the VN request rate.

Figure 4.9 shows the solving time for each method as a function of the number of VN requests per 100 time units.

Before analyzing the figure, one must consider five different aspects: i) all methods have been simulated using the same machine; ii) the time to embed a VN strongly depends on the physical characteristics (e.g. CPU) of that machine; iii) the time to embed a VN strongly depends on the nature of the embedding method (i.e. a mathematical algorithm will take just a few milliseconds, while linear programming is expected to take hundreds of milliseconds); iv) two different linear programming tools (GLPK was used to solve the MIP of G-MCF, R-ViNE, D-ViNE,D-ViNE-SP, D-ViNE-LB; and CPLEX® to solve the ILP of the VNE-NLF); v) methods R-ViNE, D-ViNE,D-ViNE-SP, and D-ViNE-LB perform two linear programming operations, i.e. one for the mapping of the virtual nodes, and another for the mapping of the links.

The fourth aspect, although important for the solving time, will not interfere with the curve behaviour, e.g. polynomial or exponential, since the same method, i.e. branch and cut,

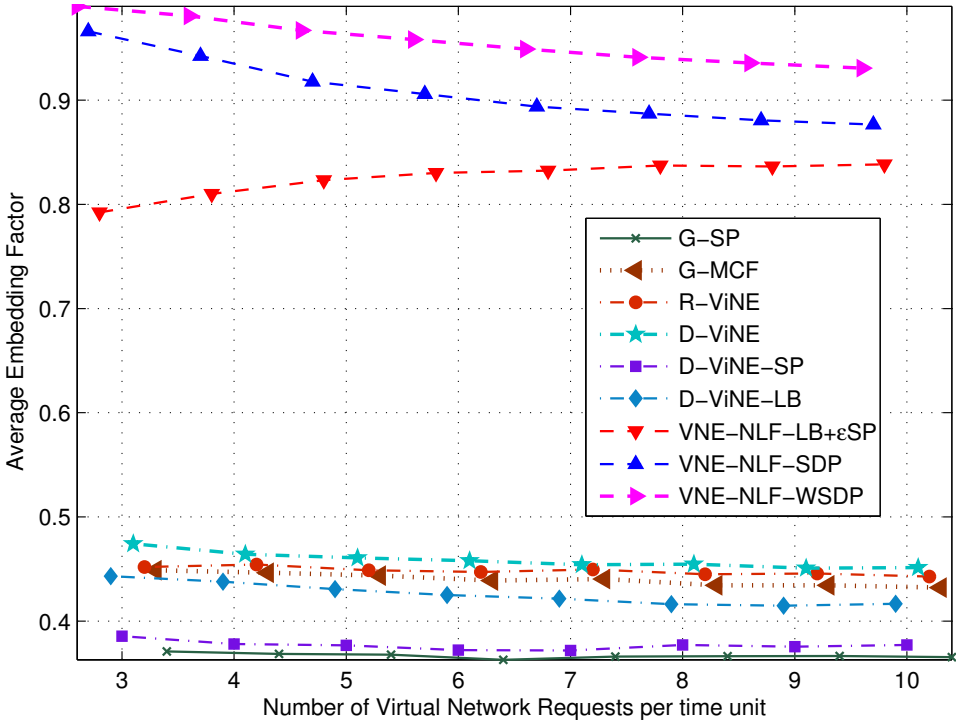


Figure 4.8: Average Embedding Factor as a function of VN Request rate.

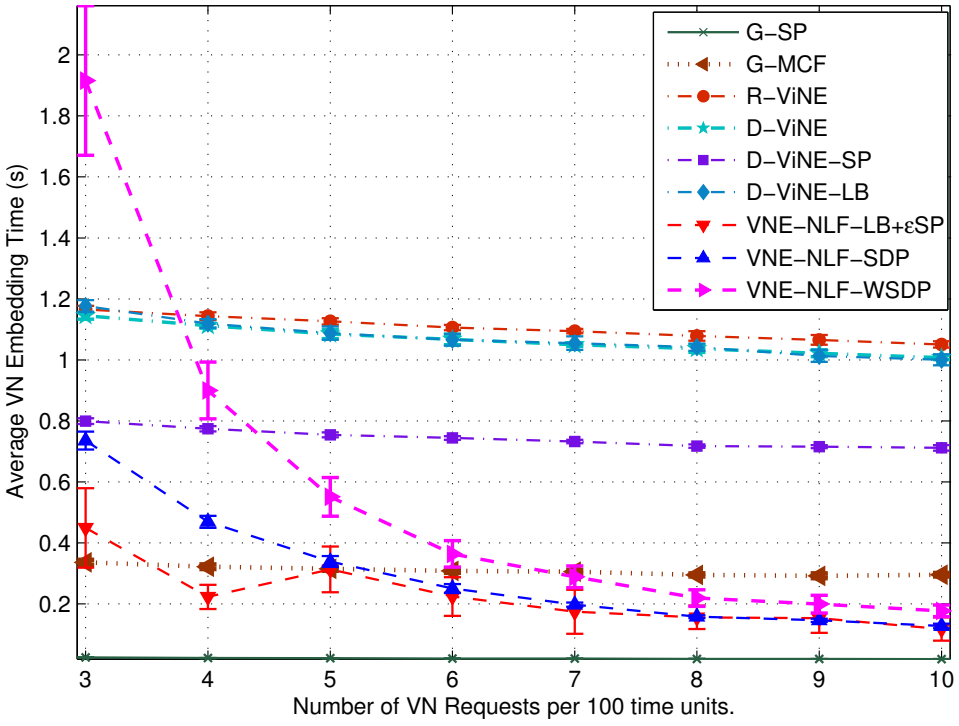


Figure 4.9: VN Solving Time as a function of VN Request rate.

is applied by both solvers (i.e. GLPK and CPLEX) to solve the VN embedding problem.

From the figure, we can observe two types of behaviours: the method VNE-NLF with three different costs functions shows a decaying behaviour with the VN request rate; for the remaining methods we observe a nearly constant behaviour.

For the first behaviour, i.e. method VNE-NLF with three different cost functions, one should take into consideration that the VN request acceptance ratio is considerably higher, e.g. 90% until 6 VN requests: more than one mapping solution per VN request is expected to exist; therefore, the optimization process takes place and will consume the majority of the solving time to obtain the optimal solution.

For the remaining methods, the VN request acceptance ratio is lower and below 70%: usually there is not more than one mapping solution per VN request, on average, which significantly reduces the solving time. This is the case of the methods G-MCF, R-ViNE, D-ViNE, D-ViNE-SP and D-ViNE-LB.

We can also add that the methods R-ViNE, D-ViNE, D-ViNE-SP, and D-ViNE-LB take twice the time on average to embed a VN compared to G-MCF. This is related with the number of MIP problems solved per VN embedding. The latter only considers one MIP problem per VN embedding, while the former ones consider two MIP problems.

The method that performs the embedding in the shortest time has the poorest performance (the G-SP), which solves each VN request embedding problem in an average of 20 milliseconds.

The method that requires the longest time to perform the embedding for the case of 3 VN requests has the highest performance (the VNE-NLF), using the WSDP cost function, which takes less than 2 seconds on average for that case. However, if we increase the load, the situation significantly changes, and the methods R-ViNE, D-ViNE and D-ViNE-LB take more time to obtain the embedding solution. On average, they take 1 second to embed a VN, while the VNE-NLF-WSDP consumes less than 200ms.

4.8.4 Impact of the Maximum Distance Between Virtual Nodes

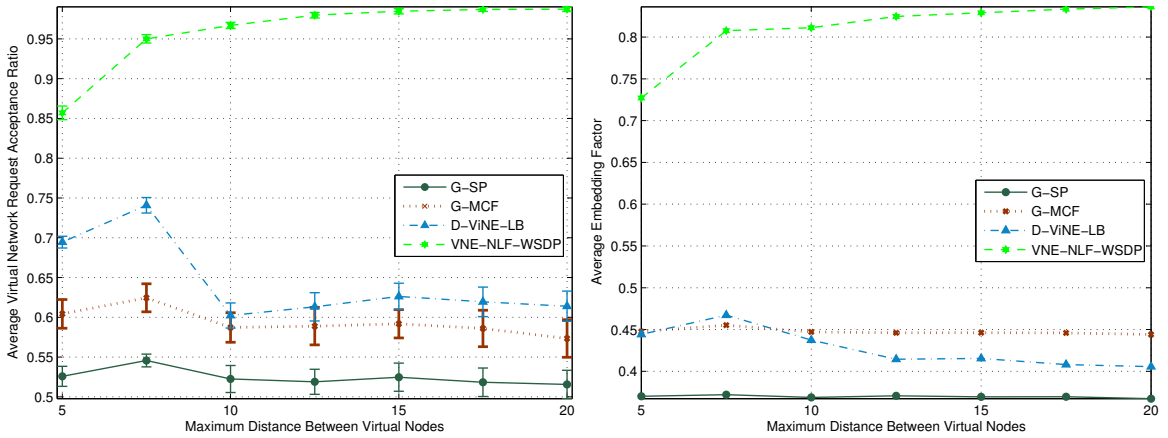
To evaluate the impact on the overall performance of the different embedding methods due to the restriction on the maximum allowed distance between virtual nodes represented in equation (4.28), a new set of simulation experiments was performed. The VN request arrival rate was fixed to 4 VN requests per 100 time units; the maximum distance between virtual nodes was set to vary between 5 and 20 within intervals of 2.5 distance units; for each considered value of maximum distance, the same set of VN requests was used. The remaining parameters i.e., virtual network size, link probability, and number of nodes were maintained. To increase the readability of all figures, only the best method of each group is presented in this section: G-SP (heuristic), G-MCF (mixed integer programming - link-path), D-ViNE-LB (mixed integer programming with better node-link embedding coordination - link-path), and VNE-NLF-WSDP (integer linear programming - node-link).

4.8.4.1 VN Request Acceptance Ratio

Figure 4.10a depicts the VN request acceptance ratio as a function of the distance between virtual nodes. Two different behaviours can be observed:

i The acceptance ratio increases with the distance between virtual nodes: this is the case of the VNE-NLF group. This behaviour is expected if we consider the cases where VN requests were initially not mapped due to the distance constraint; increasing the permitted distance between virtual nodes will in principle result in more accepted VNs.

ii Increasing the distance between virtual nodes decreases the acceptance ratio: this is the case of the assignment methods G-SP, G-MCF, D-ViNE-LB.



(a) Average VN Acceptance Ratio.

(b) Average Embedding Factor.

Figure 4.10: Resource allocation evaluation as a function of the distance between virtual nodes.

4.8.4.2 Embedding Factor

The embedding factor is depicted in Figure 4.10b, where three distinct behaviours can be observed:

i The embedding factor slightly decreases with the maximum distance: this is the case of methods G-SP and G-MCF, where these demonstrate to lose efficiency with the considered distance.

ii The embedding factor does not vary with the maximum distance: this is the case of method D-ViNE-LB.

iii The embedding factor increases with the maximum distance: this is the case of the VNE-NLF group. This demonstrates that the VNE-NLF group is able to be more efficient with the relaxation on the distance constraint equation (4.28).

4.8.5 Re-Optimization Evaluation

This sub-section presents the simulation results obtained using the optimization method VNE-NLF, and its extension for re-optimization of virtual network embedding. Our evaluation is primary focused on the minimum and maximum resource utilization on the CPU and on the memory, and on the average bandwidth utilization.

Table 4.3: Physical Network and Virtual Network Parameters.

Parameters	**Physical Network**	**Virtual Network**
N. CPUs	{2; 4; 6}	{1; 2; 3; 4 }
CPU Frequency (GHz)	{2.0 to 3.2 in 0.2 steps }	{2.0 to 2.6 in 0.1 steps }
RAM Memory (GB)	{2; 4; 6; 8}	{64; 128; 256; 512 }
Link Bandwidth (Mbps)	{500}	{2.048; 8.448; 34.368}

The simulation parameters applied here are the same as in 4.8.2, except for the physical and the VN network specifications which are presented on Table 4.3, and the VN request

arrival rate that is started with 0.8 VN requests per time unit, and increases by 0.2 VN request, up to 1.8 requests. Regarding the re-optimization process, the results were obtained using a virtual machine configured with 4 cores (Intel Xeon X5650@2.67GHz) and the CPLEX was set to use up to 4 threads, the relative gap tolerance was set to 0.05 (i.e. feasible integer solution proved to be within percent of optimal), and a time limit of 24 hours was used in order to avoid long time simulations. However, most of the re-optimizations were performed within the 24 hours' time-frame.

The maximum and minimum CPU utilization are depicted in Figure 4.11a. We can observe that the re-optimization process clearly reduces the maximum CPU utilization. The gain is higher when the physical network is not loaded and lowers when the network is almost fully loaded. The re-optimization achieves values 20% lower for a maximum CPU utilization of 0.8 VN request per time unit, and 5% lower for 1.8 VN request per time unit. We can observe also that the re-optimization process increases the minimum CPU utilization, where it achieves values 25% larger for minimum CPU utilization. The variation on the number of VN requests does not seem to affect substantially the gap between the two embedding processes.

The maximum and minimum memory utilization is shown in Figure 4.11b. The same behaviour, as for the maximum CPU utilization or minimum CPU utilization can be observed, where the re-optimization process reduces significantly the maximum memory utilization. The re-optimization achieves values 16% lower for maximum memory utilization of 0.8 VN request per time unit, and 3% lower for 1.8 VN request per time unit. We can also observe that the minimum memory utilization with the re-optimization process increases, where it achieves values 25% higher for memory utilization.

Figure 4.11c shows the average bandwidth utilization as a function of the number of virtual network requests. We can observe that applying the re-optimization reduces significantly the average bandwidth utilization on the physical links. This gain is even higher with the increase on the number of VN requests, reaching values 17.5% lower for 1.8 VN requests per time unit.

4.8.6 Energy-Aware Evaluation

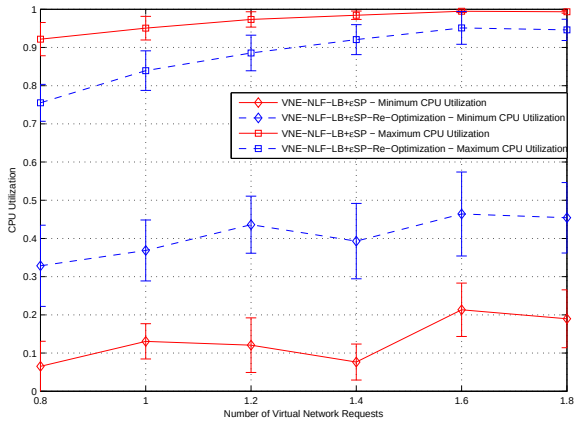
In this sub-section, we evaluate the optimisation with respect to the energy consumption. The evaluated cost functions are briefly summarized in Table 4.4.

The value of P_b is set to 175 units of power, while the value of P_l is set to 75 units of power[3]. According to the previous values, θ_b is set to 0.7, while θ_l is set to 0.3.

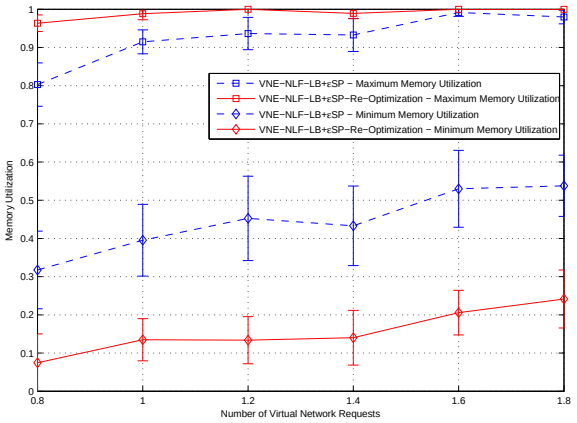
Table 4.4: Compared VN Embedding Methods - Energy Evaluation.

Notation	Method Description
VNE-NLF-WSDP	Virtual Network Embedding - Node-Link Formulation - Weighted Shortest Distance Path Minimization.
VNE-NLF-BCM	Virtual Network Embedding - Node-Link Formulation - Bandwidth Consumption Minimization
VNE-NLF-ECM	Virtual Network Embedding - Node-Link Formulation - Energy Consumption Minimization

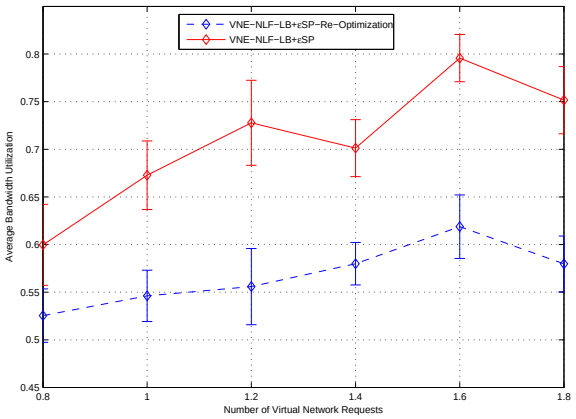
[3]These values were also considered by the authors of [BH09].



(a) Maximum and Minimum CPU Utilization



(b) Maximum and Minimum Memory Utilization



(c) Average Bandwidth Utilization

Figure 4.11: Re-optimization evaluation as a function of VN request rate.

4.8.6.1 Physical Nodes Active

Figure 4.12a depicts the average percentage of physical nodes in the active state per VN request rate. The number of active nodes increases with the rate of VN requests, with a long-term trend to reach 100%. The cost function VNE-NLF-WSDP has almost all nodes in the active state after a VN request rate of 6. The cost functions VNE-NLF-BCM and VNE-NLF-ECM have reduced the number of active nodes by 15.0% (and 3.5%) and 36.2% (and 5.8%) for a VN request rate of 3 (and 10), when compared to VNE-NLF-WSDP, respectively. These values are obtained using the overall number of physical active nodes and not the percentage, as provided in the figure.

4.8.6.2 Physical Links Active

Figure 4.12b depicts the percentage of physical active links per VN request rate. From the figure, we can observe that the percentage of active links increases with the rate of VN requests. All the objective functions proposed have reduced the percentage of active links. We also observe that the objective function VNE-NLF-ECM, aiming at minimizing energy consumption, does not always achieve the lowest percentage of active links. This occurs for a rate higher or equal to 7 VN requests, where the objective function VNE-NLF-BCM achieves the same percentage of active physical links. Although this function does not consider any energy parameters in its formulation, it implicitly reduces the number of active links by

directly minimizing the bandwidth consumption. The cost function VNE-NLF-ECM has reduced the number of active links by 26.9% (12.2%) for a VN request rate of 3 (10), when compared to VNE-NLF-WSDP, respectively.

4.8.6.3 Physical Network Energy Consumption

Figure 4.12c depicts the average physical network energy consumption as a function of the VN request rate. From the figure, we can observe that energy consumption increases with the VN request rate for all objective functions evaluated, which corroborates the results obtained on Figure 4.12a: if more VNs are being allocated per time unit, more physical resources need to be activated. We must also notice that the function VNE-NLF-BCM reduces the energy consumption; this makes sense, since this function aims at minimizing the physical bandwidth allocation, hence minimizing the overall number of forwarding nodes allocated. The cost function VNE-NLF-BCM and VNE-NLF-ECM have reduced the physical network energy consumption by 14.4% (3.1%) and 31.4% (3.6%) for a VN request rate of 3 (10), when compared to VNE-NLF-WSDP, respectively.

4.8.6.4 VN Energy Consumption

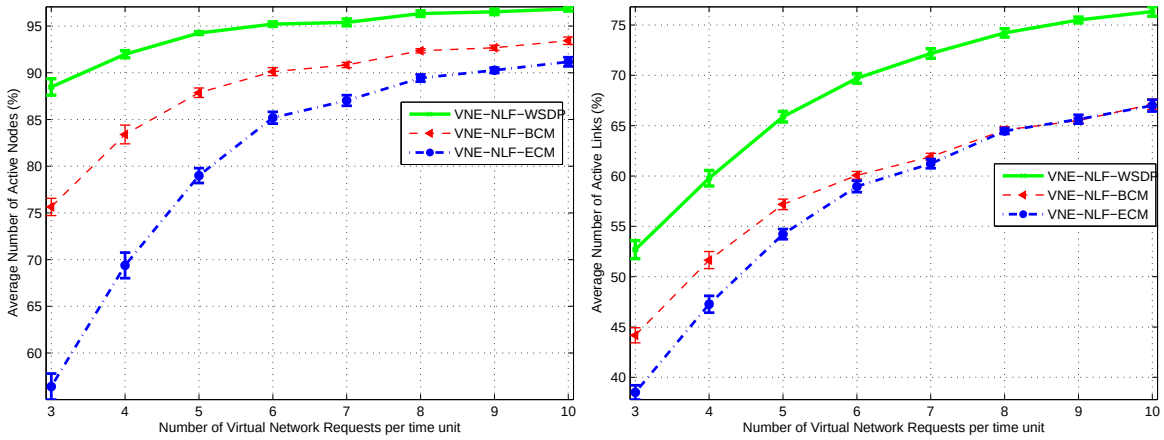
Figure 4.12d depicts the average VN energy consumption as a function of the VN request rate. Here we observe the same behaviour for all methods - the energy consumption decreases with the rate of the VN requests, and tends to a specific bound. If we consider the result obtained on Figure 4.12a, we can assume that this bound is mostly given by the second term of equation (4.17), whereas the transition between power states (i.e. power-off and power-on) is less frequent for a higher VN request rate.

The cost function VNE-NLF-BCM, which aims at minimizing the bandwidth consumption, slightly increases the energy consumption per VN allocation, when compared to VNE-NLF-WSDP. If we have in mind Figure 4.12a, we can state that the first term of equation (4.17) has a minimal impact on the objective function VNE-NLF-WSDP per VN request, since the majority of the physical nodes are already powered-up. Moreover, the cost function VNE-NLF-BCM, despite minimizing the overall number of forwarding nodes allocated, it does not minimize the number of power-up nodes per VN request, therefore requiring more energy per VN request.

The cost function VNE-NLF-ECM, which aims at minimizing the energy consumption, results in the lowest energy consumption per VN allocation, and reduces the energy consumption per VN allocation by 50.9% (6.4%) for a VN request rate of 3 (and 10), when compared to VNE-NLF-WSDP.

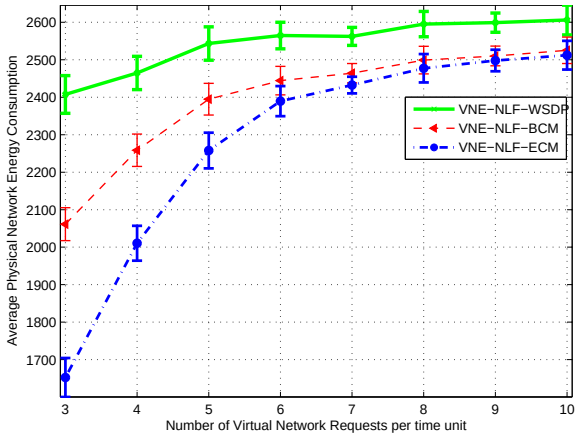
4.8.7 Virtual Network Migration Evaluation

We assume that a VN re-embedding request is triggered by a physical node maintenance event (or eventually imminent failure), according to a Poisson process of 1 event per 500 time units. A set of physical nodes that need to be power-off for maintenance is randomly generated, i.e. integers uniformly distributed between 1 and 50. We have considered sets of 1, 2 and 3 physical nodes, which correspond to a shutdown of 2%, 4%, and 6% of the physical network resources. The CPU and bandwidth capacity of the nodes and links affected is set to zero. All the simulations were performed using an Intel® Core ™ Processor i5-3210M @2.5GHz, and the time consumed per set of VNs re-embedding is registered.

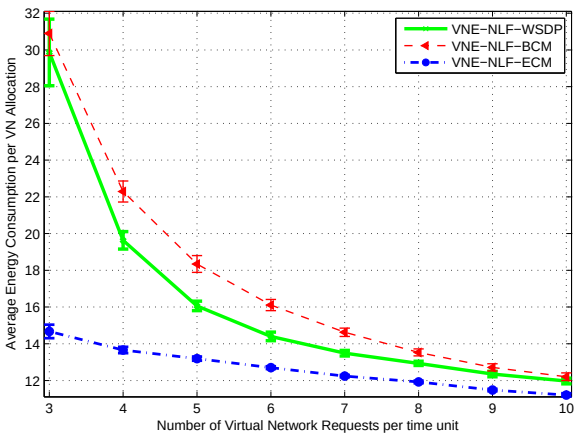


(a) Average Percentage of Physical Nodes Active.

(b) Average Percentage of Physical Links Active.



(c) Average Physical Network Energy Consumption.



(d) Average Virtual Network Energy Consumption.

Figure 4.12: Energy evaluation as a function of VN request rate.

4.8.7.1 Physical Network Resilience Factor

Figure 4.13a depicts the average physical network resilience factor as a function of the VN request rate and the percentage of physical resources shutdown. The results show that the network resilience decreases with the VN request rate and also with the percentage of resources shutdown. On one hand, with a higher VN request rate, the number of VN requests allocated in the physical network increases. On the other hand, by turning off more physical resources, the number of VNs affected by the decrease of the amount of free resources does not increase. Therefore, it penalizes the VN resilience factor. Noteworthy, the VNRE-NLF is able to re-embed all sets of VNs affected for a VN request rate of 3, and for all considered percentage of physical resources shutdown.

4.8.7.2 Virtual Nodes Migration

Virtual node migration is the component with the highest impact in the VN migration execution time (see Figure 3.11). The virtual node migration process not only consumes additional CPU of the source node and destination node, but also requires extra physical bandwidth to transfer the virtual node between the physical nodes. Figure 4.13b presents the average percentage of virtual nodes migrated per set of VNs affected and as a function

of the VN request rate. The virtual nodes previously assigned to physical nodes that need to be turned off do not count to the overall percentage of virtual nodes migrated, since their migration to new physical nodes is mandatory. Therefore, the percentage of virtual nodes migrated only incorporates the virtual nodes placed on other physical nodes, and that need to be migrated to make the assignment problem feasible. The number of nodes migrated increases with the VN request rate and with the percentage of physical resources shutdown. By increasing the number of VNs allocated on the physical network, i.e. VN request rate, and the percentage of physical resources shutdown, the spare capacity of the physical network is reduced. Hence, it is required to move more virtual nodes to make the re-embedding feasible.

4.8.7.3 Embedding Factor

Re-embedding sets of VNs is important to minimize the number of virtual nodes migrated, and also to re-embed the virtual links in the smallest set of physical links possible, to save bandwidth for the incoming VN requests. Figure 4.13c shows the embedding factor for each set of VNs re-embedded and as a function of the VN request rate. The embedding factor decreases with the VN request rate and with the percentage of physical resources shutdown. By increasing the number of VN request rate and the percentage of physical resources shutdown, we are not only reducing the spare capacity of the physical network, but also making the VN re-embedding less efficient in terms of virtual link re-assignment (i.e. less physical bandwidth is available to make it possible to re-embed the overall virtual links in the smallest amount of physical links possible). Consequently, the network has virtual links consuming more than one physical link.

4.8.7.4 Physical Bandwidth Allocation

It is also important to know how much bandwidth is additionally provisioned for each set of VNs re-embedded. Figure 4.13d depicts the average percentage of additional bandwidth allocation per set of VNs re-embedded and as a function of the VN request rate. The additional bandwidth increases with the VN request rate, and also with the percentage of physical resources shutdown. By increasing the number of VN request rate and the percentage of physical resources shutdown, we are implicitly increasing the percentage of virtual nodes migrated (as stated in sub-section 4.8.7.2); if we aim at minimizing the number of virtual nodes migrated, we will sacrifice the virtual links (i.e. longer paths are taken by the virtual links). Therefore, it is consumed an additional bandwidth per VN request rate and percentage of physical resources shutdown.

4.8.7.5 VN Re-embedding Time

An important aspect of a VN re-embedding method is the time that it requires to re-embed, on average, a set of VNs, and how it varies with respect to the different loads on the physical infrastructure, i.e. VN request rate, and with the different percentage of physical resources shutdown. Figure 4.13e shows the solving time for each set of VNs and as a function of the VN request rate. The solving time increases significantly with the VN request rate, and also with the percentage of physical resources shutdown. The solving time directly depends on the number of virtual links and virtual nodes that need to be re-embedded. By increasing the VN request rate, we are increasing the number of VNs allocated and implicitly the number of VNs potentially affected by a physical resource shutdown. By increasing the percentage of physical resources shutdown, we are directly increasing the number of VNs affected. Therefore, by increasing the number of VNs affected, we are increasing the number of virtual nodes and links to be re-embedded and implicitly increasing the solving time.

4.9 Summary

This chapter proposed the VNE-NLF to solve the VN embedding problem. The model applies optimization theory to simultaneously embed the virtual nodes and the virtual links.

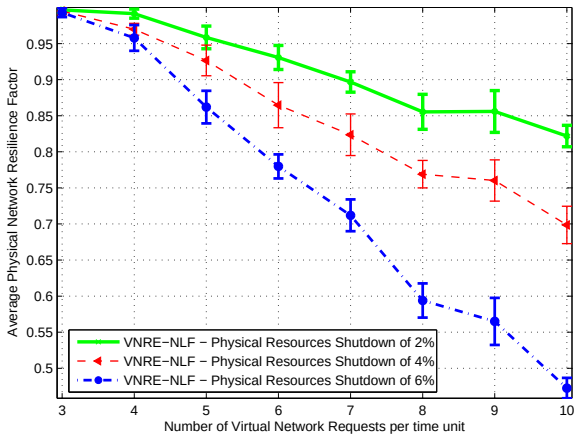
Three cost functions are proposed: the LB+ϵSP which aims to minimize the overall load on the network per VN embedding; the SDP which aims to minimize the number of physical links consumed, and at the same time it chooses physical nodes with higher availability of resources; and the WSDP which includes the demanded capacity by the VN in the objective function.

Moreover, two new objective functions are also proposed to optimise the energy consumption: BCM, which aims to minimize the bandwidth consumption, and ECM, which aims to minimize the energy consumption.

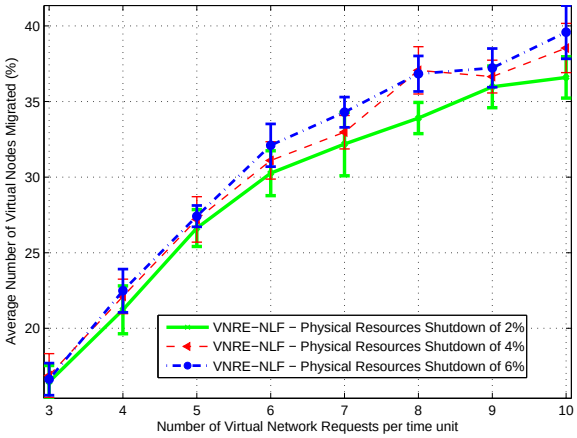
Additionally, it is proposed the VNRE-NLF to solve the online virtual network re-embedding problem as a simultaneous optimization of virtual nodes and virtual links, providing the optimal bound for each set of virtual networks migrated. This approach aims at minimizing the overall VN migration cost per re-embedding: i) number of virtual nodes migrated; ii) physical bandwidth consumption. Simulation experiments show how far the state of the art heuristics are from an ILP based optimization method. The difference between the performance of the heuristics and the VNE-NLF approach is, at least, 30% for the VN request acceptance ratio (see Figure 4.3).

The results also showed that, not only the minimization of the bandwidth allocation is important for the VN embedding, but also the load balancing has a significant impact. The minimization of the bandwidth allocation positively affects the energy consumption, and the consideration of both the bandwidth minimization allocation and the CPU load on the objective function provides a good VN acceptance ratio.

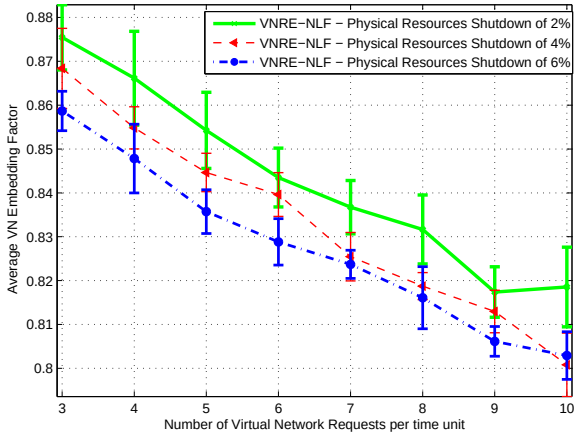
The results also show that the virtual network resilience to migration events is directly affected by the VN request rate and by the percentage of physical resources shutdown. One can also conclude that it is not only important to have enough spare capacity to re-accommodate the virtual nodes and virtual links affected by the physical resource shutdown, but also to have additional capacity to accommodate virtual link re-assignments and virtual node migrations performed to make the VN re-embedding problem feasible.



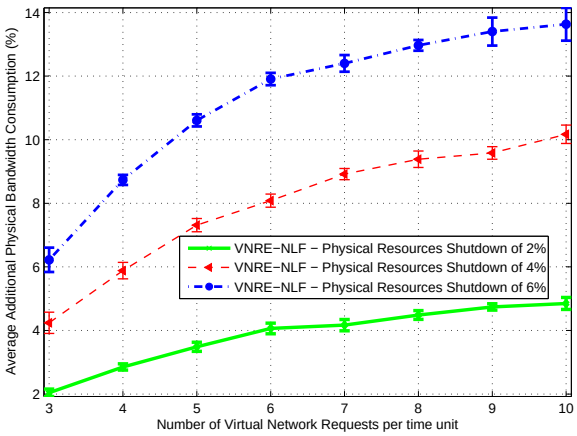
(a) Average Physical Network Resilience Factor.



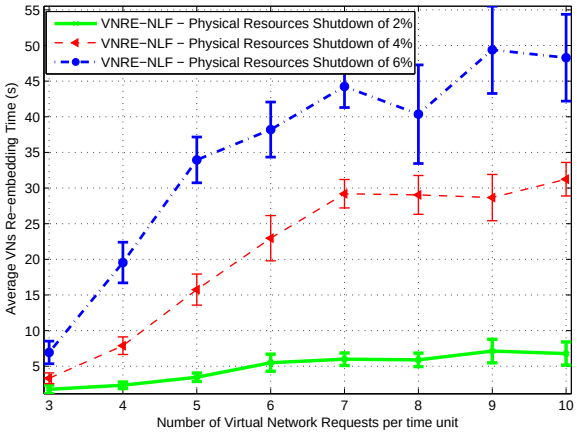
(b) Average Percentage of Virtual Nodes Migrated.



(c) Average Embedding Factor.



(d) Average Additional Physical Bandwidth.



(e) Average VN Re-embedding Time.

Figure 4.13: VN migration evaluation as a function of VN request rate.

Chapter 5 - Conclusions & Future Work

"Do not go where the path may lead, go instead where there is no path and leave a trail."

—Ralph Waldo Emerson

With such a wide problem space and a comparable solution spectrum, it is important to keep the proposed contributions in perspective, as well as understanding what can be done as follow-up work on the different aspects of this Thesis. This final chapter presents the most important conclusions on the explored topics. Guidelines for network operators are provided. Future research directions are recommended.

Network virtualisation is seen as a promising approach to overcome the so-called "Internet impasse" and bring innovation back into the Internet by allowing easier migration towards novel networking approaches, as well as the coexistence of complementary network architectures on a shared infrastructure and in a commercial context.

In this Thesis, a set of research challenges were addressed: How will the operator manage and control the virtual resources? How to efficiently embed the virtual network requests onto the physical network? How to seamlessly move the virtual networks across the physical network?

5.1 Results and Achievements

In this section, we provide the major results of this PhD Thesis aligned with the previous research challenges.

5.1.1 RAMC Framework

To fully operate and manage virtual networks, a network operator needs to have means in his possession to perform resource allocation, monitoring and control. The RAMC framework was developed to address the allocation and management of virtual networks. This framework was implemented and evaluated in a small-scale testbed as a proof of concept, demonstrating this way the feasibility to deploy virtual networks on commodity hardware.

5.1.2 Virtual Network Migration

The VN migration process is an important mechanism both for network management purposes and for the conception and deployment of new network architectures and protocols based on network virtualisation. The VN migration enables the virtual network to not be physically tied to a set of hosts previously assigned, and it makes it viable to move components of a VN or even the entire VN from one set of physical hosts to a new set on a seamless way and on-demand.

In this Thesis, we presented the VN clone migration as an alternative to the live migration approach. The proposed approach achieves no VN downtime, and it takes just a few seconds to be fully performed.

Similar research works [WKB+08, PFC+10] have addressed this issue by modifying the control of the VR. Nonetheless, these approaches are dependent on the networking protocols which are configured on the VR. If we consider the scenario were VNs are being leased by Infrastructure Providers to Virtual network Operators, it would not be viable to change the control plane of the VR according to the networking protocols configured inside of the virtual network.

5.1.3 Virtual Network Embedding

The RAMC framework allows the network operator to manage and control virtual networks. However, mechanisms are required to efficiently embed on-demand virtual networks onto the substrate network. This Thesis not only proposed a heuristic algorithm to solve the VNE problem, but also proposed a mathematical formulation to solve the VNE problem as an optimization problem, and allowing this way to increase the potential revenue of the InP by accepting more VN requests.

To endorse the efficient embedding requirement, three cost functions were proposed and evaluated to optimise the resource allocation and load balancing. This Thesis also presented

how far the existing heuristic algorithms are from the optimal bound, and it is also proven the feasibility to apply this method on the VNE problem.

Additionally, to simultaneously optimise the resource allocation and energy consumption, two cost functions are proposed and also evaluated. It was demonstrated that it was possible to save energy without affecting the resource allocation.

5.1.4 Virtual Network Re-Embedding

This Thesis also proposed the VNRE-NLF to solve the online virtual network re-embedding problem as a simultaneous optimization of virtual nodes and virtual links, providing the optimal bound for each set of virtual networks migrated. The virtual networks proved to be highly resilient to failures on the physical network.

5.2 Operator Recommendations

The decoupling of networks and infrastructure (and consequently the roles of network provider and infrastructure provider) has a potentially disruptive effect on operators' business. For incumbent operators, this brings new opportunities, already analysed before, but also new threats. Easier establishment of networks will enable increased competition from new entrants; in addition, the economic attractiveness of providing network infrastructure, which basically becomes a commodity, may be questionable in many scenarios.

Important lessons about running virtual networks should be learned from the experience with MPLS VPNs, which most operators have been developing for more than 10 years. Although VPNs should be seen as a limited form of network virtualisation, several problem spaces, such as scalability, reliability and security, face relatively similar problems.

Standardization will surely play a key role to enable interoperability between different vendors and, not less importantly, between network operators. Network virtualisation-ready commercial products from major vendors have been launched. Standardization is therefore essential to guarantee interoperability and avoid vendor lock-in. In addition, a clear regulatory environment must exist to clearly define the roles of InPs, VNOs and VNPs, and also how these players will interact.

The potential advantages of network virtualisation for operators are clear, in several business scenarios and use cases: to partition network infrastructure into multiple logical domains, to segregate different types of services (e.g. Voice Over IP (VoIP), IPTV, Internet), or to enable coexistence of different network technologies (e.g. IPv4, IPv6).

5.3 Future Research Directions

This section explores areas for future work. The potential of network virtualisation shall be analysed together with other relevant emergent trends, namely cloud computing and the convergence of IT and networking. The combination of these ongoing transformations is likely to raise an immense field of challenges to the scientific community.

One of these challenges relates to the efficient embedding of network resources and IT resources. In this PhD Thesis, we have only focused on solving the VNE problem with network constraints. Now, we shall add to the equation another type of node (i.e. server node) and new requirements may advent from it, e.g. storage requirement, OS, CPU Architecture.

Another challenge relates with the virtual link negotiation between the InP domain and the data centre domain. A distributed mechanism must be in place to self-assign virtual link identifiers between those two domains.

On the other hand, it will necessary to research on how to deal with multi-domain scenarios: e.g. multi-networks, multi-clouds, or even a combination of both, and also multi-operators. Additionally, with the emerging of cloud computing new business models are arising that need further research: e.g. infra-structure as a service, network as a service, security as a service, or even entity as service.

Moreover, it is necessary to integrate the concept of virtual networks in wireless networks. Note that these networks currently have an increasing importance to telecommunications operators, and therefore they have to engage the possibility of creating virtual networks in these environments for commercial use.

Another important open research topic relates with Software Defined Networking (SDN). The SDN architecture decouples the network control and forwarding functions enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services. It will of utmost interest to investigate on how SDN and Network Virtualisation (NV) can be combined together to bring innovation, and to alleviate the current Internet limitations.

Finally, it is necessary to investigate on to how to extend the NV of nodes and links, embraced in this PhD Thesis, to functions of the network (NFV), such as NAT, firewall, intrusion detection, DNS, caching.

Appendices

Appendix A - Network Virtualisation from an Operator Perspective

Márcio Melo, Susana Sargento, and Jorge Carapinha

in *Proceedings Conferência sobre Redes de Computadores,* Oeiras, Portugal, October, 2009.

The format has been revised.

Network Virtualisation from an Operator Perspective

Márcio Melo, Susana Sargento, and Jorge Carapinha

Abstract

Network virtualisation has been hailed by the research community as a tool to enable a smooth transition to the Future Internet, by allowing the coexistence of different architectures and protocols over the same network infrastructure. Recently, the interest from the operators and industry mainstream in network virtualisation has grown quite significantly, as the potential benefits of virtualisation, both from an economical and an operational point of view, become clear. So far, the concept has been mainly a research topic and has been materialized in small-scale testbeds and research network environments. The challenges posed by the deployment of virtualisation in operator networks are still largely unknown and require urgent study. In this paper, we present the 4WARD architecture for network virtualisation and, based on this architecture, we propose a framework for network resource control in virtualisation-based network environments. We also present the developed virtual network testbed and the first results of isolation between different virtual networks.

A.1 Introduction

Network Virtualisation has gained an increasing prominence in networking and telecommunications fields in the last few years. Initially, the interest in network virtualisation was mainly pushed by Future Internet research initiatives [PACR03, APST05b, FGR07b, ZZRR08], mainly with the objective to find a platform on which novel Internet architectures could be experimented and evaluated without limitations or constraints, namely those associated with the traditional IP model. Later on, it became clear that virtualisation could constitute a key component of next-generation Internet architecture itself [TWEF03], and not just as a mere platform for experimentation. Perhaps more importantly for network operators, it also became clear that network virtualisation could provide a number of short/medium term business advantages, with potential reduction of costs and increase of revenues, as an interesting tool from an operational point of view.

However, the large-scale deployment of network virtualisation by commercial operators faces a number of challenges, most of which have not been fully evaluated up to now. One of the key issues with network virtualisation is the management and control of network infrastructure resources - more specifically, how to map, or embed, virtual resources into physical substrate resources. Although this problem has been already addressed in the literature [ZA06, LT06, RAL03], the proposed solutions have been mainly oriented to research network environments, overlooking key constraints and requirements of commercial operator networks.

Network virtualisation has followed the usual development cycle, which started with research and testbed experimentation through a number of research initiatives. Validation for deployment in commercial operator networks is still largely unaccomplished and represents a logical continuation of the research efforts so far. This paper aims at providing a contribution in this direction. Our main focus is the management and control of network infrastructure resources in a network virtualisation environment. We present both the concept and an experimental testbed that entails the management and control of virtual networks in an operator perspective. The starting point of this paper is the network virtualisation architecture and business model developed in the framework of the 4WARD project [4WA10a].

The paper is organized as follows. Section A.2 provides a general overview of network virtualisation. The 4WARD network virtualisation business model and roles are briefly explained. In particular, the interface between the virtual network provider and the infrastructure provider is analysed in some detail. Section A.3 examines the problems of resource control in a network virtualisation environment, mainly from the point of view of the infrastructure provider, and proposes a solution for resource negotiation and control. Section A.4 briefly describes a small-scale testbed, currently under development, with a view to future testing and validation of this architecture. Finally, section A.5 concludes with a summary and possible directions for future work.

A.2 Network Virtualisation Overview

A.2.1 Historical Perspective

In general, virtualisation provides an abstraction between a user and a physical resource, in such a way that the user gets the illusion of direct interaction with that physical resource. In the last few years, significant advances in operating system virtualisation technologies have enabled the use of virtualisation in a growing number of contexts and applications. Products like XEN [XEN14] and VMware [vmw14], which enable multiple operating systems to co-

exist securely within a single machine, are now widely used, both for server consolidation and on desktop machines. Virtualisation gained strong attention, especially with the widespread deployment of server virtualisation in data centres, enabling organizations to optimize performance of large computing infrastructures. Unsurprisingly, the networking community embraced the concept of virtualisation. In reality, the concept of network virtualisation is not new – in the late 1990s, network-based IP/MPLS/VPNs [RR06] materialized the concept of building multiple virtual IP networks over a common large scale network infrastructure. Later on, the concept was extended to layer 2 technologies, such as Ethernet, through services like VPWS and VPLS. However, all these incarnations of the concept were tightly bound to a specific protocol, either layer 3 (e.g. IP) or layer 2 (e.g. Ethernet). On the other hand, at the node level, virtualisation was elusive, in the sense that it was basically just a separation of addressing spaces, but not a real separation of network resources. At that time, "full-blown" virtualisation was not seen as a major requirement, and this limitation did not prevent VPNs from being a remarkable market success. A somewhat related concept is that of overlay network, which can be loosely, defined as a network built on top of one or more existing, networks. Overlay networks have been often used to improve specific Internet characteristics, such as routing, and to deploy new features, such as multicast, or Peer-to-Peer (P2P) applications. The advantage of the overlay network approach is that deployment can be incremental, thus circumventing the need to massively upgrade all nodes in the network. But again, because of the limited visibility to the underlying network characteristics, overlays are usually ineffective or sub-optimal. Network virtualisation, as it is viewed in this paper, supersedes all the above variants and is based on two fundamental building blocks: node virtualisation and link virtualisation. The main advantages of network virtualisation realized by the industry to make network virtualisation potential going beyond the one of future Internet scenarios, mainly to operators and service providers on a short/medium term, are as follows [Jun09, Cis09a]:

- Reduction of costs: by using a single virtualized infrastructure to run multiple services, CAPEX and OPEX can be reduced, compared to the typical scenario where different types of service (e.g. voice, data, broadcast) are run in separate networks.

- Increase of revenues: by sharing infrastructure, the network operator achieves a better utilization of the network resources and optimizes profitability.

- Flexible network planning: the swift and easy establishment of virtual networks can be used as a safeguard against unpredictability of the service demand.

- Security and isolation: virtualisation can provide real isolation of network resources, with benefits in terms of fault isolation, security, and performance guarantees.

- Flexibility and programmability: virtual networks can be tuned to fulfil specific service and application requirements (e.g. security, performance, dependability), thus a "one-size-fits-all" approach is no longer required.

A.2.2 Network Virtualisation Business Models and Roles

From a business model point of view, a very significant impact of network virtualisation is the ability to cleanly decouple infrastructure from services, which has been pursued for a long time but never really accomplished. This potential separation of infrastructure and services paves the way for the creation of new business models and roles. Network virtualisation can be deployed in a number of very different scenarios and business models but, in general,

it is based on three distinct roles, as defined by the 4WARD project, and represented in Figure A.1:

1. The Infrastructure Provider (InP) deploys and runs the network physical resources, and partitions them into isolated virtual resources using some virtualisation technology. These resources are typically offered to virtual network operators and not to end users (but the customer of the InP might as well be a corporation using the virtual network for its internal use, rather than to build commercial end user services). The InP has visibility into what resources are leased to each virtual network (VN), but not into the protocols running inside.

2. The virtual network provider (VNP) is responsible to find and compose the adequate set of virtual resources from one or more infrastructure providers, in order to fulfil the virtual network operator request. The VNP leases slices of the virtualised infrastructure from one or more InPs and puts them together. In reality, what the VNP provides is not a network, but just an empty container where the virtual network operator builds the protocols that will make the VN to come alive. The role of the VNP is particularly important in scenarios where multiple InP domains are involved, but may be redundant in the case where a VN is limited to a single network infrastructure domain.

3. In each isolated network partition, the virtual network operator (VNO) is, in principle, free to deploy any protocol stack and network architecture, independently of the underlying network infrastructure technology. The VNO operates, maintains, controls and manages the virtual network. From a functional viewpoint, the role of the VNO should be indistinguishable from that of any operator running a native network infrastructure. Ideally, the fact that resources are virtual, rather than physical, should not imply any major impact from an operational point of view. VNOs have a unified view of the network, regardless of the multiple infrastructure domains on which it is built.

It should be noted that this model does not preclude the possibility of more than one role being played by a single entity. In a vertically integrated scenario, the three roles would be typically played by the same operator. Yet, even in this case, a functional separation of roles based on the model above should make sense.

A.2.3 The VNP-InP interface

The VNP-InP interface is a key aspect of the network virtualisation architecture. Through this interface, the VNP is able to request the establishment, modification or removal of virtual networks (supposedly, as a result of a corresponding request from the VNO). In its turn, the InP is supposed to acknowledge the VNP requests and notify any relevant event (e.g. a network error). The split of responsibilities and the information flow between the VNP and the InP are therefore of the utmost importance. Ultimately, this will depend on the information flowing through the VNP/InP interface in both directions, as illustrated in Figure A.2. In principle, one of two basic approaches could be taken:

1. The InP announces the resources which are available to be leased by VNPs, i.e. the internal structure of the InP infrastructure (or a virtual representation thereof) and the current state of resources. The InP is supposed to publish this information in some way, e.g. by means of a specific notice-board, such as proposed in [PJ06]. It is up to the VNP to pick one or multiple InPs amongst all possible candidates, that would be able to provide the required resources, while fulfilling any applicable constraints (e.g. performance guarantees, cost). Since the relationship between the VNP and the InP

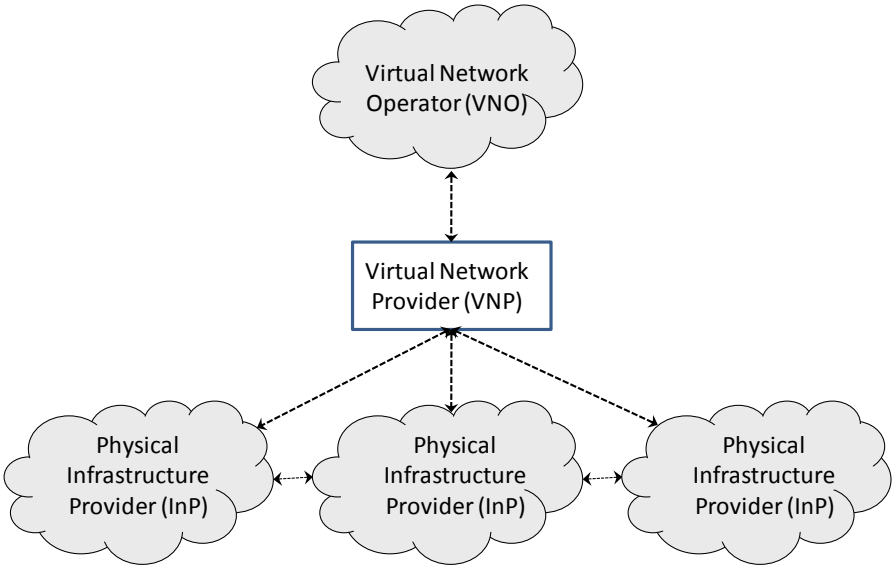


Figure A.1: Network virtualisation business roles.

is quite straightforward, the complexity of the VNP-InP interface is quite low in this case. This approach is appropriate for research testbeds, or whenever there is a trust relationship between the InP and all potential VNPs; in a commercial environment this is not likely to be the case, except perhaps in a vertically integrated scenario, in which VNP and InP have a common business affiliation.

2. The InP exposes a minimal set of resources, namely the points of presence (PoPs) and hides the internal structure and the state of resources. Because the VNP does not know in advance whether the InP is able to fulfil its request, the virtual network characteristics have to be provided to the InP and a negotiation has to be carried out through the VNP/InP interface prior to VN establishment phase, when the resources are actually reserved. In turn, the InP decides whether the request can be accommodated in the physical resources and, if so, maps virtual nodes into substrate nodes and finds the substrate path between every pair of directly connected virtual nodes. This is likely to be the approach followed in a commercial environment, where a relationship of trust between VNPs and InPs is not expected.

As stated before, in this paper we are mainly interested in commercial network environments; therefore, in the following sections we are mainly focused on the second approach.

A.3 Controlling Virtual Network Resources

This section presents an architecture for automatic virtual network creation and the corresponding approach for control of virtual network resources. It comprises the main building blocks of the network and their functionalities, and the communication required to provide the virtual network creation.

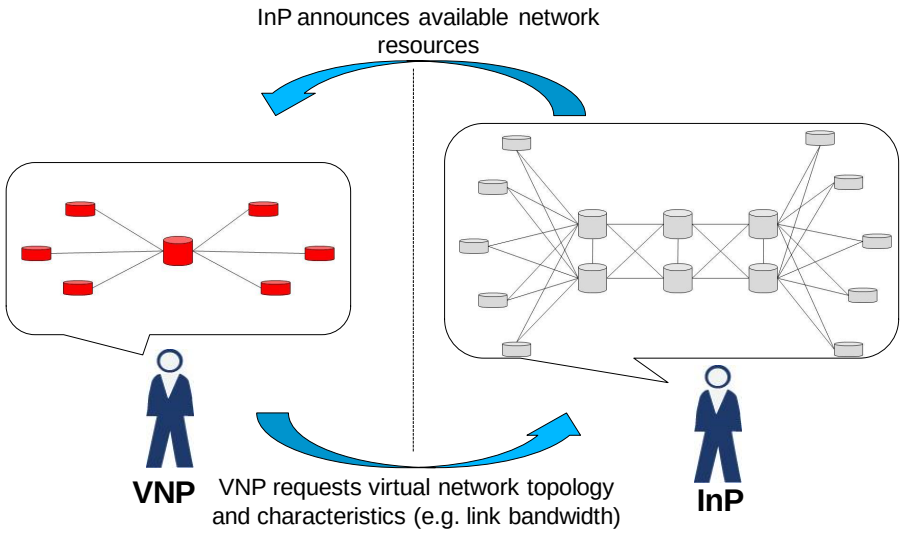


Figure A.2: VNP-InP Information Flow.

A.3.1 Building blocks

Prior to the creation of a new virtual network, the InP should find the adequate physical resources, taking into account the current state of the network and the level of occupancy of the network resources, at that moment, in the case of an "on-the-fly" reservation, or at the requested future time, in the case of an advance reservation. Several theoretical approaches have been proposed to handle this problem [ZA06, LT06, RAL03]. However, these solutions are mainly oriented to small-scale networks or research testbeds, and do not take into account the constraints that usually apply in a commercial environment. In practice, the mapping of virtual nodes into physical nodes is often constrained by physical location, in which case the selection of the physical node to associate with a specific virtual node is fixed, or limited to a small set of choices. This is the case of edge nodes, or PoPs, which for economical reasons are supposed to be physically close to customers or end-users. Typically, at least one virtual node should be located in each geographic area (e.g. city, region) where the service is to be deployed. By contrast, for other types of virtual nodes, physical location is not relevant from the VNP point of view – this is usually the case with core nodes, with no direct connection to end users. The mapping of virtual nodes and links into physical nodes and links should follow a set of constraints and optimization criteria to be defined by the InP (e.g. minimum cost, resource load balance, segregation of resources according to the service type), and can be materialized in a complex algorithm. Physical resource control and virtual resource embedding include three basic components (Figure A.3):

1. VN admission control: the InP verifies whether there are available resources to fulfil the virtual network request made by the VNP, and decides whether it can be accepted, or not. VN admission control does not necessarily find an optimal solution for a VN yet – this is supposed to be the role of resource mapping, as described below – it only verifies that, at least, one solution can be found.

2. Resource mapping: the InP identifies the set of possible substrate nodes and links to host the requested virtual network and selects the optimal solution.

3. Re-optimization: the network state keeps changing as new VNs are setup and others are torn down, or as a result of node or link failure conditions. This often leads to inefficient utilization of resources, in which some parts of the network infrastructure (either link resources or node resources) become excessively loaded, while others are under-utilized. Therefore, the capability to re-optimize the allocation of virtual resources across the substrate network without traffic disruption, either on a periodic basis or triggered by a specific event (e.g. when a specific resource availability threshold has been reached) is a key VN requirement. In addition, the resource management process typically makes use of two auxiliary components:

   (a) Discovery: this function is responsible for discovering network resources and providing them available for the admission control and mapping functions.

   (b) Monitoring: this function collects real-time information from nodes and links, and signals any significant deviation from the expected network behaviour.

A.3.2 VN Setup Negotiation Process

As explained before, it is likely that in most cases the VNP has limited knowledge of the physical resources provided by the InP. On the other hand, there will be multiple candidate InPs to provide the required network resources in many cases. Therefore, the VNP must be able to inquire a set of candidate InPs and, based on their responses, select one or more that will actually provide the network resources simultaneously and cooperatively. This requires the VNP/InP negotiation to be divided in two stages, as depicted in Figure A.4:

1. Query: the VNP inquires the InP about the availability of resources to build a specific VN. The InP is expected to provide a yes/no reply, possibly with additional information, e.g. cost, QoS parameters.

2. Commit: the VNP requests the reservation of network resources and the InP enforces the corresponding resource reservation, after establishing the mapping between virtual and physical resources. It should be noted that virtual networks can be created "on-the-fly", i.e. just before the resource is required, or in advance, i.e. at some future point in time. In either case, a time may be optionally specified for resources to be released; otherwise, the VN will only be torn down through explicit signalling.

From the InP point of view, a relevant issue is how to map the blocks represented in Figure A.3 into these two phases. The right hand side of Figure A.4 suggests a possible approach, but this will be further discussed in the next section. The VNP is expected to build the virtual network topology and define resource capacity requirements, namely link bandwidth and node computational capability. As discussed previously, other characteristics such as geographical location of the edge nodes will be needed in most cases. The information provided by the VNP to the InP must contain a model of the virtual network topology as a graph, with a set a virtual nodes and virtual links and including the applicable constraints (e.g. link bandwidth, node computational capacity, physical location). Each virtual node and virtual link must be characterized by a number of parameters. A tentative list of parameters to characterise virtual networks, nodes, and links is shown in Table A.1.

A.3.3 Signalling and Control

As explained earlier the creation of a VN involves two phases, query and commit. Figure A.4 depicts the VN creation process: the left hand side represents the message flow

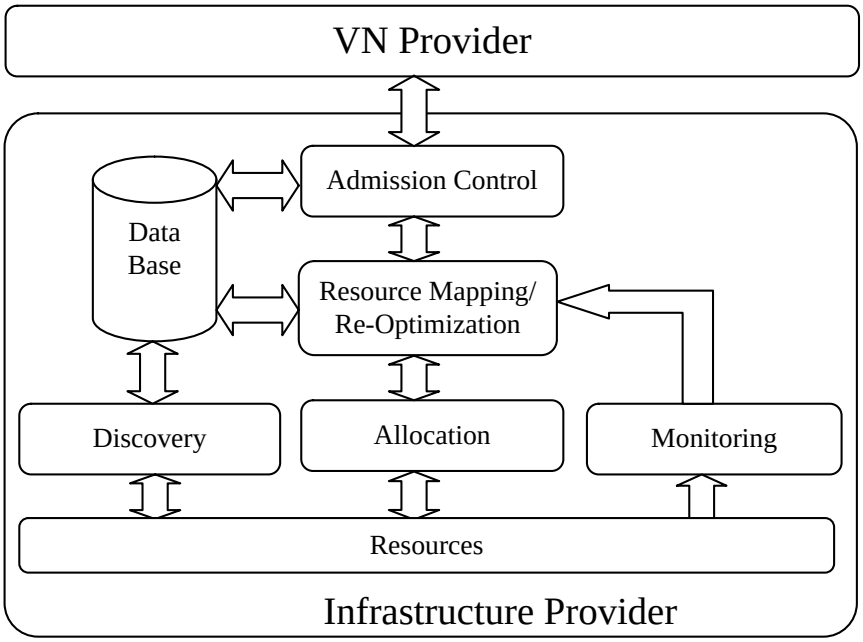


Figure A.3: InP block diagram

between the VNO and the VNP when a new VN is requested. In this example, the VNP contacts two candidate InPs to accommodate that VN, InP X and InP Y, and then decides to opt for InP X, based on some criterion, e.g. InP X provides the requested resources at lower cost than InP Y. Then, the process continues with the commit phase, in which the resources are actually reserved.

A.3.3.1 Query Phase

The process is started when the VNO sends a VN Request to the VNP, including the VN topology and its constraints, node constrains (i.e. physical location, CPU) and link constrains (i.e. bandwidth, delay). The VNP is then in charge of assigning a VN ID and an ID for each virtual resource. Then a ***VN_Query.request*** message is sent to one or more InPs. This message must contain the VN ID, the nodes/links IDs, the VN topology and its specifications, according to Table A.1. At a first stage, the InP will perform a VN Admission Control, checking that every requested virtual node and virtual link can be accommodated by at least one substrate node and substrate link, respectively. If one or more virtual nodes and/or virtual links cannot be accommodated due to lack of resources in the substrate (i.e. insufficient bandwidth or computational resources), the process should be stopped here and the ***VN_Query.response*** is sent to the VNP, indicating that the request cannot be fulfilled (optionally, indicating the reason of the failure). Otherwise, if every virtual resource can be accommodated by the substrate, then a ***VN_Query.response*** message with a positive reply, including the VN ID, should be sent to the VNP. It should be noted that the VN Admission Control is not expected to find the optimal solution for the virtual-to-physical resource mapping problem yet, but only whether at least one solution exists. This is understandable, since the resource mapping is the most complex step of this process, and in

Table A.1: Virtual Network Characteristics.

Network virtualisation components	**Parameters**
Virtual network	Virtual network ID Start time of the service End time of the service Class of service / reliability Preemption level
Virtual node	Node ID Physical location Physical node ID CPU Memory Storage
Virtual link	End points Traffic characteristics Reliability level QoS isolation level

many cases, the query will not be followed by a corresponding commit. However, this may not be the case in all circumstances, and the InP may decide to go further than just performing admission control. So, optionally, at a second step, the InP may perform a pre-reservation by mapping the VN into the network infrastructure, making use of an optimization algorithm, knowing a priori that every requested virtual resource can be accommodated. The choice of the first available or optimal solution may be based on different criteria, such as: preferring substrate nodes with more plentiful resources, selecting substrate links with more available bandwidth, link aggregation (i.e. virtual link that maps into 2 substrate links) and link segmentation (i.e. virtual link spanning through multiple substrate links). After obtaining the best solution, the InP must perform a reservation (at this stage, at logical level only) of the concerned substrate resources. This reservation will be cancelled if a specific timeout expires without any effective reservation being made by a corresponding ***VN_Commit.request*** from the VNP, and the reserved resources will be released again. Optionally, this timeout should be included in the ***VN_Query.response*** message.

A.3.3.2 Commit Phase

If the VNP receives one or more positive responses from the candidate InPs in the query phase, the process will typically continue with the selection of the InP (if more than one candidate InP answered positively), followed by a ***VN_Commit.request***, with the corresponding VN ID. After receiving the ***VN_Commit.request***, the InP verifies whether a pre-reservation exists for the given VN ID and if it is still valid. If so, it proceeds with the allocation of the virtual resources. After allocating each virtual resource, it sends a ***VN_Commit.response***, including the VN ID and the ID of each virtual resource. If the VN ID is not valid or the pre-reservation has expired, or if for some reason it cannot allocate any particular virtual resource, the InP should send a ***VN_Commit.response*** indicating the reason of the negative response. If a pre-reservation has not been performed beforehand, then the complete process has to be executed. A potential issue in this case is that, because no resources have been reserved, it may be the case that when the commit request arrives, the resources are no longer available. Thus, from the point of view of the InP, there is a trade-off between increasing the complexity of the query phase and improving the reliability of the whole process.

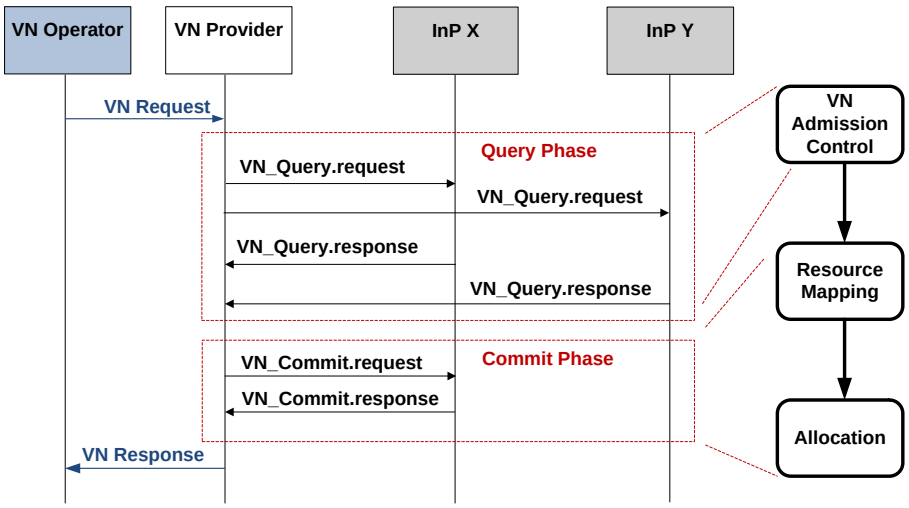


Figure A.4: VN creation sequence chart and flow diagram.

A.4 A Virtual Network Control Testbed

To demonstrate the concept of network virtualisation, it was implemented a small scale testbed with 7 substrate nodes, 1 server, 4 routers with no routing protocol running and 2 clients. Figure A.5 depicts the implemented testbed. On top of the substrate network, 2 VNs were created manually, VN 1 and VN 2; XEN hypervisor [XEN14] was used for the creation of the Virtual machines. Regarding that some virtual machines will act as routers, for instance VRouter1-1, the Extensible Open Router Platform (XORP) [xor14] was installed in each of them (i.e. virtual router) and the Open Shortest Path First (OSPF) protocol was configured as a routing protocol. In the virtual servers, a video stream will be activated through VLC media player [vlc14], and the virtual servers will stream a video across the VNs; VLC was also installed in the clients. To enable the virtual links, VLANs [VLA14] were configured in each substrate link, VLAN2 for VN1 and VLAN3 for VN2. The VNs are similar in terms of topology, and the virtual resources belonging to VN1 have the same IP address as the corresponding resources in VN2. Our experiment starts with virtual server (VServer) 1 and 2 streaming two different videos. With the OSPF protocol, stream 1 is forced to use Virtual Router1-2 (VRouter) by giving lower port cost to it in VRouter1-1, and the stream 2 is forced to use VRouter2-2. The clients will immediately start receiving the corresponding stream. After a while, the connection between Router1 and Router2 is broken, causing the breakdown of the corresponding virtual links, between VRouter1-1 and VRouter1-2 and between VRouter2-1 and VRouter2-2. Therefore, stream 1 will be interrupted, and stream 2 will continue without problems since it is not using that connection. After a couple of seconds, stream 1 will be restored and will be using the virtual link between VRouter1-1 and VRouter1-3 and between VRouter1-3 and VRouter1-4 in a different virtual network from the stream 2. The recovery time is just due to OSPF converging process. Stream 1 will have no influence in stream 2: the measured round trip time of the packets in stream 2 did not change with the recovery of stream 1. Both streams will be using the same path but in different virtual networks. This demonstrates the isolation between VN1 and VN2 in terms of performance and IP addressing.

This testbed will be the basis for more complex experiments in the future, such as resource

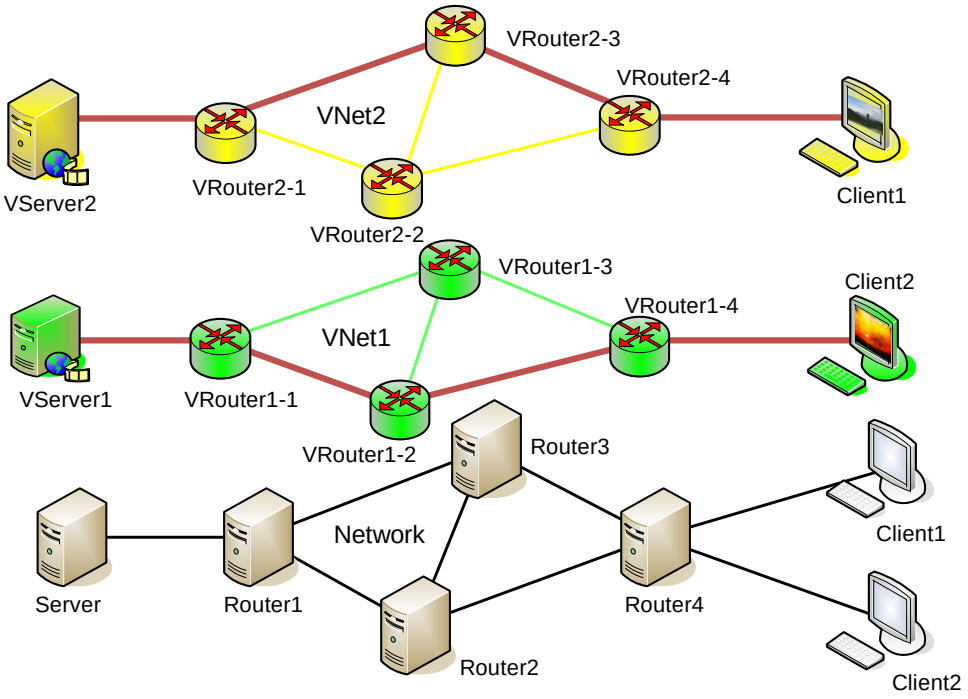


Figure A.5: Network Virtualisation Testbed

management algorithms, and the support of automatically controlled VNs.

A.5 Conclusion and Future Work

A new Architecture for Network Virtualisation based on the 4WARD project has been presented: it promotes the deployment of new protocols and enables the emergence of new players in the telecommunications market. A new role has been defined, the one of the Virtual Network Provider; this element will trigger the competition and/or cooperation among different Infrastructure Providers to provide substrate resources. The Virtual Network creation process has been explained, including the several processes of admission control, resource mapping and allocation. The entities responsible for each process were also described, and a special focus has been included in the internal composition of the Infrastructure Provider. As can be seen from the testbed description, the base testbed is running with manual intervention for virtual networks control. Currently, we are working on the deployment of a management and control solution for network virtualisation that will enable the automatic creation and deletion of Virtual Networks. Moreover, we will also work on an optimal algorithm for resource management between different virtual networks, taking into account the Infrastructure Provider constraints.

Appendix B - Virtual Network Mapping - An Optimization Problem

Márcio Melo, Jorge Carapinha, Susana Sargento, Luis Torres, Tran Phuong Nga, Andreas Timm-Giel, and Ulrich Killat

in *Mobile Networks and Management Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering,* vol. 97, pp 187-200, 2012.

The format has been revised.

Virtual Network Mapping - An Optimization Problem

Márcio Melo, Jorge Carapinha, Susana Sargento, Luis Torres, Tran Phuong Nga, Andreas Timm-Giel, and Ulrich Killat

Abstract

Network Virtualisation is acclaimed to be a key component for the Future Internet by enabling the coexistence of heterogeneous (virtual) networks on the same physical infrastructure, providing the dynamic creation and support of different networks with different paradigms and mechanisms in the same physical network. A major challenge in the dynamic provision of virtual networks resides on the efficient embedding of virtual resources into physical ones. Since this problem is known to be $\mathcal{NP}$-hard, previous research focused on designing heuristic-based algorithms; most of them do not consider a simultaneous optimization of the node and the link mapping, leading to non optimal solutions. This paper proposes an integer linear programming formulation to solve the Virtual Network embedding problem, as a simultaneous optimization of virtual nodes and links placement, providing the optimal boundary for each VN mapping. A $\boldsymbol{link-node}$ formulation is used and the multi-commodity flow constrain is applied. In addition, a heuristic algorithm for VN embedding is also proposed and compared against the optimal formulation. The performance of the ILP formulation and of the heuristic are evaluated by means of simulation. Simulation experiments show significant improvements of the VN acceptance ratio, on average additional 10% of the VN requests are accepted when using the ILP formulation, which corresponds, on average, to more 7 virtual networks accommodated on the physical network.

B.1 Introduction

Network Virtualisation has gained an increasing prominence in networking and telecommunications fields in the last few years. Initially, the interest in network virtualisation was mainly pushed by Future Internet research initiatives [PACR03, APST05b, FGR07b, ZZRR08], mainly with the objective to find a platform on which novel Internet architectures could be experimented and evaluated without limitations or constraints, namely those associated with the traditional IP model. Later on, it became clear that virtualisation could constitute a key component of next-generation Internet architecture itself [TWEF03], and not just as a mere platform for experimentation. Perhaps more importantly for network operators, it also became clear that network virtualisation could provide a number of short/medium term business advantages, with potential reduction of costs and increase of revenues, as an interesting tool from an operational point of view [CJ09, MSC09]. Although there is a large interest on virtualized networks both from the research community and network operators, several challenges still prevent it from being deployed on real environments [CB09]. One of the major obstacles lies in the efficient embedding[1] of a virtual network onto a physical network. Since this process requires the simultaneous optimization of virtual nodes and links placement, it is complex in nature and requires large amounts of computing power. Some authors, such as [ZA06, LK09, LT06, YYRC08, CRB09, FBCB10, NMCS11a], have already proposed solutions to this problem, mostly based on heuristic approaches, but have failed to provide the optimal solution for each VN mapping.

In this paper we propose a linear integer programming formulation to solve the virtual network assignment problem and to provide the optimal boundary for each VN embedding. The formulation supports heterogeneous virtual and substrate networks. In addition, we propose an heuristic algorithm based on [NMCS11a] to solve the VN assignment problem. Simulation experiments show significant improvements of the VN acceptance ratio: on average more 10% of the VN requests are accepted which corresponds to 7 more embedded virtual networks on the physical network. The paper starts with the discussion of the related work on existent mapping algorithms B.2. Section B.3 describes the network embedding problem, specifies the ILP model and shortly summarizes the enhancements proposed to a mapping heuristic based on [NMCS11a]. Section B.4 analyzes the performance of both the ILP optimization model and corresponding heuristic, and section B.5 concludes the paper and describes the future work.

B.2 Related Work

This simultaneous nodes' and links' mapping optimization can be formulated as an unsplittable flow problem [ZA06, And02], known to be $\mathcal{NP}$*-hard*, and therefore, it is only tractable for a small amount of nodes and links. In order to solve this problem, several approaches have been suggested, mostly considering the *off-line* version of the problem where the VN requests are fully known in advance.

In [LK09] a backtracking method based on subgraph isomorphism was proposed; it considers the on-line version of the mapping problem, where the VN requests are not known in advance, and proposes a single stage approach where nodes and links are mapped simultaneously, taking constraints into consideration at each step of the mapping. When a bad mapping decision is detected, a backtrack to the previous valid mapping decision is made, avoiding a costly re-map.

[1] The terms embedding, mapping and assignment are used interchangeably in this paper

The work in [LT06] defines a set of premises about the virtual topology, i.e. the backbone nodes are star-connected and the access-nodes connect to a single backbone node. Based on these premises, an iterative algorithm is run, with different steps for core and access mapping. However, the algorithm can only work for specific topologies.

A distributed algorithm was studied in [HLZ08]. It considers that the virtual topologies can be decomposed in hub-and-spoke clusters and each cluster can be mapped independently, therefore reducing the complexity of the full virtual network mapping. This proposal has lower performance and scalability, when compared with centralized approaches.

Zhu et al. [ZA06] proposed a heuristic, centralized, algorithm for dealing with VN embedding. The goal of the algorithm is to maintain a low and balanced stress of both nodes and links of the substrate network. However, the stress of nodes and links do not consider heterogeneity on their characteristics.

Yu et al. [YYRC08] propose an embedding algorithm which considers finite resources on the physical network, and enables path splitting (i.e. virtual link composed by different paths) and link migration (i.e. to change the underlying mapping) during the embedding process. However, this level of freedom can lead to a level of fragmentation that is infeasible to manage on large scale networks. In [CRB09], it was taken a formal approach to solve the on-line VN embedding problem using a mixed integer programming formulation in a two-step approach. This approach, despite providing a better coordinated node and link mapping, it does not solve the VN assignment problem as an overall, and does not support heterogeneity of nodes.

Butt et al. [FBCB10] proposed a topology awareness heuristic for VN embedding and also suggest some algorithms to avoid bottlenecks on the physical infrastructure, where they consider virtual node reallocation and link reassigning for this purpose. Nogueira et al. [NMCS11a] proposed a heuristic that takes into account the heterogeneity of the VNs and also of the physical infrastructure. The algorithm is evaluated by means of simulation and also on a small scale testbed, where it achieves mapping times of the order of tens of milliseconds.

Although all these algorithms provide a solution for the virtual network mapping problem, most of them fail to provide the optimal boundary for each VN mapping. Also, some of them fail to solve the assignment problem as a simultaneous optimization of the virtual node placement and of the virtual link placement, which lead to non optimal solutions.

B.3 Problem Description and ILP Model Formulation

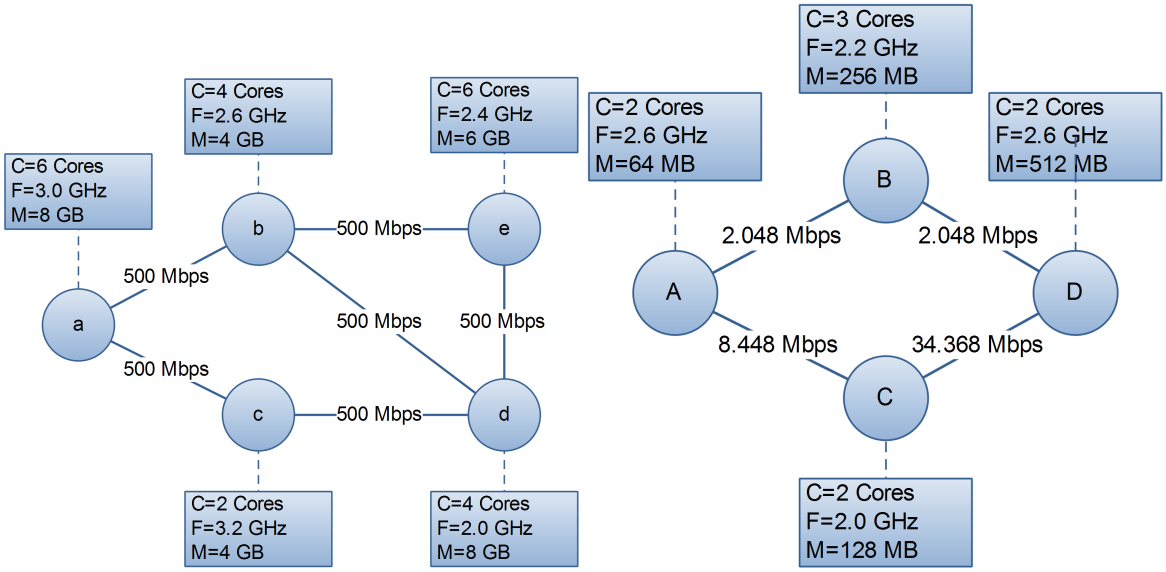
In this section, we start with the description of the VN assignment problem. The ILP model formulation is then presented, followed by a proposal of enhancement of a heuristic based on [NMCS11a].

B.3.1 Virtual Network Assignment Problem Description

First, we start with the convention used for the index notation: we use $\boldsymbol{i, j}$ for nodes and links in the physical network, and $\boldsymbol{m, n}$ for nodes and links in the virtual network.

We consider that we have a physical network with a given number of nodes, N, and with a random topology, as depicted in figure B.1a. Each node is described by the number of CPU, which correspond to letter C in the Figure, the clock CPU frequency, F, and by the RAM amount he possesses, M. With respect to the links, we consider the bandwidth capacity, B, and we assume that each link is an unidirectional link. Virtual networks are described the same way as physical networks, as shown in Figure B.1b. We use the letter P when we want

to refer to the physical resources, e.g. $\boldsymbol{C^P}$, and the letter V is used for virtual resources, e.g. $\boldsymbol{C^V}$.



(a) Example of a substrate network topology description

(b) Example of a virtual network topology description

Figure B.1: Network Topology Description

The CPU capacity, the RAM size and the CPU frequency of the nodes is stored in a array with $\boldsymbol{N^P}$ entries, e.g. $\boldsymbol{C^P \rightarrow N^P \times 1}$.

We denote the total CPU capacity (the initial capacity) by $\boldsymbol{C^{P_{total}}}$, the non allocated capacity is denoted by $\boldsymbol{C^{P_{free}}}$, and the used capacity is denoted $\boldsymbol{C^{P_{used}}}$, where $\boldsymbol{C^{P_{total}} = C^{P_{used}} + C^{P_{free}}}$. The same notation is used for the RAM. We use the adjacency matrix (B.1), $\boldsymbol{A^P \rightarrow N^P \times N^P}$, to describe the connectivity of the physical network, and (B.2) to describe the connectivity of the virtual network, $\boldsymbol{A^V \rightarrow N^V \times N^V}$.

$$\boldsymbol{A^P_{ij}} = \begin{cases} \mathbf{1}, \text{ the physical node } \boldsymbol{i} \text{ is neighbour of } \boldsymbol{j} \\ \mathbf{0}, \boldsymbol{else} \end{cases} \tag{B.1}$$

$$\boldsymbol{A^V_{mn}} = \begin{cases} \mathbf{1}, \text{the virtual node } \boldsymbol{m} \text{ is neighbor of } \boldsymbol{n} \\ \mathbf{0}, \boldsymbol{else} \end{cases} \tag{B.2}$$

B.3.2 Integer Linear Programming Problem Formulation

We use an ILP formulation [Wol00] to solve the embedding problem of VNs . Here we only use two assignment variables for the virtual network mapping: for the virtual nodes, we use the binary variable $\boldsymbol{x}$ shown in equation (B.3), where $\boldsymbol{x^m_i \rightarrow N^V \times N^P}$ matrix; for the virtual links, we use the binary variable $\boldsymbol{y}$ represented in equation (B.4), where $\boldsymbol{y^{mn}_{ij} \rightarrow (N^V)^2 \times (N^P)^2}$ matrix (4-dimensional).

Our objective function is represented in equation (B.5) and is divided into two parts. Our primary goal is to minimize the maximum load per physical resource and, in the case of having different mapping solutions with the same maximum utilization, the second part of the objective function is activated which will opt for the solution that consumes the less physical links.

The maximum load at each different resource, i.e. memory RAM load ($\boldsymbol{M_{load}}$), the CPU load ($\boldsymbol{C_{load}}$), and the link load ($\boldsymbol{B_{load}}$), are represented in equations (B.6),(B.7),(B.8), respectively. We multiply the CPU frequency by the CPU load in equation (B.6) and by the RAM load in equation (B.7), in order to firstly use physical nodes with lower frequency and to preserve the remaining for virtual nodes with higher frequency demands.

Equation (B.9) ensures that each virtual node is assigned and it is assigned to just one physical node, and equation (B.10) guarantees that each physical node can accommodate in maximum one virtual node per virtual network request, although each physical node can accommodate other virtual nodes from different VNs. We use equations (B.11) and (B.12) to make sure that we do not exceed the available capacity of all physical nodes, and we use equation (B.13) to guarantee that we do not violate that requirement on the CPU frequency.

In order to optimize the mapping of the virtual links and at the same time to cope with the optimization of the virtual nodes, we apply the multi-commodity flow constraint [EIS75] with a $\boldsymbol{node-link}$ formulation [PM04], and we also use the notion of direct flows on the virtual links, which are represented in equation (B.14). To ensure that we have enough bandwidth available at each physical link, we use equation (B.15).

Assignment Variables

$$\boldsymbol{x_i^m} = \begin{cases} \mathbf{1}, \text{virtual node } \boldsymbol{m} \text{ is allocated at physical node } \boldsymbol{i} \\ \mathbf{0}, \boldsymbol{else} \end{cases} \tag{B.3}$$

$$\boldsymbol{y_{ij}^{mn}} = \begin{cases} \mathbf{1}, \text{virtual link } \boldsymbol{mn} \text{ uses physical link } \boldsymbol{ij} \\ \mathbf{0}, \boldsymbol{else} \end{cases} \tag{B.4}$$

Optimization Function

$$\boldsymbol{minimize}\ \boldsymbol{C_{load}^{max}} + \boldsymbol{M_{load}^{max}} + \boldsymbol{B_{load}^{max}} + \boldsymbol{\epsilon} \times \sum_{m,n \in \boldsymbol{N^V}(m), n<m} \boldsymbol{y_{ij}^{mn}} \times \boldsymbol{B_{mn}^V} \tag{B.5}$$

Constraints

Derived from the Optimization Function

$$\forall \boldsymbol{i} : \boldsymbol{F_i^P} \times \frac{\boldsymbol{C_i^{P_{used}}} + \sum_m \boldsymbol{x_i^m} \times \boldsymbol{C_m^V}}{\boldsymbol{C_i^{P_{total}}}} \leq \boldsymbol{C_{load}^{max}} \tag{B.6}$$

$$\forall \boldsymbol{i} : \boldsymbol{F_i^P} \times \frac{\boldsymbol{M^{P_{used}}} + \sum_m \boldsymbol{x_i^m} \times \boldsymbol{M_m^V}}{\boldsymbol{M_i^{P_{total}}}} \leq \boldsymbol{M_{load}^{max}} \tag{B.7}$$

$$\forall \boldsymbol{i,j} \in \boldsymbol{N^P}(\boldsymbol{i}), \boldsymbol{i<j} : \frac{\boldsymbol{B_{ij}^{P_{used}}} + \sum_{m,n \in \boldsymbol{N^V}(m)} \boldsymbol{y_{ij}^{mn}} \times \boldsymbol{B_{mn}^V}}{\boldsymbol{B_{ij}^{P_{total}}}} \leq \boldsymbol{B_{load}^{max}} \tag{B.8}$$

Assignment of virtual nodes to physical nodes

$$\forall \boldsymbol{m} : \sum_{\boldsymbol{i}} \boldsymbol{x_i^m} = \mathbf{1} \tag{B.9}$$

One virtual node per physical node

$$\forall \boldsymbol{i} : \sum_{\boldsymbol{m}} \boldsymbol{x_i^m} \leq \mathbf{1} \tag{B.10}$$

CPU conservation

$$\forall \boldsymbol{i} : \sum_{\boldsymbol{m}} \boldsymbol{x}_{\boldsymbol{i}}^{\boldsymbol{m}} \times \boldsymbol{C}_{\boldsymbol{m}}^{\boldsymbol{V}} \leq \boldsymbol{C}_{\boldsymbol{i}}^{\boldsymbol{P}_{\boldsymbol{free}}} \tag{B.11}$$

Memory conservation

$$\forall \boldsymbol{i} : \sum_{\boldsymbol{m}} \boldsymbol{x}_{\boldsymbol{i}}^{\boldsymbol{m}} \times \boldsymbol{M}_{\boldsymbol{m}}^{\boldsymbol{V}} \leq \boldsymbol{M}_{\boldsymbol{i}}^{\boldsymbol{P}_{\boldsymbol{free}}} \tag{B.12}$$

Frequency limitation

$$\forall \boldsymbol{i} : \sum_{\boldsymbol{m}} \boldsymbol{x}_{\boldsymbol{i}}^{\boldsymbol{m}} \times \boldsymbol{F}_{\boldsymbol{m}}^{\boldsymbol{V}} \leq \boldsymbol{F}_{\boldsymbol{i}}^{\boldsymbol{P}} \tag{B.13}$$

Multi-commodity flow conservation with $\boldsymbol{link - node}$ formulation

$$\forall \boldsymbol{m}, \boldsymbol{n} \in \boldsymbol{N}^{\boldsymbol{V}}(\boldsymbol{m}), \boldsymbol{m} < \boldsymbol{n}, \forall \boldsymbol{i} : \sum_{\boldsymbol{j} \in \boldsymbol{N}^{\boldsymbol{P}}(\boldsymbol{i})} (\boldsymbol{y}_{\boldsymbol{ij}}^{\boldsymbol{mn}} - \boldsymbol{y}_{\boldsymbol{ji}}^{\boldsymbol{mn}}) = \boldsymbol{x}_{\boldsymbol{i}}^{\boldsymbol{m}} - \boldsymbol{x}_{\boldsymbol{i}}^{\boldsymbol{n}} \tag{B.14}$$

Bandwidth conservation

$$\forall \boldsymbol{i}, \boldsymbol{j} \in \boldsymbol{N}^{\boldsymbol{P}}(\boldsymbol{i}), \boldsymbol{i} < \boldsymbol{j} : \sum_{\boldsymbol{m},\boldsymbol{n} \in \boldsymbol{N}^{\boldsymbol{V}}(\boldsymbol{m}), \boldsymbol{m} < \boldsymbol{n}} \boldsymbol{B}_{\boldsymbol{mn}}^{\boldsymbol{V}} \times (\boldsymbol{y}_{\boldsymbol{ij}}^{\boldsymbol{mn}} + \boldsymbol{y}_{\boldsymbol{ji}}^{\boldsymbol{mn}}) \leq \boldsymbol{B}_{\boldsymbol{ij}}^{\boldsymbol{P}_{\boldsymbol{free}}} \tag{B.15}$$

B.3.3 Mapping Heuristic Algorithm

In this section we propose a heuristic for network embedding, based on the one from [NMCS11a]. A pseudo-code description of the mapping algorithm is shown in algorithm 3. With respect to the base algorithm, this one contains several changes. First, we used a different equation to determine the node stress, $\boldsymbol{S_N}$, which is reflected at line 14. The former equation represented in equation (B.16) tends to balance the number of virtual nodes per physical nodes, to favour nodes with lower CPU clock frequency and to reduce the combination of consumed RAM and CPU. However, we could have physical nodes with different capacities and also virtual nodes with different requirements, which do not cope well with the objective of distributing the virtual nodes per physical nodes uniformly; moreover, physical nodes could be highly loaded at the CPU and mostly free at the RAM or the opposite, which, for the equation is totally transparent as long as the combination of the two has the lower value. The equation proposed for the node stress, presented at line 14, tends to balance the use of both RAM and CPU, and to favour nodes with higher clock CPU frequency.

Lines 30 to 32 are used to tune the $\boldsymbol{link - path}$ cost, $\boldsymbol{D(u, v)}$ according to neighbours. We have set the value of $\boldsymbol{\beta}$ to 0.01, which reduces the $\boldsymbol{link - path}$ cost to virtual neighbours that have been already assigned. Lines 34 to 39 are the replacement of line 32 in [NMCS11a] used to calculate the node potential i.e., $\boldsymbol{\pi}$. Here, the node potential is the average of the minimum $\boldsymbol{link - path}$ cost to all the possible candidates to virtual neighbours, multiplied by the node stress, which is represented in line 41. Lines 50 to 54 are used to update in runtime the link stress, $\boldsymbol{S_{LS}}$, of the physical links that have been already assigned to virtual links.

$$\boldsymbol{S}_{\boldsymbol{N}_i} = \frac{\sum_{\boldsymbol{j}}^{\boldsymbol{N_V}} \sum_{\boldsymbol{n}}^{\boldsymbol{N}_{\boldsymbol{V}_j}} \boldsymbol{\Lambda}(\boldsymbol{n}_j, \boldsymbol{i})}{\boldsymbol{\epsilon} + \text{Free RAM} \cdot \text{CPU Freq} \cdot (\text{N.CPU - Load})} \tag{B.16}$$

Algorithm 3: Mapping Algorithm Pseudo-Code

```
input : Substrate (Substrate Network) , V_Request (Requested VN)
output: V_Map (Mapped Virtual Network)
foreach Link i in Substrate.Links do
    foreach VN j in Substrate.VNs do
        foreach Link k in j.Links do
            if Link k_j ⊇ Link i then
                S_LS(i) += S_LV_j(k_j) ;
            end
        end
    end
end
foreach Link i in Substrate.Links do
    S_LS(i) = Σ_j^{N_V} Σ_k^{L_{V_j}} ((S_LV_j(k_j) | k_j ⊇ i)) ;
end
foreach Node i in Substrate.Nodes do
    S_{N_i} = CPU_Freq × [(M^{P_used}/M^{P_total})^2 + (C^{P_used}/C^{P_total})^2];
    π(v) = 0 ;
end
foreach Node n in V_Request.Nodes do
    foreach Node i in Substrate.Nodes do
        if MeetsConstraints(n, i) then
            n.Candidates.Add(i) ;
        end
    end
end
foreach Node n in V_Request.Nodes do
    foreach Link k connected to n do
        ConnectedVNode=GetLinkDestination(k) ;
        foreach SourceCandidate v in n.Candidates do
            foreach DestCandidate u in ConnectedVNode.Candidates do
                D(v,u)= Cost(CSFP_Dijkstra(v,u));
                if u.Map then
                    D(v,u)=β × D(v,u);
                end
            end
            if π(v) then
                π(v) = mean[π(v), min(∀u ∈ V_C : D(v,u)] ;
            end
            else
                π(v) = min[∀u ∈ V_C : D(v,u)] ;
            end
        end
        π(v) = π(v) × S_{N_v};
    end
    n.Map = v : π(v) = min(π) ;
end
foreach Node n in V_Request.Nodes do
    V_Map.Nodes ∪ n ;
    foreach Link k connected to n do
        ConnVNode=GetLinkDestination(k) ;
        V_Map.Links ∪ CSFP_Dijkstra(n.Map,ConnVNode.Map) ;
        foreach Link i in Substrate.Links do
            if V_Map.Links then
                S_LS(i)+ = S_LV_n(k) ;
            end
        end
    end
end
```

B.4 Evaluation Results

In this section, we describe the simulation scenario and depict our major results. Our evaluation is primarily focused on the VN acceptance ratio according to different number of

Table B.1: Physical Nodes Pool Parameters.

N. CPUs	{2; 4; 6}
CPU Frequency (GHz)	{2.0 to 3.2 in 0.2 steps }
RAM Memory (GB)	{2; 4; 6; 8}
Link Bandwidth (Mbps)	{500}

Table B.2: Virtual Nodes Pool Parameters.

N. CPUs	{1; 2; 3; 4 }
CPU Frequency (GHz)	{2.0 to 2.6 in 0.1 steps }
RAM Memory (MB)	{64; 128; 256; 512 }
Link Bandwidth (Mbps)	{2.048; 8.448; 34.368}

demands, i.e. average number of VN requests per time unit, and also on how many VNs can be accommodated on the physical network using the proposed model. We also compare the proposed ILP model with the heuristic proposed in the previous section.

B.4.1 Simulation Parameters

In order to evaluate the ILP model according to different number of VNs requests per time unit, we have implemented a discrete event simulator in Matalab®.

The physical network topology was created using the Waxman random topology generation method [Wax88], with 30 substrate nodes. The recommended parameters for link probability, $\boldsymbol{\alpha = 0.4}$ and $\boldsymbol{\beta = 0.1}$, were used although some topologies did not have full connectivity, i.e. one physical node with no viable path to all the remaining nodes (e.g. a node with no links or non connected clusters). In order to circumvent this, after generating the topology, additional links were added to the nodes with fewer interfaces until total connectivity was reached. For each substrate node, a set of parameters was randomly attributed, from a pool of possible ones, using an uniform distribution, such as RAM amount, number of CPUs and CPU frequency. The physical link's bandwidth was set at a fixed bitrate. The set of parameters is presented on Table B.1.

The virtual networks were generated using the same model, although the number of virtual nodes follows a uniform distribution, from 2 to 10. After generating the virtual topology, the same set of specifications was assigned, with a uniform distribution. The virtual nodes specification pool can be observed on Table B.2.

We assume that VN requests arrive according to a Poisson distribution and that each VN has an associated lifetime with an average of $\boldsymbol{\mu = 100}$, following an exponential distribution. Regarding the average number of VN requests per time unit, we have started with 0.8 VN requests per time unit and we have increased by intervals of 0.2 until reaching 1.8. For each different demand, i.e. value of $1/\ \boldsymbol{\lambda}$, 10 trials were performed. A new set of VNs requests and a new physical network topology were generated for each trial and for each value of $1/\ \boldsymbol{\lambda}$. All simulations were set to run until 1000 time units. A confidence interval of 95% is used for every result presented below.

We have used CPLEX®[cpl12] version 11 to solve the linear programming problem, and a time limit of 600 seconds was defined for each VN mapping, although most VNs were embedded in hundred of milliseconds.

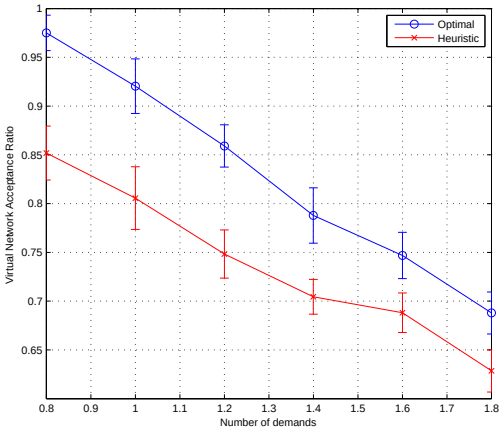
B.4.2 Simulation Results

We use several performance metrics to evaluate the optimal model and heuristic. We measure the acceptance ratio and the number of accommodated VNs as a function of the number of requests. We also measure the average memory RAM and CPU utilization on the nodes, and the average bandwidth utilization on the links. In all these cases, we plot the performance metrics as function of the number of VN requests per time unit.

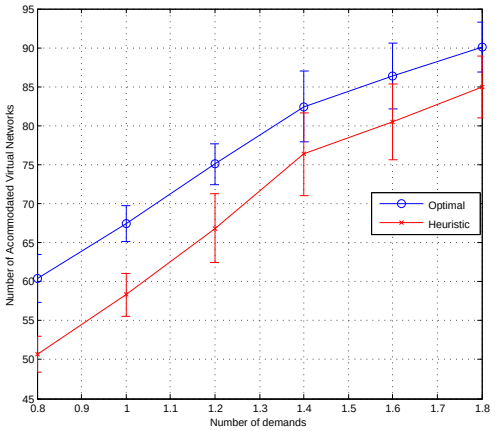
Figure B.2a presents the VN acceptance ratio of the proposed ILP model ('optimal') and of the modified heuristic ('heuristic') for different number of requests ('number of demands'). As can be observed, the acceptance ratio decays linearly with the number of requests for both mapping methods. For instance, with a $1/\boldsymbol{\lambda} = \mathbf{0.8}$, i.e. 0.8 VN request per time unit, nearly all the virtual networks are accepted when using the ILP model, while with the heuristic only 85% of the requests are accepted.

The average number of accommodated VNs per request is depicted in Figure B.2b. The number of accommodated VNs increases linearly with the number of demands for both embedding methods, although the ILP model embeds on average seven more VNs.

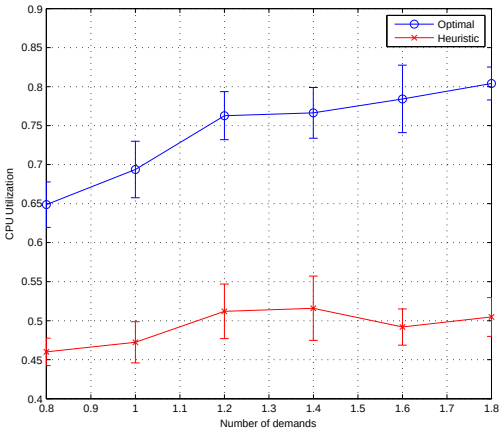
The remaining results concern the average utilization of the different types of resources: memory RAM and CPU for the physical nodes, and bandwidth for physical links according to the VN requests. Figure B.2c shows the average CPU utilization for different number of demands. With the ILP model, there is an increase of the CPU utilization with the increase of requests, reaching 80% while the heuristic maintains the utilization at 50%. The average memory RAM utilization according to different demands is depicted in Figure B.2d. Both mapping methods produce an increase of memory utilization with the number of requests, reaching a stable value at 84% for the ILP model and 71% for the heuristic. The average bandwidth utilization is depicted in Figure B.2e. Again, with both mapping methods, there is an increase of the resource utilization, with the optimal model reaching higher utilization values.



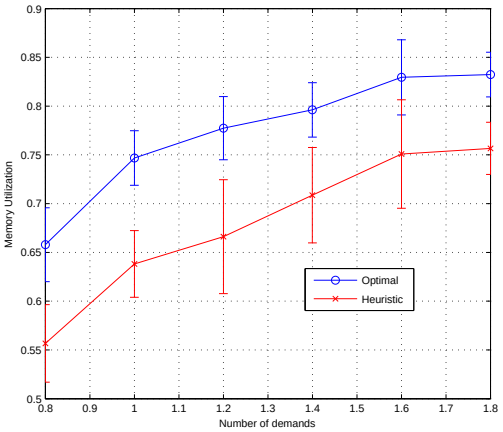
(a) Average acceptance ratio



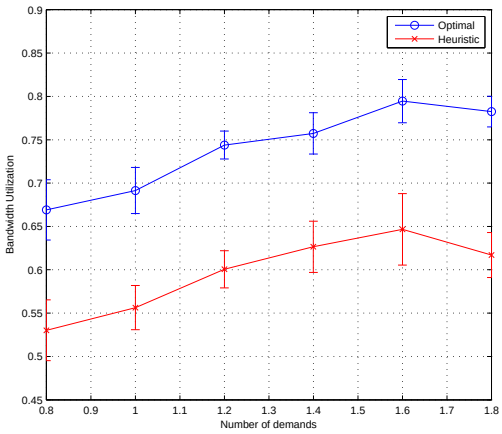
(b) Average number of accommodated virtual networks



(c) Average CPU utilization



(d) Average memory utilization



(e) Average bandwidth utilization

Figure B.2: Evaluation Metrics per demand.

B.5 Conclusion

This paper proposed an ILP model to solve the virtual network embedding problem and to provide the optimal boundary for each VN mapping. The model applies optimization theory and simultaneously optimizes the virtual nodes and the virtual links assignment, supporting heterogeneous virtual and substrate networks. This paper also proposed a heuristic algorithm that is used as comparison with the ILP model.

The obtained results show significant improvements of the VN acceptance ratio, when we compare the ILP model with the heuristic. In average, the ILP model is able to map additional 10% VN requests. Translating in the number of extra virtual networks accommodated on the physical network, on average 7 more VNs are accommodated. The ILP model is able to load the resources to a maximum of 80% on average, with a high VNs demand, while the heuristic does not go beyond 70%.

Future work will endorse the global optimal solution (which may require reassignment of previously mapped virtual nodes or links), and the migration of virtual nodes and networks. Scalability issues will also be addressed.

Appendix C - A Re-Optimization Approach for Virtual Network Embedding

Marcio Melo, Jorge Carapinha, Susana Sargento,Ulrich Killat, and Andreas Timm-Giel

in *Mobile Networks and Management Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, vol. 58, pp 271-283, 2013.

The format has been revised.

A Re-Optimization Approach for Virtual Network Embedding

Marcio Melo, Jorge Carapinha, Susana Sargento,Ulrich Killat, and Andreas Timm-Giel

Abstract

Network Virtualisation is claimed to be a key component of the Future Internet by enabling the coexistence of heterogeneous (virtual) networks in the same physical infrastructure, providing the dynamic creation and support of different networks with different paradigms and mechanisms in the same physical network. A major challenge in the dynamic provision of virtual networks resides in the optimal embedding solution of virtual resources into physical ones.

Since this problem is known to be $\mathcal{NP}$-hard, previous research focused on designing heuristic-based algorithms; most of them do not consider either a simultaneous optimization of the node and the link mapping or the re-optimization of VNs, leading to non-optimal solutions.

This paper proposes an extension of Virtual Network Embedding Node-Link Formulation to support the re-optimization of existing VNs and to provide the optimal bound. It also presents an evaluation of the proposed approach when applied to a previous heuristic in the literature. Simulation experiments show significant improvements when using the VN re-optimization process: not only the bandwidth consumption have been reduced by 17.5%, but the same is true for the maximum utilization levels on the CPU and on the memory.

C.1 Introduction

Network Virtualisation has gained an increasing prominence in the last few years. Initially, the interest in network virtualisation was mainly pushed by Future Internet research initiatives [PACR03, APST05b, FGR07b, ZZRR08], mainly with the objective to find a platform on which novel Internet architectures could be experimented and evaluated without limitations or constraints, namely those associated with the traditional IP model.

Later on, it became clear that virtualisation could constitute a key component of next-generation Internet architecture itself [TWEF03], and not just as a mere platform for experimentation. Perhaps more importantly for network operators, it also became clear that network virtualisation could provide a number of short/medium term business advantages, with potential reduction of costs and increase of revenues, as an interesting tool from an operational point of view [CJ09, MSC09].

Although there is a large interest on virtualized networks both from the research community and network operators, several challenges still prevent them from being deployed in real environments [CB09]. One of the major obstacles lies in providing the exact embedding[1] solution of a VN into a physical network. Some solutions to this problem were already proposed [ZA06, YYRC08, CRB09], mostly based on heuristic approaches; however, they have failed to provide the optimal solution for each VN mapping. An earlier publication [MCS+12] proposed a mathematical formulation, VNE-NLF, that uses ILP to provide the optimal bound. However, the optimal re-assignment of VNs previously embedded was not considered.

This paper focuses on the VN re-optimization process: it proposes an extension to VNE-NLF to support the re-optimization of VNs previously assigned, providing an optimal bound. First, the performance of VNE-NLF [MCS+12] is analyzed, where it is compared with a new proposed heuristic, the VNE-ESPH. Second, the re-optimization approach is proposed and compared to VNE-NLF. Simulation experiments show significant improvements when using the VN re-optimization process. Not only the bandwidth consumption has been reduced by 17.5%, but it has also reduced the maximum utilization levels on either the CPU and the memory, where it achieves maximum levels of utilization 20% and 16% lower, respectively.

The rest of the paper is organized as follows. After summarizing the related work in section C.2, section C.3 describes the virtual network embedding problem, and explains the mathematical formulation proposed to support the VN re-optimization on the assignment process. Section C.4 analyzes the performance of both the VNE-NLF and of the VNE-ESPH, and also evaluates the performance of the re-optimization process when compared to VNE-NLF. Section C.5 concludes the paper and describes the future work.

C.2 Related Work

The simultaneous node and link mapping optimization can be formulated as an unsplittable flow problem [ZA06], known to be $\mathcal{NP}$*-hard*, and therefore, it is only tractable for a small amount of nodes and links. In order to solve this problem, several approaches have been suggested, mostly considering the *off-line* version of the problem where the VN requests are fully known in advance.

In [LK09] a backtracking method based on sub-graph isomorphism was proposed; it considers the on-line version of the mapping problem, where the VN requests are not known

[1] The terms embedding, mapping and assignment are used interchangeably in this paper

in advance, and proposes a single stage approach where nodes and links are mapped simultaneously, taking constraints into consideration at each step of the mapping. When a bad mapping decision is detected, a backtrack to the previous valid mapping decision is made, avoiding a costly re-map.

The work in [LT06] defines a set of premises about the virtual topology, i.e. the backbone nodes are star-connected and the access-nodes connect to a single backbone node. Based on these premises, an iterative algorithm is run, with different steps for core and access mapping. However, the algorithm can only work for specific topologies.

A distributed algorithm was studied in [HLZ08]. It considers that the virtual topologies can be decomposed in hub-and-spoke clusters, and that each cluster can be mapped independently, therefore reducing the complexity of the full VN mapping. This proposal has lower performance and scalability when compared with centralized approaches.

Zhu et al. [ZA06] proposed a heuristic, centralized, algorithm to deal with VN embedding. The goal of the algorithm is to maintain a low and balanced load of both nodes and links of the substrate network. However, the load of nodes and links does not consider heterogeneity on their characteristics.

Yu et al. [YYRC08] proposed an embedding algorithm which considers finite resources on the physical network, and enables path splitting (i.e. virtual link composed by different paths) and link migration (i.e. to change the underlying mapping) during the embedding process. However, this level of freedom can lead to a level of fragmentation that is infeasible to work on large scale networks. In [CRB09], a formal approach was taken to solve the online VN embedding problem using a mixed integer programming formulation in a two-step approach. This approach, despite providing a better coordinated node and link mapping, does not solve the VN assignment problem, and does not support heterogeneity of nodes.

Butt et al. [FBCB10] proposed a topology-aware heuristic for VN embedding and also suggested a set of algorithms to avoid bottlenecks on the physical infrastructure, where they consider virtual node reallocation and link reassignment for this purpose. Nogueira et al. [NMCS11b] proposed a heuristic that takes into account the heterogeneity of the VNs and also of the physical infrastructure. The algorithm is evaluated by means of simulation and also on a small scale testbed, where it achieves mapping times of the order of tens of milliseconds.

Botero et al. [BHFM12] proposed an algorithm to solve the VN embedding problem considering also the CPU demand by the hidden hops. Lu et al. [LHkW+11] proposed an adaptive algorithm based on multi-commodity flows to solve the VN embedding. Chowdhury et al. [CRB12] extended his preliminary results [CRB09] and included a generalized window-based VN embedding to evaluate the effect of look ahead on the mapping of VNs. Melo et al. proposed in [MCS+12] a mathematical formulation to solve the VN mapping using ILP, and compared it with the heuristic in [NMCS11b].

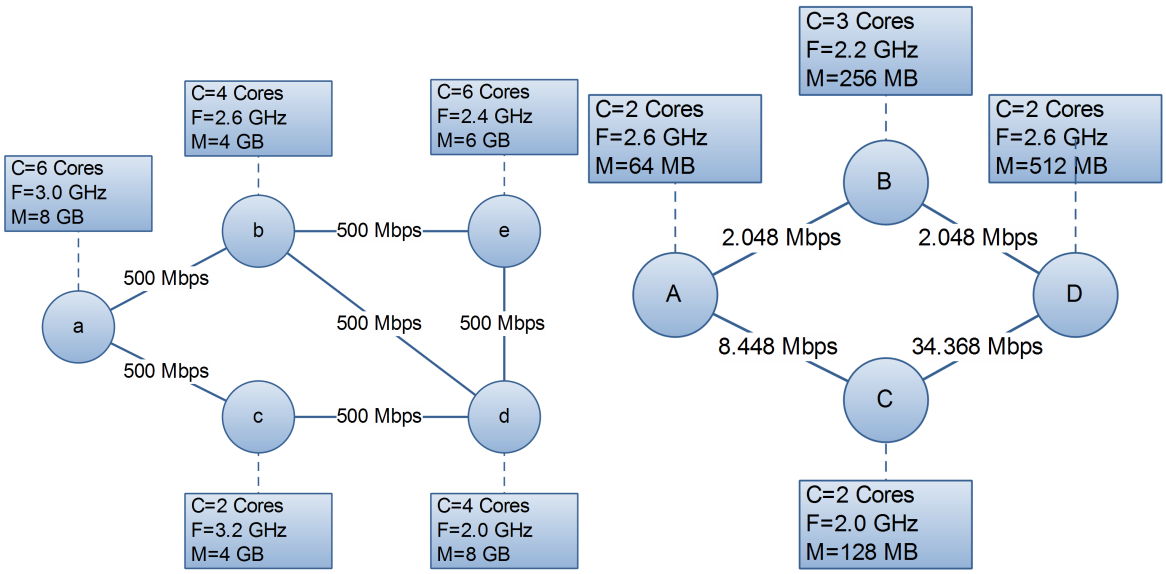
Although all these algorithms provide a solution for the VN mapping problem, the optimal re-assignment of existing VNs is not considered. This is the purpose of this paper, which evaluates both VNE-NLF ILP embedding approach and its heuristic (the VNE-ESPH), and proposes the optimal re-assignment of VNs currently embedded through VNE-NLF.

C.3 Problem Description and Mathematical Formulation Extension

In this section, we start with the description of the VN assignment problem. The mathematical model extension for VN re-optimization is then presented.

C.3.1 Network Description

We consider a physical network with a given number of nodes, N, and with a given topology, as depicted in Figure C.1a. Each node is described by the number of CPUs, which corresponds to letter C in the Figure, the clock CPU frequency, F, and by the RAM amount it possesses, M. With respect to the links, we consider the bandwidth capacity, B, and we assume that each link is a unidirectional link. Virtual networks are described the same way as physical networks, as shown in Figure C.1b. We use superscript to distinguish virtual from physical resources, where the letter P is used for the physical resources, e.g. $\boldsymbol{C^P}$, and the letter V is used for virtual resources, e.g. $\boldsymbol{C^V}$. The convention used for the index notation is: $\boldsymbol{i, j}$ for nodes and links in the physical network, and $\boldsymbol{m, n}$ for nodes and links in the VN.



(a) Substrate Network Topology (b) Virtual Network Topology

Figure C.1: Network Topology Description

C.3.2 Mathematical Formulation Extension - Re-Optimization Support

An extension to VNE-NLF [MCS+12] is proposed in order to support the embedding problem of VNs using re-optimization. The proposed formulation only uses two assignment variables for the VN mapping problem: one for the assignment of the virtual nodes to physical nodes, and another to express the assignment of virtual links into physical links. The binary variable $\boldsymbol{x}$ is used for the virtual nodes and is expressed in equation (C.1), where $\boldsymbol{x_i^m} \rightarrow \boldsymbol{W^V} \times \boldsymbol{N^P}$ matrix (and $\boldsymbol{W^V} = \sum_{\boldsymbol{k}} \boldsymbol{N^{V_k}}$ which is the sum of all virtual network size's currently embedded on the physical network). For the virtual links, the binary variable $\boldsymbol{y}$ is used and it is represented in equation (C.2), where $\boldsymbol{y_{ij}^{mn}} \rightarrow \boldsymbol{(W^V)^2} \times \boldsymbol{(N^P)^2}$ matrix.

The objective function is represented in equation (C.3) and achieves two goals: the primary goal is to minimize the maximum load per physical resource and, in the case of having different mapping solutions with the same maximum utilization, the second part of the objective function is activated which will opt for the solution which consumes the lowest bandwidth, where $\boldsymbol{\epsilon}$ represents a small constant.

Assignment Variables

$$x_i^m = \begin{cases} \mathbf{1}, \text{virtual node } \boldsymbol{m} \text{ is allocated at physical node } \boldsymbol{i} \\ \mathbf{0}, \boldsymbol{else} \end{cases} \tag{C.1}$$

$$y_{ij}^{mn} = \begin{cases} \mathbf{1}, \text{virtual link } \boldsymbol{mn} \text{ uses physical link } \boldsymbol{ij} \\ \mathbf{0}, \boldsymbol{else} \end{cases} \tag{C.2}$$

Optimization Function

$$\boldsymbol{minimize}\ \mathcal{C}_{max}^{load} + \mathcal{M}_{max}^{load} + \mathcal{B}_{max}^{load} + \epsilon \times \sum_{m,n \in N^V(m), n<m} y_{ij}^{mn} \times B_{mn}^V \tag{C.3}$$

Constraints

(Maximum) Resource Utilization of the CPU

$$\forall \boldsymbol{i} : F_i^P \times \frac{C_i^P + \sum_m x_i^m \times C_m^V}{C_i^P(\mathbf{0})} \leq \mathcal{C}_{max}^{load} \tag{C.4}$$

(Maximum) Resource Utilization of the Memory

$$\forall \boldsymbol{i} : F_i^P \times \frac{M_i^P + \sum_m x_i^m \times M_m^V}{M_i^P(\mathbf{0})} \leq \mathcal{M}_{max}^{load} \tag{C.5}$$

(Maximum) Resource Utilization of the Bandwidth

$$\forall i, j \in N^P(i), i < j : \frac{B_{ij}^P + \sum_{m,n \in N^V(m)} y_{ij}^{mn} \times B_{mn}^V}{B_{ij}^P(\mathbf{0})} \leq \mathcal{B}_{max}^{load} \tag{C.6}$$

Assignment of virtual nodes to physical nodes

$$\forall \boldsymbol{m} : \sum_i x_i^m = \mathbf{1} \tag{C.7}$$

One virtual node per physical node

$$\forall \boldsymbol{k}, \forall \boldsymbol{i} : \sum_{m \in VN_k} x_i^m \leq \mathbf{1} \tag{C.8}$$

Multi-commodity flow conservation with $\boldsymbol{node - link}$ formulation

$$\forall \boldsymbol{m}, \boldsymbol{n} \in \boldsymbol{N^V(m)}, \boldsymbol{m < n}, \forall \boldsymbol{i} : \sum_{j \in N^P(i)} (y_{ij}^{mn} - y_{ji}^{mn}) = x_i^m - x_i^n \tag{C.9}$$

CPU conservation

$$\forall \boldsymbol{i} : \sum_m x_i^m \times C_m^V \leq C^{P_i} \tag{C.10}$$

Memory conservation

$$\forall \boldsymbol{i} : \sum_m x_i^m \times M_m^V \leq M^{P_i} \tag{C.11}$$

Bandwidth conservation

$$\forall i, j \in N^P(i), i < j : \sum_{m,n \in N^V(m), m<n} B_{mn}^V \times (y_{ij}^{mn} + y_{ji}^{mn}) \leq B_{ij}^P \tag{C.12}$$

CPU frequency requisite

$$\forall \boldsymbol{i} : \sum_{\boldsymbol{m}} \boldsymbol{x_i^m} \times \boldsymbol{F_m^V} \leq \boldsymbol{F_i^P} \tag{C.13}$$

Remarks:

The (maximum) utilization per resource type, i.e. memory RAM ($\boldsymbol{\mathcal{M}_{max}^{load}}$), CPU ($\boldsymbol{\mathcal{C}_{max}^{load}}$), and bandwidth ($\boldsymbol{\mathcal{B}_{max}^{load}}$), is represented in equations (C.4),(C.5),(C.6), respectively, where $\boldsymbol{C_i^P}$ represents the currently available capacity and $\boldsymbol{C_i^P(0)}$ represents the total capacity. The resource utilization on the CPU and on the memory is also multiplied by the CPU frequency, in order to firstly use physical nodes with lower CPU frequency and to preserve the remaining for virtual nodes with higher CPU frequency demand. Therefore, the second term of (C.4) and (C.5) is not only the maximum resource utilization, but it is the CPU frequency multiplied by the resource utilization either on the CPU or on the memory.

Equation (C.7) ensures that each virtual node is assigned and it is to just one physical node.

In order to support the re-optimization process, equation (C.8) is proposed and differs from the initial formulation [MCS+12] once it takes into consideration all the VNs that are currently assigned, and not only one VN request. Equation (C.8), is also used to guarantee that each physical node accommodates, in maximum, one virtual node per VN[2], where $\boldsymbol{k}$ is used to represent all VNs running on that specific physical node. However, each physical node can accommodate, in principle, more virtual nodes from other VNs (i.e. $\boldsymbol{VN_k}$).

In order to optimize the mapping of the virtual links and at the same time to cope with the optimization of the virtual nodes, the multi-commodity flow constraint [EIS75] is applied with a $\boldsymbol{node-link}$ formulation [PM04], and the notion of direct flows on the virtual links is also used, which is represented in equation (C.9). To ensure that the available capacity at the CPU, memory and bandwidth is not exceeded, equations (C.10), (C.11) and (C.12) are used, respectively. Finally, equation (C.13) is used to guarantee that we do not violate that requirement on the CPU frequency.

C.4 Evaluation Results

In this section we depict our main results. Our evaluation is primarily focused, on the impact due the VN size and due to the physical network size, on VN acceptance ratio according to the VN size, i.e. the number of virtual nodes, and also on the number of VNs that can be accommodated on the physical network using the proposed model. We compare the VNE-NLF with the VNE-ESPH embedding approach, and we then evaluate the performance, in terms of resource utilization, of the re-optimization proposal when compared to VNE-NLF.

C.4.1 Simulation Parameters - VNE-NLF and VNE-ESPH

In order to evaluate the VNE-NLF and the VNE-ESPH, we have implemented a discrete event simulator in Matlab®. The physical network topology was created using the Waxman random topology generation method [Wax88], and the number of physical nodes was set to 30, 40 and 50, according to the evaluated scenario.

The recommended parameters for link probability, $\boldsymbol{\alpha = 0.4}$ and $\boldsymbol{\beta = 0.1}$, were used although some topologies did not have full connectivity, i.e. one physical node with no viable path to all the remaining nodes (e.g. a node with no links or non-connected clusters). In order to circumvent this, after generating the topology, additional links were added to the nodes with fewer interfaces, until total connectivity was reached.

[2]This assumption is also taken by other authors, i.e., [ZA06, YYRC08, CRB09].

For each substrate node, a set of parameters was attributed using an uniform distribution, such as RAM amount, number of CPUs and CPU frequency. The physical link's bandwidth was set to a fixed bitrate.

The VNs requests were created using the same topology generation model, and the number of virtual nodes was fixed from 2 to 10 virtual nodes with intervals of 2 nodes, where the size of the VN was changed according to the evaluated scenario. After generating the virtual topology, the same set of specifications was assigned, with a uniform distribution. Either the physical network or the virtual network specifications can be observed on Table C.1.

We assume that each VN request arrives according to a Poisson process, and that each VN has an associated lifetime with an average of $\mathbf{1/\mu = 75}$ time units, following an exponential distribution. Regarding the average number of VN requests per time unit, they are set to 1.8 VN requests per time unit. For each considered size of VN, 10 trials were performed. For each trial, a new set of VN requests and new physical network topology were generated. All simulations were set to run until 1000 time units. A confidence interval of 95% is used for every result presented below. The CPLEX [cpl12] version 12 was used to solve the linear programming problem, and a time limit of 600 seconds was defined for each VN mapping, although most VNs were embedded in hundreds of milliseconds.

Table C.1: Physical Network and Virtual Network Parameters.

Parameters	**Physical Network**	**Virtual Network**
N. CPUs	{2; 4; 6}	{1; 2; 3; 4 }
CPU Frequency (GHz)	{2.0 to 3.2 in 0.2 steps }	{2.0 to 2.6 in 0.1 steps }
RAM Memory (GB)	{2; 4; 6; 8}	{64; 128; 256; 512 }
Link Bandwidth (Mbps)	{500}	{2.048; 8.448; 34.368}

C.4.2 Simulation Results - VNE-NLF and VNE-ESPH

This section compares VNE-NLF with VNE-ESPH methods through several metrics: VN request acceptance ratio, number of VNs that were running after the simulation experiment was finished, average resource utilization, e.g. memory RAM and CPU, on the nodes, and average bandwidth utilization on the links.

The VN request acceptance ratio is depicted in Figure C.2a. It decays linearly with the VN size for both embedding methods, which is due to the fact that both try to accommodate more virtual resources with the same amount of physical resources available; the slope is sharper for the case of the VNE-ESPH. Therefore, we can infer that the performance of the heuristic is influenced by the size of the virtual network. We can also infer that the size of the physical network influences significantly the VN request acceptance ratio, where smaller physical networks are more prone to reject VN requests. Another important aspect to retain is the fact that the VNE-NLF method applied on a physical network with 40 physical nodes is able to have the same VN request acceptance ratio as the VNE-ESPH method applied on a physical network with 50 nodes, which is 20% larger.

The average number of accommodated VNs is presented in Figure C.2b. The number of VNs running on the substrate decays linearly with the VN request size for both embedding methods, and the slope is sharper in the case of the heuristic. This kind of behaviour is expected once we are trying to embed larger VNs with the same amount of physical resources, although the total number of virtual nodes is almost the same. If we consider the case of a physical network with 50 physical nodes and running the VNE-NLF as a embedding method,

we have the same number of virtual nodes running on the substrate either with VN requests of 8 nodes or with VN requests of 10 nodes, i.e. $\mathbf{8 \times 100 = 10 \times 80}$.

The average CPU utilization is shown in Figure C.2c. The same levels of average CPU utilization, i.e. 75%, are reached in the VNE-NLF, independent either on the physical network size or on the virtual network size. Regarding the VNE-ESPH, the average CPU utilization does depend on the virtual network size, and for VNs larger or equal to 4 virtual nodes, the average CPU utilization starts to decay linearly with the VN size. This is strongly related with the VN acceptance ratio which is lower for VNs with the same network size; this embedding method performs worse with larger VNs.

The average memory utilization as a function of the VN size is depicted in Figure C.2d. The same behaviour is perceived for the average memory utilization as for the average CPU utilization for both embedding methods. The average memory utilization reaches values of 80% for both embedding methods, although the average memory utilization using the VNE-ESPH starts to decay with VNs larger than 4 nodes.

The average bandwidth utilization is depicted in Figure C.2e. The same behaviour is perceived for both the average bandwidth utilization and the average memory or CPU utilization, where the average bandwidth utilization reaches values of 80%. The average memory utilization reaches values of 80% for both embedding methods, although the average bandwidth utilization using the VNE-ESPH starts to decay with VNs larger than 4 nodes.

C.4.3 Re-Optimization

This section, presents the simulation results obtained using the optimization method VNE-NLF, and its extension for re-optimization of virtual network embedding. Our evaluation is primary focused on the minimum and maximum resource utilization on the CPU and on the memory, and on the average bandwidth utilization.

The simulation parameters applied here are the same as in Section C.4.1, except for the physical and the VN size. VN requests ranging from 0.8 VN per time unit to 1.8 VN per time unit were considered. Regarding the re-optimization process, the results were obtained using a virtual machine configured with 4 cores (Intel Xeon X5650@2.67GHz) and the CPLEX was set to use up to 4 threads, the relative gap tolerance was set to 0.05 (i.e. feasible integer solution proved to be within percent of optimal), and a time limit of 24 hours was used in order to avoid long time simulations. However, most of the re-optimizations were performed within the 24 hours' time frame.

The maximum and minimum CPU utilization are depicted in Figure C.3a. We can observe that the re-optimization process clearly reduces the maximum CPU utilization. The gain is higher when the physical network is not loaded and lowers when the network is almost fully loaded. The re-optimization achieves values 20% lower for a maximum CPU utilization of 0.8 VN request per time unit, and 5% lower for 1.8 VN request per time unit. We can observe also that the re-optimization process increases the minimum CPU utilization, where it achieves values 25% larger for minimum CPU utilization. The variation on the number of VN requests does not seem to affect substantially the gap between the two embedding processes.

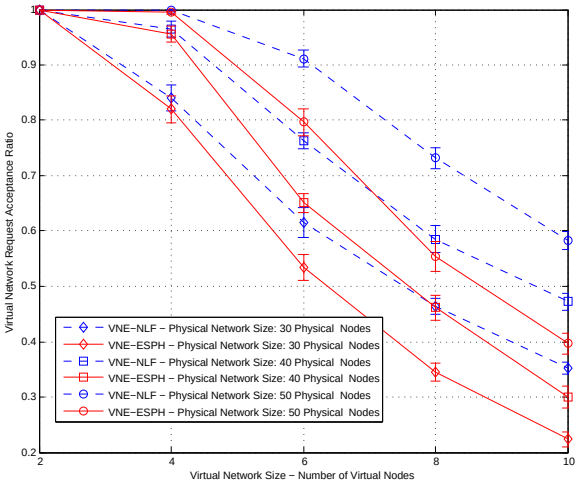
The maximum and minimum memory utilization is shown in Figure C.3b. The same behaviour, as for the maximum CPU utilization or minimum CPU utilization can be observed, where the re-optimization process reduces significantly the maximum memory utilization. The re-optimization achieves values 16% lower for maximum memory utilization of 0.8 VN request per time unit, and 3% lower for 1.8 VN request per time unit. We can also observe that the minimum memory utilization with the re-optimization process increases, where it achieves values 25% higher for memory utilization.

Figure C.3c shows the average bandwidth utilization as a function of the number of virtual network requests. We can observe that applying the re-optimization reduces significantly the average bandwidth utilization on the physical links. This gain is even higher with the increase on the number of VN requests, reaching values 17.5% lower for 1.8 VN requests per time unit.

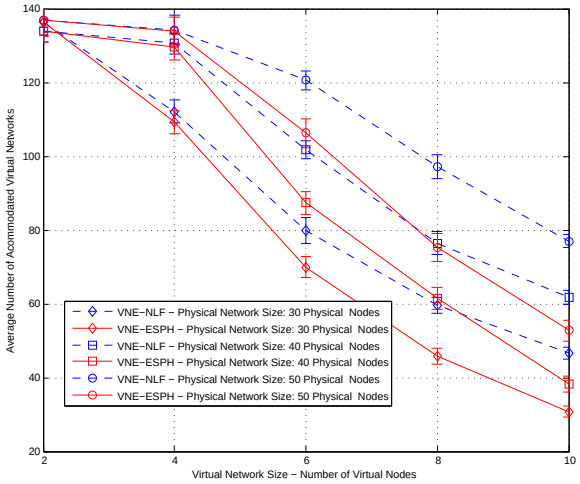
C.5 Conclusion

This paper proposed a re-optimization mechanism to enhance the performance of the virtual network embedding process. The proposed re-optimization approach proved to be a very efficient approach. It was not only able to reduce significantly the overall bandwidth consumption, where the average bandwidth utilization decreased 10%, but it also decreased clearly the maximum CPU and memory utilization levels. This is not only important from a load balancing perspective, where the load of nodes that are already at critical levels is being moved to nodes less loaded, but it is also important from a revenue perspective, where bandwidth that was previously provisioned is being released and that could be leased to new VNs.

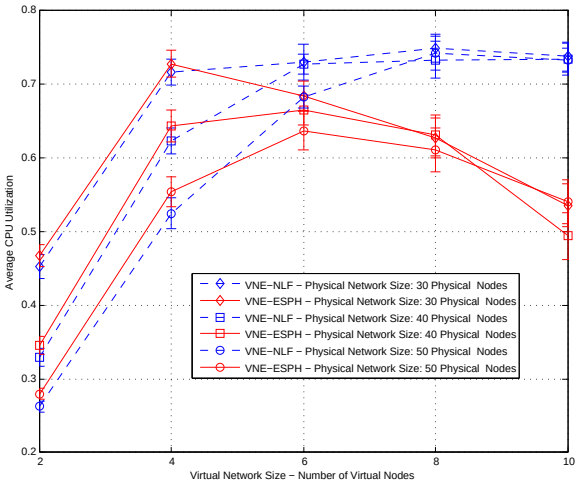
Future work will endorse the utilization of the re-optimization as a way to improve the VN request acceptance ratio. The evaluation of the VNE-NLF with different cost functions and with other heuristics available in the literature will also be performed, as well as its evaluation using other metrics, e.g. revenue or provisioning cost.



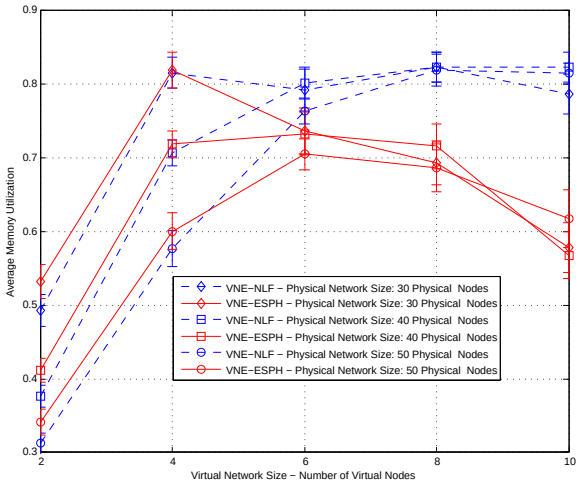
(a) VN Request Acceptance Ratio



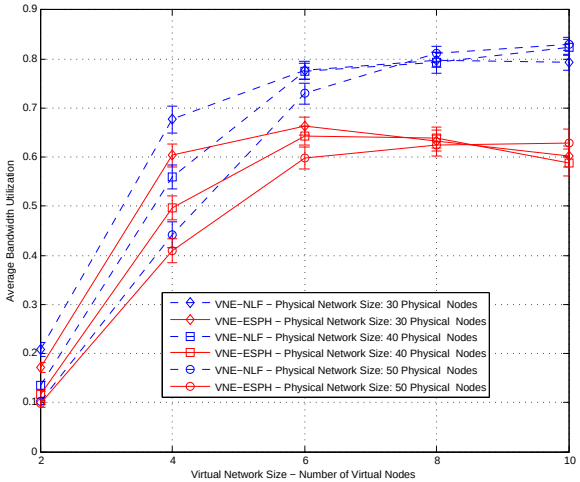
(b) Average Number of VNs



(c) Average CPU Utilization

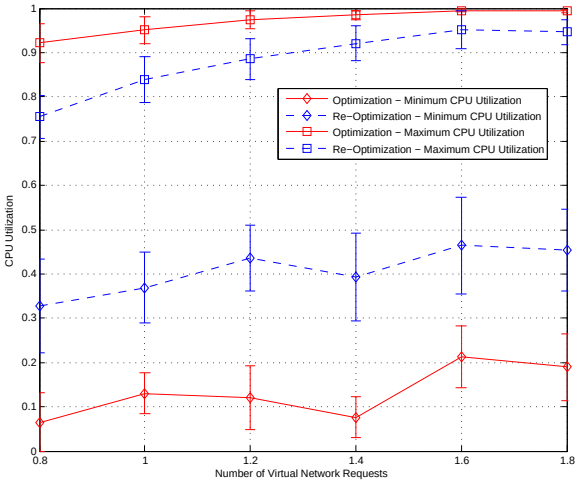


(d) Average Memory Utilization

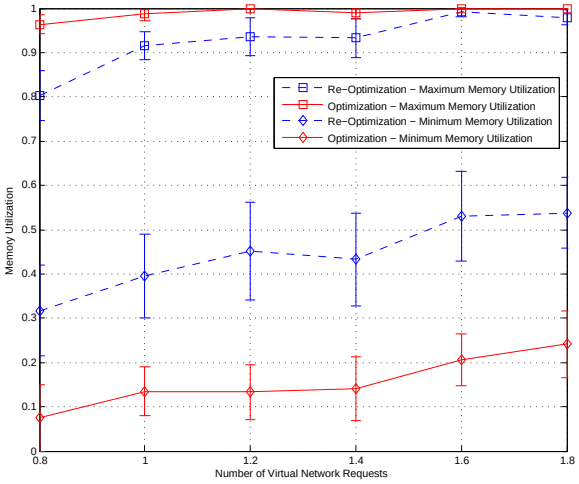


(e) Average Bandwidth Utilization

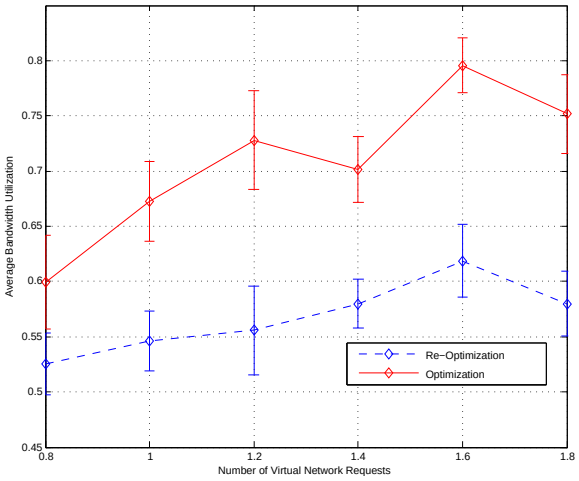
Figure C.2: VN Request Acceptance Ratio, Number of Existing VNs on the Substrate and Resource Utilization as a function of the VN Size.



(a) Maximum and Minimum CPU Utilization



(b) Maximum and Minimum Memory Utilization



(c) Average Bandwidth Utilization

Figure C.3: Resource Utilization as a function of the Number of VN requests.

Appendix D - Optimal Virtual Network Embedding: Node-Link Formulation

Márcio Melo, Susana Sargento, Ulrich Killat, Andreas Timm-Giel, and Jorge Carapinha

in *IEEE Transactions on Network and Service Management,* vol. 10, Issue: 4 2013.

The format has been revised.

Optimal Virtual Network Embedding: Node-Link Formulation

Márcio Melo, Susana Sargento, Ulrich Killat, Andreas Timm-Giel, and Jorge Carapinha

Abstract

Network Virtualisation is claimed to be a key component of the Future Internet, providing the dynamic support of different networks with different paradigms and mechanisms in the same physical infrastructure. A major challenge in the dynamic provision of virtual networks is the efficient embedding of virtual resources into physical ones. Since this problem is known to be $\mathcal{NP}$-hard, previous research focused on designing heuristic-based algorithms; most of them either do not consider a simultaneous embedding of virtual nodes and virtual links, or apply link-path formulation, leading to non-optimal solutions.

This paper proposes an integer linear programming (ILP) formulation to solve the online virtual network embedding problem as a result of an objective function striving for the minimization of resource consumption and load balancing. To this end 3 different objective functions are proposed and evaluated. This approach applies multi-commodity flow constraint to accomplish a node-link formulation that optimizes the allocation of physical network resources.

This proposal is evaluated against state of the art heuristics. The performance of the heuristics related to VN request acceptance ratio is, at least, 30% below the one of the VNE-NLF method. From the three cost functions evaluated, the Weighted Shortest Distance Path (WSDP) is the one which embeds more VNs and also requires, on average, less physical resources per embedding.

D.1 Introduction

Network Virtualisation has gained an increasing prominence in networking and telecommunications fields in the last few years. Initially, the interest in network virtualisation was mainly pushed by Future Internet research initiatives [PACR03, APST05b, FGR07b, ZZRR08], mainly with the objective to find a platform on which novel Internet architectures could be experimented and evaluated without limitations or constraints, namely those associated with the traditional IP model. Later on, it became clear that virtualisation could constitute a key component of the next-generation Internet architecture itself [TWEF03], and not just as a mere platform for experimentation. It also became clear for network operators that network virtualisation could provide a number of short/medium term business advantages, with potential reduction of costs and increase of revenues, as an interesting tool from an operational point of view [CJ09, MSC09].

Although there is a large interest on virtualized networks both from the research community and network operators, several challenges still prevent them from being deployed in real environments [CB09]. One of the major obstacles lies in the efficient embedding[1] of a Virtual Network (VN) onto a physical network. Since this process requires the simultaneous optimization of virtual nodes and links placement, it is complex in nature, both in formulation and computationally. Several works, such as [ZA06, LK09, LT06, YYRC08, CRB09, FBCB10, NMCS11b, BHFM12, CRB12], have already proposed solutions to this problem, mostly based on heuristic approaches; however, they do not provide the optimal solution for each VN mapping.

This paper focuses on the online embedding of VN requests in the physical network. An ILP formulation, the Virtual Network Embedding Node-Link Formulation (VNE-NLF), is used to solve the VN assignment problem on the basis of a minimization of resource consumption and load balancing strategy. The VNE-NLF includes link delay constraints and supports the specification of the maximum distance between virtual nodes. In addition, different cost functions are proposed and analyzed, which enforce load balancing of links and nodes, and shortest distance paths. Simulation experiments show how far the state of the art heuristics differ from an ILP method. If the VN request acceptance ratio is used as a measurement metric, the solutions obtained by the state of the art heuristics are, at least, 30% below the ones of the VNE-NLF (see Figure D.3). From the cost functions evaluated, the WSDP is the one which embeds more VNs and also requires, on average, less physical resources per embedding. Compared to our previous work in [MCS+12], this paper:

i extends the mathematical formulation to support two new constraints, i.e. link delay and maximum distance between nodes;

ii proposes three new cost functions, i.e. Load Balancing plus ϵ Shortest Path (LB+ϵSP), SDP, and WSDP;

iii defines a new evaluation metric, i.e. the embedding factor, which represents the amount of resources that have been requested over the amount of resources that have been leased;

iv and it provides a performance comparison with 6 state-of-the-art heuristics.

The contributions of this paper can be summarized as follows:

i ILP optimization for virtual network embedding, which is based on node-link formulation; it enables the simultaneous embedding of virtual nodes and links, which optimizes the

[1] The terms embedding, mapping and assignment are used interchangeably in this paper.

allocation of physical network resources, i.e. CPUs on the physical nodes and bandwidth on the physical links.

ii Analysis and evaluation of different objective goals: load balancing objective function LB+ϵSP, shortest path objective function SDP, and load balancing combined with shortest path objective function WSDP;

iii Comparison of the performance of existing VN embedding solutions, heuristics, with a pure ILP formulation.

iv Evaluation metrics that relate e.g. VN acceptance ratio and link utilization. Moreover, a new metric, the embedding factor, is proposed.

The rest of the paper is organized as follows. After summarizing the related works in section D.2, section D.3 describes the virtual network embedding problem, the notations and parameters used, and the embedding process. Section D.4 describes the proposed VNE-NLF and the applied constraints, while section D.5.1 presents and discusses the different proposed cost functions. Section D.6 analyzes the performance of the VNE-NLF with different cost functions, and compares it with six existing heuristics. Finally, section D.7 concludes the paper and describes the future work.

D.2 Related Work

This simultaneous node and link mapping optimization can be formulated as an unsplittable flow problem [ZA06], known to be $\mathcal{NP}$-hard. In order to solve this problem, several approaches have been suggested, mostly considering the *off-line* version of the problem where the VN requests are fully known in advance.

In [LK09] a backtracking method based on sub-graph isomorphism was proposed; it considers the on-line version of the mapping problem, where the VN requests are not known in advance, and proposes a single stage approach where nodes and links are mapped simultaneously, taking constraints into consideration at each step of the mapping. When a bad mapping decision is detected, a backtrack to the previous valid mapping decision is made, avoiding a costly re-map.

The work in [LT06] defined a set of premises about the virtual topology, i.e. the backbone nodes are star-connected and the access-nodes connect to a single backbone node. Based on these premises, an iterative algorithm is run, with different steps for core and access mapping. However, the algorithm can only work for specific topologies.

A distributed algorithm was studied in [HLZ08]. It considers that the virtual topologies can be decomposed in hub-and-spoke clusters and each cluster can be mapped independently, therefore reducing the complexity of the full VN mapping. This proposal has lower performance when compared with centralized approaches.

Zhu ***et al.*** [ZA06] proposed a heuristic based on a centralized algorithm to deal with VN mapping. The goal of the algorithm is to maintain a low and balanced load of both nodes and links of the substrate network. Yu ***et al.*** [YYRC08] proposed a mapping algorithm which considers finite resources in the physical network, and enables path splitting (i.e. virtual link composed by different paths) and link migration (i.e. to change the underlying mapping) during the embedding process. However, this level of freedom can lead to a level of fragmentation that is unfeasible to manage large scale networks.

In [CRB09] a formal approach is taken to solve the on-line VN mapping problem using a mixed integer programming formulation. Chowdhury ***et al.*** applied a two step approach to embed VNs on the substrate. In the first step, the virtual nodes are assigned to physical

nodes and in the second step the virtual links are assigned to physical paths. Compared to the previous state of the art heuristics, i.e. [ZA06, YYRC08], the formulation proposed by Chowdhury ***et al.*** provides a better coordination of the two phases, since an "augmented substrate graph construction" is used.

The approach in [CRB09] completely differs from the mathematical formulation proposed in this paper, which applies a node-link formulation. In our approach, the universe of embedding solutions is considered within the ILP formulation, and the VN embedding problem is solved in a single step using the multi-commodity flow constraint and by considering the notion of direction of the flows.

Butt ***et al.*** [FBCB10] proposed a topology aware heuristic for VN mapping, and also suggested algorithms to avoid bottlenecks on the physical infrastructure, where they consider virtual node reallocation and link reassignment for this purpose. Nogueira ***et al.*** [NMCS11b] proposed a heuristic that takes into account the heterogeneity of the VNs and also of the physical infrastructure. The heuristic is evaluated by means of simulation and also on a small scale testbed, where it achieves mapping times of the order of tens of milliseconds.

Botero ***et al.*** [BHFM12] proposed an algorithm to solve the VN mapping problem, which also considers the CPU demand of the hidden hops. Chowdhury et al. [CRB12] extended his preliminary results [CRB09] and included a generalized window-based VN embedding to evaluate the effect of look ahead on the mapping of VNs.

Alkmim ***et al.*** [ABdF13] proposed a mathematical formulation that aims to: i) map virtual routers and virtual links; ii) minimize the bandwidth consumption; and iii) minimize the time required to instantiate a virtual router. In contrast to this work, we also aim to optimize link load and CPU load distribution.

Although all these algorithms provide a solution for the VN mapping problem, an optimal solution for the embedding task and its efficiency is not provided. Also, some of them fail to solve the assignment problem as a simultaneous optimization of the virtual node and link placement, which leads to non-optimal solutions.

The VNE-NLF applies a node-link formulation to solve the VN embedding problem in a single step using the multi-commodity flow constraint. This approach provides the optimal solution for the objective function used, since the universe of solutions is considered within the ILP formulation.

D.3 Network Description and Problem Formulation

In this section, we introduce the virtual network embedding problem. In addition, the VN embedding notations used throughout the paper are presented, and the virtual network embedding system is explained. Finally, the mapping goals are introduced to support the mathematical formulation.

D.3.1 Network Description

We use superscript to distinguish the physical network from the virtual network, where p and v correspond to physical and virtual, respectively.

D.3.1.1 Physical network

A physical network can be described as a weighted undirected graph $\boldsymbol{G^p = \{N^p, L^p, C^p, B^p, D^p, \mathrm{Dis}^p\}}$ composed by a set of physical nodes, $\boldsymbol{N^p}$, and a set of physical links, $\boldsymbol{L^p}$. Each physical node $\boldsymbol{i}$ is characterized by its processing capacity, $\boldsymbol{C_i^p}$, commonly referred to as the CPU, and by its physical location, which can be defined by $\boldsymbol{x}$ and $\boldsymbol{y}$ coordinates. The

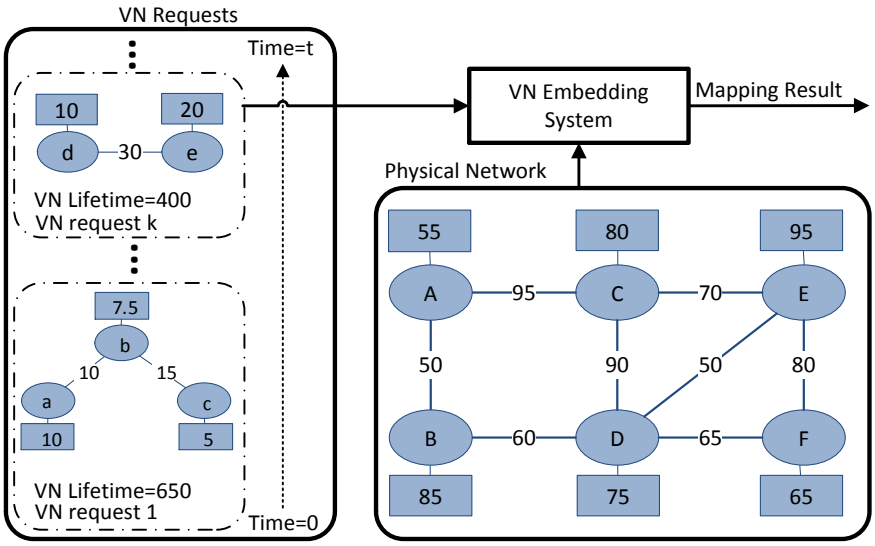


Figure D.1: VN Embedding System - Topology Example

distance between virtual nodes, $\mathbf{Dis}^{p}$, can be obtained using equation (D.1). With respect to the physical links, we consider that each link ij has a given bandwidth, $\boldsymbol{B}^{p}_{ij}$, and a given link delay, $\boldsymbol{D}^{p}_{ij}$, and we also assume that each link is an undirected link. The bottom-right of Figure D.1 illustrates a physical network topology example composed of 6 physical nodes and 8 physical links, and the corresponding capacities of the nodes and the links are presented on top of the elements.

$$\mathbf{Dis}^{p}_{ij} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \tag{D.1}$$

D.3.1.2 Virtual Network Request

VN request can be described as a weighted undirected graph $\boldsymbol{G^v = \{N^v, L^v, C^v, B^v, D^v, \mathbf{Dis}^v\}}$ composed by a set of virtual nodes, $\boldsymbol{N^v}$, and a set virtual links, $\boldsymbol{L^v}$. Each virtual node $\boldsymbol{m}$ is characterized by the amount of required CPU, $\boldsymbol{C^v_m}$, and the virtual links $\boldsymbol{mn}$ are logical connections between virtual nodes and characterized by the amount of dedicated bandwidth, $\boldsymbol{B^v_{mn}}$, and by the maximum link delay permitted, $\boldsymbol{D^v_{mn}}$. We also assume that each virtual link is an undirected link. The maximum distance between virtual nodes, $\mathbf{Dis}^v$, can be used to limit the number of intermediate hops between virtual nodes. The left part of Figure D.1 represent the example of two virtual network requests, VN request **1** on the bottom-left and VN request $\boldsymbol{k}$ on the top-left. Each VN request has a given lifetime that is, in principle, independent from each other, and each lifetime could have different time scales, since it is strongly dependent on the purpose of the virtual network request itself. If we consider a VN request for a live rock concert, the time scale will be hours, but if we consider a VN for a culinary workshop of one week, the time scale will be days.

D.3.1.3 VN Assignment Notations

First, we start with the convention used for the index notation: $\boldsymbol{N^p}$ represent the set of nodes that belong to the physical network; $\boldsymbol{L^p}$ represent the set of links that belong to the physical network; and $\boldsymbol{L^p_i}$ represent a subset of links $\boldsymbol{ij}$ that are directly connected to the node $\boldsymbol{i}$. The same type of notation is used to represent the VN using the letters $\boldsymbol{m}$ and $\boldsymbol{n}$ in the virtual network. The notations used throughout this paper for the VN assignment problem are presented in Table D.1. The table is divided into three parts: the static parameters of the physical network, the dynamic parameters of the physical network, and the virtual network requests with the demanded capacities.

D.3.2 Unfilled Physical Network Resources

The remaining capacity of each physical node at a specific time $\boldsymbol{t}$ is given by the difference between the total processing capacity and the capacity consumed by all virtual nodes allocated on that physical node, and is presented in equation (D.2), where $\boldsymbol{u}$ represents the set of all virtual nodes allocated on that precise physical node and at time $\boldsymbol{t}$.

$$\forall \boldsymbol{i} \in \boldsymbol{N^p} : \boldsymbol{C^p_i(t) = C^p_i(0) - \sum_u C^v_u(t)} \tag{D.2}$$

In parallel, the available bandwidth of each physical link at a specific time $\boldsymbol{t}$ is given by the difference between the total bandwidth and the bandwidth consumed by all virtual link segments allocated on that physical link, and is presented in equation (D.3), where $\boldsymbol{w}$ represents the set of all virtual link segments allocated on that specific physical link and at time $\boldsymbol{t}$.

A virtual link can be composed of one or more physical links, physical path. We consider that each virtual link has a single physical path, and we do not consider link aggregation (i.e. virtual link composed by different physical paths).

One physical link could accommodate one or more virtual link segments belonging to different virtual links.

$$\forall \boldsymbol{ij} \in \boldsymbol{L^p(k)} : \boldsymbol{B^p_{ij}(t) = B^p_{ij}(0) - \sum_w B^v_w(t)} \tag{D.3}$$

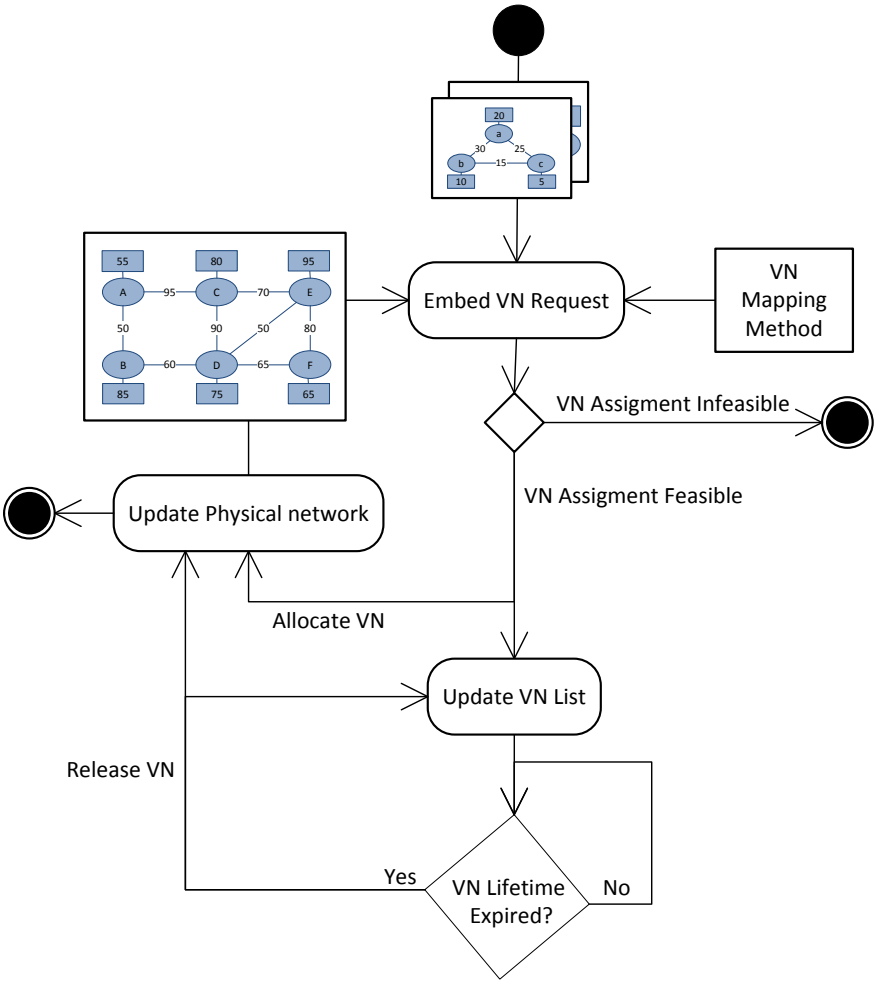


Figure D.2: VN Request Life Cycle - Activity Diagram

D.3.3 VN Request Embedding Process

The VN request embedding process can be divided into two components: the component that ensures the mapping of the virtual nodes, and the one that handles the mapping of the virtual links.

D.3.3.1 Virtual Node Mapping

Each virtual node needs to be mapped onto one physical node, this relation is given by the mapping function $\boldsymbol{\mathcal{M}[m \in N^v(k)] = i}$, where virtual node $\boldsymbol{m}$ is mapped onto physical node $\boldsymbol{i}$. Each physical node candidate needs to have, at least, the same amount of available CPU as required by the virtual node, which is represented in equation (D.4).

$$\forall i, \forall \mathcal{M}[m \in N^v(k) = i] : C_m^v(k) \leq C_i^p(t) \tag{D.4}$$

D.3.3.2 Virtual Link Mapping

Each virtual link can be mapped onto one or more physical links (i.e. physical path), this relation is given by the mapping function $\boldsymbol{\mathcal{M}[L_{mn}^v]}$, where the virtual link $\boldsymbol{mn}$ is mapped

Table D.1: VN Assignment Problem Notation.

$\boldsymbol{G^p}$	Physical Network
$\boldsymbol{N^p}$	Set of Physical Nodes
$\boldsymbol{i}$, $\boldsymbol{j}$	Physical Nodes
$\boldsymbol{ij}$	Physical Link
$\boldsymbol{L^p}$	Set of Physical Links
$\boldsymbol{L_i^p}$	Set of Physical Links directly connected to Physical Node $\boldsymbol{i}$
$\boldsymbol{C_i^p(0)}$	Total CPU of Physical Node $\boldsymbol{i}$
$\mathbf{Dis}_{\boldsymbol{ij}}^{\boldsymbol{p}}$	Distance Between Physical Nodes $\boldsymbol{ij}$
$\boldsymbol{B_{ij}^p(0)}$	Total Bandwidth of Physical Link $\boldsymbol{ij}$
$\boldsymbol{C_i^p(t)}$	Available CPU at time $\boldsymbol{t}$ on Physical Node $\boldsymbol{i}$
$\boldsymbol{B_{ij}^p(t)}$	Available Bandwidth at time $\boldsymbol{t}$ on Physical Link $\boldsymbol{ij}$
$\boldsymbol{G^v(k)}$	Virtual Network Request $\boldsymbol{k}$
$\boldsymbol{N^v(k)}$	Set of Virtual Nodes of VN Request $\boldsymbol{k}$
$\boldsymbol{L^v(k)}$	Set of Virtual Links of VN Request $\boldsymbol{k}$
$\boldsymbol{L_m^v(k)}$	Set of Virtual Links directly connected to Virtual node $\boldsymbol{m}$ of VN Request $\boldsymbol{k}$
$\boldsymbol{m}, \boldsymbol{n}$	Virtual Nodes
$\boldsymbol{mn}$	Virtual Link
$\boldsymbol{C_m^v(k)}$	CPU of Virtual Node $\boldsymbol{m}$ of VN Request $\boldsymbol{k}$
$\mathbf{Dis}_{\boldsymbol{mn}}^{\boldsymbol{v}}\boldsymbol{(k)}$	Maximum Distance Between Virtual Nodes $\boldsymbol{mn}$ of VN Request $\boldsymbol{k}$
$\boldsymbol{B_{mn}^v(k)}$	Bandwidth of Virtual Link $\boldsymbol{mn}$ of VN Request $\boldsymbol{k}$
$\boldsymbol{D_{mn}^v(k)}$	Delay of Virtual Link $\boldsymbol{mn}$ of VN Request $\boldsymbol{k}$

onto one physical path. Each physical link candidate belonging to the physical path needs to have, at least, the same amount of bandwidth available as required by the virtual link which is presented in equation (D.5).

$$\forall \boldsymbol{ij} \subseteq \boldsymbol{\mathcal{M}}[\boldsymbol{mn} \in \boldsymbol{L^v(k)}] : \boldsymbol{B_{mn}^v(k)} \leq \boldsymbol{B_{ij}^p(t)} \tag{D.5}$$

D.3.4 VN Request Life Cycle

The embedding process begins upon a new VN request arrival, which is depicted in Figure D.2. A VN mapping method (e.g., heuristic) is used to embed the VN; it takes as inputs the current status of the physical network (e.g. available CPU capacity and existing bandwidth) and the VN request itself. If the result of the mapping process is a viable solution, the mapping is considered to be feasible; if not, it is considered to be unfeasible and the VN embedding process stops.

D.3.5 Mapping Metrics

In order to assess the performance of an embedding method, and at the same time to compare it with others, different performance metrics were defined.

D.3.5.1 VN Request Acceptance Ratio

The VN request acceptance ratio, $\mathcal{A}^{\mathbf{VN}}$, is given by equation (D.6) and defines the overall performance of an embedding method: the number of VN requests accepted, k', over the number of all VN requests, k.

$$\mathcal{A}^{\mathbf{VN}} = \frac{k'}{k} \tag{D.6}$$

D.3.5.2 Embedding Factor

The embedding factor, $\mathcal{E}^{\mathbf{VN}}$, is given by equation (D.7) and represents the ratio between the amount of virtual resources that were requested for the VN and the amount of physical resources that were effectively provisioned to accommodate that VN, i.e. the efficiency on embedding. The parameters, α, β, γ and η, are used to weight the different types of resources.

$$\mathcal{E}^{\mathbf{VN}} = \frac{\alpha \sum_m C^v_m + \beta \sum_{mn} B^v_{mn}}{\gamma \sum_i C^p_i + \eta \sum_{ij} B^p_{ij}} \tag{D.7}$$

D.4 Virtual Network Embedding - Mathematical Formulation

This section describes the mathematical formulation developed to solve the online VN embedding problem with the defined constraints.

An Integer Linear Programming (ILP) approach is used to solve the online VN embedding problem; we propose a node-link formulation, and two assignment variables are applied during the embedding process. The index notation used here is the same as in section D.3.1.3.

D.4.1 Assignment Variables

The binary variable x is used in the mapping of the virtual nodes and is defined in equation (D.8), where $x^m_i \rightarrow N^V \times N^P$ matrix. With respect to the virtual links, the binary variable y is used and it is represented in equation (D.9), where $y^{mn}_{ij} \rightarrow (L^V)^2 \times (L^P)^2$ matrix.

D.4.1.1 Virtual Node Assignment

$$x^m_i = \begin{cases} \mathbf{1}, \text{virtual node } m \text{ is allocated at physical node } i \\ \mathbf{0}, else \end{cases} \tag{D.8}$$

D.4.1.2 Virtual Link Assignment

$$y^{mn}_{ij} = \begin{cases} \mathbf{1}, \text{virtual link } mn \text{ uses physical link } ij \\ \mathbf{0}, else \end{cases} \tag{D.9}$$

D.4.2 Constraints

To assure the correct mapping of the virtual nodes and of the virtual links, and also to obey to the conservation law on the capacities of the physical nodes and physical links, a set of constraints is defined.

D.4.2.1 Assignment of virtual nodes to physical nodes

Equation (D.10) ensures that each virtual node is assigned, and that it is assigned to just one physical node.

$$\forall \boldsymbol{m} : \sum_{\boldsymbol{i}} \boldsymbol{x}_{\boldsymbol{i}}^{\boldsymbol{m}} = \mathbf{1} \tag{D.10}$$

D.4.2.2 One virtual node per physical node

Equation (D.11) guarantees that each physical node can accommodate in the maximum one virtual node per VN request, although each physical node can accommodate other virtual nodes from different VNs. This constraint is used to ensure that each virtual node is assigned to a different physical node per VN embedding, and can be suitable in application scenarios where it is required to have physical node diversity for redundancy reasons.

$$\forall \boldsymbol{i} : \sum_{\boldsymbol{m}} \boldsymbol{x}_{\boldsymbol{i}}^{\boldsymbol{m}} \leq \mathbf{1} \tag{D.11}$$

D.4.2.3 CPU conservation

Equation (D.12) assures that the available CPU capacity of each physical node is not exceeded.

$$\forall \boldsymbol{i} : \sum_{\boldsymbol{m}} \boldsymbol{x}_{\boldsymbol{i}}^{\boldsymbol{m}} \cdot \boldsymbol{C}_{\boldsymbol{m}}^{\boldsymbol{v}} \leq \boldsymbol{C}_{\boldsymbol{i}}^{\boldsymbol{p}} \tag{D.12}$$

D.4.2.4 Virtual Node distance

Equation (D.13) assures that the maximum distance between virtual nodes, $\boldsymbol{D}_{\boldsymbol{mn}}^{\boldsymbol{v}}$, is not violated. The maximum distance between virtual nodes is a parameter of the VN embedding problem. The effect of this parameter on the VN embedding will be studied on a separate section (see D.6.3).

This parameter is given in distance units and can be used to express the maximum radius between virtual nodes (in the simulated scenario the location of the physical nodes is set in a grid). The distance between physical nodes, i.e. $\mathbf{Dis}_{\boldsymbol{ij}}^{\boldsymbol{p}}$, is obtained using equation (D.1), and $\boldsymbol{K}$ represents a large constant which is used only in situations where the virtual node $\boldsymbol{n}$ is not mapped at physical node $\boldsymbol{i}$, i.e. $\boldsymbol{x}_{\boldsymbol{i}}^{\boldsymbol{n}} = \mathbf{0}$.

$$\begin{aligned} &\forall \boldsymbol{m}, \boldsymbol{n} \in \boldsymbol{L}_{\boldsymbol{m}}^{\boldsymbol{v}}, \boldsymbol{m} < \boldsymbol{n}, \forall \boldsymbol{i} : \\ &\sum_{\boldsymbol{j}} \mathbf{Dis}_{\boldsymbol{ij}}^{\boldsymbol{p}} \cdot \boldsymbol{x}_{\boldsymbol{j}}^{\boldsymbol{m}} \leq \mathbf{Dis}_{\boldsymbol{mn}}^{\boldsymbol{v}} \cdot \boldsymbol{x}_{\boldsymbol{i}}^{\boldsymbol{n}} + (\mathbf{1} - \boldsymbol{x}_{\boldsymbol{i}}^{\boldsymbol{n}}) \cdot \boldsymbol{K} \end{aligned} \tag{D.13}$$

D.4.2.5 Assignment of virtual links to physical links - multi-commodity flow conservation with node-link formulation

To simultaneously optimize the mapping of virtual links and virtual nodes, the multi-commodity flow constraint [EIS75] is applied with a node-link formulation [PM04]; moreover, the notion of direct flows on the virtual links is used, which is represented in Eq. (D.14), where $\boldsymbol{L}_{\boldsymbol{m}}^{\boldsymbol{v}}$ represents all the virtual links that are directly connected with the virtual node $\boldsymbol{m}$, and $\boldsymbol{L}_{\boldsymbol{i}}^{\boldsymbol{p}}$ represent all the physical links that are directly connected with the physical node

i.

$$\forall mn \in L_m^v, m < n, \forall i : \\ \sum_{ij \in L_i^p} (y_{ij}^{mn} - y_{ji}^{mn}) = x_i^m - x_i^n \tag{D.14}$$

D.4.2.6 Bandwidth conservation

To ensure that the available bandwidth at each physical link is not surpassed, Equation (D.15) is defined.

$$\forall ij \in L_i^p, i < j : \\ \sum_{mn \in L_m^v, m<n} B_{mn}^v (y_{ij}^{mn} + y_{ji}^{mn}) \leq B_{ij}^p \tag{D.15}$$

D.4.2.7 Link delay limit

The virtual link delay, D_{mn}^v, is a parameter of the VN embedding problem, and is equal to the sum of the delay of all physical links that compose the virtual link. To ensure that the constraint on the link delay is not violated we apply equation (D.16).

$$\forall mn \in L_m^v, m < n, \forall i : \\ \sum_{ij \in L_i^p, i<j} D_{ij}^p (y_{ij}^{mn} + y_{ji}^{mn}) \leq D_{mn}^v \tag{D.16}$$

D.5 Virtual Network Assignment - Objective Function

One of the major challenges when formulating an ILP model for VN assignment resides in the definition of the objective function: the allocation of resources need to be optimized in order to support the efficiency of the corresponding VN process.

Moreover, the correct specification of the VN mapping constraints (see Section D.4) is also a challenge of this approach. In this section, we describe the main goals that need to be achieved when formulating an objective function for virtual network embedding; three different objective functions are proposed to achieve these goals.

D.5.1 Objective Goals

A primary goal for the embedding algorithm is to minimize resource consumption in order to have resources available for forthcoming VN embedding requests. Minimization of resource consumption is only possible for the bandwidth consumption depending on the number of links involved in an embedding process. The processing power has just to be installed exactly in the amount required by the VN request on some physical nodes.

Resource minimization consequently means that the VN's should exhibit minimal hop counts on their paths. This in turn means that almost every physical node should be available to host a virtual node. As long as the resources required by VN's are small compared to physical capacities of nodes and links, this availability is guaranteed with high probability by a load balancing strategy, which results in some spare capacity for each physical node or link.

Therefore, the dominating aspects in the formulation of an objective function for the ILP problem are the minimization of bandwidth consumption and load balancing.

D.5.2 Load Balancing plus ϵ Shortest Path

The objective function Load Balancing plus ϵ Shortest Path (LB+ϵSP) is proposed in equation (D.17), and it achieves two goals: the primary goal is to minimize the maximum load per physical resources; in the case of different mapping solutions with the same maximum utilization, the second part of the objective function is activated which will opt for the solution which consumes the lowest bandwidth. $\mathcal{L}^{C^p}_{max}$ represents the overall maximum node load; $\mathcal{L}^{B^p}_{max}$ represents the overall maximum link load. The parameters $C^p_i(0)$, $B^p_{ij}(0)$, $C^v_m(k)$, $B^v_{mn}(k)$ were defined in Table D.1; the parameter ϵ represents a small constant, which should be small enough to not affect the first objective; and the parameters α and β are used to weight the load cost of each type of resources.

$$
\begin{aligned}
&minimize\ \alpha \cdot \mathcal{L}^{C^p}_{max} + \beta \cdot \mathcal{L}^{B^p}_{max} + \epsilon \cdot \sum_{mn} y^{mn}_{ij} \cdot B^v_{mn}(t), \\
&\forall i \in N^p : \frac{C^p_i(t) + \sum_m x^m_i \cdot C^v_m(k)}{C^p_i(0)} \leq \mathcal{L}^{C^p}_{max} \\
&\forall ij \in L^p : \frac{B^p_{ij}(t) + \sum_{mn} y^{mn}_{ij} \cdot B^v_{mn}(k)}{B^p_{ij}(0)} \leq \mathcal{L}^{B^p}_{max}
\end{aligned}
\tag{D.17}
$$

D.5.3 Shortest Distance Path

The previous objective function (D.17) works well in situations where there are abundant resources in the physical network. Then, bandwidth consumption is of no concern and load balancing is beneficial because it gives a high degree of flexibility in the resource allocation process.

Nevertheless, in situations where the physical resources are scarce, it is desirable to reduce the number of physical links consumed to the minimum possible.

Therefore, the objective function SDP, proposed in equation (D.18), aims to minimize the number of physical links consumed due to the VN embedding, while it prefers physical links with more available bandwidth, and at the same time chooses physical nodes with more available CPU power, thereby supporting the load balancing aspect. The parameters α and β are used to weight the cost of each type of resource. (Note that the first term in equation (D.18) would result in a constant, if $C^p_i(t)$ was missing in the denominator.)

$$
minimize\ \alpha \left(\sum_m \sum_i \frac{x^m_i}{C^p_i(t)} \right) + \beta \left(\sum_{mn} \sum_{ij} \frac{y^{mn}_{ij}}{B^p_{ij}(t)} \right) \tag{D.18}
$$

D.5.4 Weighted Shortest Distance Path

The objective function WSDP, proposed in equation (D.19), is similar to equation (D.18), although here the demanded capacity by the VN is included in the objective function. This has the effect that high demands are allocated to nodes or links with a large amount of free capacity.

$$\boldsymbol{minimize}\ \boldsymbol{\alpha}\left(\sum_{m} \boldsymbol{C_m^v(k)}\left[\sum_{i} \frac{\boldsymbol{x_i^m}}{\boldsymbol{C_i^p(t)}}\right]\right) + \boldsymbol{\beta}\left(\sum_{mn} \boldsymbol{B_{mn}^v(k)}\left[\sum_{ij} \frac{\boldsymbol{y_{ij}^{mn}}}{\boldsymbol{B_{ij}^p(t)}}\right]\right) \tag{D.19}$$

D.6 Evaluation Results

In this section, we describe the simulation scenario, the evaluation metrics, and depict our major results. We compare the VNE-NLF model in its several versions with six state of the art methods.

D.6.1 Simulation Parameters

To evaluate the VNE-NLF model, we have implemented a discrete event simulator in Matlab®, with the proposed formulation using different objective functions.

The physical network topology is created using the GT-ITM tool [ZCB96], the number of physical nodes is set to 50, which is representative of a medium scale infrastructure provider, and the link probability between two physical nodes is set to 0.5. The node CPU capacity and the link bandwidth are real numbers uniformly distributed between 50 and 100. The VNs requests are also representative of either small or medium scale virtual networks, and are created using the same topology generation method; the number of virtual nodes is not fixed, but follows a uniform distribution, from 2 to 10 virtual nodes per VN topology; the virtual link probability is set to 0.5. The CPU capacity of the virtual nodes and the bandwidth of the virtual links are also real numbers uniformly distributed between 0 and 20, and between 0 and 50, respectively[2]. The considered values for the bandwidth and for the CPU are normalized, since the objective function aims at simultaneously optimizing the allocation of both types of resources.

We assume that VN requests arrive according to a Poisson process, and that each VN has an associated lifetime measured in time units with an average of $1/\boldsymbol{\mu} = \mathbf{1000}$, following an exponential distribution. The same assumption was also taken by the authors of [CRB09]. The average number of VN requests per time unit, i.e., value of $\boldsymbol{\lambda}$, is started with 3 VN requests per 100 time units, and increases by 1 VN request, up to 10 requests. This can give an insight into two opposite case scenarios, with a very high and very low acceptance ratio. For each value of $\boldsymbol{\lambda}$, 10 trials are performed. A new set of VN requests and a new physical network topology are generated for each trial. All simulations are set to run up to 50000 time units to mitigate the transient phase effect [Jai91] and to obtain the steady-state. A confidence interval of 95% is used for all results presented below.

The evaluated embedding methods are G-SP[ZA06], G-MCF[YYRC08], R-ViNE[CRB09], D-ViNE[CRB09], D-ViNE-SP[CRB09], D-ViNE-LB[CRB09], and the proposed linear programming formulation, i.e. VNE-NLF, with 3 different cost functions which were described in the previous section. All these methods are briefly summarized in Table D.2.

The state of the art methods are simulated using an existing implementation [vin12]; to solve the mixed integer programming on the methods G-MCF, R-ViNE, D-ViNE, D-ViNE-LB, and D-ViNE-SP, the GLPK [glp12] solver version 4.20 is used.

[2]These values were also considered by the authors of [YYRC08, ZA06, CRB09]

All the simulations for the different embedding methods were performed using an Intel® Xeon® CPU X3220@2.4GHz, and the time consumed per VN request embedding was registered.

Table D.2: Compared VN Embedding Methods.

Notation	Method Description
G-SP [ZA06]	Greedy Node Mapping with Shortest Path Based Link Mapping.
G-MCF [YYRC08]	Greedy Node Mapping with Splittable Link Mapping using MCF.
R-ViNE [CRB09]	Randomized Node Mapping with Splittable Link Mapping using MCF.
D-ViNE [CRB09]	Deterministic Node Mapping with Splittable Link Mapping using MCF.
D-ViNE-SP [CRB09]	Deterministic Node Mapping with Shortest Path Based Link Mapping.
D-ViNE-LB [CRB09]	Deterministic Node Mapping with Splittable Link Mapping using MCF, where $\boldsymbol{\alpha_{uv} = \beta_{uv} = 1, \forall u, v, w \in N^S}$.
VNE-NLF-LB+ϵSP	VN Embedding with node-link Formulation using overall Load Balancing; in the case of having more than one solution, it uses Shortest Path, where $\boldsymbol{\epsilon = 1.0 \times 10^{-11}}$.
VNE-NLF-SDP	VN Embedding with node-link Formulation using overall Short Distance Path.
VNE-NLF-WSDP	VN Embedding with node-link Formulation using overall Weighted Short Distance Path.

The CPLEX®[cpl12] version 12.2 was used to solve the linear programming problem of the VNE-NLF; a time limit of 600 seconds is defined for each VN mapping, although most of the VNs are embedded in hundreds of milliseconds; and the CPLEX® was set to use only one CPU core for comparison purposes with the remaining methods. The evaluation metrics are the ones defined in section D.3.5.

D.6.2 Impact of the Number of VN Requests

This sub-section presents the evaluation results as a function of the VN request rate, for all the previously described metrics. To increase the readability of all figures, we have considered different $\boldsymbol{x}$ values for different strategies, e.g.: 3.4, 4.4, 5.4 for G-SP; 3.3, 4.3, 5.3 for G-MCF; 3.2, 4.2, 5.2 for R-ViNE.

Before comparing the different embedding methods and algorithms, we should group them into four different categories according to the nature of the method itself, i.e. heuristic, heuristic combined with mixed integer programming, and linear programming:

i Heuristic - the VN embedding problem is solved using a simple algorithm; this method performs the VN embedding very ***fast*** and a possibly sub-optimal embedding solution is obtained. The method G-SP [ZA06] fits into this category;

ii Heuristic combined with Mixed Integer Programming (MIP) - the VN embedding problem is solved in two steps: in the first step a mathematical algorithm is used to map virtual nodes on physical nodes, and in the second step the MIP is performed to embed the virtual links. The method G-MCF [YYRC08] fits into this category.

iii Heuristic combined with Mixed Integer Programming (MIP) and a better coordination between mapping phases is added - the same principle is applied, as in the above category, to solve the VN embedding problem, although a better coordination between the mapping

phases is achieved using an augmented "***substrate graph construction***" [CRB09]. The methods R-ViNE, D-ViNE, D-ViNE-SP, D-ViNE-LB [CRB09] fit into this category;

iv Integer Linear Programming (ILP) - the VN embedding problem is solved using integer linear programming. This method obtains an optimal solution for a given cost function combining resource consumption minimization with a load balancing strategy. The method VNE-NLF and its different objective functions fit into this category.

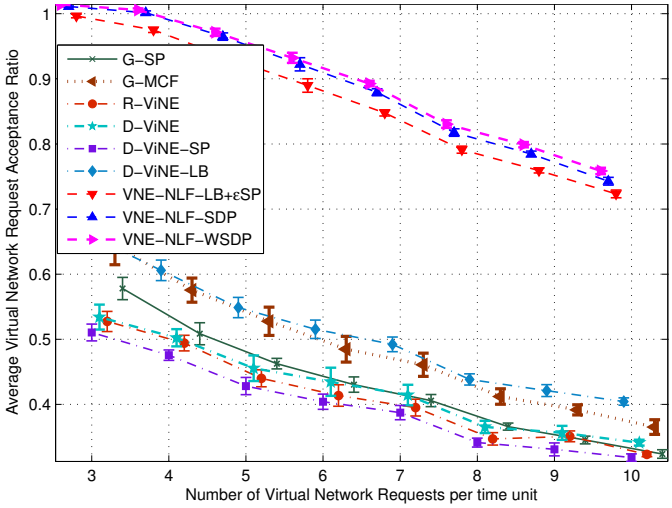


Figure D.3: Average VN Acceptance ratio as a function of VN Request rate.

D.6.2.1 VN Request Acceptance Ratio

One of the main aspects of the performance of each embedding method is the VN request acceptance ratio, which is shown in Figure D.3 and is given by equation (D.6). As can be observed, all methods show a linear behaviour with the variation on the VN requests, where the acceptance ratio decays linearly with the number of VN requests, and the slope is approximately the same for all methods. This decay represents the fact that there are no infinite physical resources.

The method VNE-NLF, with its different objective functions, achieves the highest performance, and it clearly outperforms the other approaches. This is expected since integer linear programming is applied to solve the VN embedding problem, and the optimal solution, according to the objective function considered, is obtained per VN embedding.

The reason for these results, not only resides in the usage of an integer linear programming approach, but also in the utilization of the node-link formulation by the VNE-NLF, which considers the universe of all possible embedding solutions, instead of a few solutions. If we take, for example, the first case with only 3 VN requests per 100 time units, the VNE-NLF is able to accept nearly all requests, while the remaining methods are able to accept only 70% of the requests. The embedding method that has the lowest acceptance ratio is the D-ViNE-SP, and the method with the highest VN request acceptance ratio is the VNE-NLF-WSDP.

It is expected that the VNE-NLF method will perform better in all cases. For instance, if the embedding problem is feasible, i.e., possible solutions exist, the VNE-NLF will find out the optimal solution according to the cost function. Using a heuristic approach or even a combined approach, this is not always the case: frequently only a feasible solution will be presented.

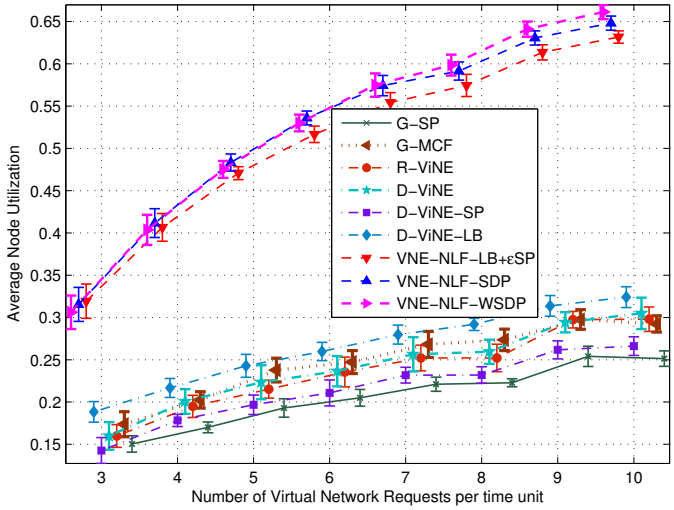


Figure D.4: Average Node Utilization as a function of VN Request rate.

D.6.2.2 Node Utilization

The average node utilization as a function of the number of VN requests is depicted in Figure D.4. With a small number of VN requests, i.e., 3 VN requests, the node utilization does not go beyond 20% and 35% for the heuristic group (i.e. groups ***i***, ***ii***, and ***iii***) and for the VNE-NLF, respectively. The VNE-NLF group is consuming more resources of the physical nodes than the heuristics, which is expected according to the acceptance ratio. When the number of VN requests is increased, the node utilization also increases, since we are trying to accommodate more VNs on the infra-structure, but with the same amount of available physical resources.

An efficient embedding method in situations of high VN demand would be able to load the nodes to their full capacity. The important aspect to retain here is how much node resources can be loaded and what kind of embedding methods tends to saturate them firstly.

The node utilization shows a dependency on the VN acceptance ratio, as it can be perceived from Figure D.3 and Figure D.4. To provide a better understanding on this issue, we plot the acceptance ratio metric times the node utilization, which is shown in Figure D.5. We observe that the methods that make use of heuristics, e.g. G-SP, or heuristics combined with MIP, e.g. G-MCF and D-ViNE-LB, show the same behaviour for all the VN requests considered, i.e. the VN acceptance metric multiplied by the node utilization metric is nearly constant. The same does not apply to the VNE-NLF, since it increases per VN request considered, until 6 VN requests per 100 time units, and beyond the 6 VN requests per 100 time units, it shows the same behaviour as its counterparts. This means that, although the VN request rate is increasing, the VNE-NLF approach is able to keep with this increase until the VN embedding problem moves from an optimization problem (i.e. there are sufficient physical resources for the demand), to a feasibility problem (i.e. there are no sufficient resources for the demand).

D.6.2.3 Link Utilization

The physical link utilization metric is plotted in Figure D.6. Here we do not have the same regular behaviour according to the number of VN requests for all the methods, as shown before for the node utilization. Also, there is no consensus in terms of clearly identifying which

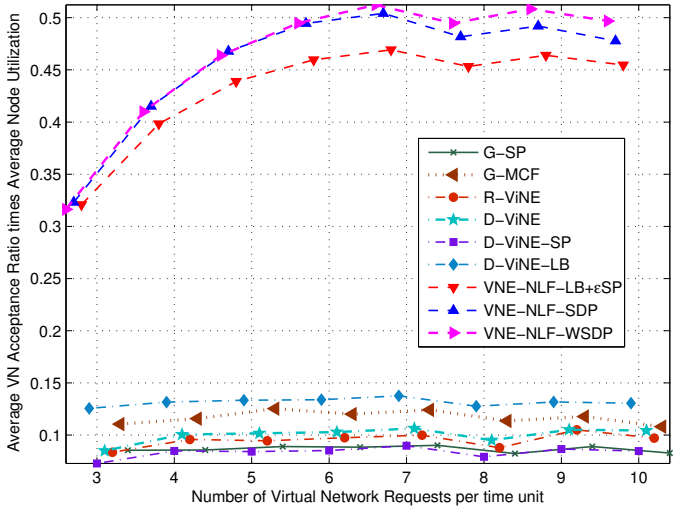


Figure D.5: Average VN Acceptance Ratio times Average Node Utilization as a function of VN Request rate.

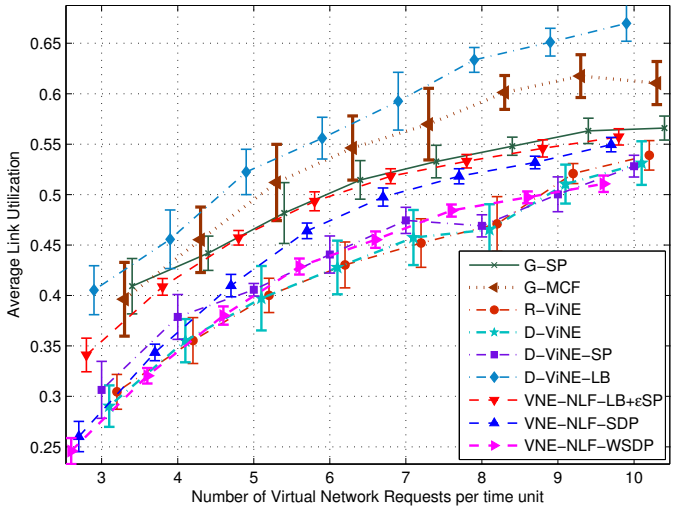


Figure D.6: Average Link Utilization as a function of VN Request rate.

group causes the highest utilization on the physical links due to the embedding process. Nevertheless, we can clearly state that, on average, either the G-MCF or D-ViNE-LB shows the highest utilization on the links; in the extreme case scenario (i.e. with 10 VN requests) they have an average link utilization of 60% and 67%, respectively. With respect to the lowest link utilization, we observe that the embedding methods R-ViNE, D-ViNE, D-ViNE-SP, and VNE-NLF-WSDP are the ones that tend to consume less bandwidth, reaching values of 50% of average link utilization, for the same considered situation.

Having in mind that one virtual link could be mapped in several ways, it is reasonable to observe different behaviours according to the strategy of the method. If the strategy is to save bandwidth, i.e. SP, the embedding will consume the least bandwidth possible per VN mapping; if the strategy is load balancing on the links, i.e. LB, it will tend to balance the utilization among all links in order to distribute the total load.

The link utilization also shows a dependency on the VN acceptance ratio, as can be observed in Figure D.3 and Figure D.6. To provide a better understanding, we plotted the acceptance ratio metric times the link utilization in Figure D.7. In contrast to the node

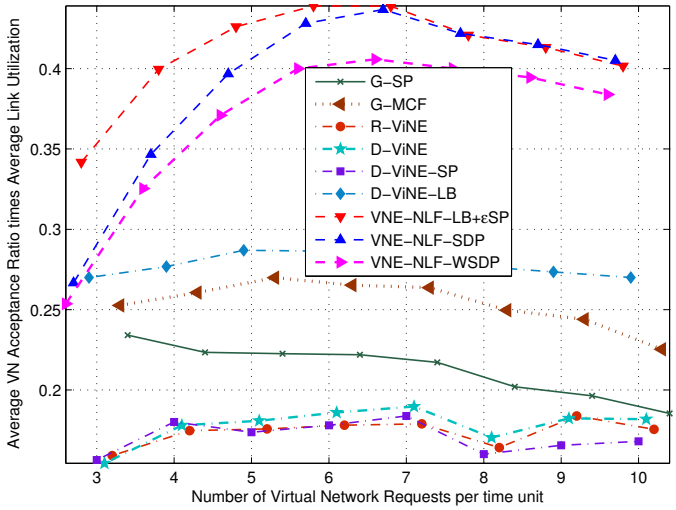


Figure D.7: Average VN Request Acceptance Ratio times Average Link Utilization as a function of VN Request rate.

utilization, the dependency factor on the link utilization shows a more complex behaviour: it still increases significantly until reaching 6 VN requests for the case of the VNE-NLF group, although it starts to decrease after 7 VN requests, in a not so expressive way. For the other methods the dependency on the VN request rate is less pronounced.

D.6.2.4 Embedding Factor

Figure D.8 shows the embedding factor as a function of the VN request rate, where the weight parameters, $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$, and $\boldsymbol{\eta}$ of equation (D.7) are set to 1. The embedding factor slightly decreases with the number of VN requests, except for the case of the VNE-NLF-LB+$\boldsymbol{\epsilon}$SP. The behaviour obtained when using the VNE-NLF-LB+$\boldsymbol{\epsilon}$SP is as expected, since it performs an overall load balancing of the physical nodes and links, choosing the solution that consumes the least bandwidth. Therefore, in the situation of only a few VN requests, the method will tend to allocate more resources than required due to the nature of the load balancing; with a higher VN request rate, this situation tends to disappear once the available resources are scarcer. Therefore, the embedding factor will increase with the number of VN requests. We can also state that the efficiency of the heuristic group, in general, is very low, lower than 50%. With respect to the VNE-NLF group, it has a good efficiency, being in most of the cases higher than 85%; the efficiency of the VNE-NLF-WSDP is closer to 100% which means that, on average, this method provisions the same amount of resources as requested per VN.

D.6.2.5 VN Embedding Time

An important aspect of all the VN embedding methods is the time that they require to embed, on average, a VN request and how it varies with respect to the different loads on the physical infrastructure, i.e. the VN request rate.

Figure D.9 shows the solving time for each method as a function of the number of VN requests per 100 time units.

Before analyzing the figure, one must consider five different aspects: i) all methods have been simulated using the same machine; ii) the time to embed a VN strongly depends on the physical characteristics (e.g. CPU) of that machine; iii) the time to embed a VN strongly

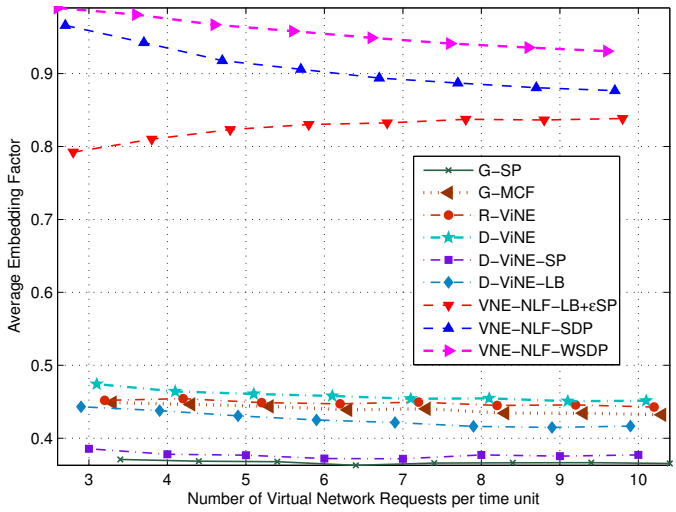


Figure D.8: Average Embedding Factor as a function of VN Request rate.

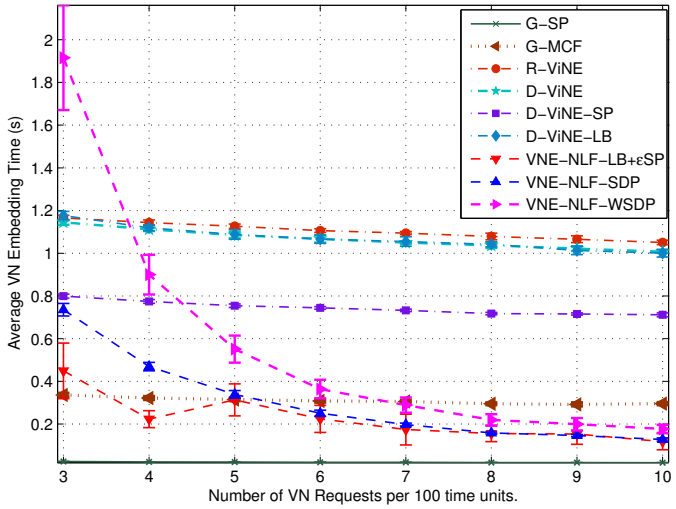


Figure D.9: VN Solving Time as a function of VN Request rate.

depends on the nature of the embedding method (i.e. a mathematical algorithm will take just a few milliseconds, while linear programming is expected to take hundreds of milliseconds); iv) two different linear programming tools (GLPK was used to solve the MIP of G-MCF, R-ViNE, D-ViNE,D-ViNE-SP, D-ViNE-LB; and CPLEX® to solve the ILP of the VNE-NLF); v) methods R-ViNE, D-ViNE,D-ViNE-SP, and D-ViNE-LB perform two linear programming operations, i.e. one for the mapping of the virtual nodes, and another for the mapping of the links.

The fourth aspect, although important for the solving time, will not interfere with the curve behaviour, e.g. polynomial or exponential, since the same method, i.e. branch and cut, is applied by both solvers (i.e. GLPK and CPLEX) to solve the VN embedding problem.

From the figure, we can observe two types of behaviours: the method VNE-NLF with three different costs functions shows a decaying behaviour with the VN request rate; for the remaining methods we observe a nearly constant behaviour.

For the first behaviour, i.e. method VNE-NLF with three different cost functions, one should take into consideration that the VN request acceptance ratio is considerably higher, e.g. 90% until 6 VN requests: more than one mapping solution per VN request is expected

to exist; therefore, the optimization process takes place and will consume the majority of the solving time to obtain the optimal solution.

For the remaining methods, the VN request acceptance ratio is lower and below 70%: usually there is not more than one mapping solution per VN request, on average, which significantly reduces the solving time. This is the case of the methods G-MCF, R-ViNE, D-ViNE, D-ViNE-SP and D-ViNE-LB.

We can also add that the methods R-ViNE, D-ViNE, D-ViNE-SP, and D-ViNE-LB take twice the time on average to embed a VN compared to G-MCF. This is related with the number of MIP problems solved per VN embedding. The latter only considers one MIP problem per VN embedding, while the former ones consider two MIP problems.

The method that performs the embedding in the shortest time has the poorest performance (the G-SP), which solves each VN request embedding problem in an average of 20 milliseconds.

The method that requires the longest time to perform the embedding for the case of 3 VN requests has the highest performance (the VNE-NLF), using the WSDP cost function, which takes less than 2 seconds on average for that case. However, if we increase the load, the situation significantly changes, and the methods R-ViNE, D-ViNE and D-ViNE-LB take more time to obtain the embedding solution. On average, they take 1 second to embed a VN, while the VNE-NLF-WSDP consumes less than 200ms.

D.6.3 Impact of the Maximum Distance Between Virtual Nodes

To evaluate the impact on the overall performance of the different embedding methods due to the restriction on the maximum allowed distance between virtual nodes represented in equation (D.13), a new set of simulation experiments was performed. The VN request arrival rate was fixed to 4 VN requests per 100 time units; the maximum distance between virtual nodes was set to vary between 5 and 20 within intervals of 2.5 distance units; for each considered value of maximum distance, the same set of VN requests was used. The remaining parameters i.e., virtual network size, link probability, and number of nodes were maintained. To increase the readability of all figures, only the best method of each group is presented in this section: G-SP (heuristic), G-MCF (mixed integer programming - link-path), D-ViNE-LB (mixed integer programming with better node-link embedding coordination - link-path), and VNE-NLF-WSDP (integer linear programming - node-link).

D.6.3.1 VN Request Acceptance Ratio

Figure D.10 depicts the VN request acceptance ratio as a function of the distance between virtual nodes. Two different behaviours can be observed:

i The acceptance ratio increases with the distance between virtual nodes: this is the case of the VNE-NLF group. This behaviour is expected if we consider the cases where VN requests were initially not mapped due to the distance constraint; increasing the permitted distance between virtual nodes will in principle result in more accepted VNs.

ii Increasing the distance between virtual nodes decreases the acceptance ratio: this is the case of the assignment methods G-SP, G-MCF, D-ViNE-LB.

D.6.3.2 Node Utilization

Figure D.11 depicts the average physical node utilization as a function of the distance between virtual nodes. The behaviour observed for each method can be compared to that of the VN acceptance ratio.

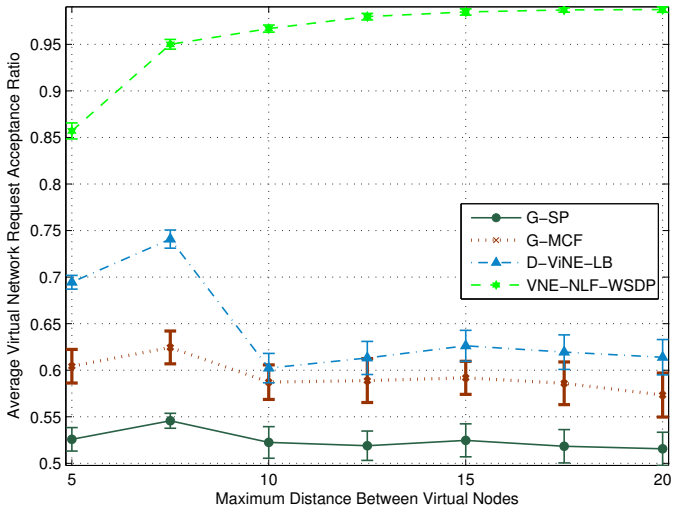


Figure D.10: Average VN Acceptance Ratio as a function of the Distance between Virtual Nodes

D.6.3.3 Link Utilization

The average physical link utilization is shown in Figure D.12, and different behaviours are observed according to the embedding method:

i G-SP and G-MCF methods reduce slightly the link utilization with the distance, and the method D-ViNE-LB maintains the link utilization, despite some fluctuations may be observed.

ii The group of VNE-NLF demonstrates to slightly increase the link utilization with the maximum distance between nodes. This is expected, considering the increase on the VN request acceptance ratio with the distance.

D.6.3.4 Embedding Factor

The embedding factor is depicted in Figure D.13, where three distinct behaviours can be observed:

i The embedding factor slightly decreases with the maximum distance: this is the case of methods G-SP and G-MCF, where these demonstrate to lose efficiency with the considered distance.

ii The embedding factor does not vary with the maximum distance: this is the case of method D-ViNE-LB.

iii The embedding factor increases with the maximum distance: this is the case of the VNE-NLF group. This demonstrates that the VNE-NLF group is able to be more efficient with the relaxation on the distance constraint equation (D.13).

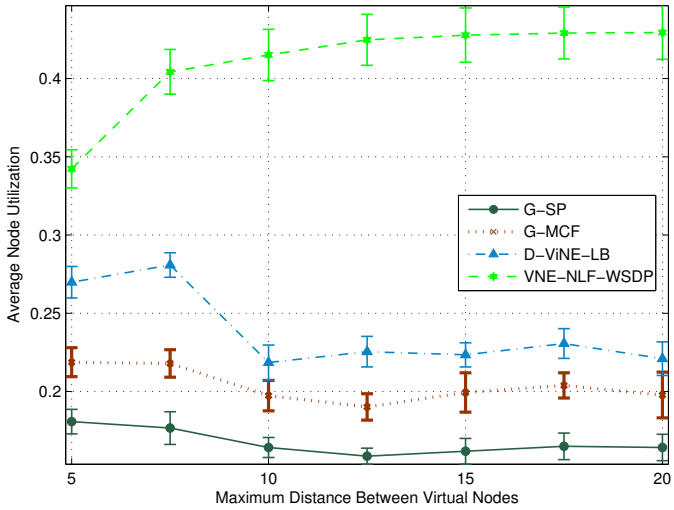


Figure D.11: Average Node Utilization as a function of the Distance between Virtual Nodes

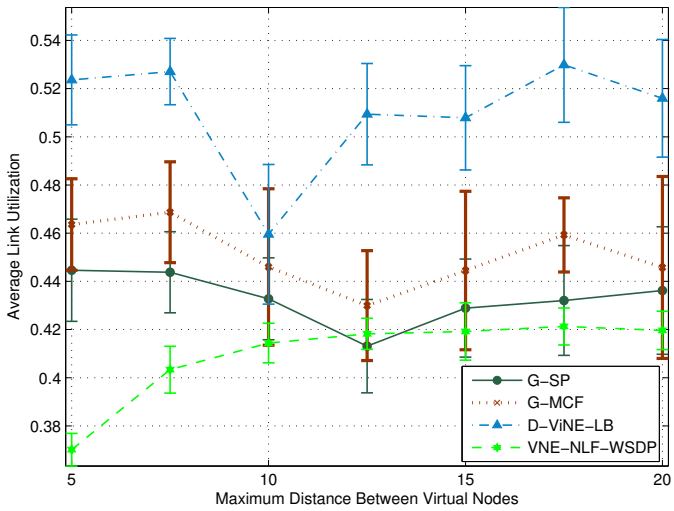


Figure D.12: Average Link Utilization as a function of the Distance between Virtual Nodes

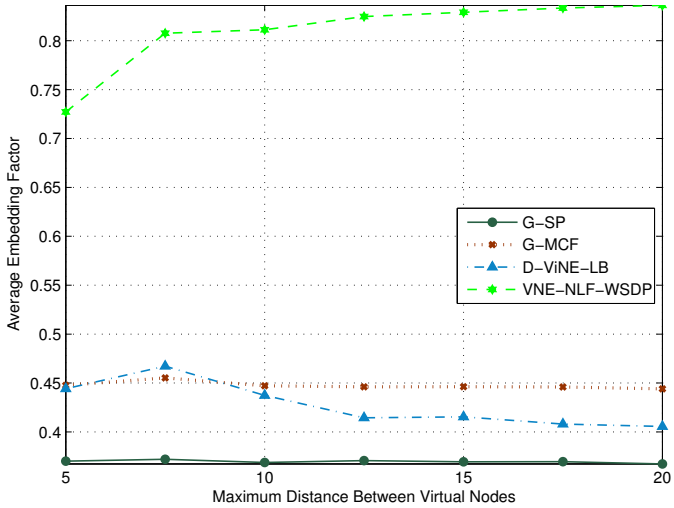


Figure D.13: Average Embedding Factor as a function of the Distance between Virtual Nodes

D.7 Conclusion

This paper proposed the VNE-NLF to solve the VN embedding problem. The model applies optimization theory to simultaneously embed the virtual nodes and the virtual links.

Three new cost functions are proposed: the LB+ϵSP which aims to minimize the overall load on the network per VN embedding; the SDP which aims to minimize the number of physical links consumed, and at the same time it chooses physical nodes with higher availability of resources; and the WSDP which includes the demanded capacity by the VN in the objective function.

Simulation experiments show how far the state of the art heuristics are from an ILP based optimization method. The difference between the performance of the heuristics and the VNE-NLF approach is, at least, 30% for the VN request acceptance ratio (see Figure D.3). The node utilization is also higher when comparing with the existing heuristics, which is expected since we are accommodating more virtual nodes on the network. However, the link utilization is similar to the ones of the heuristics, and in some cases (e.g. G-MCF, D-ViNE-LB) it is lower, which reflects the good efficiency of the embedding when using the VNE-NLF approach. The embedding factor of the VNE-NLF is very high (i.e. it is close to 1). The results also show that the maximum allowed distance between virtual nodes seems to affect differently the performance of each embedding method: for the case of R-ViNE and D-ViNE with its three variants, it does affect negatively the performance of the VN embedding; for the case of G-SP and G-MCF, it does not seem to cause a direct impact on the embedding; and regarding the VNE-NLF approach, it does affect positively the VN embedding.

The VNE-NLF with all its different cost functions proves to be an efficient embedding method. Not only it provides better results, but it also performs the VN embedding faster than the compared heuristics (with exception of G-SP) for a large number of VN requests per time unit. From all the proposed and simulated cost functions, the WSDP is the one that demonstrates the best overall performance.

Future work will endorse VN embedding with multi-objective optimization support, and re-configuration of virtual networks. The definition of cost functions which take into account energy parameters will be also addressed, through the study of the impact on energy saving, i.e. the shutdown of an interface or even an equipment, by applying optimization to solve the VN embedding problem.

Appendix E - Optimal Virtual Network Embedding: Energy Aware Formulation

Márcio Melo, Susana Sargento, Ulrich Killat, Andreas Timm-Giel, and Jorge Carapinha

submitted in *IEEE Transactions on Network and Service Management (TNSM)*, vol. NA , Issue: NA 2014.

The format has been revised.

Optimal Virtual Network Embedding: Energy Aware Formulation

Márcio Melo, Susana Sargento, Ulrich Killat, Andreas Timm-Giel, and Jorge Carapinha

Abstract

Network Virtualisation is a key component of the Future Internet, providing the dynamic support of different networks with different paradigms and mechanisms in the same physical infrastructure. A major challenge in the dynamic provision of virtual networks is the embedding approach taking energy efficiency into account, while not affecting the overall VN acceptance ratio. Previous research focused on either designing heuristic-based algorithms to address the efficient embedding problem or to address the energy impact.

This paper proposes an integer linear programming formulation, Energy Aware - Virtual Network Embedding - Node-Link Formulation (EA-VNE-NLF) that solves the online virtual network embedding as an optimization problem, providing both the minimum energy consumption and optimal resource allocation per VN mapping. Two different objective functions are proposed: i) addressing primarily the resource consumption problem - BCM; ii) addressing primarily the energy consumption problem - ECM.

The performance of each objective function is evaluated by means of simulation and compared with an existing objective function, WSDP, that is considered state of the art on the resource allocation problem. The simulation results show that the objective function BCM reduces the energy consumption of the physical network by 14.4%, and the embedding factor by 4.3%, consuming almost the same amount of resources as requested per VN, and sightly worsening the VN acceptance ratio by 2.3%. ECM reduces the energy consumption of the physical network by 31.4% and improves the embedding factor by 4.1%, without affecting the VN acceptance ratio when compared to WSDP.

E.1 Introduction

NOWADAYS, network operators are required to pay more attention to the power consumption of their networks, either due to environmental policies imposed by the local governments or due to energy costs [AN08]. In fact, the power consumption of the data plane in idle mode is 90% the one in full mode [CSB+08, LGL+11], while the power consumption of the control plane, i.e. CPU, is about 70% [BH09] the one in idle mode.

Several procedures can be taken to reduce the energy consumption, such as turning nodes that are not being used into sleep mode or by using "green" protocols. In [NPI+08a] the impact on network protocols by turning network interfaces and components into sleep mode for saving energy is discussed, and in [GS03a] two forms of power management schemes are presented and evaluated that reduce the energy consumption of networks. Not only the power consumption of each network equipment ***per se*** is important, but also the power consumption of the service itself. The energy consumption of the data plane can be derived from the virtual links allocation, and the one of the control plane can be derived from the virtual nodes, if we consider the data and control planes as two different power consumption sources.

Network virtualisation will trigger the development of green protocols and will facilitate an optimal load distribution in the network: virtual nodes may be concentrated on certain physical nodes, thereby creating unused resources which can be turned into sleep mode. It is important that all provisioned services, i.e. virtual networks, are provisioned in the smallest number of physical nodes and links to save energy, and therefore, to reduce the $\boldsymbol{CO_2}$ footprint, without affecting the reliability of the network.

One of the major obstacles for operators lies in the energy efficient embedding[1] of a Virtual Network (VN) onto a physical network, while maintaining the same levels of VN acceptance ratio. Since this is a multi-objective problem, it requires the simultaneous optimization of: i) resource allocation and ii) energy consumption. Previous research works, such as [ZA06, YYRC08, LK09, NMCS11b, CRB12] focused on the efficient embedding, while some recent research works, such as [SZC+12, BHD+12], focused on the energy consumption aspects. However, most of them either do not take into account both objectives, or do not solve it as an optimization problem, leading to non optimal embedding solutions.

This paper focuses on the online embedding of VN requests in the physical network taking energy constraints into account. An ILP formulation, the Energy Aware - Virtual Network Embedding - Node-Link Formulation (EA-VNE-NLF), is used to solve the VN assignment problem on the basis of an optimization of resource allocation and energy consumption. In addition, different cost functions are proposed and analyzed, which either primarily enforce bandwidth consumption minimization, energy consumption minimization or both objectives. The performance of each objective function is evaluated by means of simulation and compared with an existing objective function, WSDP, that is considered state of the art on the resource allocation problem. The simulation results show that both objective functions significantly reduce the energy consumption of the physical network, consuming similar amount of resources as requested per VN, and with similar VN acceptance ratios.

Compared to our previous work in [MSK+13], this paper:

i extends the mathematical formulation to support energy parameters;

ii proposes two cost functions, Bandwidth Consumption Minimization (BCM) and Energy Consumption Minimization (ECM) that take into account the energy consumption minimization;

[1] The terms embedding, mapping and assignment are used interchangeably in this paper.

iii provides a performance comparison with an objective function, i.e. Weighted Shortest Distance Path (WSDP), which is a state of the art approach for virtual network embedding.

The rest of the paper is organized as follows. After summarizing the related works in E.2, section E.3 describes the virtual network embedding problem and the evaluation metrics. Section E.4 describes the Energy Aware - Virtual Network Embedding - Node-Link Formulation (EA-VNE-NLF) and the applied constraints, while section E.5 describes the VN embedding and discusses different objective functions. Section E.6 analyzes the performance of the EA-VNE-NLF with different cost functions, and section E.7 concludes the paper and describes the future work.

E.2 Related Work

The VN embedding problem can be formulated as an un-splittable flow problem [ZA06] of resource allocation and energy consumption optimization. In order to solve this problem, several approaches have been suggested, mostly considering the resource consumption aspect.

The work in [LT06] defined a set of premises about the virtual topology, i.e. the backbone nodes are star-connected and the access-nodes connect to a single backbone node. Based on these premises, an iterative algorithm is run, with different steps for core and access mapping. However, the algorithm can only work for specific topologies.

A distributed algorithm was studied in [HLZ08]. It considers that the virtual topologies can be decomposed in hub-and-spoke clusters, and each cluster can be mapped independently, therefore reducing the complexity of the full VN mapping. This proposal has lower performance when compared with centralized approaches.

Zhu ***et al.*** [ZA06] proposed a heuristic based on a centralized algorithm to deal with VN mapping. The goal of the algorithm is to maintain a low and balanced load of both nodes and links of the substrate network. Yu ***et al.*** [YYRC08] proposed a mapping algorithm which considers finite resources in the physical network, and enables path splitting (i.e. virtual link composed by different paths) and link migration (i.e. to change the underlying mapping) during the embedding process. However, this level of freedom can lead to a level of fragmentation that is unfeasible to manage in large scale networks.

In [CRB09] a formal approach is taken to solve the on-line VN mapping problem using a mixed integer programming formulation. Chowdhury ***et al.*** applied a two step approach to embed VNs on the substrate. In the first step, the virtual nodes are assigned to physical nodes, and in the second step the virtual links are assigned to physical paths. Compared to the previous state of the art heuristics, i.e. [ZA06, YYRC08], the formulation proposed by Chowdhury ***et al.*** provides a better coordination of the two phases, since an "augmented substrate graph construction" is used.

The approach in [CRB09] completely differs from the mathematical formulation proposed in this paper, which applies a node-link formulation. In our approach, the universe of embedding solutions is considered within the ILP formulation, and the VN embedding problem is solved in a single step using the multi-commodity flow constraint and by considering the notion of direction of the flows.

Butt ***et al.*** [FBCB10] proposed a topology aware heuristic for VN mapping, and also suggested algorithms to avoid bottlenecks on the physical infrastructure, where they consider virtual node reallocation and link reassignment for this purpose. Nogueira ***et al.*** [NMCS11b] proposed a heuristic that takes into account the heterogeneity of the VNs and also of the physical infrastructure. The heuristic is evaluated by means of simulation and also on a

small scale testbed, where it achieves mapping times of the order of tens of milliseconds. Botero ***et al.*** [BHFM12] proposed an algorithm to solve the VN mapping problem, which also considers the CPU demand of the hidden hops. Chowdhury ***et al.*** [CRB12] extended his preliminary results [CRB09] and included a generalized window-based VN embedding to evaluate the effect of look ahead on the mapping of VNs. Alkmim ***et al.*** [ABdF13] proposed a mathematical formulation that aims to: i) map virtual routers and virtual links; ii) minimize the bandwidth consumption; and iii) minimize the time required to instantiate a virtual router.

In our previous research work [MCS+12], we have proposed an ILP formulation to solve the online VN embedding problem from an optimization standpoint, moreover in [MCS+13b] we have extended the Integer Linear Programming (ILP) formulation to contemplate the re-optimization of VNs currently embedded. An extended approach and the comparison with existing heuristics were recently added in [MSK+13].

The previous referred research works do not take into account energy aspects. Recent approaches focusing on the energy consumption aspects comprise the one of Zhang ***et al.*** [SZC+12], which proposed an efficient energy-aware algorithm using a consolidation technique to reduce the energy consumption. However, a comparison with the optimal solution and baselines is not provided. Botero ***et al.*** [BHD+12] extended their preliminary work and proposed a Mixed Integer Programming (MIP) with the aim of providing optimal energy efficient embeddings. This formulation despite providing the optimal solution for energy embeddings, it does not consider the resource allocation objective, thereby penalizing the VN acceptance ratio.

Although all these algorithms provide a solution for the VN mapping problem, an optimal solution for energy consumption and resource allocation is not provided. Also, some of the proposals fail to solve the assignment problem as a simultaneous optimization of the virtual node and link placement, which leads to non-optimal resource allocation solutions.

Our approach, the EA-VNE-NLF, applies a node-link formulation to solve the VN embedding problem in a single step using the multi-commodity flow constraint. This approach provides the optimal solution for the objective problem considered, both in terms of resource allocation and energy efficiency, for a given set of weights in the objective function.

E.3 Network Description and Problem Formulation

In this section, we introduce the virtual network embedding problem. In addition, the VN embedding notations used throughout the paper are presented, and the virtual network embedding system is explained. Finally, the mapping goals are introduced to support the mathematical formulation. Readers familiar with our previous paper [MSK+13] may skip paragraphs E.3.1-E.3.4.

E.3.1 Network Description

We use superscript to distinguish the physical network from the virtual network, where p and v correspond to physical and virtual, respectively.

E.3.1.1 Physical network

A physical network can be described as a weighted undirected graph $\boldsymbol{G^p = \{N^p, L^p, B^p\}}$ composed by a set of physical nodes, $\boldsymbol{N^p}$, and a set of physical links, $\boldsymbol{L^p}$. Each physical node $\boldsymbol{i}$ is characterized by its processing capacity, $\boldsymbol{C_i^p}$, commonly referred to as the CPU; The node

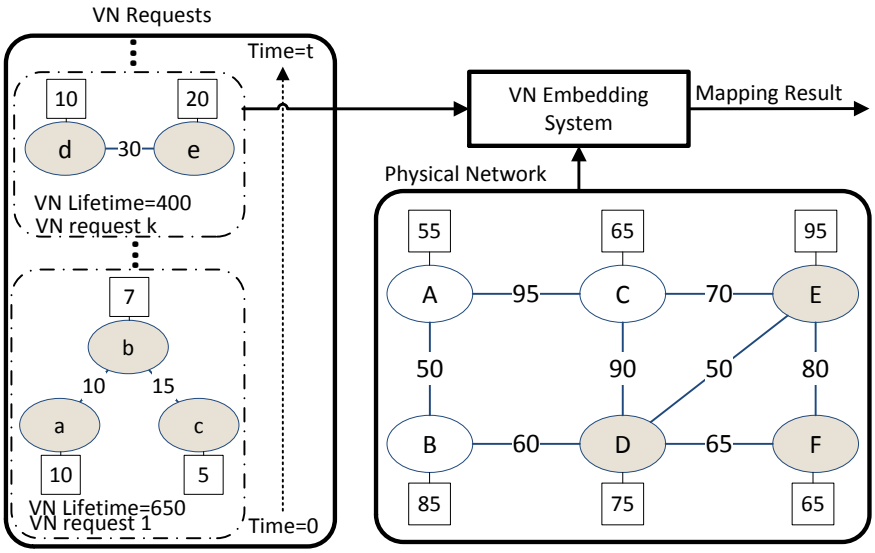


Figure E.1: VN Embedding System - Topology Example

i is also defined by its power state u_i - active if the node is power-up or inactive if the node is power-off, and by its power consumption P_i.

Additionally, the node i is characterized by the role it is performing in the network: hosting node if it accommodates virtual nodes; forwarding node if its physical links accommodate virtual links but the node itself does not accommodate virtual nodes.

With respect to the physical links, we consider that each link ij has a given bandwidth, B_{ij}^{p}; we assume that each link is an undirected link. The bottom-right of Figure E.1 illustrates a physical network topology example composed of 6 physical nodes and 8 physical links, the corresponding capacities of the nodes and the links are presented on top of the elements. The physical nodes power state is represented by white for inactive and gray for active.

E.3.1.2 Virtual Network Request

VN request can be described as a weighted undirected graph $G^v = \{N^v, L^v, B^v\}$ composed by a set of virtual nodes, N^v, and a set virtual links, L^v. Each virtual node m is characterized by the amount of required CPU, C_m^v, and the virtual links mn are logical connections between virtual nodes and characterized by the amount of dedicated bandwidth, B_{mn}^{v}. We also assume that each virtual link is an undirected link. The left part of Figure E.1 represents the example of two virtual network requests, VN request **1** on the bottom-left and VN request k on the top-left. VN requests have a given lifetime which is independent from each other.

E.3.1.3 VN Assignment Notations

First, we start with the convention used for the index notation: N^p represents the set of nodes that belong to the physical network; L^p represents the set of links that belong to the physical network; and L_i^p represents a subset of links ij that are directly connected to the node i. The same type of notation is used to represent the VN using the letters m and n in the virtual network. The notations used throughout this paper for the VN assignment problem are presented in Table E.1. The table is divided into three parts: the static parameters of the

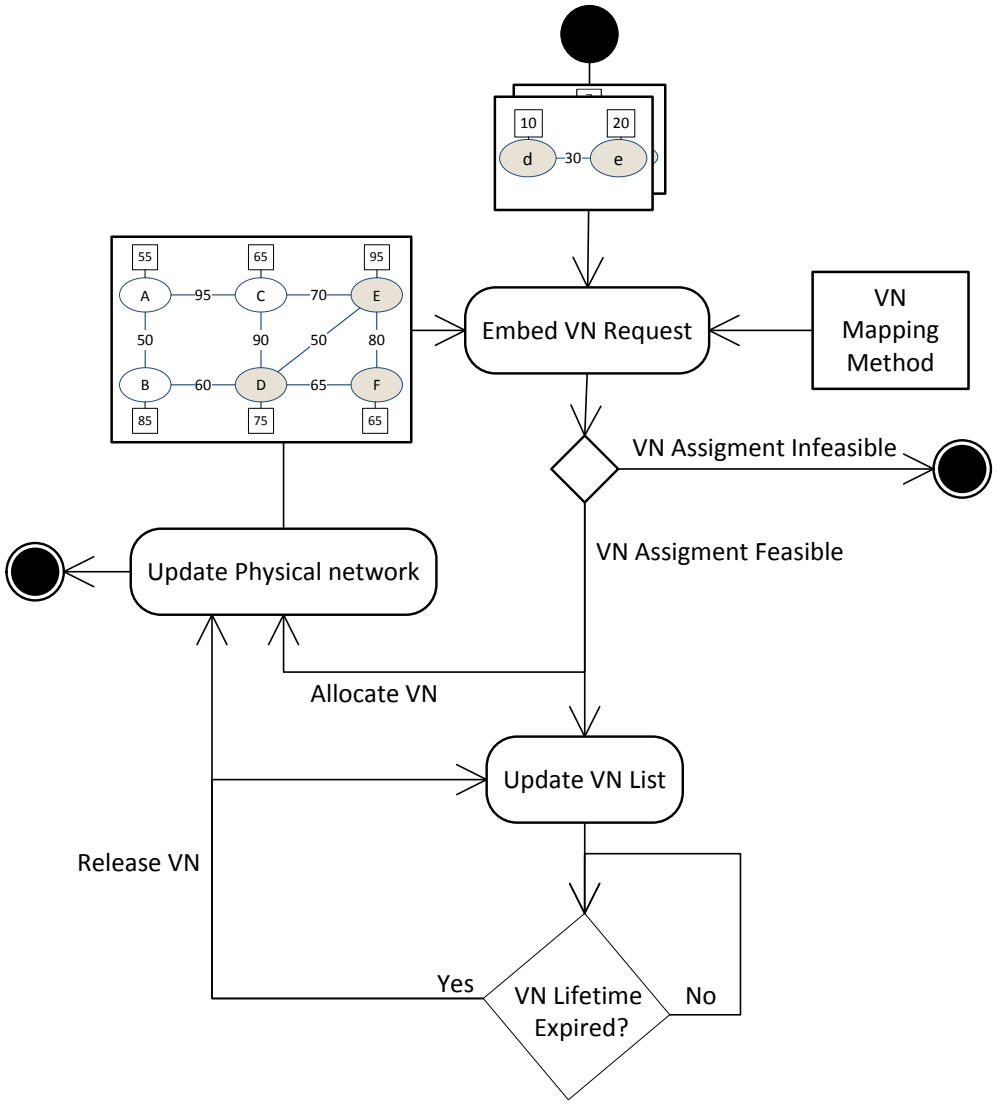


Figure E.2: VN Request Life Cycle - Activity Diagram

physical network, the dynamic parameters of the physical network, and the virtual network requests with the demanded capacities.

E.3.2 Unfilled Physical Network Resources

The remaining capacity of each physical node at a specific time $\boldsymbol{t}$ is given by the difference between the total processing capacity and the capacity consumed by the virtual nodes previously allocated on that physical node up to time $\boldsymbol{t}$, and is presented in equation (E.1), where $\boldsymbol{U}$ represents the set of virtual nodes previously allocated to that physical node.

$$\forall \boldsymbol{i} \in \boldsymbol{N^p} : \boldsymbol{C_i^p(t)} = \boldsymbol{C_i^p(t_0)} - \sum_{\boldsymbol{u \in U}} \boldsymbol{C_u^v(t)} \tag{E.1}$$

The available bandwidth of each physical link at a specific time $\boldsymbol{t}$ is given by the difference between the total bandwidth and the bandwidth consumed by all virtual link segments allocated to that physical link, and is presented in equation (E.2), where $\boldsymbol{W}$ represents the set

Table E.1: VN Assignment Problem Notation.

G^p	Physical Network
N^p	Set of Physical Nodes
$i,\ j$	Physical Nodes
ij	Physical Link
L^p	Set of Physical Links
L^p_i	Set of Physical Links directly connected to Physical Node i
$C^p_i(t_0)$	Available CPU of Physical Node i at time t_0
$B^p_{ij}(t_0)$	Available Bandwidth of Physical Link ij at time t_0
t	Time
$C^p_i(t)$	Available CPU of Physical Node i at time t
$u_i(t)$	Power State of Physical Node i at time t
$P_i(t)$	Power Consumption of Physical Node i at time t
$B^p_{ij}(t)$	Available Bandwidth of Physical Link ij at time t
t_k	Time of Virtual Network Request Event k
k	Virtual Network Request
G^v	Virtual Network
$N^v(t_k)$	Set of Virtual Nodes at time t_k
$L^v(t_k)$	Set of Virtual Links at time t_k
$L^v_m(t_k)$	Set of Virtual Links directly connected to Virtual node m at time t_k
m, n	Virtual Nodes
mn	Virtual Link
$C^v_m(t_k)$	CPU of Virtual Node m at time t_k
$B^v_{mn}(t_k)$	Bandwidth of Virtual Link mn at time t_k

of all virtual link segments allocated to that specific physical link at time t.

A virtual link can be composed of one or more physical links, i.e. a physical path. We consider that each virtual link has a single physical path, and we do not consider link aggregation (i.e. virtual link composed by different physical paths). One physical link can accommodate one or more virtual link segments belonging to different virtual links.

$$\forall ij \in L^p(t) : B^p_{ij}(t) = B^p_{ij}(t_0) - \sum_{w \in W} B^v_w(t) \tag{E.2}$$

E.3.3 VN Request Embedding Process

The VN request embedding process can be divided into two components: the component that ensures the mapping of the virtual nodes, and the one that handles the mapping of the virtual links.

E.3.3.1 Virtual Node Mapping

Each virtual node needs to be mapped onto one physical node. This relation is given by the mapping function $\mathcal{M}[m \in N^v(t_k)] = i$, where virtual node m is mapped onto physical

node i. Each physical node candidate needs to have, at least, the same amount of available CPU as required by the virtual node, which is represented in equation (E.3).

$$\forall i, \forall \mathcal{M}[m \in N^v(t_k)] = i : C^v_m(t_k) \leq C^p_i(t_k) \tag{E.3}$$

E.3.3.2 Virtual Link Mapping

Each virtual link can be mapped onto one or more physical links (i.e. physical path). This relation is given by the mapping function $\mathcal{M}[mn \in L^v_{mn}(t_k)]$, where the virtual link mn is mapped onto one physical path. Each physical link candidate belonging to the physical path needs to have, at least, the same amount of bandwidth available as required by the virtual link, which is presented in equation (E.4).

$$\forall ij \subseteq \mathcal{M}[mn \in L^v(t_k)] : B^v_{mn}(t_k) \leq B^p_{ij}(t_k) \tag{E.4}$$

E.3.4 VN Request Life Cycle

The embedding process begins upon a new VN request arrival, which is depicted in Figure E.2. A VN mapping method is used to embed the VN; it takes as inputs the current status of the physical network (e.g. available CPU capacity, existing bandwidth, and physical node power state: active or inactive) and the VN request itself. If the result of the mapping process is a viable solution, the mapping is considered to be feasible; if not, it is considered to be unfeasible and the VN embedding process stops.

E.3.5 Mapping Metrics

In order to assess the performance of an embedding method and at the same time to evaluate the energy impact, different metrics were defined.

E.3.5.1 VN Request Acceptance Ratio

The VN request acceptance ratio, $\mathcal{A}(t_k)$, is given by equation (E.5) and defines the overall performance of an embedding method: the sum of all VN requests accepted, k', over the sum of all VN requests, k.

$$\mathcal{A}(t_k) = \frac{k'}{k} \tag{E.5}$$

E.3.5.2 Embedding Factor

The embedding factor, $\mathcal{E}(t_k)$, is given by equation (E.6) and represents the ratio between the amount of virtual resources (i.e. bandwidth) that were requested for the VN, and the amount of physical resources that were effectively provisioned to accommodate that VN, i.e. the efficiency of embedding.

$$\mathcal{E}(t_k) = \frac{\sum_{mn \in L^v(t_k)} B^v_{mn}}{\sum_{mn \in L^v(t_k)} \sum_{ij \subseteq \mathcal{M}[mn \in L^v(t_k)]} B^v_{mn}} \tag{E.6}$$

E.3.5.3 Physical Node Power State

The physical node power state is given by equation (E.7). The value of $\boldsymbol{u_i(t)}$ is set to 1 if physical node $\boldsymbol{i}$ hosts one or more virtual links. With this equation, we can take into account the situation where a physical node is being used as a forwarding node only.

$$\boldsymbol{u_i(t)} = \begin{cases} \mathbf{1}, \text{ if physical link } \boldsymbol{ij} \in \boldsymbol{L_i^p} \text{ hosts a virtual link} \\ \mathbf{0} \text{ otherwise} \end{cases} \tag{E.7}$$

E.3.5.4 Power Consumption

The power consumption, $\boldsymbol{P_i}$, of physical node $\boldsymbol{i}$ is given by equation (E.8), and it is obtained by summing the power required to maintain the node active, which is denoted in the equation as the baseline power, $\boldsymbol{P_b}$, with the load power, $\boldsymbol{P_l}$, currently allocated. The baseline power is given by equation (E.9), and the load power is obtained with equation (E.10). Both expressions are multiplied by the parameter $\boldsymbol{\theta}$, that is used to weight the cost of each power consumption source given in equation (E.11).

In equation (E.8), we assume that the baseline power already incorporates the power required to operate the data plane. This assumption is relevant for the case where a physical node is powered up to perform only the role of forwarding, i.e. its physical links host one or more virtual links, but no virtual nodes are being hosted in it. Therefore, the power consumption of this specific physical node will be equal to $\boldsymbol{P_b}$ only. According to [LGL+11], it is plausible to assume that the power required to power up all physical links of a given physical node is much smaller than the power required to power up the physical node itself [BH09]. Therefore, we are not considering the power required to power up each physical link individually in the equation.

$$\boldsymbol{P_i(t)} = \boldsymbol{P_{bi}} + \boldsymbol{P_{li}} \tag{E.8}$$

$$\boldsymbol{P_{bi}(t)} = \boldsymbol{\theta_b} \times \boldsymbol{u_i(t)} \tag{E.9}$$

$$\boldsymbol{P_{li}(t)} = \boldsymbol{\theta_l} \times [\boldsymbol{C_i^p(t_0)} - \boldsymbol{C_i^p(t)}] \tag{E.10}$$

$$\boldsymbol{\theta_b} + \boldsymbol{\theta_l} = \mathbf{1} \tag{E.11}$$

E.3.5.5 Physical Network Energy Consumption

The energy consumption of the physical network, $\boldsymbol{E^p(t)}$, is given by equation (E.12) and it represents the amount of energy that will be required to operate the physical network. This is obtained by summing the amount of energy required to power up (i.e. $\boldsymbol{P_b}$) the physical nodes and the amount of energy required to host (i.e. $\boldsymbol{P_l}$) the virtual nodes.

$$\boldsymbol{E^p(t)} = \boldsymbol{\theta_b} \sum_{\boldsymbol{i \in N^p}} \boldsymbol{u_i(t)} + \boldsymbol{\theta_l} \sum_{\boldsymbol{i \in N^p}} \boldsymbol{C_i^p(t_0)} - \boldsymbol{C_i^p(t)} \tag{E.12}$$

E.3.5.6 VN Energy Consumption

The energy consumption of a VN, $\boldsymbol{E^v(t_k)}$, is given by equation (E.13) and it represents the amount of energy that will be required to host a new VN request. This is obtained by summing the amount of energy required to power up (i.e. $\boldsymbol{P_b}$) new physical nodes at time $\boldsymbol{t_k}$, and the amount of energy required to host new virtual nodes (i.e. $\boldsymbol{P_l}$), where $\boldsymbol{\epsilon}$ is an arbitrarily small value.

$$\begin{aligned} \boldsymbol{E^v(t_k)} &= \boldsymbol{\theta_b} \sum_{\boldsymbol{i \in N^p}} \boldsymbol{(u_i(t_k) - u_i(t_k - \epsilon))} \\ &+ \boldsymbol{\theta_l} \sum_{\boldsymbol{m \in N^v(t_k)}} \boldsymbol{C_m^v(t_k)} \end{aligned} \tag{E.13}$$

E.4 Virtual Network Embedding - Mathematical Formulation

This section describes the mathematical formulation used to solve the online VN embedding problem with the defined constraints. For convenience of the reader, we repeat equations describing the constraints which are explained in our previous paper [MSK+13].

An Integer Linear Programming (ILP) approach is used to solve the online VN embedding problem; we propose a node-link formulation, and two assignment variables are applied during the embedding process. The index notation used here is the same as in section E.3.1.3.

E.4.1 Assignment Variables

E.4.1.1 Virtual Node Assignment

$$\boldsymbol{x_i^m} = \begin{cases} \boldsymbol{1,} \text{ virtual node } \boldsymbol{m} \text{ is allocated at physical node } \boldsymbol{i} \\ \boldsymbol{0, else} \end{cases} \tag{E.14}$$

E.4.1.2 Virtual Link Assignment

$$\boldsymbol{y_{ij}^{mn}} = \begin{cases} \boldsymbol{1,} \text{ virtual link } \boldsymbol{mn} \text{ uses physical link } \boldsymbol{ij} \\ \boldsymbol{0, else} \end{cases} \tag{E.15}$$

E.4.2 Constraints

E.4.2.1 Assignment of virtual nodes to physical nodes

Equation (E.16) ensures that each virtual node is assigned, and that it is assigned to just one physical node.

$$\boldsymbol{\forall m : \sum_i x_i^m = 1} \tag{E.16}$$

E.4.2.2 One virtual node per physical node

Equation (E.17) guarantees that each physical node can accommodate in the maximum one virtual node per VN request, although each physical node can accommodate other virtual nodes from different VNs.

$$\boldsymbol{\forall i : \sum_m x_i^m \leq 1} \tag{E.17}$$

E.4.2.3 CPU conservation

Equation (E.18) assures that the available CPU capacity of each physical node is not exceeded.

$$\forall \boldsymbol{i} : \sum_{\boldsymbol{m}} \boldsymbol{x_i^m} \cdot \boldsymbol{C_m^v}(\boldsymbol{t_k}) \leq \boldsymbol{C_i^p}(\boldsymbol{t_k}) \tag{E.18}$$

E.4.2.4 Assignment of virtual links to physical links - multi-commodity flow conservation with node-link formulation

To simultaneously optimize the mapping of virtual links and virtual nodes, the multi-commodity flow constraint [EIS75] is applied with a node-link formulation [PM04]; moreover, the notion of direct flows on the virtual links is used, which is represented in Eq. (E.19), where $\boldsymbol{L_m^v}$ represents all the virtual links that are directly connected to the virtual node $\boldsymbol{m}$, and $\boldsymbol{L_i^p}$ represents all the physical links that are directly connected to the physical node $\boldsymbol{i}$.

$$\begin{aligned} &\forall \boldsymbol{mn} \in \boldsymbol{L_m^v}, \boldsymbol{m} < \boldsymbol{n}, \forall \boldsymbol{i} : \\ &\sum_{\boldsymbol{ij} \in \boldsymbol{L_i^p}} (\boldsymbol{y_{ij}^{mn}} - \boldsymbol{y_{ji}^{mn}}) = \boldsymbol{x_i^m} - \boldsymbol{x_i^n} \end{aligned} \tag{E.19}$$

E.4.2.5 Bandwidth conservation

To ensure that the available bandwidth at each physical link $\boldsymbol{ij}$ is not surpassed, Equation (E.20) is defined.

$$\begin{aligned} &\forall \boldsymbol{ij} \in \boldsymbol{L_i^p j} : \\ &\sum_{\boldsymbol{mn} \in \boldsymbol{L_m^v}, \boldsymbol{m} < \boldsymbol{n}} \boldsymbol{B_{mn}^v}(\boldsymbol{t_k})(\boldsymbol{y_{ij}^{mn}}) \leq \boldsymbol{B_{ij}^p}(\boldsymbol{t_k}) \end{aligned} \tag{E.20}$$

E.5 Objective Functions - Energy Aware

The VN embedding problem requires the mapping of virtual nodes and virtual links, i.e. it requires decision variables to handle the mapping. With respect to the nodes, we can only optimize the distribution of the load among the physical nodes (i.e. we have a matching of one virtual node per physical node; therefore the total CPU consumption will be independent of the mapping). With respect to the links, we are not only able to optimize the link load, but also the physical bandwidth consumption, since a virtual link can be mapped in different ways: it can either be directly embedded in a single physical link or eventually span across several physical links. Another important objective is related with the energy consumption of the physical network and how it can be minimized by concentrating the load on the minimum amount of physical nodes.

Having enumerated all the dimensions of our problem, we can also realize that they are not independent, since embedding a virtual node directly affects the virtual links and therefore the bandwidth consumption; and aiming for load balancing directly conflicts with an energy saving strategy. Therefore, the objectives are intrinsically related. In this section, we describe three objective functions to address the nature of the VN embedding problem from an optimization perspective, that aims at performing one or more objectives: i) load balancing; ii) resource consumption minimization; iii) and energy consumption minimization.

E.5.1 Weighted Shortest Distance Path

The objective function WSDP, proposed in equation (E.21), aims to minimize the number of physical links, while it prefers physical links with more available bandwidth, and at the same time chooses physical nodes with more available CPU power, thereby supporting the load balancing aspect. This has the effect that high demands are allocated to nodes or links with a large amount of free capacity. The parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are used to weight the cost of each type of resource. This is the approach proposed in [MSK+13] and that we consider as baseline for our energy-aware extensions.

$$\textbf{minimize: } \boldsymbol{\alpha} \left(\sum_{m} \boldsymbol{C^v_m(t_k)} \left[\sum_{i} \frac{\boldsymbol{x^m_i}}{\boldsymbol{C^p_i(t)}} \right] \right) + \boldsymbol{\beta} \left(\sum_{mn} \boldsymbol{B^v_{mn}(t_k)} \left[\sum_{ij} \frac{\boldsymbol{y^{mn}_{ij}}}{\boldsymbol{B^p_{ij}(t)}} \right] \right) \tag{E.21}$$

E.5.2 Bandwidth Consumption Minimization

In situations where the bandwidth resource is scarce or more expensive when compared to the CPU, it is preferable to obtain the minimum bandwidth allocation. The objective function BCM which is proposed in equation (E.22) fulfills this objective: bandwidth allocation minimization per VN embedding request.

$$\textbf{minimize: } \sum_{mn \in L^v, n<m} \boldsymbol{B^v_{mn}(t_k)} \sum_{ij \in L^p} \boldsymbol{y^{mn}_{ij}} \tag{E.22}$$

E.5.3 Energy Consumption Minimization

One can realize from the previous objective functions that they are both agnostic to energy consumption aspects, i.e. the power-up of new nodes.

The objective function ECM, which is proposed in equation (E.23), is energy consumption oriented and fulfils three objectives: i) to minimize the power-up of new physical nodes including the forwarding nodes per VN embedding - this is achieved using the parameter $\boldsymbol{P_{ij}(t)}$ in the first term that represents the power state of the physical nodes $\boldsymbol{i}$, $\boldsymbol{j}$ immediately prior to the embedding, and it is used to penalize the allocation of virtual links on physical links attached to inactive physical nodes; ii) to minimize the number of physical links required per VN embedding - this is achieved by summing the decision variable $\boldsymbol{y^{mn}_{ij}}$, which is used to represent the mapping of virtual links; iii) to minimize the load power - this is obtained by using the second term that considers the current CPU allocation $\boldsymbol{[C^p_i(t_0) - C^p_i(t_k)]}$ on physical node $\boldsymbol{i}$, plus the CPU demand $\boldsymbol{[C^v_m(t_k)]}$ of virtual node $\boldsymbol{m}$ over the total CPU capacity $\boldsymbol{[C^p_i(t_k)]}$. This gives the CPU ratio which is then multiplied by $\boldsymbol{P_l}$. To not jeopardize the second objective when physical nodes $\boldsymbol{i}$ and $\boldsymbol{j}$ are active, we consider $\boldsymbol{P_{ij}(t)}$ to be equal to $\boldsymbol{P_l}$. The parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are used to weight the cost of each term of the objective

function.

$$\textbf{minimize: } \boldsymbol{\alpha} \left(\sum_{mn \in L^v, n<m} \sum_{ij \in L^p} \boldsymbol{y}_{ij}^{mn} \times \boldsymbol{P}_{ij}(\boldsymbol{t}) \right) \tag{E.23}$$

$$+ \boldsymbol{\beta} \left(\sum_{m \in N^v} \boldsymbol{P_l} \sum_{i \in N^p} \boldsymbol{x}_i^m \frac{\boldsymbol{C}_i^p(\boldsymbol{t_0}) - \boldsymbol{C}_i^p(\boldsymbol{t_k}) + \boldsymbol{C}_m^v(\boldsymbol{t_k})}{\boldsymbol{C}_i^p(\boldsymbol{t_0})} \right), \text{where}$$

$$\boldsymbol{P}_{ij}(\boldsymbol{t}) = \begin{cases} \boldsymbol{P_l}, \text{ if physical node } \boldsymbol{i} \text{ and } \boldsymbol{j} \text{ are active} \\ \boldsymbol{P_b}, \text{ if physical node } \boldsymbol{i} \text{ and } \boldsymbol{j} \text{ have different states} \\ \boldsymbol{2P_b}, \text{ if physical nodes } \boldsymbol{i} \text{ and } \boldsymbol{j} \text{ are not active} \end{cases}$$

E.6 Evaluation Results

In this section, we describe the simulation scenario and depict our major results. The evaluated cost functions are briefly summarized in Table E.2.

Due to space limitations and also to obtain a better perception on the results, we do not present here a comparison with other embedding approaches, i.e. mostly heuristics, since this was already performed for the WSDP approach in [MSK+13].

E.6.1 Simulation Parameters

To evaluate the EA-VNE-NLF model, we have implemented a discrete event simulator in Matlab® with the proposed formulation using different objective functions; the values of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ were set to 0.5.

The physical network topology is created using the GT-ITM tool [ZCB96], the number of physical nodes is set to 50, which is representative of a medium scale infrastructure provider, and the link probability between two physical nodes is set to 0.5. The node CPU capacity and the link bandwidth are real numbers uniformly distributed between 50 and 100. The VN requests are also representative of either small or medium scale virtual networks, and are created using the same topology generation method; the number of virtual nodes is not fixed, but follows an uniform distribution, from 2 to 10 virtual nodes per VN topology; the virtual link probability is set to 0.5. The CPU capacity of the virtual nodes and the bandwidth of the virtual links are also real numbers uniformly distributed between 0 and 20, and between 0 and 50, respectively[2]. The considered values for the bandwidth and for the CPU are in the same range, since the objective functions aim at simultaneously optimizing the allocation of both types of resources. The value of $\boldsymbol{P_b}$ is set to 175 units of power, while the value of $\boldsymbol{P_l}$ is set to 75 units of power[3]. According to the previous values, $\boldsymbol{\theta_b}$ is set to 0.7, while $\boldsymbol{\theta_l}$ is set to 0.3.

[2]These values were also considered by the authors of [YYRC08, ZA06, CRB09, MSK+13].

[3]These values were also considered by the authors of [BH09].

Table E.2: Compared VN Embedding Methods.

Notation	Method Description
WSDP [MSK+13]	Weighted Shortest Distance Path Minimization.
BCM	Bandwidth Consumption Minimization
ECM	Energy Consumption Minimization

We assume that VN requests arrive according to a Poisson process, and that each VN has an associated lifetime measured in time units with an average of $\mathbf{1/\mu = 1000}$, following an exponential distribution. The same assumption is also taken by the authors of [MSK+13]. The VN request rate, i.e., value of $\boldsymbol{\lambda}$, is started with a rate of 3 VN requests per 100 time units, and increases by 1 VN request, up to a rate of 10 VN requests. This can give an insight into two opposite case scenarios, with a very high and very low acceptance ratio. For each value of $\mathbf{1/\lambda}$, 10 trials are performed. A new set of VN requests and a new physical network topology are generated for each trial. All simulations are set to run up to 50000 time units to obtain the steady-state and to preclude the transient phase effect [Jai91]. A confidence interval of 95% is used for all results presented below.

The CPLEX®[cpl12] version 12.2 was used to solve the linear programming problem of the EA-VNE-NLF. A time limit of 600 seconds is defined for each VN mapping, although most of the VNs are embedded in hundreds of milliseconds. The CPLEX® is set to use only one CPU core for comparison purposes with the remaining methods. All the simulations are performed using an AMD® Opteron™ Processor 4238@3.3GHz, and the time consumed per VN request embedding is registered. The evaluation metrics are the ones defined in section E.3.5.

E.6.2 Simulation Results

This sub-section presents the evaluation results as a function of the VN request rate for all the previously described metrics.

E.6.2.1 VN Request Acceptance Ratio

One of the main aspects of the performance of each embedding method is the VN request acceptance ratio, which is shown in Figure E.3. From the figure, we can observe that the three objective functions demonstrate the same behaviour, i.e. the acceptance ratio decreases with the VN request rate. This is justified by the fact that there are more virtual resources to allocate in the same amount of physical resources. We can also observe that the objective function that aims at reducing the bandwidth consumption per VN embedding, BCM, produces slightly worse results when compared to the remaining curves. Noteworthy, the objective function that aims at reducing the energy consumption per VN embedding, ECM, has similar results when compared to WSDP.

E.6.2.2 Embedding Factor

Figure E.4 shows the embedding factor as a function of the VN request rate. The embedding factor decreases with the number of VN requests for all curves. This behaviour is expected and understandable, if we have in mind one key point: when there are few virtual network requests, the embedding problem is mainly one of optimization; when the situation changes, i.e. there are many VN requests, the embedding problem is mainly one of feasibility. All the objective functions have a good effectiveness per VN embedding, i.e. higher than 88%. The two objective functions proposed, BCM and ECM, improve the embedding factor by 1.4% (4.3%) and 0.7% (4.1%) for a VN request rate of 3 (10), when compared to WSDP.

E.6.2.3 Physical Nodes Active

Figure E.5 depicts the average percentage of physical nodes in the active state per VN request rate. The number of active nodes increases with the rate of VN requests, with a long-term trend to reach 100%. The cost function WSDP has almost all nodes in the active

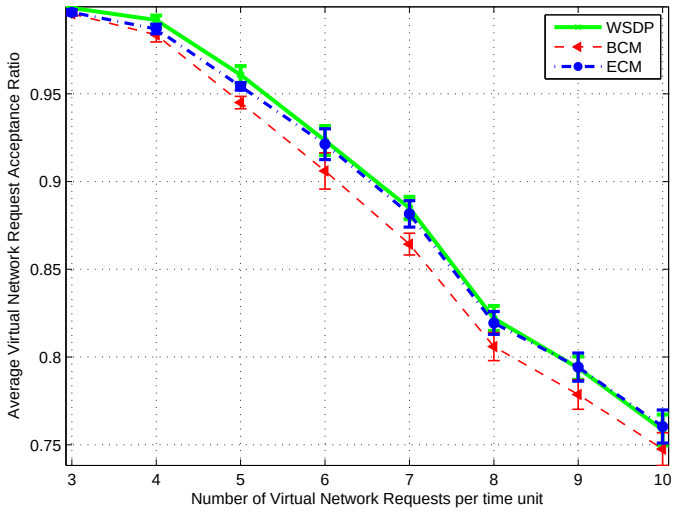


Figure E.3: Average VN Acceptance ratio per VN request.

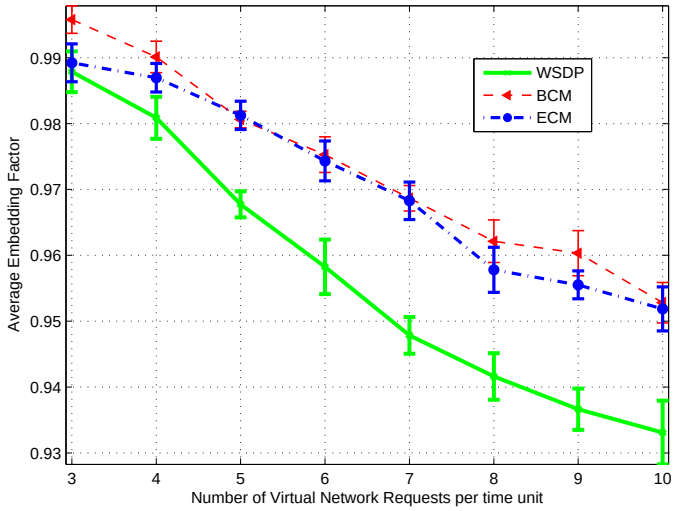


Figure E.4: Average Embedding Factor per VN request.

state after a VN request rate of 6. The cost functions BCM and ECM have reduced the number of active nodes by 15.0% (and 3.5%) and 36.2% (and 5.8%) for a VN request rate of 3 (and 10), when compared to WSDP, respectively. These values are obtained using the overall number of physical active nodes and not the percentage, as provided in the figure.

E.6.2.4 Physical Links Active

Figure E.6 depicts the percentage of physical active links per VN request rate. From the figure, we can observe that the percentage of active links increases with the rate of VN requests. All the objective functions proposed have reduced the percentage of active links. We also observe that the objective function ECM, aiming at minimizing energy consumption, does not always achieve the lowest percentage of active links. This occurs for a rate higher or equal to 7 VN requests, where the objective function BCM achieves the same percentage of active physical links. Although this function does not consider any energy parameters in its formulation, it implicitly reduces the number of active links by directly minimizing the

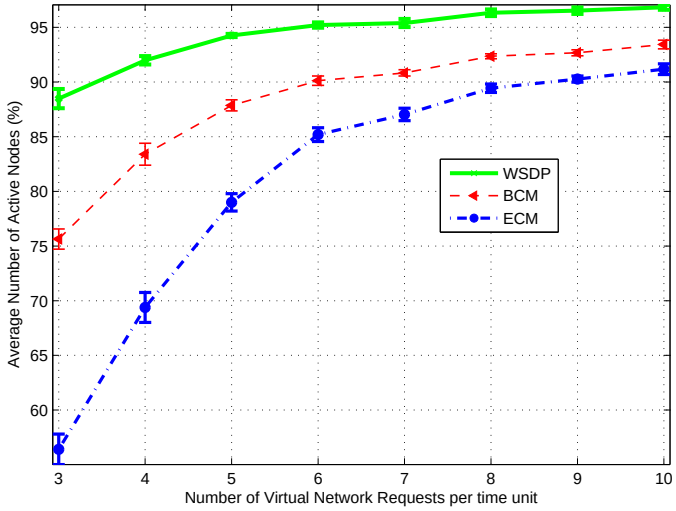


Figure E.5: Average Percentage of Physical Nodes Active per VN request.

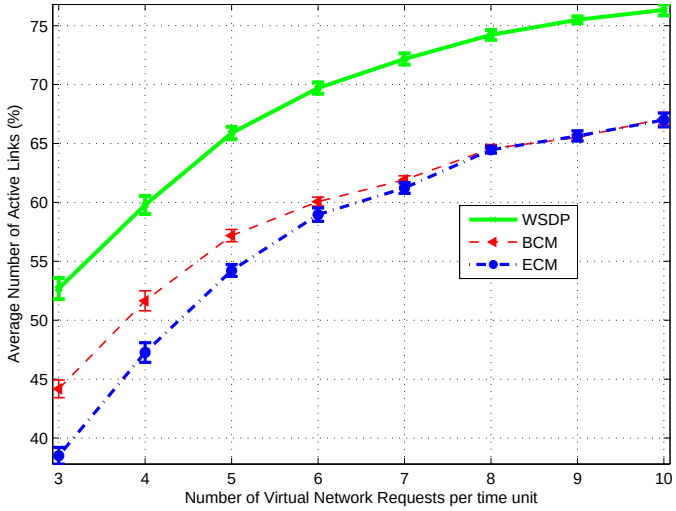


Figure E.6: Average Percentage of Physical Links Active per VN request.

bandwidth consumption. The cost function ECM has reduced the number of active links by 26.9% (12.2%) for a VN request rate of 3 (10), when compared to WSDP, respectively.

E.6.2.5 Physical Network Energy Consumption

Figure E.7 depicts the average physical network energy consumption as a function of the VN request rate. From the figure, we can observe that energy consumption increases with the VN request rate for all objective functions evaluated, which corroborates the results obtained on Figure E.5: if more VNs are being allocated per time unit, more physical resources need to be activated. We must also notice that the function BCM reduces the energy consumption; this makes sense, since this function aims at minimizing the physical bandwidth allocation, hence minimizing the overall number of forwarding nodes allocated. The cost function BCM and ECM have reduced the physical network energy consumption by 14.4% (3.1%) and 31.4% (3.6%) for a VN request rate of 3 (10), when compared to WSDP, respectively.

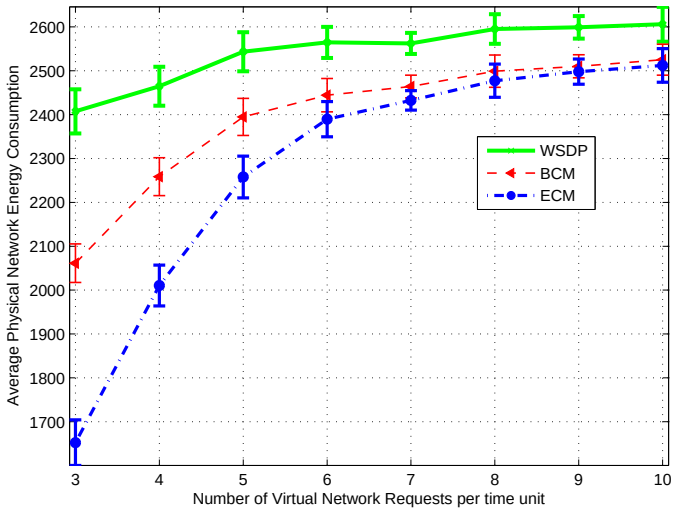


Figure E.7: Average Physical Network Energy Consumption per VN request.

E.6.2.6 VN Energy Consumption

Figure E.8 depicts the average VN energy consumption as a function of the VN request rate. Here we observe the same behaviour for all methods - the energy consumption decreases with the rate of the VN requests, and tends to a specific bound. If we consider the result obtained on Figure E.5, we can assume that this bound is mostly given by the second term of equation (E.13), whereas the transition between power states (i.e. power-off and power-on) is less frequent for a higher VN request rate.

The cost function BCM, which aims at minimizing the bandwidth consumption, slightly increases the energy consumption per VN allocation, when compared to WSDP. If we have in mind Figure E.5, we can state that the first term of equation (E.13) has a minimal impact on the objective function WSDP per VN request, since the majority of the physical nodes are already powered-up. Moreover, the cost function BCM, despite minimizing the overall number of forwarding nodes allocated, it does not minimize the number of power-up nodes per VN request, therefore requiring more energy per VN request.

The cost function ECM, which aims at minimizing the energy consumption, results in the lowest energy consumption per VN allocation, and reduces the energy consumption per VN allocation by 50.9% (6.4%) for a VN request rate of 3 (and 10), when compared to WSDP.

E.6.2.7 VN Embedding Time

An important aspect of all the VN embedding methods is the time that they require to embed, on average, a VN request, and how it varies with respect to the different loads on the physical infrastructure, i.e. VN request rate.

Figure E.9 shows the solving time for each objective function as a function of the VN request rate. From the figure, we can observe that the embedding time is in the order of seconds for a lower VN request rate, while it is in the order of milliseconds for a higher VN request rate. The obtained behaviour is to be expected: having a smaller number of VN requests will result in an optimization problem, while increasing the VN request rate will result in a feasibility problem.

The objective function ECM showed the best performance with respect to embedding time. This objective function, compared to the WSDP one, does not consider the bandwidth demanded by the VN, i.e. $\boldsymbol{B_m^v n(t_k)}$, neither the available bandwidth of the physical network,

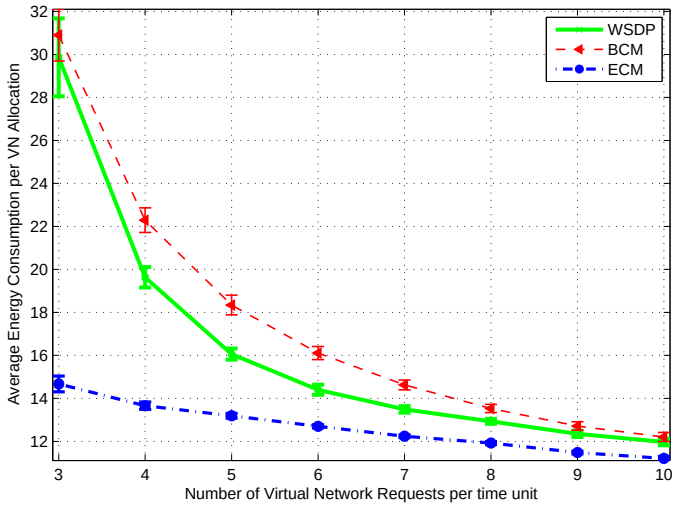


Figure E.8: Average Virtual Network Energy Consumption per VN request.

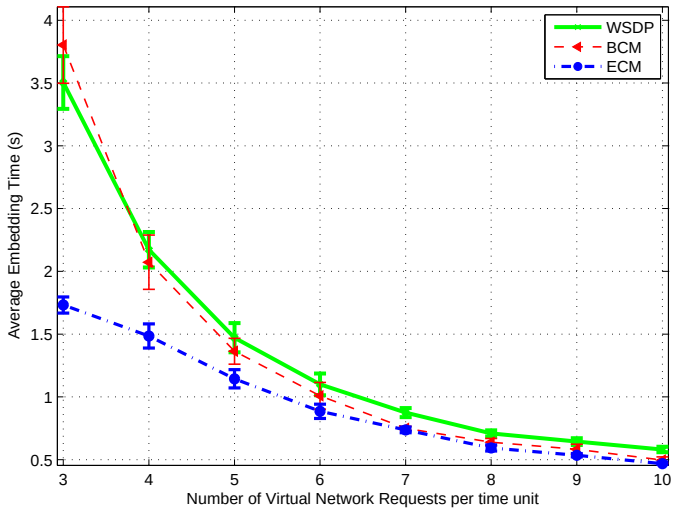


Figure E.9: Average VN Embedding Time per VN request.

i.e. $B_i^p j(t_k)$. Therefore, we can state that, removing those two parameters from the cost function, improves the embedding time.

E.7 Conclusion

This paper proposed the EA-VNE-NLF ILP formulation to solve the VN embedding problem. The model applies optimization theory to simultaneously embed the virtual nodes and the virtual links. Two new objective functions are proposed: BCM, which aims to minimize the bandwidth consumption, and ECM, which aims to minimize the energy consumption.

From the two objective functions proposed and evaluated one can conclude the following: BCM increases the embedding factor per VN request, and it simultaneously reduces the physical network energy consumption, while it obtains a slightly smaller VN acceptance ratio when compared to WSDP; ECM reduces significantly the physical and virtual network energy consumption, and keeps a similar VN request acceptance ratio, and it additionally takes less time to embed each VN request. The results also showed that, not only the minimization of

the bandwidth allocation is important for the VN embedding, but also the load balancing has a significant impact. The minimization of the bandwidth allocation positively affects the energy consumption, and the consideration of both the bandwidth minimization allocation and the CPU load on the objective function provides a good VN acceptance ratio.

Future work will endorse the VN embedding problem applied to virtual node seamless migration, and re-configuration of virtual networks taking into account energy parameters.

Appendix F - Optimal Virtual Network Migration: A Step Closer For Seamless Resource Mobility

Márcio Melo, Susana Sargento, and Jorge Carapinha

submitted in *IEEE/ACM Transactions on Networking*, vol. NA , Issue: NA 2014.

The format has been revised.

Optimal Virtual Network Migration: A Step Closer For Seamless Resource Mobility

Márcio Melo, Susana Sargento, and Jorge Carapinha

Abstract

One of the key problems with Network Virtualisation is the ability to move components of the virtual network, or even the entire virtual network, between physical hosts, in real-time and seamlessly to the end-users.

This paper addresses virtual resource mobility from a new perspective: i) it proposes Virtual Network (VN) Clone migration, which requires no assumptions regarding the protocols running inside the virtual networks or its own architecture, leaving space for different types of protocols and architectures to be implemented, tested and used in commercial scenarios; ii) and it proposes VNRE-NLF, an integer linear programming formulation to solve the online virtual network re-embedding problem as a simultaneous optimization of virtual nodes and virtual links, providing the optimal bound for the migration of virtual networks. This approach aims at minimizing the overall VN migration cost per re-embedding: i) number of virtual nodes migrated; ii) physical bandwidth consumption.

The results are very promising: the VN Clone migration achieves no VN downtimes and it takes just a few seconds to be fully performed. This makes the VN Clone approach suitable both for non- and real-time traffic. The obtained results of VNRE-NLF show that the VN is highly resilient to migration events if no more than 2% of the physical resources need to be shut-down (i.e. resilience factor higher than 0.8). Moreover, it shows that it is not only important to have enough spare capacity to re-accommodate the virtual nodes and virtual links affected by the physical resource shut-down, but also to have additional capacity to accommodate virtual link re-assignments and virtual node migrations.

F.1 Introduction

ONE of the major features of Network Virtualisation is the possibility to move virtual resources, i.e. VR, on-demand and seamlessly from one physical host to another without losing network connectivity. The virtual resource migration was initially proposed by Wang ***et al.*** [WvdMR07, WKB+08] as a primitive for network management tasks; later on, Lo ***et al.*** [LAZ12] proposed three algorithms to schedule which resources should be migrated firstly to minimize the overall migration cost.

The VN migration itself can be triggered by several reasons: i) equipment/facilities maintenance; ii) network performance; iii) end-user requirement/service level agreement; iv) energy saving; v) security protection; vi) fault management; vii) end-user mobility. The VN migration process is composed by two distinct phases: i) VN re-embedding - to obtain a new embedding solution of an existing VN, but taking into account the current constraints of the physical network (available bandwidth and available processing capacity); ii) virtual resource migration - the operation of effectively moving the virtual resources, e.g. virtual links and virtual nodes.

This paper endorses the VN migration process. To address the first phase, the VN re-embedding, an integer linear programming formulation is proposed, i.e. Virtual Network Re-Embedding Node-Link Formulation (VNRE-NLF), to obtain the optimal bound per VN re-embedding taking into account the VN migration cost: i) number of virtual nodes migrated; ii) overall bandwidth consumption. To endorse the second phase, the virtual resource migration, it is proposed VR Cloning as an alternative to VR live migration [WvdMR07]. VR Cloning is based on saving the current state of the VR and transferring the VR clone to the new physical host. Different triggers for VN migration are described and analysed, which can be used as an input to define new heuristics or even mathematical formulations to perform the VN re-embedding, taking into account the current constraints of the physical network (available bandwidth and available processing capacity).

The performance of the VR Cloning is evaluated by means of real experiments. This proposed approach achieves no VN downtime and it takes 2.75 seconds to be fully performed. The performance of the Virtual Network Re-Embedding Node-Link Formulation (VNRE-NLF) is evaluated by means of simulation. The simulation results show that the VN resilience to migration events is dependent on the VN request rate and also on the percentage of physical resources shut-down. The obtained results point show that the VN is highly resilient to migration events if no more than 2% of the physical resources need to be shut-down (i.e. resilience factor higher than 0.8).

The rest of the paper is organized as follows. After summarizing the related work in section F.2, section F.3 presents the VN migration triggers and describes the virtual router migration process. Section F.4 presents the Network Virtualisation architecture proposed to support the VR cloning migration. Section F.5 describes the virtual network re-embedding problem and the evaluation metrics. Section F.6 describes the Virtual Network Re-Embedding Node-Link Formulation (VNRE-NLF) and presents one objective function that aims at minimizing the VN migration cost. Section F.7 analyzes the performance of the VR clone method and evaluates the VNRE-NLF, and section F.8 concludes the paper and describes the future work.

F.2 Related Work

F.2.1 VN Migration

The VM migration was initially proposed for data centres as a way to move the CPU load from one physical server to another. This feature is not only important for load balancing purposes, but it can also be applicable for planned maintenance operations, i.e. moving away all the VMs from one physical server that needs to be rebooted or shutdown to different physical servers.

In order to reduce the downtime due to the VM migration process, Clark ***et al.*** [CFH+05] proposed the live migration of VMs, within the same LAN, which allows a VM to be migrated while still running. This not only reduces significantly the downtime provoked by the VM migration, but it also makes the migration process seamless to the end-users or to the running applications. Ma ***et al.*** [MLL10] proposed some improvements on the live VM migration process, which both reduces the overall migration time and the total data transmitted in the order of 30%.

The VM migration approach is also applicable and important to the networking area, where it can be used for networking maintenance operations or for networking service deployment. For instance, it can be used to move critical (or non-critical) VRs from physical hosts that need some kind of maintenance operation without disrupting the routing protocols. This is much preferable and human error safe than manually configuring routing protocol metrics to move away the networking traffic from that physical router.

With that in mind, Wang ***et al.*** [WvdMR07] proposed VROOM as a primitive for networking management tasks which makes it possible to move virtual routers freely without changing the IP-layer topology. Wang ***et al.*** [WKB+08] extended their prior work and proposed to decouple the data plane from the control plane of the VRs, which results in no performance impact on the data traffic when a hardware data plane is used, and very low impact when a software date plane is used.

The VR migration feature was also considered and evaluated on Internet Protocol Television (IPTV) scenarios [MNG+09]. Pisa ***et al.***[PFC+10] proposed a new migration model for XEN, using also data and control plane separation, which outperforms the XEN standard migration model. Lo ***et al.*** [LAZ12] used the virtual router migration [WKB+08] as a primitive to perform the virtual network migration, i.e. the migration of an entire VN, and also proposed three algorithms to address the VN migration scheduling problem and to minimize the total migration cost.

Although the separation of the data plane from the control plane seems to be a very effective approach on the VR migration, once it reduces the downtime of the VN, it is also a limiting factor on the conception of new network architectures and on the deployment of new network protocols. We argue that not only the VN migration process should be independent of the networking protocols that are running on the virtual network, but also no assumption should be taken on the router architecture itself. Therefore, we consider each VR as a black-box and propose the VR Cloning as an alternative to the current VR live migration process [WvdMR07].

F.2.2 Virtual Network Re-Embedding Problem

The VNRE problem can be formulated as an un-splittable flow problem [ZA06] of resource re-allocation. In order to solve this problem, several centralized algorithms have been suggested [LT06, ZA06, YYRC08, FBCB10, NMCS11b, CRB12], to endorse the online VNE problem, i.e. static provisioning of VNs; others consider, instead, a distributed algorithm

[HLZ08] to better handle scalability issues. However, these approaches neither take into account the VNRE problem, i.e. dynamic provisioning of VNs, nor consider the VN migration cost, i.e. virtual node migration and bandwidth re-allocation.

Moreover, centralized algorithms have been suggested [YQA+10, RAB10] to address the SVNE problem. The aim is to reserve additional (or redundant) computing and bandwidth capacity for each VN when it is first provisioned, which can be used in situations of node and link failures. Nonetheless, these algorithms do not endorse the scenario where the backup physical resources are not available, nor consider the VN migration cost.

Recently, Xiaolin ***et al.*** [LWD+14] proposed an algorithm, based on the work of Chowdhury ***et al.*** [CRB12], to endorse the VNRE problem. Despite, endorsing the VN re-embedding the authors do not consider the bandwidth re-allocation nor provide the optimal bound per set of VNs re-embedded.

In our previous research work [MCS+12], we have proposed an ILP formulation to solve the online VN embedding problem from an optimization standpoint. Moreover in [MCS+13b] we have extended the Integer Linear Programming (ILP) formulation to contemplate the re-optimization of VNs currently embedded. An extended approach and the comparison with existing heuristics was recently added in [MSK+13].

Although all these algorithms provide a solution for the VN mapping problem, an optimal solution for VNRE taking into account the migration costs is not provided. Also, some of the proposals fail to solve the assignment problem as a simultaneous optimization of the virtual node and link placement, which leads to non-optimal resource re-allocation solutions. Our approach, the VNRE-NLF, applies a node-link formulation to solve the VNRE problem in a single step using the multi-commodity flow constraint. This approach provides the optimal solution for the objective problem considered, both in terms of node migration and bandwidth re-allocation, for a given set of weights in the objective function.

F.3 Seamless Approach for VN Migration

The VN migration process can be started from a predicted event in the sense that it can be scheduled in time, i.e. planned maintenance, or it can be triggered from a non-predicted event, i.e. VN security attack or hardware failure. In this section, we start with a description of the several factors that can lead to a VN migration process, and we finalize the section by explaining the different actions required in our approach for VN Cloning.

F.3.1 Triggers for VN Migration

The time to perform all the VN migration operations can be considered critical or non-critical depending on the event that leads to the migration process.

F.3.1.1 Equipment/Facilities Maintenance

The migration process can be triggered due to maintenance reasons within the physical hosts or external to the physical hosts: due to the replacement or upgrade of hardware modules, firmware updates or patches where the physical host needs to reboot or shutdown and disassembled; or due to the need to temporary or permanent change the location of the physical host, or UPS upgrade or power grid maintenance.

F.3.1.2 Network Performance

The VN migration can also be started due to networking performance reasons. If we consider the fact that VNs are dynamically provisioned and constantly being created and removed, the consumption of network resources is likely to become unbalanced due to the dynamics of the VNs arrivals and departure events. In order to optimize the physical resources and at the same time to distribute the network load equally per all physical hosts, or even to alleviate the load at some physical hosts that are reaching critical levels, a re-optimization process on the existing VNs should be performed.

F.3.1.3 End-user Requirement/Service Level Agreement

It may also be required to move a VR from one Provider Edge (PE) to another PE which is much closer to the end-user. It may also be required to reduce the overall round trip delay of a VN throughout the migration of parts, i.e. VRs or VL, of the VN or even the entire VN to physical places and hosts which provide smaller round trip delays.

F.3.1.4 Energy Saving

The migration of VNs can also be triggered due to power consumption reasons. Either to move VNs closer to greener power grids, or to move VNs to physical hosts that consume less energy; or even to concentrate VNs on the minimum number of physical hosts as possible, without interfering on the service levels reliability, in order to reduce the number of active physical hosts. This is not only due to diurnal traffic patterns, where the traffic is much smaller in the night, and therefore the number of pieces of network equipment required to operate in the night is smaller, but is also due to the fact that a physical router in idle mode consumes over 90% as if it was in full mode [CSB+08].

F.3.1.5 Security Protection

The migration of a VN can be even required as a way to provide higher security protection, or a way to move VNs from physical hosts which are currently under attack [AS11]. A VN can evade detection or attach by changing its location in the physical network.

F.3.1.6 Fault Management

The VN migration process can also be triggered due to fault management decisions. If we consider a reactive fault management, the VN migration is triggered when the hardware fails. If we consider a proactive fault management, the VN migration process is started when the hardware that is prone to fail is flagged.

F.3.1.7 Service Deployment

In order to deploy new services on production networks, VNs running on trial scenarios can be phased migrated to physical hosts located within these networks or only just parts of the VNs need to be moved away.

F.3.1.8 End-User Mobility

The VN migration process can be even required due to mobility reasons. An end-user can be moving away from his home network to a foreign network and, in order to avoid the

connection break during the handover process, i.e. change the IP address and or the access technology of the end-user, the VR migration process can be used as an complementary mechanism (or even in some situations in the absence of any mobility process, as an alternative) to mobile IP implementations [Per02] and IEEE802.21 - Media Independent Handover Services [GWC+09]. In this way, the IP addresses of the end-users are preserved and the TCP sessions are maintained during the transition of one network to another.

On Table F.1 we summarize all types of triggers considered, and we also express them in terms of event periodicity, i.e. when they are expected to occur, event duration, i.e. the expected time that the event will consume, and priority, i.e. this can be used as a classification of the event in terms of urgency to be taken.

Table F.1: VN migration types of trigger events, event duration and event priority.

Event Type	Event Periodicity	Event Duration	Event Priority
Physical Maintenance	Monthly/Annually	Hours/Days	Medium/Low
Network Performance	Daily/Weekly	N/A	High/Medium
Energy Saving	Daily	Hours	Medium/Low
Security Protection	N/A	Minutes/Hours	Urgent/High
Fault Management	N/A	Minutes/Hours	Urgent/High
Service Deployment	Weekly/Monthly	Hours/Days	Medium/Low
End-user Mobility	Hourly/Daily	Minutes/Hours	Urgent/High

As a result of the previously described triggers we can foresee three different use cases where VN migration is required:

i In the first situation the host is considered to be removed from the physical network topology. Therefore all the VNs using that specific host need to be reallocated to other physical hosts.

ii In the second situation the host still exists, but one or more physical links have been removed from the physical network topology, e.g. hardware failure. Likely some VNs need to be reallocated to other physical hosts; although it does not strictly mean that it needs also to migrate VRs (it may be possible that only VL migration can solve the problem).

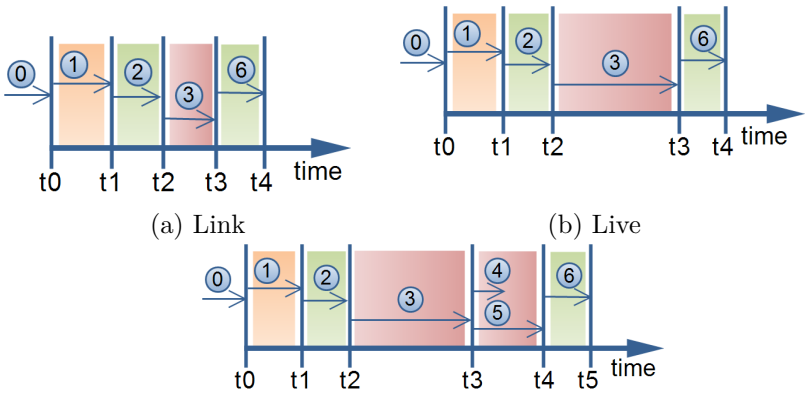
iii In the third scenario the host exists on the physical network, but it is only capable of doing switching operations [nex12]. This implies only the migration of the VRs; the migration of the VLs is not mandatory.

F.3.2 VN Clone Migration Procedure

As a result of the three use cases, one of two things can take place in the migration process: VL migration, or VL and VR migration. Each migration process encompasses a set of different operations. Two different approaches are evaluated for the VR migration: the VR live migration and the VR clone migration (our proposed approach).

Figure F.1 represents the VN migration timeline, which is not at scale; it is intended to reflect the amount of time that each action takes when comparing with others, to reveal its level of criticality to the overall VN migration process, and also to identify the common actions, which are represented in the timeline by numbers, taken in the different VN migration processes (i.e. link, live and clone migration).

The VN link migration timeline is represented in the left (Figure F.1a); the center (Figure F.1b) depicts the live migration timeline; the right (Figure F.1c) depicts the clone migration timeline.



(a) Link (b) Live

(c) Clone

Figure F.1: VN Migration Timeline

In the Figure F.1, we can observe that the VN link migration takes less time to be fully performed when compared to the live or the clone migration. This is due to the migration of the VR itself, since it requires moving the VR content (e.g. memory RAM) from one physical host to another, while in the case of the virtual link migration, no data needs to be transferred across physical nodes.

The different operations which compose the clone migration process are described here:

0. ***VN Migration Trigger*** - The VN migration process starts after receiving a VN migration trigger (i.e. which can be one of the triggers presented on Table F.1), and it corresponds to step 0.

1. ***Compute VN Mapping*** - The time that this action takes to be performed can be considered as critical or non-critical to the overall migration time, and it will strictly depend on the event which causes the VN migration (see Table F.1), and that involves the computation of a new VN mapping (i.e. VNRE).

   The VN mapping takes as input the VN nodes and links that need to be migrated, and the output will be the new location of those nodes and links[1]. The VN mapping can be performed either using a heuristic approach [NMCS11b] which performs the VN embedding in a faster and efficient way, or using mixed integer linear programming approach [MCS+12], which takes relatively more time to perform the VN embedding, but achieves the optimal bound.

   The "available" time to perform the VN (re-)embedding will be dictated by the event type that leads to the VN migration process, i.e. a critical event (e.g. fault management) requires that the VN embedding is performed as fast as possible, and a non-critical event (e.g. network performance) requires that the VN (re-)embedding is as good as possible (i.e. optimal bond).

2. ***VLs Setup*** - This operation comprises the setup of new virtual links, e.g. setup of VLAN interfaces and virtual bridges, and it is considered as non-critical, since it is performed beforehand and in the time frame of the order of milliseconds (or even nanoseconds with optical switches).

[1]It may also be possible that it is less costly (i.e. shorter virtual links=>less provisioned bandwidth) to move not only the nodes and links that were initially considered to be migrated, but also other nodes and links.

3. ***Clone/Move/Restore*** - This is the most critical and time consuming task to the overall VN migration process, and it can be divided into three sub-operations:

   a) ***Clone VR*** - The cloning of the VR involves saving the current state, i.e. memory RAM, to the physical host hard-disk or even to a RAM-disk. This can be considered as critical or non-critical action, if the VR is put into suspend while being cloned in the first case, or if the VR is still running while being cloned in the latter. The time required to perform this process is given by equation (F.1), where $\boldsymbol{VR_{memory}}$ is the memory RAM size of the VR.

   b) ***Move VR Clone*** - This part encompasses the transfer of the VR clone to the new physical host and it is, in principle, the most time consuming task of the three sub-operations, and of the overall VN clone migration process. Despite contributing to the overall VN migration execution time, it does not affect the VN downtime, since the VN is still running while the VR clone is being relocated. The time required to perform this task is given by equation (F.2), where $\boldsymbol{BW_{reserved}}$ is the bandwidth which is effectively reserved by the operator to this kind of operations, and $\boldsymbol{BW_{free}}$ is the bandwidth that is not provisioned (or available) on the physical path between the physical hosts and at that time period. In theory, the reserved bandwidth can be equal to zero, although operators do tend to reserve bandwidth for these kind of operations. Note that, if this phase takes too much time to be performed, the VR clone can easily become outdated.

   c) ***Restore VR Clone*** - The restore of the VR clone is performed after it is allocated on the new host, and it does not influence the VN downtime, since the VR clone is not yet connected to the VN. This operation is performed in the time-frame of high hundreds of milliseconds.

4. ***Add Virtual Bridge Int. & Remove Original VR*** - This is a critical action and encompasses adding a new virtual interface (e.g. VLAN) to the virtual bridge that will be used to connect the VR clone to the VN. It also includes the shutdown of the original VR, which is performed in order to avoid duplicated VRs operating on the virtual network. The VN transition, from the old VLs and VRs to the new ones, is signaled by the execution of these two operations. The VR shutdown (destroy) is executed in the time-frame of low hundreds of milliseconds.

5. ***Remove VLs*** - The removal of old virtual links is a non-critical action and is performed in the same timeframe as the setup of virtual links (i.e. milliseconds). This phase does not count to the execution time, though it is part of the VN migration process.

The VN downtime due to the migration process can be obtained using equation (F.3), where the $\boldsymbol{VN_{downtime}}$ is mostly given by the VR cloning operation (i.e. $\boldsymbol{t_{clone}}$). With equation (F.4), we can obtain the VN migration execution time, where the ***action 3b*** contributes the most.

$$\boldsymbol{t_{clone} = \frac{VR_{memory\_size}}{RamDisk_{write\_speed}}} \tag{F.1}$$

$$\boldsymbol{t_{move} = \frac{VR_{memory\_size}}{BW_{reserved} + BW_{free}}} \tag{F.2}$$

$$\boldsymbol{t_{downtime} \cong t_{clone}} \tag{F.3}$$

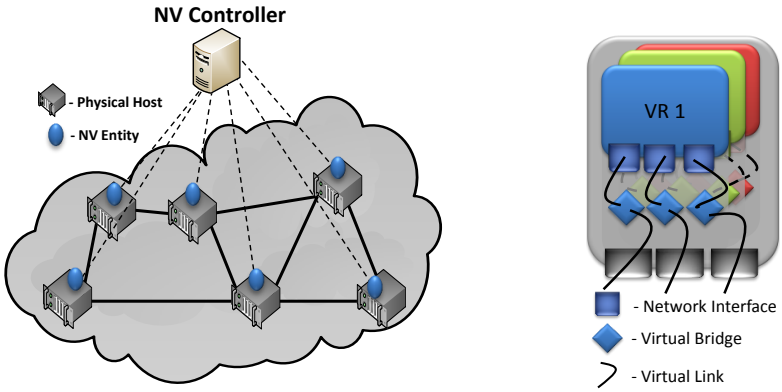
$$\boldsymbol{t_{execution} = t_4 - t_0} \tag{F.4}$$

F.4 VN Clone Migration Architecture

In this section we describe the overall architecture, which was considered to support the VN clone migration, and also the building blocks of the VR.

F.4.1 VN Clone Migration Architecture

Figure F.2a presents the NV architecture which is proposed to support VR migration, comprising the NVC and the NVE. The NVC is responsible to coordinate the VN migration process, and is also responsible to perform the VN mapping, choosing the new location of the VLs and VRs. Each NVE is responsible to enforce the NVC commands, e.g. ***CloneVR***. The list of possible commands performed by the NVC is shown in Table F.2.



(a) Network Virtualisation Architecture. (b) Virtual Router Building Blocks.

Figure F.2: VN Clone Migration Architecture.

All communications between the NVC and the NVEs are secured and performed using the SSH protocol, and the Session Control Protocol (SCP) protocol is also used to move the VR clone.

F.4.2 Virtual Router Implementation

The VR architecture considered is shown in Figure F.2b. It is composed by the VR instance, (virtual) network interfaces and virtual bridges. The virtual bridges are used either to interconnect network interfaces within the physical host, or to interconnect virtual network interfaces of VRs running inside of the host. The Linux Bridge Utils tool [bri12] is used to setup virtual bridges, and the Linux VLAN implementation [VLA14] is used to setup the virtual links.

F.5 Network Re-Embedding Problem Formulation

In this section, we introduce the virtual network re-embedding problem. In addition, the VN embedding notations used throughout the paper are presented, and the virtual network re-embedding system is explained. Finally, the mapping goals are introduced to support the mathematical formulation.

Table F.2: NV Controller - List of Commands.

Command Name	Command Description
AddVInt *IntId VIntId*	Add virtual Interface ***VIntId*** on physical interface ***IntId***
RemVInt *VIntId*	Remove virtual interface ***VIntId***
AddVBridge *BrId*	Add a Virtual Bridge on the Physical Host
RemVBridge *BrId*	Remove Virtual Bridge on the Physical Host
AddVBridgeInt *BrId IntId*	Add a (virtual) Interface to the Virtual Bridge
RemVBridgeInt *BrId IntId*	Remove a (virtual) Interface on the Virtual Bridge
CloneVR *VRId*	Clone Virtual Router ***VRId***
MoveVR *VRId HId*	Move VR ***VRId*** to physical host ***HId***
RestoreVR *VRId*	Restore Virtual Router ***VRId***
RemoveVR *VRId*	Remove Virtual Router ***VRId***

F.5.1 Network Description

We use superscript to distinguish the physical network from the virtual network, where p and v correspond to physical and virtual, respectively.

F.5.1.1 Physical network

A physical network can be described as a weighted undirected graph $\boldsymbol{G^p = \{N^p, L^p, C^p, B^p\}}$ composed by a set of physical nodes, $\boldsymbol{N^p}$, and a set of physical links, $\boldsymbol{L^p}$. Each physical node $\boldsymbol{i}$ is characterized by its processing capacity, $\boldsymbol{C_i^p}$, commonly referred to as the CPU and that can be expressed in units of CPU. A set of physical nodes $\boldsymbol{M(t)}$ requires maintenance at a given time $\boldsymbol{t}$.

With respect to the physical links, we consider that each link $\boldsymbol{ij}$ has a given bandwidth, $\boldsymbol{B_{ij}^p}$, that can be expressed in units of bandwidth; we assume that each link is an undirected link. The bottom-right of Figure F.3 illustrates a physical network topology example composed of 6 physical nodes and 8 physical links; the corresponding capacities of the nodes and the links are presented on top of the elements. In the presented figure, we have physical node $\boldsymbol{A}$ that requires maintenance at time $\boldsymbol{t}$, that is represented through a white circle. Either the processing capacity of physical node $\boldsymbol{A}$ or the bandwidth capacity of the concerned physical links is set to zero.

F.5.1.2 Virtual Network

VN can be described as a weighted undirected graph $\boldsymbol{G^v = \{N^v, L^v, C^v, B^v\}}$ composed by a set of virtual nodes, $\boldsymbol{N^v}$, and a set virtual links, $\boldsymbol{L^v}$. Each virtual node $\boldsymbol{m}$ is characterized by the amount of required CPU, $\boldsymbol{C_m^v}$, and the virtual links $\boldsymbol{mn}$ are logical connections between virtual nodes and characterized by the amount of dedicated bandwidth, $\boldsymbol{B_{mn}^v}$. We also assume that each virtual link is an undirected link.

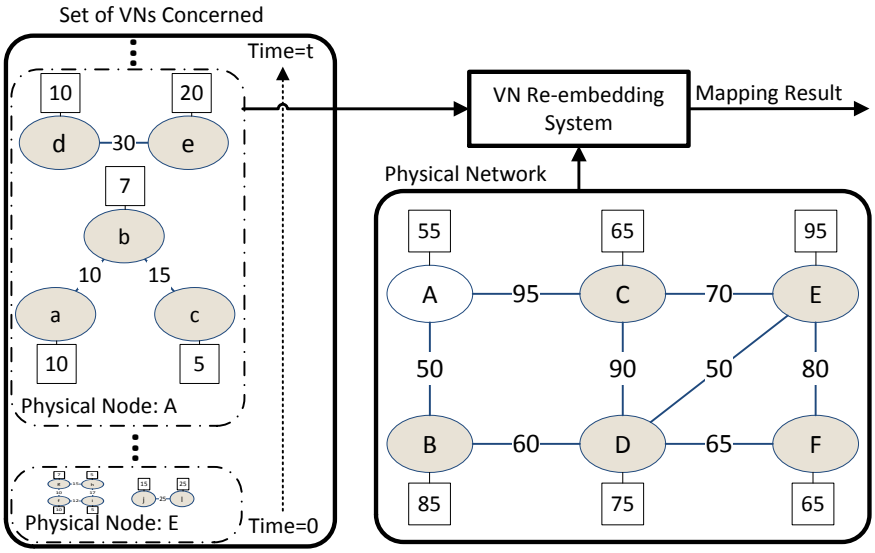


Figure F.3: VN Re-embedding System - Topology Example

The left part of Figure F.3 represents the example of two sets of virtual networks, VN request **1** on the bottom-left and VN request $\boldsymbol{k}$ on the top-left.

F.5.1.3 VN Assignment Notations

First, we start with the convention used for the index notation: $\boldsymbol{N^p}$ represents the set of nodes that belong to the physical network; $\boldsymbol{L^p}$ represents the set of links that belong to the physical network; and $\boldsymbol{L_i^p}$ represents a subset of links $\boldsymbol{ij}$ that are directly connected to the node $\boldsymbol{i}$. The same type of notation is used to represent the VN using the letters $\boldsymbol{m}$ and $\boldsymbol{n}$ in the virtual network. The notations used throughout this paper for the VN assignment problem are presented in Table F.3. The table is divided into three parts: the static parameters of the physical network, the dynamic parameters of the physical network, and the virtual network requests with the demanded capacities.

F.5.2 Unfilled Physical Network Resources

The remaining capacity of each physical node at a specific time $\boldsymbol{t}$ is given by the difference between the total processing capacity and the capacity consumed by the virtual nodes previously allocated on that physical node at time $\boldsymbol{t}$, and is presented in equation (F.5), where $\boldsymbol{U}$ represents the set of virtual nodes previously allocated on that precise physical node.

$$\forall \boldsymbol{i} \in \boldsymbol{N^p} : \boldsymbol{C_i^p(t)} = \boldsymbol{C_i^p(t_0)} - \sum_{\boldsymbol{u \in U}} \boldsymbol{C_u^v(t)} \tag{F.5}$$

In parallel, the available bandwidth of each physical link at a specific time $\boldsymbol{t}$ is given by the difference between the total bandwidth and the bandwidth consumed by all virtual link segments allocated on that physical link, and is presented in equation (F.6), where $\boldsymbol{W}$ represents the set of all virtual link segments allocated on that specific physical link at time $\boldsymbol{t}$.

A virtual link can be composed by one or more physical links, i.e. a physical path. We consider that each virtual link has a single physical path, and we do not consider link aggregation (i.e. virtual link composed by different physical paths). One physical link can accommodate one or more virtual link segments belonging to different virtual links.

Table F.3: VN Re-assignment Problem Notation.

G^p	Physical Network
N^p	Set of Physical Nodes
$M^p(t)$	Set of Physical Nodes that require maintenance at time t
$i,\ j$	Physical Nodes
ij	Physical Link
L^p	Set of Physical Links
L_i^p	Set of Physical Links directly connected to Physical Node i
$C_i^p(t_0)$	Available CPU of Physical Node i at time t_0
$B_{ij}^p(t_0)$	Available Bandwidth of Physical Link ij at time t_0
t	Time
$C_i^p(t)$	Available CPU of Physical Node i at time t
$u_i(t)$	Power State of Physical Node i at time t
$P_i(t)$	Power Consumption of Physical Node i at time t
$B_{ij}^p(t)$	Available Bandwidth of Physical Link ij at time t
t_k	Time with Virtual Network Request Event k
k	Virtual Request
G^v	Virtual Network
$N^v(t_k)$	Set of Virtual Nodes at time t_k
$L^v(t_k)$	Set of Virtual Links at time t_k
$L_m^v(t_k)$	Set of Virtual Links directly connected to Virtual node m at time t_k
m, n	Virtual Nodes
mn	Virtual Link
$C_m^v(t_k)$	CPU of Virtual Node m at time t_k
$B_{mn}^v(t_k)$	Bandwidth of Virtual Link mn at time t_k

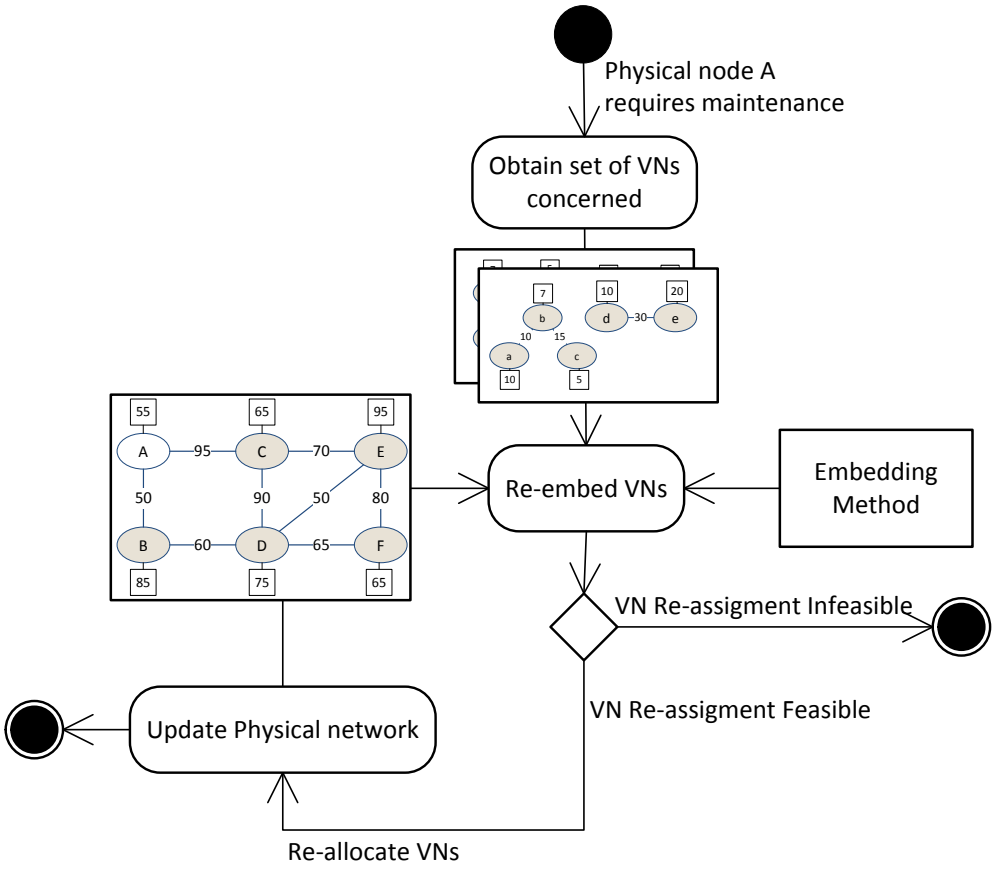


Figure F.4: VN Re-embedding - Activity Diagram

$$\forall ij \in \boldsymbol{L^p(t)} : \boldsymbol{B^p_{ij}(t)} = \boldsymbol{B^p_{ij}(t_0)} - \sum_{\boldsymbol{w \in W}} \boldsymbol{B^v_w(t)} \tag{F.6}$$

F.5.3 VN Request Re-embedding Process

The VN re-embedding process can be divided into two components: the component that ensures the re-embedding of the virtual nodes, and the one that handles the re-embedding of the virtual links.

F.5.3.1 Virtual Node Re-embedding

Each virtual node needs to be re-mapped onto one physical node; this relation is given by the mapping function $\boldsymbol{\mathcal{M}'[m \in N^v(t)_k] = i}$, where virtual node $\boldsymbol{m}$ is mapped onto physical node $\boldsymbol{i}$. The current virtual node assignment is given by the mapping function $\boldsymbol{\mathcal{M}}$.

Each physical node candidate needs to have, at least, the same amount of available CPU as required by the virtual node, which is represented in equation (F.7).

$$\forall i, \forall \boldsymbol{\mathcal{M}'[m \in N^v(t_k)] = i} : \boldsymbol{C^v_m(t_k) \leq C^p_i(t_k)} \tag{F.7}$$

F.5.3.2 Virtual Link Re-embedding

Each virtual link can be re-mapped onto one or more physical links (i.e. physical path); this relation is given by the mapping function $\mathcal{M}'[L^v_{mn}]$, where the virtual link mn is mapped onto one physical path. The current virtual link assignment is given by the mapping function $\mathcal{M}$. Each physical link candidate belonging to the physical path needs to have, at least, the same amount of bandwidth available as required by the virtual link, which is presented in equation (F.8).

$$\forall ij \subseteq \mathcal{M}'[mn \in L^v(t_k)] : B^v_{mn}(t_k) \leq B^p_{ij}(t_k) \tag{F.8}$$

F.5.4 VN Re-Embedding - Activity Diagram

The re-embedding process begins upon a new VN migration request arrival, e.g. physical node maintenance requirement, which is depicted in Figure F.4. A VN mapping method is used to re-embed the set of VNs; it takes as inputs the current status of the physical network (e.g. available CPU capacity, existing bandwidth) and the set of VNs affected. If the result of the re-embedding process is a viable solution, the mapping is considered to be feasible; if not, it is considered to be unfeasible and the VN re-embedding process stops.

F.5.5 Re-Embedding Metrics

In order to assess the performance of an embedding method, different metrics are defined:

F.5.5.1 Physical Network Resilience Factor

The physical network resilience factor, $\mathcal{R}(t)$, is given by equation (F.9) and defines the overall resilience of the virtual network to migration events, i.e. the ratio of successfully re-embedded sets of VNs, k', over the sum of sets of VNs, k.

$$\mathcal{R}(t) = \frac{k'}{k} \tag{F.9}$$

F.5.5.2 Embedding Factor

The embedding factor, $\mathcal{E}(t_k)$, is given by equation (F.10) and represents the ratio between the amount of bandwidth that was requested for the VNs and the amount of physical bandwidth that was effectively provisioned to re-accommodate the VNs, i.e. the efficiency on embedding.

$$\mathcal{E}(t) = \frac{\sum_{k \in K} \sum_{mn \in L^v_k} B^v_{mn}}{\sum_{k \in K} \sum_{mn \in L^v_k} \sum_{ij \subseteq \mathcal{M}'[mn \in L^v_k]} B^v_{mn}} \tag{F.10}$$

F.5.5.3 Bandwidth Allocation

The (additional) bandwidth allocation, $\mathcal{B}(t)$, is given by equation (F.11) and represents the ratio between the amount of bandwidth that was provisioned to re-accommodate the VNs and the amount of bandwidth that was initially provisioned, i.e. prior to the VN migration event.

$$\mathcal{B}(t) = \frac{\sum_{k \in K} \sum_{mn \in L^v_k} \sum_{ij \subseteq \mathcal{M}'[mn \in L^v_k]} B^v_{mn}}{\sum_{k \in K} \sum_{mn \in L^v_k} \sum_{ij \subseteq \mathcal{M}[mn \in L^v_k]} B^v_{mn}} - 1 \tag{F.11}$$

F.6 Virtual Network Re-Embedding Node-Link Formulation

This section describes the mathematical formulation and the objective function developed to solve the online VN re-embedding problem with the defined constraints. An Integer Linear Programming (ILP) approach is used to solve the online VN re-embedding problem. We propose a node-link formulation, and two assignment variables are applied during the re-embedding process. The index notation used here is the same as in section F.5.1.3.

F.6.1 Assignment Variables

This section presents both virtual node and link assignment variables:

F.6.1.1 Virtual Node Assignment

$$x_i^m = \begin{cases} \mathbf{1}, \text{virtual node } \boldsymbol{m} \text{ is allocated at physical node } \boldsymbol{i} \\ \mathbf{0}, \boldsymbol{else} \end{cases} \tag{F.12}$$

F.6.1.2 Virtual Link Assignment

$$y_{ij}^{mn} = \begin{cases} \mathbf{1}, \text{virtual link } \boldsymbol{mn} \text{ uses physical link } \boldsymbol{ij} \\ \mathbf{0}, \boldsymbol{else} \end{cases} \tag{F.13}$$

F.6.2 Constraints

This section presents the formulation constraints.

F.6.2.1 Assignment of virtual nodes to physical nodes

Equation (F.14) ensures that each virtual node is assigned, and that it is assigned to just one physical node.

$$\forall \boldsymbol{m} \in \boldsymbol{N^v(k)} : \sum_{\boldsymbol{i} \in \boldsymbol{N^p}} \boldsymbol{x_i^m} = \mathbf{1} \tag{F.14}$$

F.6.2.2 One virtual node per physical node

Equation (F.15) guarantees that each physical node can accommodate in the maximum one virtual node per VN, although each physical node can accommodate more than one virtual node per different VN.

$$\forall \boldsymbol{k} \in \boldsymbol{N^v}, \forall \boldsymbol{i} \in \boldsymbol{N^p} : \sum_{\boldsymbol{m} \in \boldsymbol{N^v(k)}} \boldsymbol{x_i^m} \leq \mathbf{1} \tag{F.15}$$

F.6.2.3 CPU conservation

Equation (F.16) assures that the available CPU capacity of each physical node is not exceeded.

$$\forall \boldsymbol{i} \in \boldsymbol{N^p} : \sum_{\boldsymbol{m} \in \boldsymbol{N^v}} \boldsymbol{x_i^m} \times \boldsymbol{C_m^v(t_k)} \leq \boldsymbol{C_i^p(t_k)} \tag{F.16}$$

F.6.2.4 Assignment of virtual links to physical links - multi-commodity flow conservation with node-link formulation

To simultaneously optimize the mapping of virtual links and virtual nodes, the multi-commodity flow constraint [EIS75] is applied with a node-link formulation [PM04]; moreover, the notion of direct flows on the virtual links is used, which is represented in Eq. (F.17), where $\boldsymbol{L_m^v}$ represents all the virtual links that are directly connected to the virtual node $\boldsymbol{m}$, and $\boldsymbol{L_i^p}$ represent all the physical links that are directly connected to the physical node $\boldsymbol{i}$

$$\begin{aligned} &\forall \boldsymbol{mn} \in \boldsymbol{L^v}, \boldsymbol{n} < \boldsymbol{m} \in \boldsymbol{L_m^v}, \forall \boldsymbol{i}: \\ &\sum_{\boldsymbol{ij} \in \boldsymbol{L_i^p}} (\boldsymbol{y_{ij}^{mn}} - \boldsymbol{y_{ji}^{mn}}) = \boldsymbol{x_i^m} - \boldsymbol{x_i^n} \end{aligned} \tag{F.17}$$

F.6.2.5 Bandwidth conservation

To ensure that the available bandwidth at each physical link is not surpassed, Equation (F.18) is defined.

$$\begin{aligned} &\forall \boldsymbol{ij} \in \boldsymbol{L^p}, \boldsymbol{j} < \boldsymbol{i} \in \boldsymbol{L_i^p}: \\ &\sum_{\boldsymbol{mn} \in \boldsymbol{L^v}, \boldsymbol{n} < \boldsymbol{m} \in \boldsymbol{L_m^v}} \boldsymbol{B_{mn}^v}(\boldsymbol{t_k})(\boldsymbol{y_{ij}^{mn}} + \boldsymbol{y_{ji}^{mn}}) \leq \boldsymbol{B_{ij}^p}(\boldsymbol{t_k}) \end{aligned} \tag{F.18}$$

F.6.3 Objective Function

The VN re-embedding problem requires the re-mapping of virtual nodes and virtual links, i.e. decision variables. On the nodes we can minimize the virtual nodes that need to be migrated, while on the virtual links we can minimize the overall bandwidth allocation. In this sub-section, we propose one objective function to address the VN re-embedding problem from a cost migration standpoint: i) virtual node migration minimization; ii) bandwidth allocation optimization.

F.6.3.1 Node Migration and Bandwidth Consumption Minimization

The objective function NM-BCM, proposed in equation (F.19), aims to minimize the overall number of virtual nodes migrated, which is achieved by using the first term of the equation. This objective function also aims to minimize the overall bandwidth consumption in the second term. The parameter $\boldsymbol{\alpha}$ is used to weight the cost of each virtual node migration.

$$\begin{aligned} &\boldsymbol{minimize} \sum_{\boldsymbol{k} \in \boldsymbol{K}} \sum_{\boldsymbol{m} \in \boldsymbol{N_k^v}} \sum_{\boldsymbol{i} \in \boldsymbol{N^p}} \boldsymbol{x_i^m} \times \boldsymbol{X_i^m} + \\ &\sum_{\boldsymbol{k} \in \boldsymbol{K}} \sum_{\boldsymbol{mn} \in \boldsymbol{L_k^v}, \boldsymbol{n} < \boldsymbol{m}} \boldsymbol{B_{mn}^v} \sum_{\boldsymbol{ij} \in \boldsymbol{L^p}} \boldsymbol{y_{ij}^{mn}}, \text{where} \\ &\boldsymbol{X_i^m} = \begin{cases} \boldsymbol{0}, \text{ if virtual node } \boldsymbol{m} \text{ is allocated} \\ \text{at physical node } \boldsymbol{i} \text{ or } \boldsymbol{i} \subseteq \boldsymbol{M} \\ \boldsymbol{\alpha}, \text{ otherwise} \end{cases} \end{aligned} \tag{F.19}$$

F.7 Evaluation Results

In this section, we start with a description of the network virtualisation testbed implemented to evaluate the proposed VN migration process. Two different VN migration methods are evaluated: the live migration and the clone migration with VR suspend and without VR suspend. The evaluation metrics considered are the downtime and the migration execution time. Subsection F.7.4 presents the evaluation results of the VN re-embedding process as a function of the VN request rate for all the previously described metrics (see sub-section F.5.5).

F.7.1 VN Migration - Testbed

The testbed used is composed by 6 physical hosts, which are depicted on the bottom of Figure F.5. The physical hosts are running Fedora Core 8 with Xen Hypervisor [BDF+03] version 3.1, and the virtual hosts are also running Fedora Core 8 with para-virtualisation. The top part of Figure F.5 depicts one VN, the VN ***Vegas***, which is deployed using the Network Virtualisation System Suite (NVSS) [NMCS11a], and which is used to perform the VN migration process. All VRs are running the Quagga routing suite [qua12], and are using the Open Shortest Path First (OSPF) as the routing protocol which is configured with the default parameters.

In order to generate traffic across the virtual network ***Vegas***, two physical machines are connected: one physical server that will be acting as a traffic generator, and one physical client that will consume the traffic. The physical links used by the server and the client are of 100 Mbps each.

The dashed lines in Figure F.5 represent the new location of the VR ***Nick*** and the new location of the VLs. The VR ***Nick*** is the one that will be triggered to be migrated from the physical host ***Mary*** to physical host ***Bree***. The physical hosts, ***Mary*** and ***Bree***, are rack servers with CPU Intel® Xeon® X3330@2.66GHz and E3110@3.00GHz, respectively, configured with 6GB of RAM and both using Gigabit interfaces.

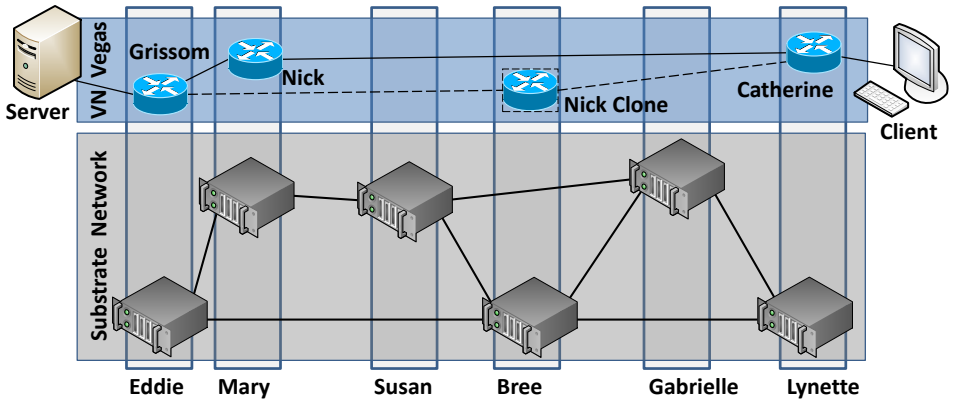


Figure F.5: Network Virtualisation Testbed: Virtual Router Migration Scenario.

F.7.2 VN Migration - Experiment Parameters

To analyse the different VN migration methods, the traffic generator D-ITG [AGE+04] is used to evaluate the impact on the traffic carried by the virtual network due the VN migration process. The total experiment time is set to 100 seconds, where the traffic is continuously generated before, during and after the VR migration event. It is considered UDP traffic with

a packet size of 1000 Bytes, and either with a bitrate of 10Mbps (i.e. 1250 packets/s) or of 20Mbps (i.e. 2500 packets/s) to evaluate the influence of the traffic bitrate on the VN migration process.

The memory RAM of the VRs, i.e. the size of the routing tables, is set from 64MB to 256MB with intervals of 32MB. During each experiment, it is measured the number of packets sent and the number of packets lost. The execution time of each Linux command is also measured: bridge setup, VLAN setup, bridge interface setup, VR cloning, VR move, VR clone restore, VR destroy, VR live migration (only for the live migration process). For each experiment, 10 trials are performed, and confidence intervals of 95% are used for every plot.

F.7.3 VN Migration - Experimental Results

F.7.3.1 VN Downtime

If we assume that there is no packet loss before and after the VN migration, we can easily obtain the VN downtime, which is the total time where the VN was inactive due to the VN migration, using equation (F.20). If we consider that the total experiment time, i.e. $\boldsymbol{T_{traffic}}$, is equal to 100 seconds, which is the case in all our experiments, we will end up having the VN downtime equal to the percentage of dropped packets.

$$\boldsymbol{VN_{downtime}} = \frac{\sum \boldsymbol{Dropped\ Packets}}{\sum \boldsymbol{Sent\ Packets}} \times \boldsymbol{T_{traffic}}\ \boldsymbol{(s)} \tag{F.20}$$

Figure F.6 shows the VN downtime as a function of the VR memory size. According to the figure, we can observe that the VN downtime exhibits two distinct behaviours: either it does not depend on the VR memory or it does strongly depend on it. In the live migration and the clone migration without VR suspend, the VN downtime does not strongly depend on the VR memory size and it is almost constant for all the memory sizes evaluated. The live migration has a VN downtime (or percentage of dropped packets) of 400 milliseconds (0.4%), and the clone migration has no downtime when using UDP traffic at 10Mbps. This behaviour is expected for both methods, where the downtime experienced on the VN live migration is mainly due to the XEN live migration procedure [CFH+05], which is an iterative process based on memory copy of dirty pages (i.e. blocks of memory which are constantly changing due to the running processes inside of the VR). In the VN clone approach, the VR memory RAM is copied at once, while the VR is still running, and at memory RAM write speeds (e.g. DDR3-800 with a peak transfer rate of 6400MBps [JED12]).

In the case of the VN clone migration with VR suspend, the VN downtime does vary with the VR memory size and increases linearly with it. This is in fact due to the cloning phase of the VR, where the VR is put into suspend mode while its memory RAM is being copied. It performs better than the live migration, i.e. achieves lower VN downtime or has lower packet loss, in situations where the VR that needs to be migrated has a memory RAM smaller than 96MB; in the remaining cases, the live migration process performs better. The VN clone migration without VR suspend outperforms the VN live migration approach and achieves no VN downtimes.

The VN clone migration with VR suspend, and with a memory size of 64MB, achieves a VN downtime of 270 milliseconds, which still makes this approach suitable for non-real time traffic and for voice over IP traffic, once it follows the ITU-T G.114[tim03] recommendation which limits the maximum acceptable round trip delay time to 300ms.

The traffic carried out by the VN does not significantly affect the $\boldsymbol{VN_{downtime}}$, although a slightly higher percentage of dropped packets is registered for the UDP traffic with higher bitrate (i.e. 20Mbps).

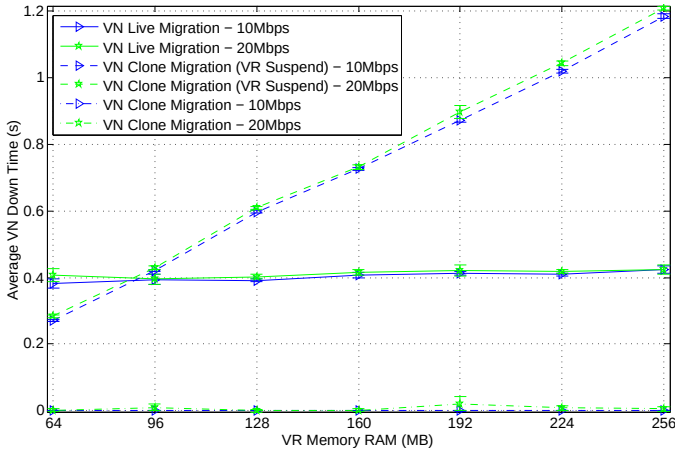


Figure F.6: Virtual Network downtime (or Percentage of Dropped Packets) as a Function of the VR Memory RAM

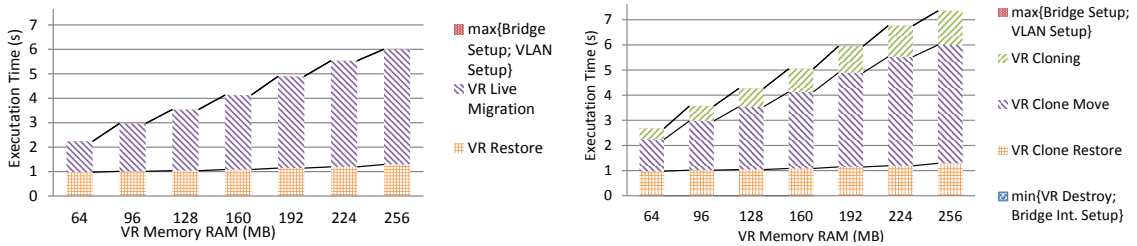
F.7.3.2 VN Migration Execution Time

The VN migration execution time is illustrated in Figure F.7. On the left side (Figure F.7a), it is represented the execution time of the live migration process, and on the right side (Figure F.7b), it is represented the execution time of the clone migration process.

The VN execution time grows linearly for both migration methods with the VR memory size. The live migration process is the one that takes less time to be fully performed, since it is performed internally by the XEN hypervisor.

In the clone migration process, the most time consuming operation is the VR Clone move (i.e. the relocation of the VR clone to the new physical host). This operation is influenced both by the VR memory size and by the available bandwidth at the physical path, between the physical host source and the physical host destination (***Mary-Susan-Bree***), which in our experiment is of 1Gbps (see equation F.2). The second most time consuming operation is the VR cloning operation, which is also dependent on the VR memory size. Both the bridging of virtual interfaces and the VLAN setup are the less time consuming operations, and take up to 4 and 8 milliseconds, respectively, to be performed.

The total execution time of the clone migration process, with a VR having 64 MB of memory RAM, is 2.75 seconds, while the total execution time of the live migration process is 2.36 seconds. Notice that the total execution time of the clone migration process can be reduced, if it is also performed internally by the hypervisor. Moreover, this time can be further reduced if the VR clone relocation takes place in parallel with the VR cloning phase.



(a) Live (b) Clone

Figure F.7: VN Migration Execution Time as Function of the VR Memory Size

F.7.4 VN Re-embedding - Simulation Parameters

To evaluate the VNRE-NLF model, we have used a discrete event simulator implemented in Matlab®, with the proposed formulation and the objective function considered.

The physical network topology is created using the GT-ITM tool [ZCB96], the number of physical nodes is set to 50, which is representative of a medium scale infrastructure provider, and the link probability between two physical nodes is set to 0.5. The node CPU capacity and the link bandwidth are real numbers uniformly distributed between 50 and 100. The VN requests are also representative of either small or medium scale virtual networks, and are created using the same topology generation method. The number of virtual nodes is not fixed, but follows an uniform distribution, from 2 to 10 virtual nodes per VN topology; the virtual link probability is set to 0.5. The CPU capacity of the virtual nodes and the bandwidth of the virtual links are also real numbers uniformly distributed between 0 and 20, and between 0 and 50, respectively[2].

We assume that a VN re-embedding request is triggered by a physical node maintenance event (or eventually imminent failure), according to a Poisson process of 1 event per 500 time units. A set of physical nodes that need to be power-off for maintenance is randomly generated, i.e. integers uniformly distributed between 1 and 50. We have considered sets of of 1, 2 and 3 physical nodes, which correspond to a shutdown of 2%, 4%, and 6% of the physical network resources. The CPU and bandwidth capacity of the nodes and links affected is set to zero.

To populate the physical network, we assume VN requests arriving according to a Poisson process, and that each VN has an associated lifetime measured in time units with an average of $\mathbf{1/\mu = 1000}$, following an exponential distribution. The same assumption was also taken by the authors of [MSK+13]. The VN request rate, i.e., value of $\boldsymbol{\lambda}$, is started with a rate of 3 VN requests per 100 time units, and increases by 1 VN request, up to a rate of 10 VN requests.

For each value of $\boldsymbol{\lambda}$, 10 trials are performed, and the same set of re-embedding events is considered. A new set of VN requests, a new physical network topology, and new set of VN re-embedding events are generated for each trial. All simulations are set to run up to 50000 time units to mitigate the transient phase effect [Jai91] and to obtain the steady-state. A confidence interval of 95% is used for all results presented below.

The CPLEX®[cpl12] version 12.2 is used to solve the linear programming problem of the VNRE-NLF; a time limit of 300 seconds is defined for each set of VN mapping, although most of the VNs are re-embedded in less than 60 seconds; the CPLEX® is set to use up to two threads. All the simulations were performed using an Intel® Core ™ Processor i5-3210M @2.5GHz, and the time consumed per set of VNs re-embedding is registered.

The evaluation metrics are the ones defined in section F.5.5.

F.7.5 VN Re-embedding - Simulation Results

This sub-section presents the evaluation results of the process of VN re-embedding as a function of the VN request rate for all the previously described metrics (see sub-section F.5.5).

F.7.5.1 Physical Network Resilience Factor

Figure F.8 depicts the average virtual network resilience factor as a function of the VN request rate and the percentage of physical resources shutdown. The results show that the

[2]These values were also considered by the authors of [YYRC08, ZA06, CRB09, MSK+13].

network resilience decreases with the VN request rate and also with the percentage of resources shutdown. On one hand, with a higher VN request rate, the number of VN requests allocated in the physical network increases. On the other hand, by turning-off more physical resources, the number of VNs affected by the decrease of the amount of free resources does not increase. Therefore, it penalizes the VN resilience factor. Noteworthy, the VNRE-NLF is able to re-embed all sets of VNs affected for a VN request rate of 3, and for all considered percentage of physical resources shutdown.

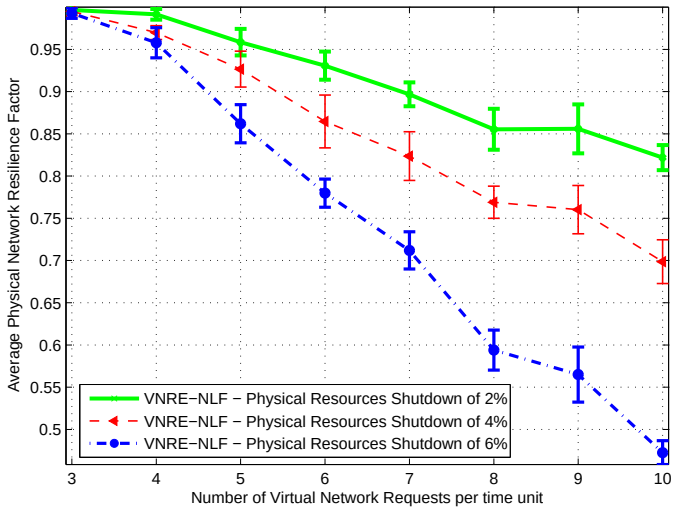


Figure F.8: Average Physical Network Resilience Factor per VN request.

F.7.5.2 Virtual Nodes Migration

Virtual node migration is the component with the highest impact in the VN migration execution time (see Figure F.7). The virtual node migration process not only consumes additional CPU of the source node and destination node, but also requires extra physical bandwidth to transfer the virtual node between the physical nodes. Figure F.11 presents the average percentage of virtual nodes migrated per set of VNs affected and as a function of the VN request rate. The virtual nodes previously assigned to physical nodes that need to be turned off do not count to the overall percentage of virtual nodes migrated, since their migration to new physical nodes is mandatory. Therefore, the percentage of virtual nodes migrated only incorporate the virtual nodes placed on other physical nodes, and that need to be migrated to make the assignment problem feasible. The number of nodes migrated increases with the VN request rate and with the percentage of physical resources shutdown. By increasing the number of VNs allocated on the physical network, i.e. VN request rate, and the percentage of physical resources shutdown, the spare capacity of the physical network is reduced. Hence, it is required to migrate more virtual nodes to make the re-embedding feasible.

F.7.5.3 Embedding Factor

Re-embedding sets of VNs is important to minimize the number of virtual nodes migrated, and also to re-embed the virtual links in the smallest set of physical links possible, to save bandwidth for the incoming VN requests. Figure F.10 shows the embedding factor for each

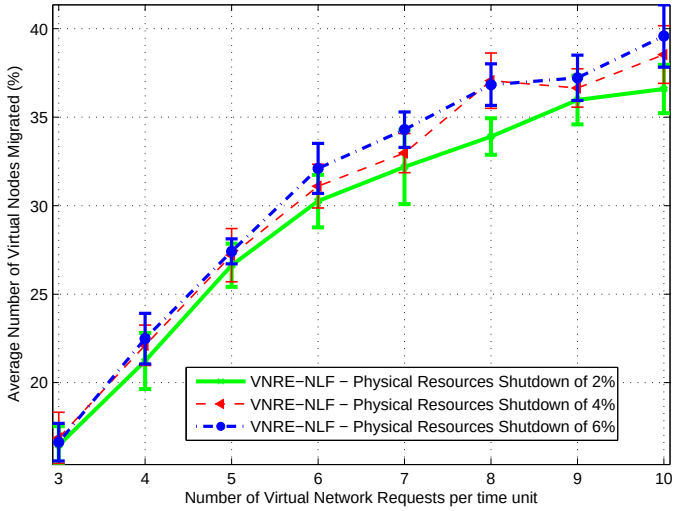


Figure F.9: Average Percentage of Virtual Nodes Migrated per VN request.

set of VNs re-embedded and as a function of the VN request rate. The embedding factor decreases with the VN request rate and with the percentage of physical resources shutdown. By increasing the number of VN request rate and the percentage of physical resources shutdown, we are not only reducing the spare capacity of the physical network, but also making the VN re-embedding less efficient in terms of virtual link re-assignment (i.e less physical bandwidth is available to make it possible to re-embed the overall virtual links in the smallest amount of physical links possible). Consequently, the network has virtual links consuming more than one physical link.

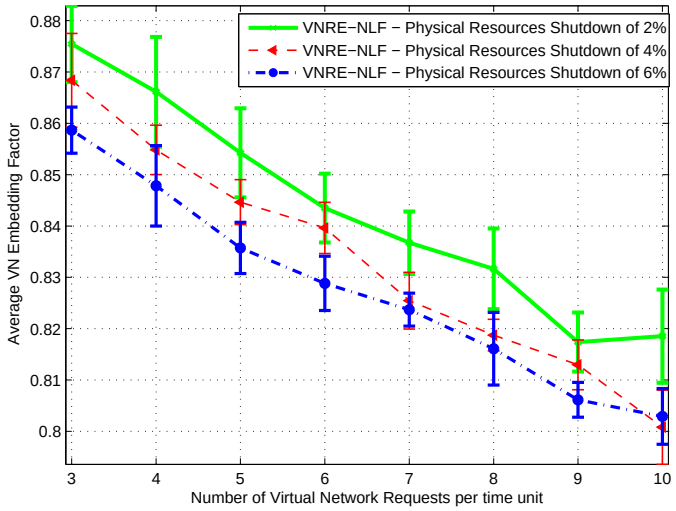


Figure F.10: Average Embedding Factor per VN request.

F.7.5.4 Physical Bandwidth Allocation

It is also import to know how much bandwidth is additionally provisioned for each set of VNs re-embedded. Figure F.11 depicts the average percentage of additional bandwidth allocation per set of VNs re-embedded and as a function of the VN request rate. The addi-

tional bandwidth increases with the VN request rate, and also with the percentage of physical resources shutdown. By increasing the number of VN request rate and the percentage of physical resources shutdown, we are implicitly increasing the percentage of virtual nodes migrated (as stated in F.7.5.2); if we aim at minimizing the number of virtual nodes migrated, we will sacrifice the virtual links (i.e. longer paths are taken by the virtual links). Therefore, it is consumed an additional bandwidth per VN request rate and percentage of physical resources shutdown.

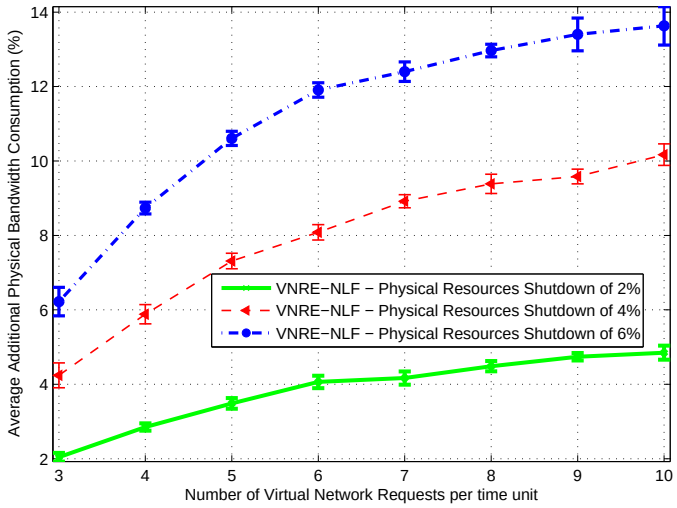


Figure F.11: Average Percentage of Additional Physical Bandwidth per VN request.

F.7.5.5 VN Re-embedding Time

An important aspect of a VN re-embedding method is the time that it requires to re-embed, on average, a set of VNs, and how it varies with respect to the different loads on the physical infrastructure, i.e. VN request rate, and with the different percentage of physical resources shutdown. Figure F.12 shows the solving time for each set of VNs and as a function of the VN request rate. The solving time increases significantly with the VN request rate, and also with the percentage of physical resources shutdown. The solving time directly depends on the number of virtual links and virtual nodes that need to be re-embedded. By increasing the VN request rate, we are increasing the number of VNs allocated and implicitly the number of VNs potentially affected by a physical resource shutdown. By increasing the percentage of physical resources shutdown, we are directly increasing the number of VNs affected. Therefore, by increasing the number of VNs affected, we are increasing the number of virtual nodes and links to be re-embedded and implicitly increasing the solving time.

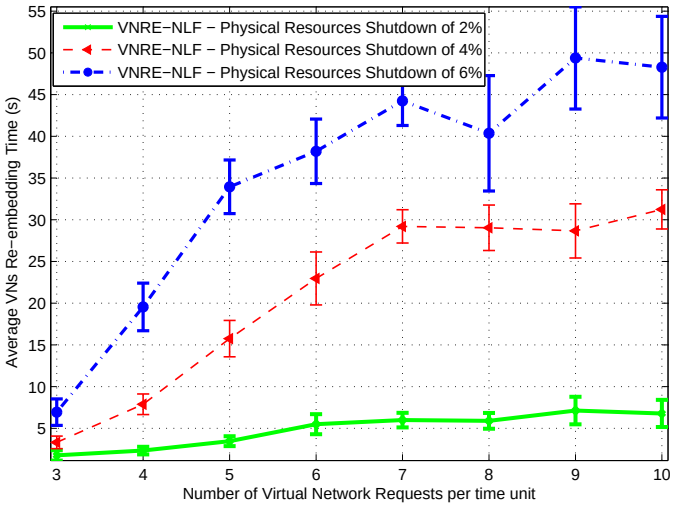


Figure F.12: Average VN Re-embedding Time.

F.8 Conclusion and Future Work

The VN migration process is an important mechanism both for network management purposes and for the conception and deployment of new network architectures and protocols based on network virtualisation. The VN migration enables the virtual network to not be physically tied to a set of hosts previously assigned, and it makes it viable to move components of a VN or even the entire VN from one set of physical hosts to a new set on a seamless way and on-demand.

This paper proposed the VN clone migration as an alternative to the live migration approach. The VN clone migration performs cloning of the VR and transfers the VR clone to the new physical host. This approach requires no restrictions on the virtual network itself or on the networking protocols running inside of the virtual network. The results show that the proposed approach achieves no VN downtime, and it takes just a few seconds to be fully performed.

Additionally, it is proposed the VNRE-NLF to solve the online virtual network re-embedding problem as a simultaneous optimization of virtual nodes and virtual links, providing the optimal bound for each set of virtual networks migrated. This approach aims at minimizing the overall VN migration cost per re-embedding: i) number of virtual nodes migrated; ii) physical bandwidth consumption. The results show that the virtual network resilience to migration events is directly affected by the VN request rate and by the percentage of physical resources shut-down. From the obtained results we can conclude that the VN is highly resilient to migration events for the VN request rates considered if no more than 2% of the physical resources need to be shut-down. We can also conclude that it is not only important to have enough spare capacity to re-accommodate the virtual nodes and virtual links affected by the physical resource shut-down, but also to have additional capacity to accommodate virtual link re-assignments and virtual node migrations performed to make the VN re-embedding problem feasible.

Future work spans from the integration of the VR cloning approach with the hypervisor, to the investigation and inclusion of new parameters on the objective function: memory RAM of the virtual nodes (i.e. to not only minimize the number of virtual nodes migrated, but also to minimize the migration of virtual nodes with higher amount of memory RAM); number of hops between physical nodes (to minimize the number of hops a virtual node needs to cross

to be migrated).

References

[4WA09] 4WARD Consortium. Virtualisation approach: Concept. Technical report, FP7-ICT-4WARD project, Deliverable D3.1.1, Sep. 2009.

[4WA10a] 4WARD Consortium. Virtualisation approach: Evaluation and integration. Technical report, FP7-ICT-4WARD project, Deliverable D3.2.0, Jan. 2010.

[4WA10b] 4WARD Consortium. Virtualisation approach: Evaluation and integration - update. Technical report, FP7-ICT-4WARD project, Deliverable D3.2.1, Jun. 2010.

[4WA14] 4WARD - Architecture and design for the future Internet. `http://www.4ward-project.eu`, Jan. 2014.

[ABdF13] Gustavo P. Alkmim, Daniel M Batista, and Nelson L. S. da Fonseca. Mapping virtual networks onto substrate networks. *Journal of Internet Services and Applications*, 4(1):1–15, 2013.

[ABKM01] David Andersen, Hari Balakrishnan, Frans Kaashoek, and Robert Morris. *Resilient overlay networks*, volume 35. ACM, 2001.

[ABSdF11] Gustavo Prado Alkmim, Daniel Macêdo Batista, and NL Saldanha da Fonseca. Optimal mapping of virtual networks. In *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pages 1–6. IEEE, 2011.

[AGE+04] S. Avallone, S. Guadagno, D. Emma, A. Pescape, and G. Ventre. D-ITG distributed internet traffic generator. In *Quantitative Evaluation of Systems, 2004. QEST 2004. Proceedings. First International Conference on the*, pages 316 – 317, September 2004.

[AN08] Tohru Asami and Shu Namiki. Energy consumption targets for network systems. In *Optical Communication, 2008. ECOC 2008. 34th European Conference on*, pages 1–4. IEEE, 2008.

[And02] David G. Andersen. Theoretical approaches to node assignment. Unpublished Manuscript, December 2002.

[APST05a] T. Anderson, L. Peterson, S. Shenker, and J. Turner. Overcoming the internet impasse through virtualization. *Computer*, 38(4):34 – 41, April 2005.

[APST05b] Thomas Anderson, Larry Peterson, Scott Shenker, and Jonathan Turner. Overcoming the Internet Impasse through Virtualization. *Computer*, 38:34–41, April 2005.

[AS11] Ehab Al-Shaer. Toward network configuration randomization for moving target defense. In Sushil Jajodia, Anup K. Ghosh, Vipin Swarup, Cliff Wang, and X. Sean

Wang, editors, *Moving Target Defense*, volume 54, pages 153–159. Springer New York, New York, NY, 2011.

[BDF+03] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177, Bolton Landing, NY, USA, 2003. ACM.

[BFH+06] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford. In VINI veritas: realistic and controlled network experimentation. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, page 14, 2006.

[BH09] Luiz André Barroso and Urs Hölzle. The datacenter as a computer: An introduction to the design of warehouse-scale machines. *Synthesis Lectures on Computer Architecture*, 4(1):1–108, 2009.

[BHD+12] J.F. Botero, X. Hesselbach, M. Duelli, D. Schlosser, A. Fischer, and H. De Meer. Energy efficient virtual network embedding. *Communications Letters, IEEE*, 16(5):756–759, 2012.

[BHFM12] Juan Felipe Botero, Xavier Hesselbach, Andreas Fischer, and Hermann Meer. Optimal mapping of virtual networks with hidden hops. *Telecommunication Systems*, 51(4):273–282, 2012.

[BHK12] Abdeltouab Belbekkouche, Md. Mahmud Hasan, and Ahmed Karmouch. Resource discovery and allocation in network virtualization. *Communications Surveys Tutorials, IEEE*, 14(4):1114–1128, 2012.

[bri12] 802.1D Bridge implementation for Linux. `http://www.linuxfoundation.org/collaborate/workgroups/networking/bridge`, August 2012.

[Bun00] H. Bunke. Recent developments in graph matching. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 2, pages 117–124 vol.2, 2000.

[CB09] N. Mosharaf K. Chowdhury and Raouf Boutaba. Network virtualization: State of the art and research challenges. *IEEE Communications Magazine*, 47(7):20–26, July 2009.

[CFH+05] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, pages 273–286, 2005.

[CHM+03] J. Crowcroft, S. Hand, R. Mortier, T. Roscoe, and A. Warfield. Plutarch: an argument for network pluralism. In *Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*, page 266, 2003.

[Cis09a] Cisco. Cisco Visual Networking Index Forecast and Methodology 2008-2013. White paper, June 2009.

[Cis09b] Cisco. Router virtualization in service providers,white paper. `http://www.cisco.com/en/US/solutions/collateral/ns341/ns524/ns562/ns573/white_paper_c11-512753.html`, July 2009.

[CJ09] Jorge Carapinha and Javier Jiménez. Network virtualization: a view from the bottom. In *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, VISA '09, pages 73–80, New York, NY, USA, 2009. ACM.

[CLW+10] Yang Chen, Jianxin Li, Tianyu Wo, Chunming Hu, and Wantao Liu. Resilient virtual network service provision in network virtualization environments. In *Parallel and Distributed Systems (ICPADS), 2010 IEEE 16th International Conference on*, pages 51–58, 2010.

[CLW12] Xuzhou Chen, Yan Luo, and Jie Wang. Virtual network embedding with border matching. In *Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on*, pages 1–8, 2012.

[CLX+10] Zhiping Cai, Fang Liu, Nong Xiao, Qiang Liu, and Zhiying Wang. Virtual network embedding for evolving networks. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pages 1–5, 2010.

[con03] Connectix Virtual Server. `http://www.connectix.com/products/vs.html`, 2003.

[Con06] VPN Consortium. VPN Technologies: Definitions and Requirements. `http://www.cisco.com/en/US/solutions/collateral/ns341/ns524/ns562/ns573/white_paper_c11-512753.html`, March 2006.

[cpl12] IBM ILOG Optimization Products. `www-01.ibm.com/software/websphere/products/optimization`, Sep. 2012.

[CRB09] N.M.M.K. Chowdhury, M.R. Rahman, and R. Boutaba. Virtual network embedding with coordinated node and link mapping. In *INFOCOM 2009, IEEE*, pages 783 –791, april 2009.

[CRB12] M. Chowdhury, M.R. Rahman, and R. Boutaba. Vineyard: Virtual network embedding algorithms with coordinated node and link mapping. *Networking, IEEE/ACM Transactions on*, 20(1):206 –219, feb. 2012.

[CRSZ01] Yang Chu, Sanjay Rao, Srinivasan Seshan, and Hui Zhang. Enabling conferencing applications on the internet using an overlay muilticast architecture. In *ACM SIGCOMM Computer Communication Review*, volume 31, pages 55–67. ACM, 2001.

[CSB+08] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright. Power awareness in network design and routing. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 457–465, 2008.

[CSB10] Mosharaf Chowdhury, Fady Samuel, and Raouf Boutaba. Polyvine: policy-based virtual network embedding across multiple domains. In *Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures*, pages 49–56. ACM, 2010.

[CSZ+11] Xiang Cheng, Sen Su, Zhongbao Zhang, Hanchi Wang, Fangchun Yang, Yan Luo, and Jie Wang. Virtual network embedding through topology-aware node ranking. *ACM SIGCOMM Computer Communication Review*, 41(2):38–47, 2011.

[CSZ+12] Xiang Cheng, Sen Su, Zhongbao Zhang, Kai Shuang, Fangchun Yang, Yan Luo, and Jie Wang. Virtual network embedding through topology awareness and optimization. *Computer Networks*, 56(6):1797 – 1813, 2012.

[DBR02] S. W Devine, E. Bugnion, and M. Rosenblum. *Virtualization system including a virtual machine monitor for a computer with a segmented architecture.* Google Patents, May 2002. US Patent 6,397,242.

[EFI14] European Future Internet Portal. `http://www.future-internet.eu`, Jan. 2014.

[EIS75] S. Even, A. Itai, and A. Shamir. On the complexity of time table and multi-commodity flow problems. In *Proceedings of the 16th Annual Symposium on Foundations of Computer Science*, SFCS '75, pages 184–193, Washington, DC, USA, 1975. IEEE Computer Society.

[Epp98] David Eppstein. Finding the k shortest paths. *SIAM Journal on computing*, 28(2):652–673, 1998.

[ETS] Network Functions Virtualisation – Introductory White Paper. `http://portal.etsi.org/NFV/NFV_White_Paper.pdf`, Oct.

[ETS13a] ETSI. Gs nfv 001 - network functions virtualisation (nfv); use cases, October 2013.

[ETS13b] ETSI. Gs nfv 002 - network functions virtualisation (nfv); architectural framework, October 2013.

[ETS13c] ETSI. Gs nfv 003 - network functions virtualisation (nfv); terminology for main concepts in nfv, October 2013.

[ETS13d] ETSI. Gs nfv 004 - network functions virtualisation (nfv); virtualisation requirements, October 2013.

[ETS13e] ETSI. Gs nfv-per 002 - network functions virtualisation (nfv); proofs of concepts; framework, October 2013.

[FA06] J. Fan and M.H. Ammar. Dynamic topology configuration in service overlay networks: A study of reconfiguration policies. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–12, 2006.

[FAPZ11a] I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann. Vne-ac: Virtual network embedding algorithm based on ant colony metaheuristic. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–6, 2011.

[FAPZ11b] I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann. Vnr algorithm: A greedy approach for virtual networks reconfigurations. In *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pages 1–6, 2011.

[FBCB10] Nabeel Farooq Butt, Mosharaf Chowdhury, and Raouf Boutaba. Topology-awareness and reoptimization mechanism for virtual network embedding. In *Proceedings of the 9th IFIP TC 6 international conference on Networking*, NETWORKING'10, pages 27–39, Berlin, Heidelberg, 2010. Springer-Verlag.

[FBTB+13] A. Fischer, J.F. Botero, M. Till Beck, H. de Meer, and X. Hesselbach. Virtual network embedding: A survey. *Communications Surveys Tutorials, IEEE*, 15(4):1888–1906, 2013.

[FDM11] Andreas Fischer and Hermann De Meer. Position paper: Secure virtual network embedding. *Praxis der Informationsverarbeitung und Kommunikation*, 34(4):190–193, 2011.

[FED14] FEDERICA. `http://www.fp7-federica.eu`, Jan. 2014.

[Fel07] A. Feldmann. Internet clean-slate design: what and why? *ACM SIGCOMM Computer Communication Review*, 37(3):64, 2007.

[FFF99] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. In *ACM SIGCOMM Computer Communication Review*, volume 29, pages 251–262. ACM, 1999.

[FGR07a] N. Feamster, L. Gao, and J. Rexford. How to lease the internet in your spare time. *ACM SIGCOMM Computer Communication Review*, 37(1):64, 2007.

[FGR07b] Nick Feamster, Lixin Gao, and Jennifer Rexford. How to lease the internet in your spare time. *SIGCOMM Comput. Commun. Rev.*, 37(1):61–64, 2007.

[FIN14] National Science Fondation Network Technology and Systems Future INternet Design Initiative (NSF NeTS FIND). `http://www.nets-find.net`, Jan. 2014.

[Fou14] Open Network Foundation. Software-Defined Networking: The New Norm for Networks. White paper, Jan. 2014.

[gen14] GENI - Global Environment for Network Innovations. `http://www.geni.net/`, Jan. 2014.

[gey14] GEYSERS - Generalized Architecture for Dynamic Infrastructure Services. `http://www.geysers.eu`, Jan. 2014.

[GH88] David E Goldberg and John H Holland. Genetic algorithms and machine learning. *Machine learning*, 3(2):95–99, 1988.

[glp12] GLPK (GNU Linear Programming Kit) . `http://www.gnu.org/software/glpk/`, Sep. 2012.

[Gol74] R. P Goldberg. Survey of virtual machine research. *IEEE Computer*, 7(6):34–45, 1974.

[Gom13] R. Gomes. Demonstrador de uma rede com tecnologia openflow. Master's thesis, Universidade de Aveiro, 2013.

[GS03a] Maruti Gupta and Suresh Singh. Greening of the internet. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 19–26, Karlsruhe, Germany, 2003. ACM.

[GS03b] Maruti Gupta and Suresh Singh. Greening of the internet. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '03, pages 19–26, New York, NY, USA, 2003. ACM.

[GS11] T. Ghazar and N. Samaan. Hierarchical approach for efficient virtual network embedding based on exact subgraph matching. In *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pages 1–6, 2011.

[GWC+09] V. Gupta, M. Williams, A. Chan, X. Liu, D. Cypher, Y. Y. An, et al. IEEE802. 21 standard and metropolitan area networks: Media independent handover services. *Draft P*, 21:D00, 2009.

[GWMT11] Tao Guo, Ning Wang, K. Moessner, and R. Tafazolli. Shared backup network provision for virtual network embedding. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–5, 2011.

[GYA+10] Xiujiao Gao, Hongfang Yu, Vishal Anand, Gang Sun, and Hao Di. A new algorithm with coordinated node and link mapping for virtual network embedding based on lp relaxation. volume 7988, pages 79881Y–79881Y–9, 2010.

[HKA13] Sandra Herker, Ashiq Khan, and Xueli An. Survey on survivable virtual network embedding problem and solutions. In *ICNS 2013, The Ninth International Conference on Networking and Services*, pages 99–104, 2013.

[HKM+98] Michael Hicks, Pankaj Kakkar, Jonathan T Moore, Carl A Gunter, and Scott Nettles. Plan: A packet language for active networks. In *ACM SIGPLAN Notices*, volume 34, pages 86–93. ACM, 1998.

[HLAZ11] Ines Houidi, Wajdi Louati, Walid Ben Ameur, and Djamal Zeghlache. Virtual network provisioning across multiple substrate networks. *Computer Networks*, 55(4):1011 – 1023, 2011. <ce:title>Special Issue on Architectures and Protocols for the Future Internet</ce:title>.

[HLZ08] I. Houidi, W. Louati, and D. Zeghlache. A distributed virtual network mapping algorithm. In *Communications, 2008. ICC '08. IEEE International Conference on*, pages 5634 –5640, may 2008.

[HLZ+10] Ines Houidi, Wajdi Louati, Djamal Zeghlache, Panagiotis Papadimitriou, and Laurent Mathy. Adaptive virtual network provisioning. In *Proceedings of the Second ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures*, VISA '10, pages 41–48, New York, NY, USA, 2010. ACM.

[HLZB09] I. Houidi, W. Louati, D. Zeghlache, and S. Baucke. Virtual resource description and clustering for virtual network discovery. In *Communications Workshops, 2009. ICC Workshops 2009. IEEE International Conference on*, pages 1–6, 2009.

[HPN09] Aun Haider, Richard Potter, and Akihiro Nakao. Challenges in resource allocation in network virtualization. In *20th ITC Specialist Seminar*, volume 18, page 20, 2009.

[IEE06] IEEE. 802.1ad - provider bridges. `http://www.ieee802.org/1/pages/802.1ad.html`, May 2006.

[IEE11] IEEE. 802.1q - virtual lans. `http://www.ieee802.org/1/pages/802.1Q.html`, Aug. 2011.

[IR11] Johannes Inführ and Günther R. Raidl. Introducing the virtual network mapping problem with delay, routing and location constraints. In Julia Pahl, Torsten Reiners, and Stefan Voß, editors, *Network Optimization*, volume 6701 of *Lecture Notes in Computer Science*, pages 105–117. Springer Berlin Heidelberg, 2011.

[IRT14] Internet Research Task Force -Virtual Networks Research Group (VNRG). `http://www.irtf.org/charter?gtype=rg&group=vnrg`, Jan. 2014.

[ITU11] ITU-T. Rec. y.3001 - future network vision — objectives and design goals, 2011.

[ITU12a] ITU-T. Rec. y.3011 - framework of network virtualiza-tion for future networks, 2012.

[ITU12b] ITU-T. Rec. y.3021 - framework of energy saving for future networks, 2012.

[ITU12c] ITU-T. Rec. y.3031 - identification framework in future networks, 2012.

[ITU14] Focus Group on Future Networks (FG FN). `http://www.itu.int/en/ITU-T/focusgroups/fn/Pages/Default.aspx`, Jan. 2014.

[Jai91] R. Jain. *The art of computer systems performance analysis*, volume 182. John Wiley & Sons New York, 1991.

[JED12] JEDEC. 204-Pin DDR3 SDRAM SO-DIMM Specification, August 2012.

[Jun09] Juniper. Virtualization in the core of the network, white paper. White paper, July 2009.

[Ken10] James Kennedy. Particle swarm optimization. In *Encyclopedia of Machine Learning*, pages 760–766. Springer, 2010.

[KJV83] Scott Kirkpatrick, D. Gelatt Jr., and Mario P Vecchi. Optimization by simmulated annealing. *science*, 220(4598):671–680, 1983.

[kvm14] Kernel-based Virtual Machine. `http://www.linux-kvm.org`, Jan. 2014.

[LAZ12] Samantha Lo, Mostafa Ammar, and Ellen Zegura. Design and analysis of schedules for virtual network migration. *Georgia Institute of Technology SCS Technical Report*, GT-CS-12-05, July 2012.

[LFDM14] Gergö Lovász, Andreas Fischer, and Hermann De Meer. Network virtualization and energy efficiency, Jan. 2014.

[LGL+11] Guohan Lu, Chuanxiong Guo, Yulong Li, Zhiqiang Zhou, Tong Yuan, Haitao Wu, Yongqiang Xiong, Rui Gao, and Yongguang Zhang. Serverswitch: a programmable and high performance platform for data center networks. In *Proceedings of the 8th USENIX conference on Networked systems design and implementation*, pages 2–2. USENIX Association, 2011.

[LHCL11] Jiang Liu, Tao Huang, Jian-ya Chen, and Yun-jie Liu. A new algorithm based on the proximity principle for the virtual network embedding problem. *Journal of Zhejiang University SCIENCE C*, 12(11):910–918, 2011.

[LHkW+11] Bo LÜ, Tao HUANG, Zhen kai WANG, Jian ya CHEN, Yun jie LIU, and Jiang LIU. Adaptive scheme based on status feedback for virtual network mapping. *The Journal of China Universities of Posts and Telecommunications*, 18(5):87 – 94, 2011.

[LK09] Jens Lischka and Holger Karl. A virtual network mapping algorithm based on subgraph isomorphism detection. In *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, pages 81–88, Barcelona, Spain, 2009. ACM.

[LPP12] A. Leivadeas, C. Papagianni, and S. Papavassiliou. Socio-aware virtual network embedding. *Network, IEEE*, 26(5):35–43, 2012.

[LT06] Jing Lu and Jonathan Turner. Efficient mapping of virtual networks onto a shared substrate. Technical report, Washington University in St. Louis, 2006.

[LWD+14] Xiaoling Li, Huaimin Wang, Bo Ding, Xiaoyong Li, and Dawei Feng. Resource allocation with multi-factor node ranking in data center networks. *Future Generation Computer Systems*, 32(0):1 – 12, 2014. <ce:title>Special Section: The Management of Cloud Systems, Special Section: Cyber-Physical Society and Special Section: Special Issue on Exploiting Semantic Technologies with Particularization on Linked Data over Grid and Cloud Architectures</ce:title>.

[LWH+10] Bo Lv, Zhenkai Wang, Tao Huang, Jianya Chen, and Yunjie Liu. Virtual resource organization and virtual network embedding across multiple domains. In *Multimedia Information Networking and Security (MINES), 2010 International Conference on*, pages 725–728, 2010.

[MAB+08] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[Mak12] Andrew Makhorin. Glpk (gnu linear programming kit). `http://www.gnu.org/software/glpk/`, 2012.

[MCS+12] Márcio Melo, Jorge Carapinha, Susana Sargento, Luis Torres, Nga Phuong-Tran, Ulrich Killat, and Andreas Timm-Giel. Virtual network mapping - an optimization problem. In Kostas Pentikousis, Rui Aguiar, Susana Sargento, and Ramón Agüero, editors, *Mobile Networks and Management*, volume 97 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 187–200. Springer Berlin Heidelberg, 2012.

[MCS13a] M. Melo, J. Carapinha, and S. Sargento. Network virtualization: A step closer for seamless resource mobility. In *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, pages 692–695, 2013.

[MCS+13b] Márcio Melo, Jorge Carapinha, Susana Sargento, Ulrich Killat, and Andreas Timm-Giel. A re-optimization approach for virtual network embedding. In Andreas Timm-Giel, John Strassner, Ramón Agüero, Susana Sargento, and Kostas Pentikousis, editors, *Mobile Networks and Management*, volume 58 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 271–283. Springer Berlin Heidelberg, 2013.

[Mel10] Márcio Melo. A network virtualisation framework in operator perspective. *In Revista do Departamento de Eletrónica, Informática e Telecomunicações, Universidade de Aveiro*, 5(2):226–234, 2010.

[MEN+13] D. Matsubara, T. Egawa, N. Nishinaga, V.P. Kafle, Myung-Ki Shin, and A. Galis. Toward future networks: A viewpoint from itu-t. *Communications Magazine, IEEE*, 51(3):112–118, 2013.

[MGNB10] C.C. Marquezan, L.Z. Granville, G. Nunzi, and M. Brunner. Distributed autonomic resource management for network virtualization. In *Network Operations and Management Symposium (NOMS), 2010 IEEE*, pages 463–470, 2010.

[MLL10] Fei Ma, Feng Liu, and Zhen Liu. Live virtual machine migration based on improved pre-copy approach. In *2010 IEEE International Conference on Software Engineering and Service Sciences (ICSESS)*, pages 230 –233, July 2010.

[MNG+09] C.C. Marquezan, J.C. Nobre, L.Z. Granville, G. Nunzi, D. Dudkowski, and M. Brunner. Distributed reallocation scheme for virtual network resources. In *IEEE International Conference on Communications, 2009. ICC '09*, pages 1 –5, June 2009.

[Mon11] R. Monteiro. Criação e reconfiguração de redes virtuais na perspetiva do operador. Master's thesis, Universidade de Aveiro, 2011.

[MS97] Q. Ma and P. Steenkiste. On path selection for traffic with bandwidth guarantees. In *Network Protocols, 1997. Proceedings., 1997 International Conference on*, pages 191–202, 1997.

[MSC09] M. Melo, S. Sargento, and J. Carapinha. Network Virtualisation from an Operator Perspective. *Proc Conf. sobre Redes de Computadores - CRC*, October, 2009.

[MSC14] M. Melo, S. Sargento, and J. Carapinha. Optimal virtual network migration: a step closer for seamless resource mobility. January, 2014.

[MSK+13] Marcio Melo, Susana Sargento, Ulrich Killat, Andreas Timm-Giel, and Jorge Carapinha. Optimal virtual network embedding: Node-link formulation. *Network and Service Management, IEEE Transactions on*, 10(4):356–368, 2013.

[MSK+14] M. Melo, S. Sargento, U. Killat, A. Timm-Giel, and J. Carapinha. Optimal virtual network embedding: Energy aware formulation. January, 2014.

[nex12] Cisco nexus 5000 series architecture: The building blocks of the unified fabric. Cisco White Paper, August 2012.

[NMCS11a] J. Nogueira, M. Melo, J. Carapinha, and S. Sargento. Network virtualization system suite: Experimental network virtualization platform. In *TridentCom 2011, 7th International ICST Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities*, 2011.

[NMCS11b] J. Nogueira, M. Melo, J. Carapinha, and S. Sargento. Virtual network mapping into heterogeneous substrate networks. In *Computers and Communications (ISCC), 2011 IEEE Symposium on*, pages 438 –444, 28 2011-july 1 2011.

[Nog10] J. Nogueira. Demonstrador de criação de redes virtuais no Âmbito do operador. Master's thesis, Universidade de Aveiro, 2010.

[NPI+08a] Sergiu Nedevschi, Lucian Popa, Gianluca Iannaccone, Sylvia Ratnasamy, and David Wetherall. Reducing network energy consumption via sleeping and rate-adaptation. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, pages 323–336, San Francisco, California, 2008. USENIX Association.

[NPI+08b] Sergiu Nedevschi, Lucian Popa, Gianluca Iannaccone, Sylvia Ratnasamy, and David Wetherall. Reducing network energy consumption via sleeping and rate-adaptation. In *NSDI*, volume 8, pages 323–336, 2008.

[ORB14] ORBIT. `http://www.orbit-lab.org`, Jan. 2014.

[PAB+06] L. Peterson, T. Anderson, D. Blumenthal, D. Casey, D. Clark, D. Estrin, J. Evans, D. Raychaudhuri, M. Reiter, J. Rexford, et al. GENI design principles. *IEEE Computer*, 39(9):102–105, 2006.

[PACR03] Larry Peterson, Tom Anderson, David Culler, and Timothy Roscoe. A blueprint for introducing disruptive technology into the internet. *SIGCOMM Comput. Commun. Rev.*, 33:59–64, January 2003.

[pan14] Pan-European Laboratory. `http://www.panlab.net`, Jan. 2014.

[Par12] B. Parreira. Integração da cloud com rede na perspectiva de operador. Master's thesis, Universidade de Aveiro, 2012.

[Per02] C. E. Perkins. Mobile ip. *Communications Magazine, IEEE*, 40(5):66–82, 2002.

[PFC+10] P. Pisa, N. Fernandes, H. Carvalho, M. Moreira, M. Campista, L. Costa, and O. Duarte. Openflow and xen-based virtual network migration. *Communications: Wireless in Developing Countries and Networks of the Future*, pages 170–181, 2010.

[PJ06] V. Prevelakis and A. Jukan. How to buy a network: Trading of resources in the physical layer. *IEEE Communications Magazine*, 44(12):94–102, 2006.

[Pla14] PlanetLab. PlanetLab - An Open Platform for Developing, Deploying, and Accessing Planetary-Scale Services. `http://www.planet-lab.org/`, 2014.

[PLP+13] C. Papagianni, A. Leivadeas, S. Papavassiliou, V. Maglaris, C. Cervello-Pastor, and A. Monje. On the optimal allocation of virtual resources in cloud computing networks. *Computers, IEEE Transactions on*, 62(6):1060–1071, 2013.

[PM04] M. Pióro and D. Medhi. *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Elsevier/Morgan Kaufmann, July 2004.

[PW06] L. Peterson and J. Wroclawski. Overview of the GENI architecture. *GENI Design Document*, pages 06–11, 2006.

[qua12] Quagga routing suite. `http://www.nongnu.org/quagga/`, August 2012.

[RAB10] MuntasirRaihan Rahman, Issam Aib, and Raouf Boutaba. Survivable virtual network embedding. In Mark Crovella, LauraMarie Feeney, Dan Rubenstein, and S.V. Raghavan, editors, *NETWORKING 2010*, volume 6091 of *Lecture Notes in Computer Science*, pages 40–52. Springer Berlin Heidelberg, 2010.

[RABdF12] Esteban Rodriguez, Gustavo Alkmim, Daniel M Batista, and Nelson LS da Fonseca. Green virtualized networks. In *Communications (ICC), 2012 IEEE International Conference on*, pages 1970–1975. IEEE, 2012.

[RAL03] Robert Ricci, Chris Alfeld, and Jay Lepreau. A solver for the network testbed mapping problem. *ACM SIGCOMM Computer Communication Review*, 33(2):65–81, 2003.

[RR06] E. Rosen and Y. Rekhter. BGP/MPLS IP Virtual Private Networks (VPNs). Technical report, RFC 4364, February 2006, 2006.

[RR10] A. Razzaq and M.S. Rathore. An approach towards resource efficient virtual network embedding. In *Evolving Internet (INTERNET), 2010 Second International Conference on*, pages 68–73, 2010.

[RSH11] A. Razzaq, P. Sjodin, and M. Hidell. Minimizing bottleneck nodes of a substrate in virtual network embedding. In *Network of the Future (NOF), 2011 International Conference on the*, pages 35–40, 2011.

[RVC+01] Eric Rosen, Arun Viswanathan, Ross Callon, et al. Multiprotocol label switching architecture. 2001.

[SAI14] SAIL - Scalable & Adaptive Internet soLutions. `http://www.sail-project.eu`, Jan. 2014.

[SH00] Thomas Stützle and Holger H Hoos. Max–min ant system. *Future generation computer systems*, 16(8):889–914, 2000.

[SHT12] D. Stezenbach, M. Hartmann, and K. Tutschku. Parameters and challenges for virtual network embedding in the future internet. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pages 1272–1278, 2012.

[SKD+13] T Sridhar, Lawrence Kreeger, Dinesh Dutt, Chris Wright, Mike Bursell, Mallik Mahalingam, Puneet Agarwal, and Kenneth Duda. Vxlan: A framework for overlaying virtualized layer 2 networks over layer 3 networks. 2013.

[SPG+13] Murari Sridharan, Mark Pearson, Ilango Ganga, Geng Lin, Patricia Thaler, Chait Tumuluri, Albert Greenberg, Kenneth Duda, and Yu-Shun Wang. Nvgre: Network virtualization using generic routing encapsulation. 2013.

[ST02] Sherlia Y Shi and Jonathan S Turner. Multicast routing and bandwidth dimensioning in overlay networks. *Selected Areas in Communications, IEEE Journal on*, 20(8):1444–1455, 2002.

[SYL+10] Gang Sun, Hongfang Yu, Lemin Li, Vishal Anand, Hao Di, and Xiujiao Gao. Efficient algorithms for survivable virtual network embedding. volume 7989, pages 79890K–79890K–7, 2010.

[SZC+12] Sen Su, Zhongbao Zhang, Xiang Cheng, Yiwen Wang, Yan Luo, and Jie Wang. Energy-aware virtual network embedding through consolidation. In *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, pages 127–132, 2012.

[TCTG12] Phuong Nga Tran, Leonardo Casucci, and Andreas Timm-Giel. Optimal mapping of virtual networks considering reactive reconfiguration. In *Cloud Networking (CLOUDNET), 2012 IEEE 1st International Conference on*, pages 35–40, 2012.

[tim03] ITU-T Recommendation G. 114. March 2003.

[TRI14] Trilogy: Re-Architecting the Internet. `http://www.trilogy-project.org`, Jan. 2014.

[TT05] J. Turner and D. Taylor. Diversifying the internet. In *Proc. IEEE GLOBECOM*, page 755–760, 2005.

[TTG13] P.N. Tran and A. Timm-Giel. Reconfiguration of virtual network mapping considering service disruption. In *Communications (ICC), 2013 IEEE International Conference on*, pages 3487–3492, 2013.

[TWEF03] J. Touch, Y. Wang, L. Eggert, and G. Finn. A virtual internet architecture. *ISI Technical Report ISI-TR-2003-570*, 2003.

[Vet95] Ronald J Vetter. Atm concepts, architectures, and protocols. *Communications of the ACM*, 38(2):30–ff, 1995.

[vin12] ViNE-Yard. `http://mosharaf.com/ViNE-Yard.tar.gz`, Sep. 2012.

[VLA14] IEEE802.1Q VLAN implementation for Linux. `http://www.candelatech.com/~greear/vlan.html`, Jan. 2014.

[vlc14] VLC media player - Open Source Multimedia Framework and Player. `http://www.videolan.org/vlc`, Jan. 2014.

[vmw14] VMware. `http://www.vmware.com/`, Jan. 2014.

[Wax88] B.M. Waxman. Routing of multipoint connections. *Selected Areas in Communications, IEEE Journal on*, 6(9):1617 –1622, dec 1988.

[WKB+08] Yi Wang, Eric Keller, Brian Biskeborn, Jacobus van der Merwe, and Jennifer Rexford. Virtual routers on the move: live router migration as a network-management primitive. *SIGCOMM Comput. Commun. Rev.*, 38(4):231–242, August 2008.

[WLS+02a] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. *ACM SIGOPS Operating Systems Review*, 36(SI):270, 2002.

[WLS+02b] Brian White, Jay Lepreau, Leigh Stoller, Robert Ricci, Shashi Guruprasad, Mac Newbold, Mike Hibler, Chad Barb, and Abhijeet Joglekar. An integrated experimental environment for distributed systems and networks. *ACM SIGOPS Operating Systems Review*, 36(SI):255–270, 2002.

[Wol00] L.A. Wolsey. Integer programming. *IIE Transactions*, 32:273–285, 2000.

[WSG02] A. Whitaker, M. Shaw, and S. D Gribble. Scale and performance in the denali isolation kernel. *ACM SIGOPS Operating Systems Review*, 36:195–209, 2002.

[WSYX11] Liu Wenzhi, Li Shuai, Xiang Yang, and Tang Xiongyan. Virtual network embedding based on shuffled frog leaping algorithm in tunie. *International Journal of Advancements in Computing Technology*, 3(11), 2011.

[WvdMR07] Y. Wang, J. van der Merwe, and J. Rexford. VROOM: virtual routers on the move. In *Proc. ACM SIGCOMM Workshop on Hot Topics in Networking*, 2007.

[XEN14] XEN hypervisor. `http://www.xenproject.org/`, Jan. 2014.

[xor14] XORP - Open Source Router. `http://www.xorp.org`, Jan. 2014.

[YLZ+12] Mao Yang, Yong Li, Lieguang Zeng, Depeng Jin, and Li Su. Karnaugh-map like online embedding algorithm of wireless virtualization. In *Wireless Personal Multimedia Communications (WPMC), 2012 15th International Symposium on*, pages 594–598, 2012.

[YQA+10] Hongfang Yu, Chunming Qiao, Vishal Anand, Xin Liu, Hao Di, and Gang Sun. Survivable virtual infrastructure mapping in a federated computing and networking system under single regional failures. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pages 1–6, 2010.

[YWK11] Wai-Leong Yeow, Cédric Westphal, and Ulas C Kozat. Designing and embedding reliable virtual infrastructures. *ACM SIGCOMM Computer Communication Review*, 41(2):57–64, 2011.

[YY11] Donggyu Yun and Yung Yi. Virtual network embedding in wireless multihop networks. In *Proceedings of the 6th International Conference on Future Internet Technologies*, CFI '11, pages 30–33, New York, NY, USA, 2011. ACM.

[YYRC08] Minlan Yu, Yung Yi, Jennifer Rexford, and Mung Chiang. Rethinking virtual network embedding: substrate support for path splitting and migration. *SIGCOMM Comput. Commun. Rev.*, 38(2):17–29, March 2008.

[ZA06] Yong Zhu and M. Ammar. Algorithms for assigning substrate network resources to virtual network components. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–12, 2006.

[ZCB96] E.W. Zegura, K.L. Calvert, and S. Bhattacharjee. How to model an internetwork. In *INFOCOM'96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE*, volume 2, pages 594–602. IEEE, 1996.

[ZCS+13] Zhongbao Zhang, Xiang Cheng, Sen Su, Yiwen Wang, Kai Shuang, and Yan Luo. A unified enhanced particle swarm optimization-based virtual network embedding algorithm. *International Journal of Communication Systems*, 26(8):1054–1073, 2013.

[ZLJ+10] Ye Zhou, Yong Li, Depeng Jin, Li Su, and Lieguang Zeng. A virtual network embedding scheme with two-stage node mapping based on physical resource migration. In *Communication Systems (ICCS), 2010 IEEE International Conference on*, pages 761–766, 2010.

[ZQ11] Shun-li Zhang and Xue-song Qiu. A novel virtual network mapping algorithm for cost minimizing. *Cyber Journals: Journal of Selected Areas in Telecommunications (JSAT)*, 2011.

[ZQGL11] Sheng Zhang, Zhuzhong Qian, Song Guo, and Sanglu Lu. Fell: A flexible virtual network embedding algorithm with guaranteed load balancing. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–5, 2011.

[ZXB10] F.E. Zaheer, Jin Xiao, and R. Boutaba. Multi-provider service negotiation and contracting in network virtualization. In *Network Operations and Management Symposium (NOMS), 2010 IEEE*, pages 471–478, 2010.

[ZZRR08] Yaping Zhu, Rui Zhang-Shen, Sampath Rangarajan, and Jennifer Rexford. Cabernet: connectivity architecture for better network services. In *Proceedings of the 2008 ACM CoNEXT Conference*, CoNEXT '08, pages 64:1–64:6, New York, NY, USA, 2008. ACM. ACM ID: 1544076.