STUDY, EVALUATION AND CONTRIBUTIONS TO NEW ALGORITHMS FOR THE EMBEDDING PROBLEM IN A NETWORK VIRTUALIZATION ENVIRONMENT

JUAN FELIPE BOTERO

Advisor: Xavier Hesselbach-Serra
Telematics Engineering Department
Universitat Politècnica de Catalunya

April 2013 –

“One day you will do things for me that you hate.
That is what it means to be family“

— Jonathan Safran Foer, *Everything is Illuminated*

Dedicated to my family and my future wife
whose love is immeasurable

ABSTRACT

Network virtualization is recognized as an enabling technology for the future Internet. It aims to overcome the resistance of the current Internet to architectural change and to enable a new business model decoupling the network services from the underlying infrastructure. The problem of embedding virtual networks in a substrate network is the main resource allocation challenge in network virtualization and is usually referred to as the Virtual Network Embedding (VNE) problem. VNE deals with the allocation of virtual resources both in nodes and links. Therefore, it can be divided into two sub-problems: Virtual Node Mapping where virtual nodes have to be allocated in physical nodes and Virtual Link Mapping where virtual links connecting these virtual nodes have to be mapped to paths connecting the corresponding nodes in the substrate network.

Application of network virtualization relies on algorithms that can instantiate virtualized networks on a substrate infrastructure, optimizing the layout for service-relevant metrics. This class of algorithms is commonly known as VNE algorithms.

This thesis proposes a set of contributions to solve the research challenges of the VNE that have not been tackled by the research community. To do that, it performs a deep and comprehensive survey of virtual network embedding.

The first research challenge identified is the lack of proposals to solve the virtual link mapping stage of VNE using single path in the physical network. As this problem is NP-hard, existing proposals solve it using well known shortest path algorithms that limit the mapping considering just one constraint. This thesis proposes the use of a mathematical multi-constraint routing framework called *paths algebra* to solve the virtual link mapping stage. Besides, the thesis introduces a new demand caused by virtual link demands into physical nodes acting as intermediate (hidden) hops in a path of the physical network.

Most of the current VNE approaches are centralized. They suffer of scalability issues and provide a single point of failure. In addition, they are not able to embed virtual network requests arriving at the same time in parallel. To solve this challenge, this thesis proposes a *distributed*, *parallel* and *universal* virtual network embedding framework. The proposed framework can be used to run any existing embedding algorithm in a distributed way. Thereby, computational load for embedding multiple virtual networks is spread across the substrate network.

Energy efficiency is one of the main challenges in future networking environments. Network virtualization can be used to tackle this problem by sharing hardware, instead of requiring dedicated hardware for each instance. Until now, VNE algorithms do not consider energy as a factor for the mapping. This thesis introduces the energy aware VNE where the main objective is to switch off as many network nodes and interfaces as possible by allocating the virtual demands to a consolidated subset of active physical networking equipment.

To evaluate and validate the aforementioned VNE proposals, this thesis helped in the development of a software framework called ALgorithms for Embedding VIrtual Networks (ALEVIN). ALEVIN allows to easily implement, evaluate and compare different VNE algorithms according to a set of metrics, which evaluate the algorithms and compute their results on a given scenario for arbitrary parameters.

PUBLICATIONS

Some ideas and figures present in this monograph have appeared previously in [BH08, BH09, MMM+10, GFB+10, BHD+11, BH11, FBD+11, DSS+11, DSB+11, BHD+12, BBF+13, BHFDM13, BH13, BMHSA13, FBB+13]

ACKNOWLEDGMENTS

The research work accomplished throughout this thesis was carried out between the fall of 2008 and the fall of 2012 at the Department of Telematics Engineering (ENTEL), Technical University of Catalonia (UPC), Barcelona, Spain. This work would not have been possible had it not been for the help and continuous support of several people.

Initially, I would like to express my sincere gratitude to my supervisor, Dr. Xavier Hesselbach, for the continuous support, guidance, commitment, patience, understanding and valuable feedback I received during these years. Thanks to him, I've been involved in many interesting and challenging research tasks that have broadened my scientific scope.

My special thanks go to all of my colleagues at UPC. Guys, thank you for your friendship, the fruitful informal talks and your contribution to create a pleasant working environment. Particularly, I would like to mention: Carolina Tripp, Ernesto García and August Betzler.

I would not like to miss the opportunity to thank Prof. Hermann de Meer for enabling my four months internship at the Computer Networks and Communications Chair of the University of Passau in 2011. Inside the Chair, I would like to thank Andreas Fischer and Michael Till Beck for the excellent research results obtained from the collaboration between UPC and the University of Passau.

I would like to thank Prof. José Roberto Amazonas from the University of São Paulo for the valuable collaboration that gave rise to excellent research results.

I want also to thank my family. They are the builders of what I am. Thanks to my father and mother for their continuous example and for always being there despite the distance. Thanks to my sister, her tenacity and resolution have always been a model.

I would like to thank my best friend; my brother. Thanks for the advices, support and tolerance, but especially, thank you for your friendship. It would not have been that fun without you.

I want to thank my future wife, Yani, for always being with me either from the distance or in person. Thank you for your love, support, understanding, endless discussions and helpful advices. With you, everything is easier.

I would also like to thank all the people that have contributed to make my stay in this country such a nice and enriching experience.

Finally, thanks to the Agència de Gestió d'Ajuts Universitaris i de Recerca (AGAUR) for the scolarship that supported the development of this thesis.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

VNE	Virtual Network Embedding
ALEVIN	ALgorithms for Embedding VIrtual Networks
ISP	Internet Service Provider
NV	Network Virtualization
VN	Virtual Network
VPN	Virtual Private Network
SN	Substrate Network
QoS	Quality of Service
VNREAL	Virtual Network Resource Embedding ALgorithms
OS	Operating System

DPVNE	Distributed, Parallel and Universal Virtual Network Embedding framework
IaaS	Infrastructure as a Service
InP	Infrastructure Provider
SP	Service Provider
VNP	Virtual Network Provider
VNO	Virtual Network Operator
SDN	Software Defined Networks
G-Lab	German Lab
VINI	VIrtual Network Infrastructure
VNR	Virtual Network Request
VNoM	Virtual Node Mapping
VLiM	Virtual Link Mapping
CPU	Central Processing Unit
VNRe	Virtual Network Reconfiguration
VoIP	Voice over IP
WiNE	Window-based virtual Network Embedding
DViNE	Deterministic Virtual Network Embedding
RViNE	Randomized Virtual Network Embedding
MIP	Mixed Integer Programming
BFS	Breadth-First Search
RW-BFS	Random Walk-BFS
LP	Linear Programming
ILP	Integer Linear Programming
SID	Subgraph Isomorphism Detection
PSO	Particle Swarm Optimization
GT-ITM	Georgia Tech Internetwork Topology Models
EA-VNE	Energy-Aware Virtual Network Embedding
EA-VNE-LB	Energy-Aware Virtual Network Embedding with Load Balancing

ICT	Information and Communications Technology
WMN	Wireless Mesh Network
VON	Virtual Optical Network
GRASP	Greedy Randomized Adaptive Search Procedures
BW	Bandwidth
AS	Autonomous System
THRU	Throughput
PDT	Packet Delay Transfer
PDV	Packet Delay Variation
PLR	Packet Loss Rate
LADN	Loop Avoidance by the DestiNation
FIFO	First In First Out
C/R	Cost/Revenue
MCF	Most Consuming First
LCF	Least Consuming First
ASID	Advanced Subgraph Isomorphism Detection
NO	Network Operator
DAVNM	Distributed Virtual Network Mapping
ALR	Adaptive Link Rate
EA-RH	Energy-Aware Reconfiguration Heuristic
SPTs	Shortest Path Trees
ESIR	Energy Saving IP Routing
OCA	Optimal Cost Approach
NS	Node Suitability
PS	Paths Suitability
SEP	Shortest Energy Path
GUI	Graphical User Interface
GPL	GNU General Public License
LGPL	GNU Lesser General Public License

VDE	Virtual Datacenter Embedding
IPv6	Internet Protocol version 6
DiffServ	Differentiated Services
IntServ	Integrated Services
IPsec	Internet Protocol Security
GRE	Generic Routing Encapsulation
MPLS	Multiprotocol Label Switching
IP	Internet Protocol
MFP	Multicommodity Flow Problem
IT	Information Technology
NRENs	National Research & Education Networks
GENUS	GÉaNt virtUalization Service
EU	European Union
SAIL	Scalable and Adaptive Internet Solutions
ICN	Information Centric Networking
GEYSERS	Generalized Architecture for Dynamic Infrastructure Services
US	United States
GENI	Global Environment for Network Innovation
NSF	National Science Foundation
ENTEL	Department of Telematics Engineering
UPC	Technical University of Catalonia
TARIFA	The Atomic Redesign of the Internet Future Architecture
RINA	Recursive InterNetwork Architecture
CCN	Content Centric Networking
TDM	Time-Division Multiplexing
FDM	Frequency-Division Multiplexing
EPI	Energy Proportionality Index
AGAUR	Agència de Gestió d'Ajuts Universitaris i de Recerca

Part I

PROBLEM STATEMENT AND STATE OF THE ART

1 INTRODUCTION

"It's easy to know what you want to say, but not to say it"
Mario Vargas Llosa

The Internet has been very successful providing us the access to exchange information in the modern world. Over the course of past three decades, the Internet architecture has shown its robustness by supporting a great bunch of applications and a large variety of network technologies over which it currently runs. However, nowadays, there is a big impediment to its further growth. Due to its multi-provider nature, adopting a new architecture or modification of the existing one requires consensus among a Internet Service Provider (ISP) and its competition. Therefore, alterations to the current Internet architecture have become restricted to simple incremental updates and deployment of new network technologies have become increasingly difficult [1, 2] – e.g. Internet Protocol version 6 (IPv6) or Differentiated Services (DiffServ) –.

To overcome this situation, Network Virtualization (NV) [3, 4] has been proposed as a main attribute of the future Internet. By allowing multiple heterogeneous network architectures to cohabit on a shared physical substrate, NV provides flexibility, promotes diversity, promises security and increased manageability. NV can eradicate the current Internet forces that have restricted changes to incremental updates, and consequently stimulate innovation.

Network virtualization is outlined as one of the preferred technologies to model the architecture of future Internet, it will allow new services and protocols to be tested and deployed much more rapid than today. End to end customized applications will be offered by instantiating a proper Virtual Network (VN) that fulfills the demanded Quality of Service (QoS). A bunch of opportunities to develop and test new protocols and services will appear with such an architecture. However, the introduction of virtual networks to the current environment is not a straight-forward task; there are many research challenges to deal with before NV is capable of running over the current Internet architecture.

Network virtualization is outlined as one of the preferred technologies to model the architecture of future Internet

The problem of embedding virtual networks in a substrate network is the main resource allocation challenge in network virtualization and is usually referred to as the VIRTUAL NETWORK EMBEDDING problem. Through dynamic mapping of virtual resources onto physical hardware, the benefit gained from existing hardware can be maximized. Optimal dynamic resource allocation, leading to the self-configuration and organization of future networks, will be necessary to provide customized end-to-end guaranteed services to end users. This optimality can be computed with regard to different objectives, ranging from QoS, economical profit, or survivability over energy-efficiency to security of the networks.

Virtual Network Embedding (VNE) consists on the mapping of the virtual network resource demands onto the physical resources in an optimal way

The aim of this thesis is to study and evaluate current contributions and propose new strategies to solve the VNE problem, improving current research and exploring new application fields.

1.1 SCIENTIFIC CONTRIBUTION

Figure 1.1 gives an overview of the contributions generated out of this thesis. There are four main covered research topics inside VNE: The *modeling, categorization and classification* of the existing approaches intending to solve VNE, the implementation of a *software framework* that allows to develop, compare and analyze VNE algorithms, a set of new *VNE algorithms* that overcome some of the shortcomings of existing proposals and the introduction of an entire new field: *Energy-efficient* VNE. Figure 1.1 comprises annotations of the form $[x]^y$, denoting that scientific publication $[x]$ contributes to Chapter y. The main publications that give rise to the most of the content of this monograph are typed in boldface whereas the remaining publications are early contributions that have contributed to the maturity of this content.

The first part of this monograph is devoted to the presentation of the main concepts needed to understand both the problem this thesis faces and the set of existing proposals that have been made until the current date. This is a contribution of this thesis to state of the art of NV, particularly it surveys the VNE problem and its variants. The main result of this contribution is published in [FBB+13] and was made in close collaboration with the Chair of Computer Networks and Communications of the University of Passau. This publication was matured using elements of previous publications [BH09, GFB+10, MMM+10].

The second part of this thesis details the contributions to the resource allocation problem in network virtualization: In particular, the author of this thesis has collaborated in the development of a software framework called ALEVIN [FBD+11, DSS+11] within the Virtual Network Resource Embedding ALgorithms (VNREAL) project [DSB+11]. ALEVIN allows to easily implement, evaluate and compare different VNE algorithms according to a set of metrics, which evaluate the algorithms and compute their results on a given scenario for arbitrary parameters. Several subsequent contributions of this thesis used ALEVIN to implement new VNE algorithms and to evaluate them against existing proposals (also implemented in ALEVIN).

This thesis also proposes new algorithms and constraints for the VNE problem. In particular, *hidden hops* [BHFDM13] are introduced.

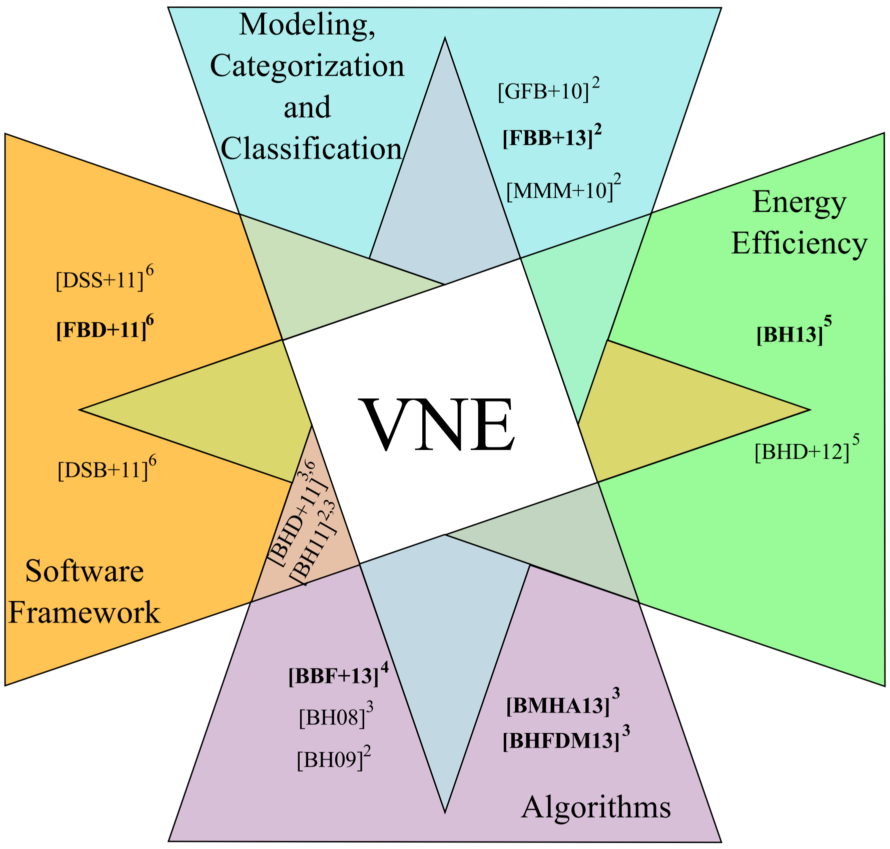


Figure 1.1: Contribution of this work illustrated as a classification of the research studies conducted by the author

They make reference to the intermediate nodes of a directed path in the Substrate Network (SN) that is mapping a specific virtual link. Hidden hops entail a resource demand because they have to perform packet forwarding of the traffic that will pass through this virtual link. Existing VNE approaches are not able to provide optimal solutions with non-linear constraints (e.g. packet loss) and are limited to mono-constraint shortest path algorithms to solve the single-path virtual link mapping. For this reason, this thesis introduces a VNE strategy based on the mathematical multi-constraint framework called paths algebra [5] that provides the flexibility to introduce an unlimited number of linear and non-linear constraints and metrics to the problem while finding all the eligible paths in the physical network to perform the virtual link mapping resulting in better and more flexible embeddings [BMHSA13].

This thesis also introduced a Distributed, Parallel and Universal Virtual Network Embedding framework (DPVNE) that overcomes the shortcomings of the existing distributed VNE algorithms. DPVNE can be used as a framework to run any existing embedding algorithm in a distributed way. Thereby, computational load for embedding multiple

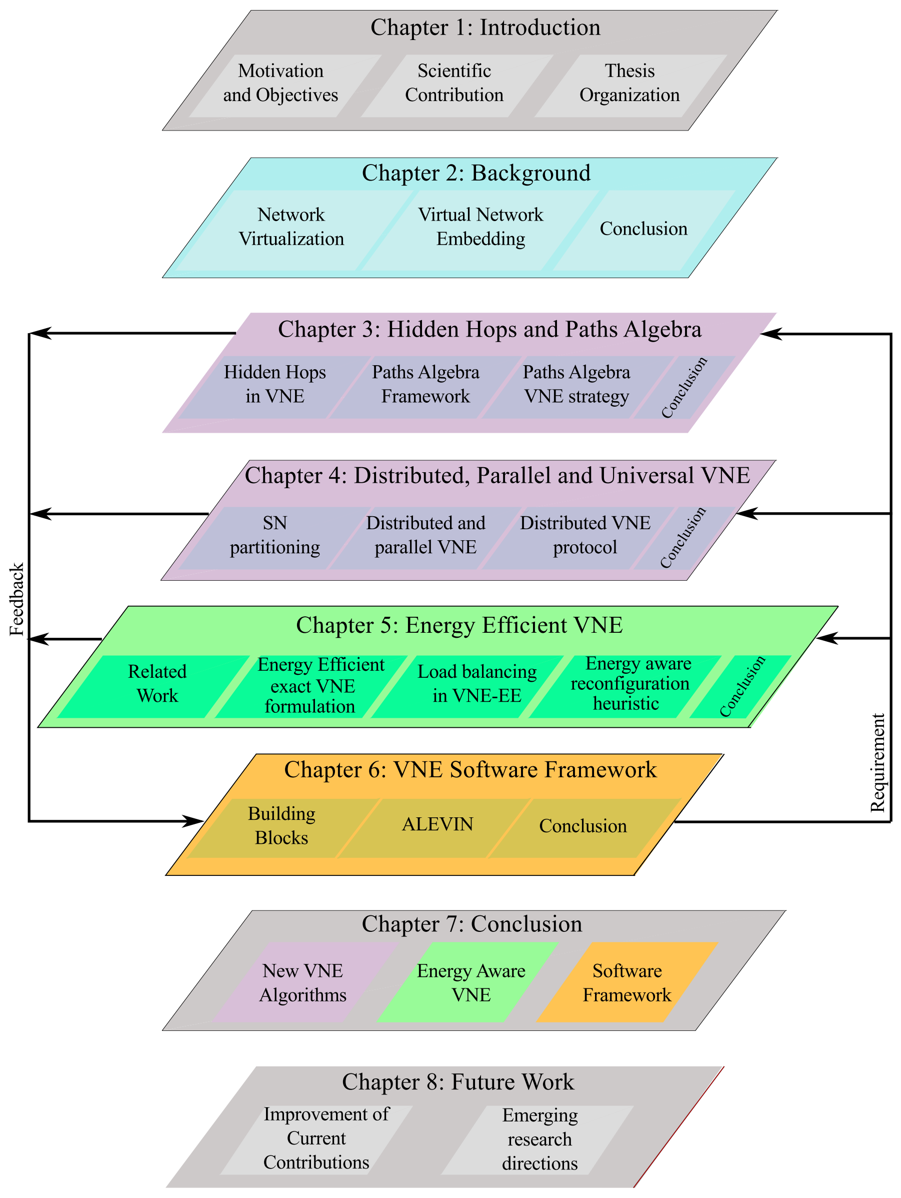


Figure 1.2: Thesis organization

virtual networks is spread across the SN. This reduces work load of individual nodes and enables the network to embed multiple virtual networks in parallel [BBF+13].

Finally, this thesis contributes to introduce the energy-aware VNE. It presents the first foray in the energy topic when mapping VN demands to the SN. The proposed strategy takes advantage of network virtualization to enable a smart energy-aware network provisioning. The objective is to switch off as many network nodes and interfaces as possible by allocating the virtual demands to a consolidated subset of active physical networking equipment. This important contribution is supported by two publications [BHD+12, BH13].

1.2 ORGANIZATION

The organization of this thesis is shown in Figure 1.2. There, the individual chapters and their interdependences are shown. The first part of the thesis devoted to the VNE problem statement and to the presentation of the state of the art is confined to the Chapter 2. The second part presenting a set contributions to the VNE problem is compiled in Chapters 3, 4, 5 and 6: On the one hand, the results presented in Chapter 3, Chapter 4 and Chapter 5 come from the implementation of the software framework presented in Chapter 6 (requirement); on the other hand, those chapters provided feedback for improvement of this software framework. The remainder of this monograph is organized as follows.

In Chapter 2, a deep background in the main concepts of this thesis is presented. In particular, network virtualization and its main resource challenge, virtual network embedding, are widely surveyed. The survey of VNE is a result of a collaboration between Juan Felipe Botero and Xavier Hesselbach from the Telematics Department of the Universitat Politècnica de Catalunya and Andreas Fischer, Michael Till Beck and Hermann de Meer from the Chair of Computer Networks and Communications of the University of Passau.

Chapter 3 presents two contributions to the virtual link mapping stage of centralized VNE: The definition of new hidden hop demands and a new algorithm to solve the virtual link mapping stage of VNE based on the paths algebra framework. The analysis of VNE with Hidden hops was delved thanks to the collaboration of Juan Felipe Botero and Xavier Hesselbach from the Telematics Department of the Universitat Politècnica de Catalunya with Andreas Fischer and Hermann de Meer from the Chair of Computer Networks and Communications of the University of Passau. The VNE proposal using the paths algebra framework was developed in strong collaboration with José Roberto Amazonas and Miguel Molina from the University of São Paulo.

Chapter 4 details DPVNE: A Distributed, parallel and universal framework to perform Virtual Network Embeddings overcoming the limitations of previous distributed VNE algorithms. DPVNE was developed in strong collaboration with Michael Till Beck, Andreas Fischer and Hermann de Meer from the Chair of Computer Networks and Communications of the University of Passau.

Chapter 5 describes the energy-aware VNE problem and proposes exact and heuristic approaches to optimally solve it. Part of the outcomes of this investigation were performed in collaboration with the Institute of computer science of the University of Würzburg and the

Chair of Computer Networks and Communications of the University of Passau.

Chapter 6 presents the software framework ALEVIN, developed within the VNREAL project in collaboration with the Institute of computer science of the University of Würzburg and the Chair of Computer Networks and Communications of the University of Passau.

Finally, Chapter 7 presents the conclusions and main results of this thesis and Chapter 8 depicts the possible improvements of the presented contributions and the emerging research branches that can be subject of future work.

2 BACKGROUND

"Wisdom comes to us when it can no longer do any good"
Gabriel García Marquez

This chapter is devoted to present the basic knowledge required to develop this thesis. The concepts of Network Virtualization and its well recognized embedding problem are deeply explored here.

In particular, the main contribution presented in this chapter is a comprehensive survey of Virtual Network Embedding (Section 2.2), including a novel VNE classification scheme, a formal and generic mathematical VNE formulation, the different parameters that can be considered in the embedding for substrate as well as for virtual networks, the different metrics used to evaluate the performance of the VNE solutions, the possible variants to solve the coordination between node and link mapping, the main optimization objectives which relate to VNE, the set of possible algorithm strategies to solve VNE, a detailed classification of the existing approaches based on the employed strategy, the different available VNE related software tools and the emerging VNE research directions. This contribution was published in [FBB+13] as a collaboration of Juan Felipe Botero and Xavier Hesselbach from the Telematics Department of the Universitat Politècnica de Catalunya with Andreas Fischer, Michael Till Beck and Hermann De Meer from the Chair of Computer Networks and Communications of the University of Passau.

2.1 NETWORK VIRTUALIZATION

Internet is, nowadays, the main communication medium and has changed the way humans interact. Internet success is due to its capacity of accommodate new services as they have been required. However, this trend might come to its end.

Current Internet Architecture is not prepared to overcome the emerging requirements coming from new services

Internet architecture was developed 35 years ago and, incrementally, it has changed until the current one. However, this architecture is not prepared to overcome the emerging challenges coming from new services being requested by users. The lack of cooperation among stakeholders does not allow radical changes to it. Nowadays, it is difficult to test and implement new protocols and innovations, proposed by the research community, that could help Internet to get over the mentioned challenges (e.g. IPv6, Integrated Services (IntServ) or DiffServ); this trend is called ossification [2]. One solution to this problem would be to reach an agreement among ISPs on any architectural change. However, this is almost an impossible mission due to two reasons: In first place, the huge number of ISPs makes difficult to arrange an agreement among all of them; in second place, such an agreement would remove the competitive advantage from architectural innovation.

Following the point of view stated by Peterson et al. in [2], there is a debate between architectural "purists" and "pluralists". While *purists* plead for incremental architectural modifications to meet legitimate needs that the architecture itself can not afford, the *pluralists* argue that these architectural *barnacles* may serve as short-term valid solutions, but in the long-term they affect the flexibility, reliability and manageability of the Internet. They (pluralists), instead, propose a new architecture to overcome the ossification impasse.

The central technology of this architecture is Network Virtualization, seen as the alternative to face up ossification [1]. A network virtualization environment is an accumulation of heterogeneous network technologies cohabiting on a shared physical substrate, also known as substrate network. On top of the SN, several virtual networks run offering end-to-end customized QoS to the end users.

A network virtualization environment is an accumulation of heterogeneous network technologies cohabiting on a shared physical substrate

The concept of virtual networks is realized in several different implementations. Examples are the Virtual Private Network (VPN), Peer-to-Peer networks and networks virtualized with System Virtualization.

In VPNs, a dedicated (usually encrypted) Virtual Link is set up between two routers, possibly spanning several physical links. VPNs can be set up on different layers in the network stack, with the IPSec implementation on the network layer being one of the more popular ones [6]. VPNs provide just point to point virtual links and do not consider the virtualization of nodes.

In Peer-to-peer (P2P) overlay networks [7] an entire network structure is created on top of an existing network. P2P nodes correspond to Virtual Nodes, setting up Virtual Links between them, that do not reflect the underlying network infrastructure, but rather the logical grouping of nodes within the P2P network. However, P2P overlay networks are created with the end hosts and do not imply the virtualization of the core network infrastructure.

Network virtualization can be answered to solve the shortcomings of today's Internet due to its ability of combining the main Internet achievements [8]: *P2P-Overlays* that achieve high performance and provide better reliability than other network architectures; the *diversity of networked resources* with respect to connectivity and quality; and *Operating System (OS) virtualization* that provides the opportunity to consolidate safely multiple networks in one physical platform.

To realize NV, physical network nodes and links are virtualized creating isolated VNs that coexist on top of a shared physical substrate.

Virtualization of nodes can be performed using several techniques, mainly "System Virtualization" [9] that virtualizes the resources of the physical node and creates virtual machines hosting an isolated guest OS with the same functionalities of a normal OS. In networks virtualized with System Virtualization, core routers host several OSs with routing functionality. The routers are interconnected with Virtual Links that are virtualized using encapsulation techniques (e.g. Internet Protocol Security (IPsec), Generic Routing Encapsulation (GRE), Multiprotocol Label Switching (MPLS), etc.) and may span several physical links. This approach allows for easy management and even mobility of Virtual Routers [10]. Moreover, it becomes possible to deploy different network protocols alongside each other with the concept of system virtualization providing a clear compartmentalization. Besides, network virtualization provides reusable topology and an energy-efficient scheme.

The following definitions of virtual nodes, virtual links, and virtual networks are used throughout this monograph [9]:

A VIRTUAL NODE/ROUTER is a software component with routing functionality, for example an operating system encapsulated in a virtual machine that is managed by a virtual machine monitor.

A VIRTUAL LINK is a logical interconnection of two virtual routers, appearing to them as a direct physical link with dynamically changing properties.

A VIRTUAL NETWORK consists of virtual routers and virtual links that form a directed connected graph using the infrastructure of the underlying physical network. Several independent virtual networks can exist in parallel on top of a physical network.

To be implemented, Network Virtualization needs a sustainable economic model. Infrastructure as a Service (IaaS) [11] is a business model where the hardware (server, storage and network), and associated software (operating systems virtualization technology, file system), are delivered as a service. Based on IaaS, Feamster et al. propose a model distinguishing between the two functions provided by current ISPs: Network infrastructure and network services [12]. This business model clearly differentiates infrastructure and service roles, as follows:

To be implemented, Network Virtualization needs a sustainable economic model distinguishing between the two functions provided by current ISPs: network infrastructure and network services

- Infrastructure Provider (InP): Owner of network infrastructure. It offers the owned infrastructure as a service, that is, it “leases” a slice of its infrastructure to the service provider. This slice is provided as a virtual network.

- Service Provider (SP): It offers network services to end users. It does not own network infrastructure; its infrastructure is a set of virtual networks leased from one (or several) InPs.

With such an architecture, the innovations (network protocols and services) proposed by the research community can be handled. A SP

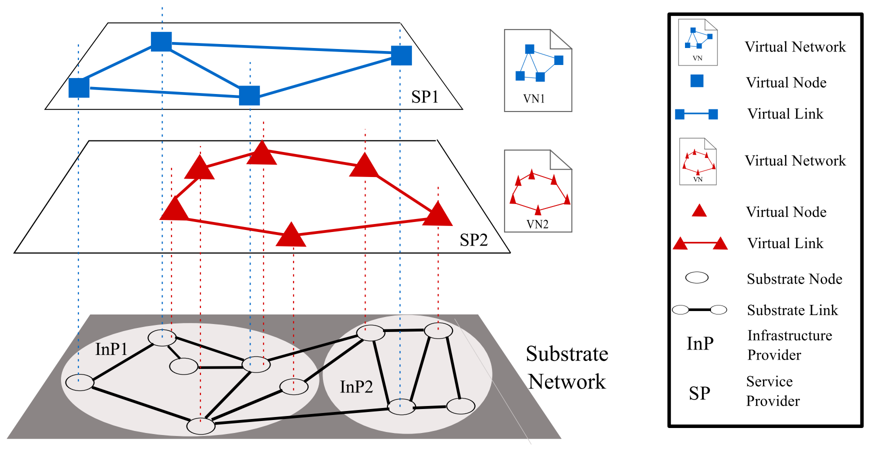


Figure 2.1: Network virtualization environment

offers isolated virtual networks to realize tests (e.g. new protocols) in it. Instead of just assisting to test purposes, an architecture based on NV can help also to provide end to end guaranteed services to end users; the service provider would behave as a virtual operator building its virtual network by borrowing the physical network resources of several infrastructure providers (independently of their location).

Figure 2.1 shows two virtual networks managed by different SPs that run on top of a SN composed of physical network infrastructure owned by two different InPs.

The 4WARD project [13, 14] proposed a further decoupling of business roles in network virtualization by means of the separation of SP's management and business roles [15]. Three main players are introduced (see Figure 2.2): The *Virtual Network Provider (VNP)* which assembles virtual resources from one or more InPs, the *Virtual Network Operator (VNO)* which installs, manages and operates the VN according to the needs of the SP, and the SP which is free of management and concentrates on business by using the VNs to offer customized services.

One of the advantages of an architecture based on virtual networks is its suitability to be programmable. A programmable network [16] is a network with separation of communication hardware (i.e., switching fabrics, routing engines) from control software. The introduction of new services into existing networks is usually a manual, time consuming and costly process. The goal of programmable networking is to simplify the deployment of new network services, leading to networks that explicitly support the process of service creation and deployment. Typically, service providers do not have access to switch/router control environments (e.g. Cisco IOS operating system), algorithms (e.g. routing protocols) or states (e.g. routing tables, flow states) consuming and costly process. However, this tendency has been changing in the last years; programmable routers

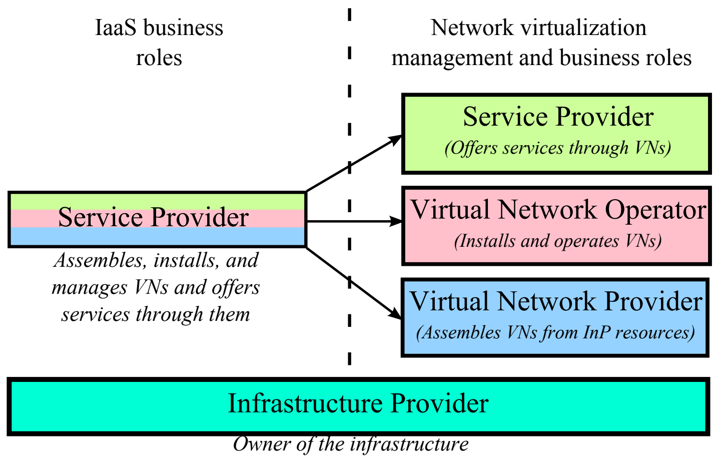

IaaS business roles
Network virtualization management and business roles
Service Provider
(Offers services through VNs)
Service Provider
Assembles, installs, and manages VNs and offers services through them
Virtual Network Operator
(Installs and operates VNs)
Virtual Network Provider
(Assembles VNs from InP resources)
Infrastructure Provider
Owner of the infrastructure

Figure 2.2: Future Internet business model

have started to emerge [17, 18], and more recently the programmability of the whole network is becoming possible by the introduction of Software Defined Networks (SDN) [19].

Virtual networks using programmable routers are an excellent tool to perform live experimentation (real testbeds) of new network protocols and services fulfilling the growing new requirements and demands.

Introduced as a means to evaluate new protocols and services [2], network virtualization has already been actively used in research testbeds like German Lab (G-Lab) [20] or VIrtual Network Infrastructure (VINI) [21], applied in distributed cloud computing environments [22] and is, by now, seen as a tool to overcome the resistance of the current Internet to fundamental changes. Indeed, even today network virtualization approaches are applied in the telecommunication market. An example for this is OpenFlow [23], which experiences strong support by the industry within the Open Networking Foundation [24]. However, NV imposes several challenges to solve for its implementation, the following are the more important ones [3].

NV imposes several challenges to solve for its implementation, the are mainly: Instantiation, Operation, Management and Interaction between players

1. INSTANTIATION: It groups the set of functionalities needed to set-up a virtual network. It can be divided into the following aspects:
    - *Interfacing*: Standardization of interfaces between different actors in the architecture must be handled.
    - *Signaling and Bootstrapping*: Guarantee the connectivity between both service and infrastructure providers (signaling must be handled to perform this task). Bootstrapping capabilities must be managed to allow service providers to customize virtual nodes and links.
    - *Admission Control and Usage Policing*: Infrastructure providers must make sure that they have enough available resources

to accept virtual network requests on the part of service providers.

- ***Virtual Network Embedding or Mapping***: As stated in the definition of virtual network, a virtual link may span several physical links, so it is possible to realize multiple mappings of one Virtual Network Request (VNR). It is very important to know how to embed VN requests made by service providers because the physical resources will be optimized depending on the way this embedding is carried out.

2. Operation: It refers to the capabilities needed by the network resources to operate virtual ones. The following capabilities have to be developed.
    - *Virtual Routers*: Guarantee the virtualization capability in the router equipment.
    - *Virtual Links*: The ability of creating tunnels (already available in VPNs) must be supported in VNs.
    - *Naming and Addressing*: These two features have to be managed to allow correct operation of virtual networks and to reach universal connectivity in a global network virtualization environment.
    - *Resource Scheduling*: Guarantee of determined virtual attributes must be provided by means of the implementation of scheduling techniques.
    - *Topology Discovery*: Infrastructure providers must be able to discover its available network resources and their status.

3. Management: It refers to the activities, methods, procedures and tools needed to operate, administrate, maintain and provision virtual networks. To ensure flexibility and manageability, the programmability of network elements is an indispensable requirement.
    - *VN Configuration and Monitoring*: This task must be performed by service providers.
    - *VN Management Frameworks*: These frameworks must be created to aggregate information of the set of infrastructure providers that are leasing their resources to a single virtual network.
    - *Mobility Management*: Mobility of devices must be allowed.
    - *Failure Handling*: Prevention, detection, propagation and isolation of failures should be handled, because a single failure in one physical link or router could result in the unworkability of several virtual networks.

4. INTERACTION BETWEEN PLAYERS: It refers to the coordination needed between the InP and the SP. It has to be maintained in two fronts: Heterogeneity of the underlying network technologies and each end-to-end VN. Enabling virtualization in each of heterogeneous substrate technologies requires specific solutions for provisioning, operation, and maintenance. Interactions between such contrasting underlying infrastructures, while providing a generic and transparent management interface for SPs to easily compose and manage VNs, remains an enormous task.

This thesis is devoted to the study of the virtual network embedding (or mapping) problem. The next section presents a deep investigation that defines the VNE problem and a categorization that allows to classify the different VNE variants and to survey the existing approaches aiming to solve the problem.

2.2 VIRTUAL NETWORK EMBEDDING

As already exposed in the previous section, the problem of embedding virtual networks in a substrate network is the main resource allocation challenge in network virtualization [25] and is usually referred to as the *Virtual Network Embedding* problem. Through dynamic mapping of virtual resources onto physical hardware, the benefit gained from existing hardware can be maximized. Optimal dynamic resource allocation, leading to the self-configuration and organization of future networks, will be necessary to provide customized end-to-end guaranteed services to end users. This optimality can be computed with regard to different objectives, ranging from QoS, economical profit, or survivability over energy efficiency to security of the networks. Figure 2.3 illustrates how network virtualization makes use of embedding algorithms in order to allocate virtual resources on a physical infrastructure in an optimal way.

Virtual Network Embedding is the main resource allocation challenge in Network Virtualization

Virtual network embedding deals with the allocation of virtual resources both in nodes and links. Therefore, it can be divided into two sub-problems: *Virtual Node Mapping (VNoM)* where virtual nodes have to be allocated in physical nodes and *Virtual Link Mapping (VLiM)* where the virtual links connecting these virtual nodes have to be mapped to paths connecting the corresponding nodes in the substrate network.

A VNE general formulation, main parameters-metrics, coordination variants, optimization objectives, algorithm strategies, classification, software tools and emerging research directions are the main contributions presented in this chapter

A recent survey by Belbekkouche et al. [26] provides a description of the main approaches proposed for resource discovery and allocation (including VNE) in network virtualization. The contributions presented in this chapter goes beyond just describing the main VNE approaches: A novel VNE classification scheme is presented. The VNE problem is considered in all its variants and current proposals coming from the research community are classified. In particular: A formal and generic mathematical *formulation* of the VNE problem, the differ-

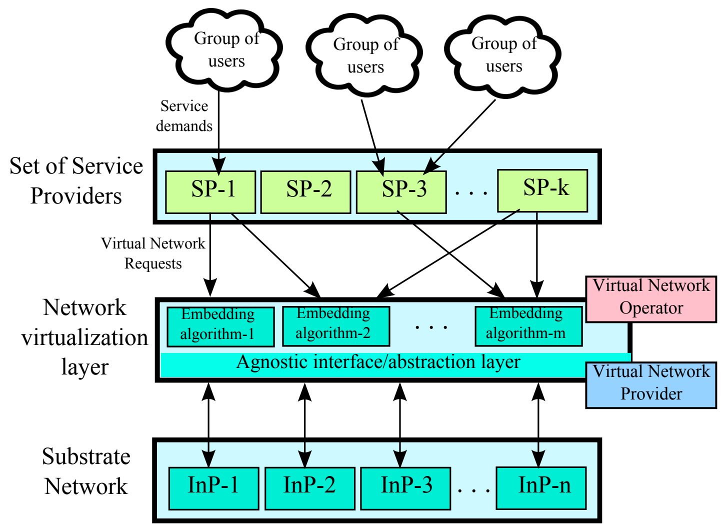


Figure 2.3: Resource allocation in future Internet

ent *parameters* that can be considered in the embedding for substrate as well as for virtual networks, the main *embedding objectives* which relate to VNE, the possible alternatives to *decompose* the VNE problem and to solve the *coordination* between VNoM and VLiM, the set of possible *optimization strategies* to solve VNE, the different *metrics* used to evaluate the performance of the solutions, a number of VNE *simulators*, a detailed *classification* of the existing approaches based on the employed strategy, and the emerging *research directions* currently considered by the research community are presented.

The remainder of this section is organized as follows: Section 2.2.1 formulates the VNE problem and presents its different categories. Different variants, strategies, parameters and metrics used by VNE algorithms as well as the main software tools are presented in Section 2.2.2. A classification of the existing VNE approaches is presented in Section 2.2.3. Finally, Section 2.2.4 highlights emerging research directions in the VNE field.

2.2.1 *Formulation and categories*

Several papers have provided specific formulations for VNE. In this section, a general VNE formulation and a novel categorization that allows to easily identify each variant of the problem, are introduced.

2.2.1.1 *Problem formulation*

The application of virtualization mechanisms to network resources leads to the question of how the virtualized resources should be realized by the substrate resources. It is important to note that substrate resources can be virtual themselves. This is commonly referred to as

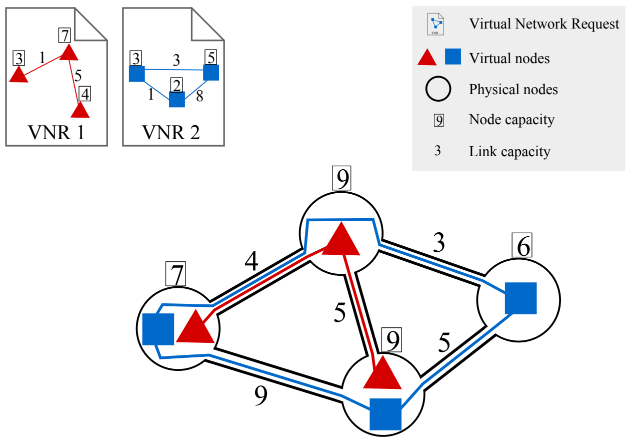


Figure 2.4: Two virtual networks mapped onto one substrate network

nested virtualization. In that case, only the lowest layer has to consist of physical resources.

The hardware abstraction provided by the virtualization solution provides a common denominator, allowing any substrate resource to host virtual resources of the same type. Typically, a substrate resource is partitioned to host several virtual resources. For example, a virtual node can, in principle, be hosted by any available substrate node. Moreover, a single substrate node can host several virtual nodes. Thus, the mapping of virtual nodes to substrate nodes describes a $n : 1$ relationship (a strict partition of substrate resources).

In some cases, substrate resources can also be combined to create new virtual resources. This is the case for a virtual link which spans several links (i.e. a path) in the substrate network. In this case, a virtual link between two virtual nodes v and w is mapped to a path in the substrate network that connects the substrate hosts of v and w. Each substrate link may then be part of several virtual links. As such, the mapping of virtual links to substrate paths describes a $n : m$ relationship (both, a partition and a combination of substrate resources).

Figure 2.4 depicts a scenario, where two virtual networks with three nodes each are hosted on one substrate network with four nodes. It can be seen that substrate nodes can host several virtual nodes (up to two in this example). Likewise, substrate links can host more than one virtual link. Moreover, one of the virtual links spans two substrate links, thus representing a virtual resource combined from several substrate resources.

Typically, there are some restrictions to be considered during the mapping. Most obviously, the candidate substrate resources for a mapping have to be able to support the performance requirements of the virtual resources. For example, a 1000 MBit/s virtual link can not be mapped to a path containing a 100 MBit/s substrate link. Likewise, the Central Processing Unit (CPU) power requested by a virtual node has to be less than (or equal to) the CPU power actually provided by a substrate node. Of course, substrate resources should be spent economically. Therefore, the mapping has to be optimized. This problem of mapping virtual resources to substrate resources in an optimal way is commonly known as the Virtual Network Embedding problem. This is typically modeled by annotating a Virtual Network Request with node and link demands. Likewise, the substrate network is annotated with node and link resources (also depicted in Figure 2.4). Demands and resources then have to be matched in order to complete the embedding. This means that virtual resources are first *mapped* to candidate substrate resources. Only if all virtual resources can be mapped, the entire network is then *embedded* and substrate resources are actually spent.

Table 2.1: Terminology used throughout this monograph

Term	**Description**
$SN = (N, L)$	SN is a substrate network, consisting of nodes N and links L
$VNR^k = (N^k, L^k)$	VNR^k denotes the k^{th} Virtual Network Request, consisting of nodes N^k and links L^k
$\dot{R} = \prod_{j=1}^{m} R_j$	$\dot{R}$ contains resource vectors for all resources $R_1, ..., R_m$
$cap : N \cup L \to \dot{R}$	The function cap assigns a capacity to an element of the substrate network (either node or link)
$dem_k : N^k \cup L^k \to \dot{R}$	The function dem_k assigns a demand to an element of VNR^k (either a node or a link)
$f_k : N^k \to N$	f_k is the function that maps a virtual node of VNR^k to a substrate node (VNoM)
$g_k : L^k \to SN' \subseteq SN$	g_k is the function that maps a virtual link of VNR^k to a path in the substrate network (VLiM)

Formally, the VNE problem can be described as follows (see Table 2.1): Let $SN = (N, L)$ be a substrate network where N represents the set of substrate nodes and L the set of substrate links and let $VNR^k = (N^k, L^k)$ be a set of $i = 1, ..., n$ Virtual Network Requests where N^k and L^k represent the set of virtual nodes and virtual links of the VNR i respectively. Furthermore, let $\dot{R} = \prod_{j=1}^{m} R_j$ be a vector space of resource vectors over resource sets $R_1, ..., R_m$ and let $cap : N \cup L \to \dot{R}$ be a function that assigns available resources to elements of the substrate network. Finally, for each VNR^k, let $dem_k : N^k \cup L^k \to \dot{R}$ be a function that assigns demands to elements of all Virtual Network Requests. Then, a Virtual Network Embedding consists of two functions $f_k : N^k \to N$ and $g_k : L^k \to SN' \subseteq SN$ for each VNR^k such that $\forall n^k \in N^k : dem_k(n^k) \leqslant cap(f_k(n^k))$ and $\forall l^k \in L^k : \forall l \in g_k(l^k) : dem_k(l^k) \leqslant cap(l)$. f_k is then called a node mapping function (VNoM) and g_k is called a link mapping function (VLiM). Together, they form an embedding for VNR^k.

Solving the VNE problem is NP-hard. Thus, current research is focused on heuristic and meta-heuristic approaches

Solving the VNE problem is NP-hard, as it is related to the multi-way separator problem [27]. Even with a given virtual node mapping, the problem of optimally allocating a set of virtual links to single substrate paths reduces to the unsplittable flow problem [28, 29], and thus also is NP-hard. Therefore, truly optimal solutions can only be gained for small problem instances. Thus, currently the main focus of work within the research community is on heuristic or meta-heuristic approaches.

2.2.1.2 *A Virtual Network Embedding taxonomy*

The generic problem introduced in the previous paragraph is described by three further constraints. Depending on the scenario, modification and relocation of virtual resources may be necessary. As such, VNE approaches either have to be static (i.e. with unchanging infrastructures) or dynamic (taking changes in virtual and substrate infrastructure into account). Moreover, virtual networks might be spread over the substrate infrastructure of multiple InPs. In this case, VNE has to be performed in a distributed way with multiple entities contributing to the mapping. Finally, depending on the scenario, virtual resources may be realized either concise, i.e. minimizing substrate resource usage, or redundant, combining multiple substrate resources to realize one virtual resource.

Instead of a particular property of a VNE algorithm, these constraints are rather different variants of the underlying VNE problem. Thus, all VNE approaches proposed in the literature can be categorized according to whether they are *Static* or *Dynamic*, *Centralized* or *Distributed*, and *Concise* or *Redundant*. These six concepts will be described here and used later on in Section 2.2.3 as a taxonomy to classify current VNE approaches.

2.2.1.3 *Static vs. Dynamic*

Existing VNE approaches can be categorized according to whether they are Static or Dynamic, Centralized or Distributed, and Concise or Redundant

In most real-world situations, VNE has to be tackled as online problem. That is, VNRs will not be known in advance. Instead, they arrive to the system dynamically and can stay in the network for an arbitrary amount of time. To be realistic, the VNE algorithm has to handle the VNRs as they arrive, rather than attending a set of VNRs at once (offline VNE). While in principle, all approaches can be operated in an online manner, static VNE approaches do not contemplate the possibility of remapping one or more VNRs to improve the performance of the embedding in the SN. Several effects lead to a need for relocation of parts of (or even complete) virtual networks:

- *Fragmentation of SN's resources*: Over time, as new VNRs arrive and are embedded and others expire and release their resources

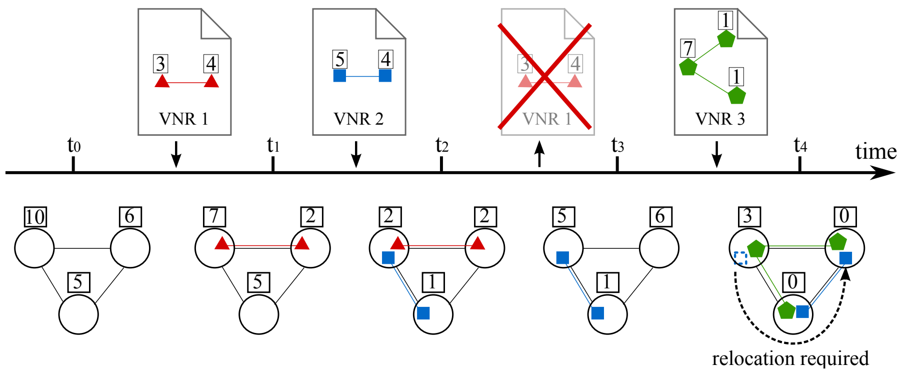


Figure 2.5: Relocation of mapped VNRs in online VNE

from the SN, the embedding becomes fragmented and the ratio of accepted VNRs diminishes, resulting in a long-term revenue abatement.

- *Changes in the VN*: Before its lifetime expires, a VN may change in terms of topology, size and resources due to new requirements demanded by its users.
- *Changes in the SN*: The SN can also suffer of long-term changes. InPs must be updating its networking infrastructure from time to time to cope with scalability issues and, hence, SN extends its size and current VNEs can find different and more optimal allocations.

Dynamic VNE approaches try to reconfigure the mapped VNRs in order to reorganize the resource allocation and optimize the utilization of SN resources. For example, Figure 2.5 shows the online embedding of three VNRs that arrive and leave the SN at different times. The last one of them (VNR 3) cannot be embedded if the resources of the SN are not first defragmented by reconfiguring the already mapped VN 2.

A good example of dynamic VNE can be found in [30]. Here, the authors realize that most of the VNR rejections are caused by the bottlenecked substrate links. To improve the rejection rate and the load balance in the SN, they propose a reactive and iterative algorithm called Virtual Network Reconfiguration (VNRe). The algorithm is reactive because it just acts when a VNR is rejected and works as follows: In first place it sorts the mapped virtual nodes by their suitability for migration, then it migrates the most suitable virtual node and its attached virtual links to another substrate node and tries to map again the VNR. If the network can not be mapped, the next iteration of the algorithm migrates the following virtual node and the process is repeated until the VNR is mapped or until a predefined number of iterations. Performance results show a significant increase of mapped VNRs after VNRe is applied.

2.2.1.4 *Centralized vs. Distributed*

The VNE problem can be solved in either a centralized or in a distributed way. Both approaches are fundamentally different, each having its own advantages and disadvantages.

CENTRALIZED In a centralized approach there will be one entity which is responsible for performing the embedding. This can be a dedicated machine computing optimal solutions to the problem. The advantage of this approach lies in the fact that the mapping entity is at every step of the embedding process aware of the overall situation of the network (i.e. it has global knowledge). This facilitates more optimal embeddings. On the other hand, a centralized entity presents a single point of failure - if it fails, the entire mapping process fails. Moreover, there may be scalability problems in large networks, where a single mapping entity may be overwhelmed by the number of Virtual Network Requests to handle.

DISTRIBUTED Contrary to centralized solutions, a distributed approach utilizes multiple entities for computing the embeddings. There may be some internal organization in how the mapping is distributed among the participating entities, or it may be organized completely ad-hoc. The advantage of such an approach lies in its better scalability. Since the load is distributed among several nodes, each individual node will be better able to cope with the embeddings. However, one has to pay for this with synchronization overhead. In particular, each node needs sufficient information about the global state of the network. The more information is available to a node, the better the results will be. However, there is also increased overhead – as such, the situation becomes a trade-off between communication cost and quality of the embeddings.

One specialization of this approach is the situation, where multiple Infrastructure Providers each only map part of a Virtual Network (InterInP VNE). In this case, even if individual Infrastructure Providers use a centralized approach within their own network, the overall procedure is to be considered distributed.

2.2.1.5 *Concise vs. Redundant*

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 might 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 redundant to be resilient regarding node and/or link failures. Otherwise, the embedding result is referred to be "concise" if there is no redundancy.

CONCISE The embedding results of *concise* approaches only use as much substrate resources as they are necessary to meet the demands of the virtual networks. There is no reservation of additional, redundant resources. This means that in case some substrate devices fail, there is no guarantee that the substrate network can recover from failure. However, since the approaches aim to be concise and to only use as much resources as necessary, the saved resources can be used to embed further virtual network requests.

REDUNDANT A redundant approach, however, reserves additional resources for the virtual entities that can be used in case some substrate resources fail at run-time. In general, there is a trade-off between the reliability of an embedding and its embedding costs: The higher the degree of reliability, the higher the embedding costs. The more resources are used, the less virtual entities can be embedded.

After the embedding has been done, the substrate entities have to be monitored. In case an instance fails, there needs to be a fail-back mechanism that is able to switch the primary instance that has failed to one of its backup instances (e.g. update routing tables) and activate it.

One can also include embedding algorithms that map virtual link demands to multiple paths of the SN inside the redundant category. Single-path approaches map virtual links to exactly one communication path within the substrate network. In contrast, multi-path approaches might split demanded Bandwidth (BW) of virtual links to multiple substrate paths. In case a virtual link is mapped to multiple substrate paths and some of these paths fail due to a failure within the network, packages can still be routed through the remaining communication links that have been set up between the substrate nodes by changing the splitting ratio. No reconfiguration has to be done, so it is fully transparent. However, bandwidth constraints have to be considered to avoid overloading of substrate link capacity.

2.2.1.6 *A notation scheme for VNE algorithms*

The aforementioned categories are mutually independent. An algorithm can be, for example, centralized, dynamic, and redundant at the same time. Consequently, this thesis introduces a generic notation to determine the class a specific VNE algorithm belongs to. VNE algorithms are described with the following syntax:

[C | D]/[S | D]/[C | R]

The first character denotes, whether the algorithm is **C**entralized or **D**istributed. Likewise, the second character denotes whether the algorithm is **S**tatic or **D**ynamic. Finally, the third character denotes whether the algorithm is **C**oncise or **R**edundant. So, an algorithm denoted as **C/D/R** will be a centralized, dynamic, redundant algorithm.

This will allow to quickly categorize any given algorithm and properly compare it with similar approaches.

2.2.2 *Computing optimal embeddings*

This section elaborates on: The different *parameters* that can be taken into account for VNE, the possible *objectives* that the embedding may pursue, the *problem decomposition and coordination* applied by existing VNE approaches, the *optimization strategies* used by those approaches to solve VNE, the various *metrics* that can be used to judge the quality of VNE, and a set of publicly available *tools* to evaluate the performance of VNE algorithms.

2.2.2.1 *VNE parameters*

The resources investigated in the VNE problem are attributed with parameters. On the one hand, substrate resources have individual capacities and qualities. On the other hand, virtual resources each have their respective requirements. For example, a substrate node can provide a certain computing capacity due to the CPU available to it. Per contra, a virtual node will require a certain computing capacity in order to properly compute routing information. These parameters are of paramount importance in order to achieve a valid embedding. Here, only the different kinds of parameters are discussed. A list of possible parameters is given both in [FBD+11] and [31].

One important distinction is that between *linear* and *nonlinear* parameters. The Linear Programming (LP) techniques described in Section 2.2.2.4 require linear parameters to find optimal solutions. If non-linear parameters (e.g. path loss probability) are considered, LP can not be used.

Apart from classifying linear and non-linear parameters, parameters can also be categorized according to several other dimensions. As a first step, one can distinguish between *node* and *link* parameters. Node parameters are attributes that refer to nodes, like computing power. Link parameters are attributes that refer to links, like bandwidth. However, there is a problem arising with such a strict model. One of the contributions of this thesis is the identification of the hidden hop's demand that refers to the computing power of substrate nodes on the path that may have an impact on the bandwidth of the virtual link being mapped [BHFDM13] (see Chapter 3). As such, virtual nodes are somewhat easier to map than virtual links, since the latter consist of a combination of substrate node and links, whereas the former take only substrate nodes into account.

VNE parameters can be classifies in several dimensions: Linear vs. Non-Linear, Primary vs. Secondary, Consumable vs. Static and Functional vs. Non-Functional

Realizing this, one can further distinguish between *primary* and *secondary* parameters [FBD+11]. Primary parameters are parameters that can be directly assigned to a substrate resource and, likewise, can be explicitly required by a virtual network request. Secondary parameters, on the other hand, depend on other (primary) parameters. For example, the probability of packet loss at a node depends both on its computing power and the size of its memory. Primary parameters will be easier to regard during the embedding, since they can be directly matched, whereas secondary parameters have to be calculated first.

Packet loss can also be a property of links (e.g. wireless links). In that case, one can realize that there is also a difference between *consumable* resources and *static* resources. Consumable resources are resources that are consumed when a virtual entity is mapped. The standard examples for this are computing power and bandwidth. In some VNE approaches though, the variability of the demanded bandwidth in a mapped virtual link is taken into account either by considering it a stochastic variable [32] or by opportunistically sharing it among different flows [33, 34]. Static resources, on the other hand, do not depend on the number of virtual resources mapped to a substrate resource. The loss probability of a wireless link in the substrate network will stay the same, no matter how many virtual links are mapped to it (provided the consumable resource "bandwidth" is not depleted). Static resources are easy to match during the embedding. Consumable resources, on the other hand, are what makes the embedding typically NP-complete.

Finally, one can further distinguish between *functional* and *non-functional* parameters. Functional parameters specify low-level functionality. Computing power and bandwidth are again good examples. Non-functional parameters, on the other hand, are high-level properties of the respective entities. Examples are security or resilience of an entity. The matching of functional parameters is typically straightforward (although possibly hard to compute), whereas non-functional parameters will require more elaborate approaches in an embedding algorithm.

2.2.2.2 *Main embedding objectives*

VNE consists of finding the optimal f_k and g_k functions in order to solve VNoM and VLiM with respect to a particular objective. This section describes the objectives that have been pursued by existing approaches to solve VNE.

Currently, the main VNE's objectives are to; comply with QoS requests, maximize InP's profit and provide resilent mappings

PROVIDE QOS-COMPLIANT EMBEDDINGS VNRs are installed and operated by the VNO according to a set of quality of service constraints defined by the service provider. These QoS requirements must be fulfilled by the virtual network embedding performed by the VNO.

There are several situations where these requirements are explicit in the request. For instance, a virtual network that provides Voice over IP (VoIP) services needs to count on medium bandwidth, low delays and high CPU requirements. Another example could be a vir-

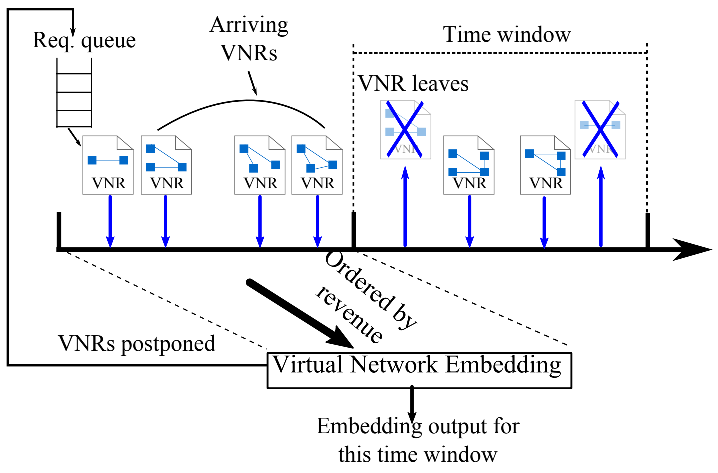


Figure 2.6: Ordering by revenue in online VNE

tual network offering P2P services that must provide medium bandwidth requirements, no relevant delay bounds and medium CPU requirements [35]. This can be achieved, e.g. by minimizing substrate resource stress (i.e. by distributing load equally across the substrate network).

MAXIMIZE THE ECONOMICAL PROFIT OF THE INP From the InP point of view, a natural objective of an online embedding algorithm would be to maximize the economic benefit of accepting VNRs (long-term average revenue). This objective is directly proportional to maximize the number of embedded VNRs (acceptance ratio). In order to reach this goal, VNE approaches should try to minimize the resources spent by the SN to map a VNR, also known as embedding cost. In this way, it is easier to embed the next VNRs, resulting in an increment of the VN acceptance ratio. However, as the revenue depends on the VNRs, a pre-ordering of the set of VNRs to be embedded may benefit the long-term average revenue. This pre-ordering must be carried out taking into account the online nature of the virtual network embedding problem.

The behavior of a typical online VNE algorithm is shown in Figure 2.6. Arriving VNRs are processed within time windows as well as in a request queue. If a request within a time window can not be embedded, it is deferred to the request queue and, in the next time window, tried again. This procedure is repeated until the time the VNR is willing to wait expires, in this case, the request is dropped.

To maximize the revenue, inside the time window, the VNRs can be decreasingly ordered by revenue. In [36], Chowdhury et al. present an approach: Window-based virtual Network Embedding (WiNE) algorithm that implements this procedure. Obtained results show that WiNE improves the mapped revenue when compared against online algorithms performing the embedding at the arrival of each VNR.

PROVIDE SURVIVABLE VNES Resilience in terms of VNE can be brought into play by integrating fallback resources within the substrate network. Backup nodes/links can be setup either for all or just for some specific primary nodes/links that may fail. At any time, consistency of the network topology has to be guaranteed, especially regarding the resources that were defined to be resilient towards failure. Recovery from failures should be transparent for the user. That is, he should not notice that the network switched to the backup resources. Even when using time-sensitive applications, the user should not notice that something went wrong. This especially requires that for selecting backup resources, all QoS requirements of the primary entities have to be considered.

The backup resource itself can be either dedicated or shared [37]. Dedicated means that for each virtual network, a complete backup network can be set up and backup resources are fully dedicated to the virtual networks and independent from each other. However, this is resource inefficient, since for each virtual resource that gets embedded, a dedicated substrate entity is needed. But in some cases, it might also be acceptable to share and reuse the backup resources in order to reduce the footprint on the substrate network caused by the additional backup resources. Usually, a higher degree of reused backup resources results in lower reliability and vice versa.

Furthermore, (shared) backup resources can be allocated either in advance (i.e., before the *first* virtual network embedding request arrived) or "on-demand" (i.e., allocated for each embedding request). Shared on-demand backup resources can be assigned at embedding time, that is, each time a virtual network request arrives [38, 37, 39]. Shared pre-allocation algorithms, however, define some specific backup resources in the configuration phase, i.e., before any virtual network request arrives [40].

2.2.2.3 *Problem decomposition and coordination*

As it has been already stated in the formulation section (see Section 2.2.1.1), the VNE problem is solved when its two sub-problems: VNoM, represented by the f_k function, and VLiM, represented by the g_k function, are solved. Looking at different InPs, one can decompose the InterInP problem (i.e. the embedding across several InPs) into a set of IntraInP problems (i.e. a set of embeddings within each InP). If these problem decompositions are not coordinated, optimization in one part can jeopardize optimization in another part. This subsection will discuss the options to handle this kind of coordination.

Coordination of the VNE's sub-problems (VNoM and VLiM) have 4 different variants: Uncoordinated, Two stages coordinated, One stage coordinated and InterInP Coordination

One alternative for VNoM/VLiM coordination is to solve each subproblem in an isolated and independent way. In this case, VNoM must be solved in first place because it provides the input to solve VLiM. This variant is called *uncoordinated VNE*. On the contrary, some VNE approaches have improved the performance of the solution by providing a coordination between the two phases. This variant, called *coordinated VNE*, can be solved in two different and coordinated stages, or in just one stage. Finally, some VNE approaches look for the mapping of a VNR across heterogeneous InPs. This variant, called *InterInP coordination*, aims to split the VNRs in different sub-requests and find the most adequate InP to map each of them.

UNCOORDINATED VNE A lack of coordination between VNoM and VLiM implies that the solution is generated in two different stages. The first stage solves VNoM and provides the function f_k to the VLiM, which is then solved in a second stage.

An example that demonstrates this lack of coordination between the VNE sub-problems was proposed in [41]. The main goal of this approach is to maximize the long-term average revenue. VNE is solved in two independent phases: VNoM follows a greedy algorithm, that chooses, for each virtual node, a set of eligible substrate nodes and then assigns one of them based on its amount of available resources. Depending on the assumption taken for the SN, VLiM is solved in two different ways: Single path mapping using one k-shortest path [42] solution for increasing k, when each virtual link must be mapped just to a single path in the SN and multiple path mapping when each virtual link demand can be carried by several paths in the SN. In the latter case, VLiM is reduced to the Multicommodity Flow Problem (MFP) [43] that provides a multi-path routing solution for each virtual link using optimal linear programming algorithms [44].

However, the lack of coordination between node and link mapping might result in neighboring virtual nodes being actually widely separated in the substrate topology. This fact increases the cost of single/multi paths used to solve the virtual link mapping phase resulting in low acceptance ratio and, therefore, low long-term revenue.

COORDINATED VNE A coordination between node and link mapping is desirable. If VNoM is preselected without considering its relation with link mapping, the solution space is restricted and the overall performance of the embedding decreases. Coordination of VNE can be performed in *two stages* when f_k is provided trying to obtain a result that optimizes the result of g_k. Alternatively, coordination can be also performed in *one stage* by solving the VNoM and VLiM at the same time.

Two stages coordinated VNE An approach that illustrates the coordination in two stages was first proposed by Chowdhury et al. in [45]. Its objective is to minimize the embedding cost. A new set of node constraints is added: Geographical location for substrate and virtual nodes and a non-negative distance per VNR indicating how far a virtual node of the VNR can be of its demanded location.

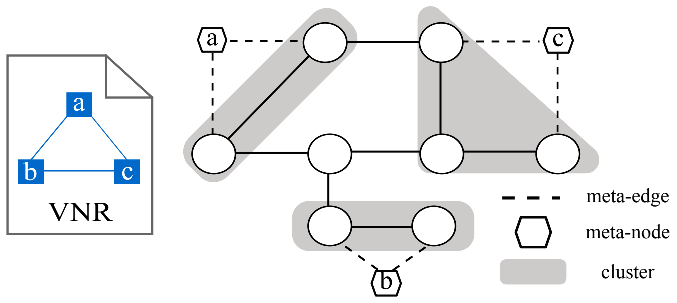


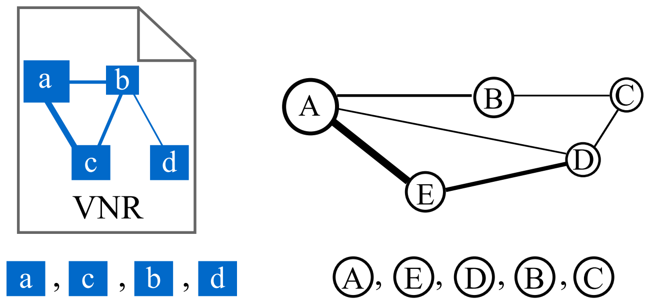
Figure 2.7: Augmented substrate graph with meta-nodes and meta-edges for a VNR

The node mapping stage starts by creating an augmented graph over the SN, introducing a set of meta-nodes, one per virtual node, each connected to a cluster of candidate SN nodes obeying location and capacity constraints. Over this augmented graph, the algorithm solves VNoM by proposing a Mixed Integer Programming (MIP) formulation. The MIP main goal is to solve the VNE trying to minimize the embedding cost, considering that the added meta-nodes map the requested virtual nodes (see Figure 2.7). To avoid the NP-completeness of the MIP, its linear programming relaxation is solved and virtual nodes are mapped to real substrate nodes (not meta-nodes) by rounding the obtained solution in two different ways: Deterministically or randomly -each resulting in a different algorithm: Deterministic Virtual Network Embedding (DViNE) and Randomized Virtual Network Embedding (RViNE)-. After that, VLiM is performed following the same two solutions proposed in [41].

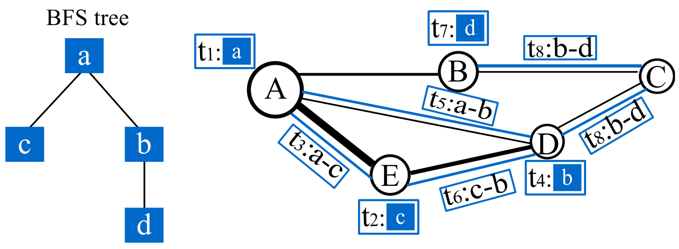
Coordination between both stages is strong, since the MIP formulation used for the virtual node mapping also considers the mapping of virtual links in the augmented SN between source and destination meta-nodes. Therefore, the substrate nodes chosen to map the nodes of the VNR are likely suited to provide virtual link mappings with low embedding cost.

One stage coordinated VNE Solving the VNE in one single stage implies that virtual links are mapped at the same time as virtual nodes. When the first virtual node pair is mapped, the virtual link between them is also mapped and, as each virtual node is mapped, the virtual links connecting it with already mapped virtual nodes are also mapped.

One good example for this variant is the approach proposed in [46] where one of the most important parameters of a network node (substrate or virtual) is its position inside the topology. Topological attributes of a node have a direct impact on the efficiency of the embedding. Taking that into account, a node ranking approach, inspired by the PageRank algorithm used by Google's search engine, is proposed to measure the topology incidence of a node. When topology



a) node ranking



b) one stage VNE

Figure 2.8: VNE in one stage

attributes are incorporated in node mapping, the acceptance ratio and the link mapping efficiency are improved. This is due to the fact that, given two nodes that are equal in resources, the node with the more capable neighborhood will be chosen, leading to a higher success probability for the embedding.

Figure 2.8 shows how the approach works. Based on Breadth-First Search (BFS), the proposed algorithm (*Random Walk-BFS* (*RW-BFS*)) solves VNE in one stage considering the impact of node mapping on the link mapping stage. The first step of *RW-BFS* is to calculate node ranks for substrate and virtual nodes as shown in Figure 2.8a. It then builds a BFS tree of the VNR, where the root node is the virtual node with greatest rank and the children are placed from left to right based on their rank. VNE is performed by going through the BFS tree and mapping each virtual node in the first feasible substrate node of its list and, at the same time, mapping the virtual links incident to that virtual node onto the substrate shortest paths that satisfy the BW demands. Figure 2.8b shows how the one stage VNE is performed. At t_1 and t_2 the virtual nodes a and c are mapped on top of the substrate node A and E respectively. At t_3 the virtual link between already mapped nodes (a and c) is mapped. Subsequent virtual nodes and links are mapped following the same procedure, combining, in this way, the virtual node and link mapping in one single stage.

INTERINP COORDINATION It is worth noting that previous algorithm variants consider the VNE in the single-InP scenario (IntraInP VNE). However, in realistic settings, VNRs must be mapped on top of

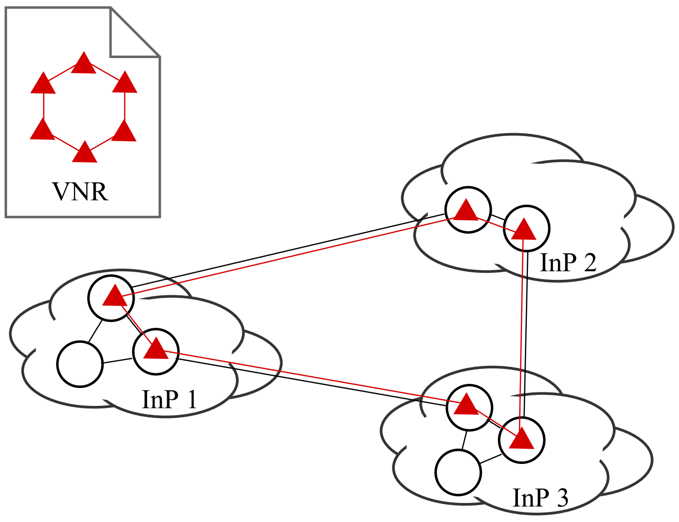


Figure 2.9: Inter-InP VNE process

a set of SNs managed by different InPs. Each InP provider should be able to embed parts of the virtual network and connect them using the external links among InPs. Therefore, to minimize the embedding cost, the SP splits the VNRs in several sub-requests and map each of them in the most convenient SN. Inside each InP, the sub-requests are mapped using ordinary VNE algorithms. Figure 2.9 shows a split VNR and its mapping to a set of SNs belonging to different InPs.

A good example of InterInP VNE is presented in [47]. Here, the authors address the conflict of interest between SPs and InPs. On the one hand, each InP strives to optimize the allocation in its equipment by getting requests for their high-margin equipment while offloading unprofitable work onto their competitors. On the other hand, the SPs are interested in satisfying their demands while minimizing their expenditure. The approach proposes a distributed protocol that coordinates the InPs and ensures competitive embedding pricing for the SPs.

2.2.2.4 *Optimization strategies*

To solve the VNE, three optimization strategies have been proposed: Exact, heuristic-based and metaheuristic solutions

The VNE problem is NP-hard (see Section 2.2.1.1). Therefore, for large problem sizes (i.e. large SN and VNRs size) the time to find the optimal solution becomes unaffordable. Taking this into account, three different types of approaches have been used to solve VNE. *Exact solutions* propose optimal techniques to solve small instances of the problem and to create baseline solutions that represent an optimal bound for heuristic-based VNE solutions. *Heuristic-based solutions* are not fixed on obtaining the global optimum. Instead, they try to find a good solution while keeping execution time low. Usually, heuristic solutions suffer from the problem that they can get stuck in a local optimum that may be far away from the real optimum. *Metaheuris-*

tic solutions improve the quality of the result by escaping from local optima in reasonable time.

EXACT SOLUTIONS Optimal VNE solutions can be achieved by means of Linear Programming. More exactly, Integer Linear Programming (ILP) can be used to optimally formulate the VNE including the virtual node and link mapping sub-problems in the same formulation. Although integer linear programs are in many practical situations NP-complete, there are exact algorithms (e.g. branch and bound, branch and cut and branch and price [48]) that solve small instances of the problem in reasonable time. Software tools implementing these algorithms, commonly called *solvers*, are available either as open-source (e.g. GLPK [49]) or proprietary (e.g. CPLEX [50]).

Some approaches have used ILP to solve VNE. For example, in [51], the VNE ILP formulation seeks for the minimization of the embedding cost and the maximization of the acceptance ratio. In [35], the ILP formulation of the problem introduces a set of new constraints: delay in links, routing and location in nodes. To deal with multiple connections between the same pair of nodes, the SN is described by a multigraph. One of the contributions of this thesis, the novel energy aware VNE introduced in [BHD+12] uses an ILP exact formulation where the goal is to embed the VNs' demand in a reduced set of equipment in the SN to save power by switching off the remaining SN resources (see Chapter 5).

HEURISTIC SOLUTIONS Execution time is crucial in VNE. Network virtualization deals with dynamic online environments where VNRs arrival time is not known in advance. Therefore, to avoid delay in the embedding of a new VNR, the execution time of VNE algorithms should be minimized. Accordingly, heuristic-based VNE solutions are proposed. They attempt to find an acceptable solution, compromising optimality for short execution time.

One good example of heuristic-based VNE approaches is presented in [52]. It proposes virtual node and link mapping in a single stage. To do so, VNE is reduced to the well known NP-hard Subgraph Isomorphism Detection (SID) problem. In graph theory, an isomorphism of graphs G and H ($G \simeq H$) is a bijection between the vertex sets of G and H, $m : V(G) \to V(H)$, such that any two vertices i and j of G are adjacent in G if and only if $m(i)$ and $m(j)$ are adjacent in H. The NP-complete SID problem tries to find a subgraph G_{sg} of G ($G_{sg} \subset G$) such that $G_{sg} \simeq H$. A modification of an existing heuristic is proposed to solve the VNE. It consists of finding an isomorphic subgraph (representing the VNR), accomplishing the VNR demands inside the substrate network, and using a hop limit constraint for the substrate paths used for the mapping of virtual links. The hop limit in the mapping of virtual link mapping determines the performance of the embedding. If the hop limit is too small, the acceptance ratio decreases, as there are less virtual link mapping possibilities. Conversely, if the limit is too large, the embedding cost is increased.

METAHEURISTIC SOLUTIONS VNE can be seen as a combinatorial optimization problem where an optimal solution is sought over a discrete search-space. As the optimal solution for large instances of these problems is hard to find, metaheuristics like simulated annealing [53], genetic algorithms [54], ant colony optimization [55], Particle Swarm Optimization (PSO) [56] or tabu search [57] can be used to find near-optimal solutions by trying to improve a candidate solution with regard to a given measure of quality. The following two approaches provide good examples of metaheuristic-based solutions for VNE:

An approach based on the Max-Min Ant Colony metaheuristic [58] has been recently proposed to solve the VNE in [59]. The problem is divided into a set of solution components (equivalent to small parts of the overall solution) and then, a set of parallel artificial ants are launched to iteratively explore the search space until a predefined number of iterations is reached. During each iteration, each ant builds incrementally the solution by transiting from one solution component to another. After all ants finish their full solution, the best one is selected as the solution of that iteration. Finally, the best solution among all iterations is chosen as the final embedding solution.

A PSO based approach is proposed in [60]. PSO is a population-based stochastic global optimizer that generates near-optimal solutions in low computing time with stable convergence. In [60], a PSO based VNE approach is proposed where each particle is a possible VNE solution that iteratively improves its position according to a fitness function (embedding cost). Finally the approximate optimal VNE solution is obtained through the evolution process of the particles. Each time, particles change their positions by modifying VNoM, VLiM is used then to guarantee the feasibility of the solution. This virtual link mapping can be performed using a single path (the shortest path) or using a multi-path approach (greedy k-shortest paths). Simulation results show the improvement in long term average revenue, acceptance ratio and Cost/Revenue (C/R) when compared against the proposal in [45].

2.2.2.5 *Embedding metrics*

The following categories of metrics can be used to evaluate the quality of a VNE: QoS, Cost-related, Resilience and Other metrics

Metrics are necessary to evaluate the quality of a successful embedding. They are used to compare different VNE approaches and to quantify advances in optimization. Within this section, different metrics are structured according to the embedding objectives indicated in Section 2.2.2.2 with "Other metrics" denoting metrics that don't exactly match one of the three objectives.

- Quality of Service metrics

- Cost-related metrics
- Resilience metrics
- Other metrics

Table 2.2 gives an overview over the various metrics discussed in this section. Depending on the application scenario, one can compute for most metrics the average, the maximum, and/or the minimum value.

quality of service metrics Quality of Service metrics aim to measure the impact of an embedding with respect to the service quality when using the virtual network. For example, when a user wants to run real-time applications like video telephony on its virtual network, the impact of the actual embedding that was chosen should not be perceptible. In addition to typical QoS metrics, metrics like the average length of paths between nodes should be taken into account because of the additional forwarding delay. Furthermore, utilization of network resources should also be considered: When a lot of different virtual entities are mapped upon a physical resource, the physical resource has to manage the load somehow. This could also include some additional scheduling overhead for switching between different virtual allocations.

Path length The path length metric measures the number of links between two substrate nodes that are mapped to two interconnected virtual nodes. The longer a corresponding path, the more resources had to be reserved for the embedding of the virtual link. This means that the path length has a direct impact to the embedding costs. Since every substrate node (except the receiving node) that is part of a path will take some time to forward packages sent via this path, the quality of service is influenced by the path length. In general, the package delay increases in connection with the length of a path.

Stress level The stress level of a substrate entity reflects the number of virtual entities that are mapped onto it. The more virtual entities use the same substrate resource, the higher the impact regarding possible side effects. For example, mapping many virtual nodes onto a single substrate node keeps the CPU of the host operating system busy. A high substrate link stress might result in some additional packet delay because the resources of the substrate link and the host interfaces communicating through this link have to be shared between virtual entities.

Utilization Utilization measures, in each SN entity (node or link), the sum of the spent substrate resources due to the mapping of virtual entities divided by the total amount of resources. This metric

Table 2.2: Metrics for virtual network embedding

Optimization goal	Metric	Description
Quality of Service	Path length	Describes the number of substrate links that are spanned by a virtual link on average
	Stress level	Describes the number of virtual entities realized by a substrate entity
	Utilization	Describes the sum of all spent substrate resources due to VNE divided by the sum of all provided substrate resources
	Throughput	Describes the data rate achievable between virtual nodes
	Delay	Describes the time a packet needs to travel across a virtual link
	Jitter	Describes the variance in inter-arrival times of packets on a virtual link
Resource spending	Cost	Describes the sum of all spent substrate resources for embedding VNRs
	Revenue	Describes the sum of all demanded resources of VNRs
	Cost/Revenue	Describes the ratio between spent substrate resources and provided virtual resources
	Acceptance ratio	Describes the number of VNRs that could be embedded
Resilience	Number of backups	Describes the number of available backup resources
	Path redundancy	Describes the diversity of paths in multi-path embeddings
	Cost of resilience	Describes the number of additional nodes required to maintain resiliency
	Recovery blocking probability	Describes the ratio of unrecoverable failure scenarios vs. all failure scenarios
	Migrations number	Describes the number of virtual nodes that have to be moved in case of failure
Other	Runtime	Describes the time a VNE algorithm will take for an embedding of a certain size
	Number of messages	Describes the number of messages that have to be exchanged in a distributed environment in order to complete the embedding
	Active substrate nodes	Describes the number of substrate nodes that have to be powered on in order to realize the hosted virtual infrastructures

is a more precise measurement tool than the stress level metric, because it also takes into account the magnitude of the resource usage instead of simply counting the number of virtual entities that use a resource. For example, to measure the utilization of a substrate node, the sum of mapped CPU resources divided by the CPU capacity of the node could be used. The usage of a substrate link could be based on the sum of mapped bandwidth resources divided by the total link bandwidth capacity.

Throughput Throughput is measured after the embedding of a virtual network has been performed. For each virtual node pair, packages between the corresponding substrate node pair are generated and sent through the path connecting these nodes. Then, the maximal data rate that can be transmitted through all connections between source-destination pairs is determined.

Delay Delay describes the amount of time needed for a packet to go from one node in the network to another node. With regard to VNE, algorithms can optimize the mapping of virtual resources so as to minimize delay between virtual nodes.

Jitter Jitter measures the variance in packet inter-arrival times. In a network virtualization scenario, jitter can be inherent to the substrate network (e.g. due to unreliable links) or introduced by virtualization itself (e.g. due to concurrent resource usage by two different virtual entities). The latter is strongly dependent on the employed virtualization solution and cannot be solved by VNE. Jitter that is inherent to the substrate network, on the other hand, can be taken into account by VNE algorithms. However, so far (to the best of the authors knowledge) there is no VNE approach that focuses on minimizing jitter.

RESOURCE SPENDING METRICS

Cost Cost in the VNE context refers to the amount of substrate resources that were used for embedding virtual networks. Cost is usually determined by summing up, in a weighted manner, all CPU and bandwidth resources of the substrate network that have been reserved for VNRs. However, besides CPU and bandwidth, additional types of resources can also be taken into account, and different types of resources can optionally be weighted in dependence on their value range. Cost is directly related to the length of substrate paths: The longer the length of a path, the more substrate resources are used and the higher the costs for an embedding.

Revenue Revenue refers to the sum of virtual resources that were actually requested by the virtual entities. This value is usually computed by applying the same scheme that was used to determine Cost.

Cost/Revenue To compare algorithms with respect to their embedding results, typically many different network topologies with varying size and properties are generated. Depending on these random topologies, Costs vary and therefore impede a fair comparision.

So in addition to Costs, Revenue is typically also taken into account: By dividing Cost by Revenue, varying Cost values are balanced. The higher the value, the more resources were needed to embed the VNs.

Acceptance ratio The acceptance ratio metric measures the number of virtual network requests that could be completely embedded by the embedding algorithm, divided by the total number of virtual network requests.

RESILIENCE-RELATED METRICS

Number of backups The number of backups metric counts the number of backup resources that are set up for a virtual entity. Several additional substrate entities can be reserved to serve as replacement in case the entity hosting the virtual entity fails.

Path Redundancy Path Redundancy measures the ratio between the number of backup paths to the number of direct paths. Some redundancy algorithms set up backup paths that could be used in case some parts of the network break down and cannot be used to forward data. Therefore, the metric refers to the amount of additional resources that are used to backup the embedded network.

Cost of resilience Related to the number of additional nodes required to maintain resilience: This metric measures the ratio of total number of running nodes and number of backup nodes. In contrast to Path Redundancy, this metric does not focus on connectivity resources but includes demanded node resources.

Recovery blocking probability When a substrate entity fails, the substrate network has to perform re-organizing actions to recover from failure. Especially, compensatory resources have to be allocated to catch the outage. Some approaches do not reserve these extra capacities in advance, so the system has to identify suitable backup resources at run time. Due to limited capacities of the entities inside the substrate network, this might fail which results in a failure of recovery. The recovery blocking probability metric measures the ratio of the number of unrecoverable failure scenarios to the total number of failure scenarios.

Number of migrations The number of migrations refers to the number of virtual nodes that need to be migrated to new facility nodes in case corresponding substrate nodes fail. Typically, at least the virtual nodes hosted on the failing substrate nodes have to be moved. Other constraints, like maximum path length, might however trigger even more migrations. Since migrations are resource intensive, they should be kept to a minimum.

OTHER METRICS

Runtime of the algorithm The runtime of algorithms can be used to compare algorithms with respect to the execution time that they need to compute an actual embedding result. For most real-life systems, runtime is a crucial factor, with a direct tradeoff between timely completion and quality of embedding results (e.g. cost/revenue).

Number of coordination messages For a distributed embedding approach, various messages between substrate nodes need to be exchanged for coordination purposes. The number of coordination messages is one related metric that can be used to determine and compare the communication overhead between different distributed approaches.

Active substrate nodes The number of active substrate nodes is related to the average length of substrate paths, because additional nodes are used to forward communication data between end nodes. Therefore, the probability that previously switched off nodes are selected to forward data rises. Regarding energy efficiency, the number of nodes that need to be turned on for an embedding can be a rough estimation of power consumption. In general, the ratio between running nodes and the total number of substrate nodes could be taken into account.

2.2.2.6 *Simulation tools*

In order to evaluate VNE algorithms, simulation tools have been developed by a number of authors. Typically, algorithms are run with a randomly generated set of scenarios. Each of these scenarios consists of a SN and a number of VNRs to be embedded. Appropriate parameters are assigned to both substrate and virtual resources. After the algorithms have tried to embed the VNRs, the results are evaluated using one or more of the metrics described in Section 2.2.2.5. VNE practitioners can use these simulation tools to develop and test their own VNE algorithms or to compare the performance of existing algorithms with regard to new metrics. This subsection presents three exemplary implementations of VNE simulation tools. Although this list is far from being comprehensive, it still indicates where one can start with practical VNE experiments.

The first example is the Vineyard VNE simulation tool by Chowdhury[1]. It has been used to validate the DViNE and RViNE VNE al-

1 http://www.mosharaf.com/ViNE-Yard.tar.gz

gorithms [45, 36]. Vineyard uses the GT-ITM network topology generator for random scenario generation. It is written in C++ with some accompanying shell scripts. It can evaluate revenue, cost, stress, and acceptance ratios. It has been extended in several other papers [40, 61, 62] to cover new VNE aspects.

Another example for a VNE simulation tool is the VNE Simulator developed by Yu[2]. Similar to Vineyard, the tool is written in the C programming language with some supporting shell scripts. It also uses the Georgia Tech Internetwork Topology Models (GT-ITM) network topology generator to generate network scenarios. It has been introduced in [41] to demonstrate the advantages of path splitting in virtual networks. Since then it has also been used and extended in a number of other publications [46, 63, 64].

One contribution of this thesis is the participation in the development of ALEVIN, a VNE simulation environment to implement, analyze and compare VNE algorithms

One of the contributions of this thesis is the easy to use VNE simulation environment called "ALEVIN" [FBD+11]. The results have been made available online[3]. ALEVIN uses a Waxman generator to create random network topologies. ALEVIN is a Java project with well-defined interfaces to implement new VNE algorithms and metrics. It comes with a number of pre-implemented algorithms drawn from several VNE papers. Moreover, it comes with extensive users and developers documentation, making it easy for others to jumpstart their evaluation. Several of the metrics mentioned in Section 2.2.2.5 have already been implemented in ALEVIN. By now, ALEVIN has been used in a number of publications [BHD+11, DSB+11, BHD+12] to compare VNE algorithms and evaluate new approaches to the VNE problem.

2.2.3 *A taxonomy of VNE approaches*

In this section, the criteria summarized in Table 2.3 are applied to classify the different approaches proposed to solve VNE by the research community. The categories discussed in Section 2.2.1.6 are used to group similar approaches. Table 2.4 and Table 2.5 list approaches in their respective categories. Each of them is further characterized, considering the coordination of VNE subproblems, as introduced in Section 2.2.2.3. Moreover, for each approach the applied optimization strategy is shown (see Section 2.2.2.4). Further comments describe individual contributions by each proposal. Approaches that appear in more than one category are marked with an asterisk (*). This happens, when an approach can be implemented in two ways (e.g. either static or dynamic).

The **C/S/C** category groups the set of proposals taking the straightforward approach, where aspects such as distributed behavior, dynamicity and redundancy are not considered. Algorithms in this cat-

2 `https://github.com/minlanyu/embed`
3 `http://alevin.sf.net/`

Table 2.3: VNE classification criteria

Criterion	**Values**	**Comments**
Category	**C/S/C**, **C/S/R**, **C/D/C**, **C/D/R** **D/S/C**, **D/S/R**, **D/D/C**, **D/D/R**	Categories defined in Section 2.2.1.6: **C**entralized or **D**istributed / **S**tatic or **D**ynamic / **C**oncise or **R**edundant
Coordination	Uncoordinated, One Stage, Two Stages, InterInP	Variants of the algorithms specified in Section 2.2.2.3
Strategy	Heuristic, Metaheuristic, Exact	Strategies used by the proposed approaches to solve VNE (see Section 2.2.2.4)

egory will operate in a centralized manner, expecting full knowledge of VNRs in advance, and providing concise solutions to the problem. The first VNE algorithms that have been proposed took this approach, with many other algorithms following. A taxonomy of these approaches is presented in the upper part of Table 2.4.

The VNE approaches belonging to the **C/D/C** category are grouped in the second part of Table 2.4. This set of proposals is able to perform a dynamic reconfiguration of currently mapped VNRs once a new VNR arrives. Mappings are concise and the algorithms run on a central instance. Dynamic approaches typically are more challenging – this leads to fewer publications in this category, compared to centralized, static approaches.

The VNE approaches belonging to the **C/S/R** category are again centralized and static. However, they consider redundancy. An example for this are embeddings that use multi-path virtual link mapping solutions. Redundancy can be used to increase resiliency when failures in nodes and links are foreseen. The VLiM stage is NP-hard if each virtual link must be allocated to a single path in the SN (cf. Section 2.2.1.1). However, if the virtual link can be mapped to multiple paths, VLiM may be reduced to the well-known multicommodity flow problem that can be solved in affordable execution time. As a consequence, a significant number of the existing approaches solve the VNE using multi-path for VLiM. The first part of Table 2.5 shows the taxonomy of the approaches in this category.

The second part of Table 2.5 groups the set of approaches that use redundancy to perform the reconfiguration of already mapped VNRs. The algorithms still run on a central instance. It can be seen that only few approaches try to combine redundancy and dynamicity.

Category	Reference	Strategy	Coordination	Contribution
C/S/C	[35] Inführ and Raidl (2011)	Exact	One Stage	Provides delay, location and routing constraints
	[65] Liu et al. (2011)	Exact	One Stage	Exact VNE based on correspondence matrices
	[66] Trinh et al. (2011)	Exact	One Stage	Exact VNE problem with SLA QoS guarantees
	[67] Pages et al. (2012)	Exact/Metaheu	One Stage	Introduces the VNE for optical networks
	[52] Lischka and Karl (2009)	Heuristic	One Stage	Provides one stage VNE. Based on SID
	[68] Di et al. (2010)	Heuristic	One Stage	Improvement of the approach in [52]
	[69] Ghazar and Saaman (2011)	Heuristic	One Stage	Introduces hierarchical management of the SN
	[70, 71] Yun et al. (2011-2012)	Heuristic	One Stage	First VNE approach in wireless multihop networks. Introduces metrics and feasibility measures for wireless VNE
	[64] Chen et al. (2012)	Heuristic	One Stage	Reduces resource fragmentation
	[72] Yu et al. (2012)	Heuristic	One Stage	One step VNE that increases coordination
	[73] Liu et al. (2011)	Heuristic	Two Stages	Improves coordination based on nodes proximity
	[33, 34] Sheng et al. (2011-2012)	Heuristic	Two Stages	Opportunistic resource sharing to deal with load fluctuation
	[74] Li et al. (2012)	Heuristic	Two Stages	Topology awareness to enforce VNE coordination
	[75] Lu and Turner (2006)	Heuristic	Uncoordinated	Embedding in specific backbone-star VN topologies
	[41] Yu et al.* (2008)	Heuristic	Uncoordinated	Utilizes the KSP algorithm [42] for VLiM
	[76] Razzaq and Siraj (2010)	Heuristic	Uncoordinated	Different K values in KSP based VLiM
	[77] Razzaq et al. (2011)	Heuristic	Uncoordinated	Investigates the VNE impact of bottlenecked nodes
	[78] Nogueira et al. (2011)	Heuristic	Uncoordinated	VNE considering SN resources heterogeneity
	[79] Leivadeas et al.* (2011)	Heuristic	Uncoordinated	Introduces VNE for wireless network testbeds
	[BHFDM13, BHD$^+$11] Botero et al. (2011-2013)	Heuristic	Uncoordinated	Introduces hidden hop constraints
	[80] Zhu and Ammar* (2006)	Heuristic	Uncoordinated	Provides a balanced link and node stress in the SN
	[59] Fajjari et al. (2011)	Metaheuristic	One Stage	Max-Min Ant Colony metaheuristic VNE approach
	[81] Cheng et al. (2012)	Metaheuristic	One Stage	Accelerates convergence of PSO VNE metaheuristic with topology aware node ranking [46]
	[82] Zhang et al. (2012)	Heuristic	Uncoordinated	Maps one virtual node in several substrate nodes
	[83] Di et al. (2012)	Heuristic	One Stage	Coordinated VNE reducing the number of backtracks by carefully chosing the first virtual node to map
	[84] Abedifar and Eshghi (2012)	Heuristic	Uncoordinated	Introduces VNE in the optical domain trying to minimize the number of λs per link
	[85] Aris Leivadeas et al. (2012)	Heuristic	Coordinated	Considers importance of virtual nodes for embedding
	[86] Tae-Ho Lee et al. (2012)	Heuristic	InterInP	clustering of virtual networks in multi-provider environment
C/D/C	[30] Fajjari et al. (2011)	Heuristic	One Stage	Migration of nodes with bottlenecked adjacent links
	[87] Bienkowski et al. (2010)	Heuristic	Two Stages	Migration when service access position changes

Category	Reference	Strategy	Coordination	Contribution
	[80] Zhu and Ammar* (2006)	Heuristic	Uncoordinated	Reduce the cost of periodic reconfigurations
	[88] Fan and Ammar (2006)	Heuristic	Uncoordinated	Reduces the cost of VNRs reconfiguration
	[89] Cai et al. (2010)	Heuristic	Uncoordinated	Reconfiguration based on SN evolution
	[90] Shun-li and Xue-song (2011)	Heuristic	Uncoordinated	Identifies mapped virtual nodes and links with not optimal mapping and migrate them to save SN resources
	[91] Sun et al. (2012)	Heuristic	Uncoordinated	Introduces the VNE problem for evolving VNRs
D/S/C	[92, 93] Houidi et al. (2010)	Heuristic	Uncoordinated	First distributed approach to solve VNE. Proposes a VNE protocol to manage the communication among substrate nodes
	[94] Xin et al. (2011)	Heuristic	InterInP	Introduces the InterInP VNE for networked clouds
	[95] Lv et al. (2011)	Heuristic	InterInP	InterInP VNE using hierarchical virtual resource organization
	[51] Houidi et al.* (2011)	Exact	InterInP	VNR is split assigning each subVN in different InPs. Provides exact and heuristic splitting approaches
	[96] Leivadeas et al.* (2012)	Heuristic	InterInP	Graph partitioning InterInP VNE using a heuristic integrating a min k-cut algorithm followed by subgraph isomorphism
D/D/C	[97] Marquezan et al. (2010)	Heuristic	Uncoordinated	First distributed dynamic approach. Reorganizes the SN when VNs demands change

Table 2.4: Taxonomy of concise VNE approaches

Category	Reference	Strategy	Coordination	Contribution
C/S/R	[51] Houidi et al.* (2011)	Exact	One Stage	First approach providing an ILP exact solution
	[98] Zhang et al. (2011)	Exact	One Stage	Optimal resilient solution attaining an enhanced QoS mapping. Provides diversified substrate back-up paths
	[BHD$^+$12] Botero et al. (2012)	Exact	One Stage	Introduces the energy-aware VNE
	[99] Wang and Wolf (2011)	Exact	One Stage	Redefines the VNR as a traffic matrix
	[100, 101, 102] Shamsi and Brockmeyer (2007-2009)	Heuristic	One Stage	Recover link failures by providing backup paths with intermediate nodes
	[103] Koslovski et al. (2010)	Heuristic	One Stage	Introduces reliabilty as a service offered by the InP. Reliable VNEs based on subgraph isomorphism detection
	[104] Yu et al. (2010)	Heuristic	One Stage	Introduces failure-dependent protection with a back-up solution for each regional failure
	[105] Lv et al. (2012)	Heuristic	One Stage	Introduces looses to multicast VNE in wireless mesh networks
	[45, 36] Chowdhury et al. (2009-2011)	Heuristic	Two Stages	Coordination in VNE using multi-path for VLiM
	[40] Rahman et al. (2010)	Heuristic	Two Stages	Upon a failure, the economic penalty is minimized by the pre-reservation of a bandwidth quota for back-up in SN links
	[61] Butt et al.* (2010)	Heuristic	Two Stages	VNE awareness of the SN bottlenecked resources

Category	Reference	Strategy	Coordination	Contribution
	[38] Yeow et al. (2010)	Heuristic	Two Stages	Introduces sharing among back up resources. Reduces resources allocated for redundancy
	[106] Sun et al. (2011)	Heuristic	Two Stages	Resilient VNE optimizing the embedding cost and reducing computational complexity
	[39] Yu et al. (2011)	Heuristic	Two Stages	Resilient VNE analyzing failures in substrate nodes
	[41] Yu et al.* (2008)	Heuristic	Uncoordinated	Introduces the multi-path approach for VLiM
	[107] Gao et al. (2010)	Heuristic	Uncoordinated	Improvement of the approach in [45]
	[108] Yang et al. (2010)	Heuristic	Uncoordinated	Divides the SN in regions to reduce VNE complexity
	[109] Zho et al. (2010)	Heuristic	Uncoordinated	Maps one virtual node to multiple substrate nodes
	[110] Chen et al. (2010)	Heuristic	Uncoordinated	Reactive resiliency protection approach against failures during the online VNE process. Considers just substrate link failures
	[111] Yu et al. (2011)	Heuristic	Uncoordinated	Proactive VNE approach offering protection against SN link failures for links with high stress
	[32] Sun et al. (2011)	Heuristic	Uncoordinated	Introduces stochastic BW demand to the VNE
	[63] Lu et al. (2011)	Heuristic	Uncoordinated	Introduces load balancing in links
	[37] Guo et al. (2011)	Heuristic	Uncoordinated	Proactive resilient VLiM approach sharing back-up paths
	[46] Cheng et al. (2011)	Metaheuristic	Two Stages	Introduces topology-awareness in VNE
	[112] Sheng et al. (2011)	Metaheuristic	Two Stages	Embedding time depends on VNR lifetime. Uses simulated annealing metaheuristic
	[60] Zhang et al. (2012)	Metaheuristic	Two Stages	Introduces particle swarm optimizaton (PSO) metaheuristic
	[113] Sun et al. (2012)	Metaheuristic	Two Stages	Introduces VNE in multi-datacenter environments
	[114] Lv et al. (2012)	Metaheuristic	Uncoordinated	Introduces VNE in wireless mesh networks
	[96] Leivadeas et al.* (2012)	Heuristic	Two Stages	Uses the approach in [36] to solve the VNE for an arbitrary pool of heterogeneous resources
	[62] Masti and Raghavan (2012)	Heuristic	Two Stages	VNE considering the residual capacity of the substrate links
	[115] Zhang et al. (2012)	Exact/Heuristic	One Stage	Recover link failures providing disjoint SN backup paths
C/D/R	[61] Butt et al.* (2010)	Heuristic	Two Stages	Reactive reconfiguration of virtual links and nodes causing rejection to less critical SN regions
	[41] Yu et al.* (2010)	Heuristic	Uncoordinated	Reconfigure the embedding by changing the splitting ratio in the multipath VLiM solution
	[116] Schaffrath et al. (2010)	Exact	One Stage	ILP-based VNE. Dynamically reconfigures existing mappings
	[117] Chen et al. (2011)	Heuristic	Two Stages	Periodic reconfiguration of SN nodes with high utilization
D/S/R	[47] Chowdhury et al. (2010)	Heuristic	InterInP	First InterInP VNE proposal. Mediates between InP and SP interests. VNR is split across InPs and embedded locally
D/D/R	[118] Houidi et al. (2010)	Heuristic	Two Stages	Fault-tolerant VNE that acts upon node and link failures

Table 2.5: Taxonomy of redundant VNE approaches

Until now, there has been little interest of the research community in finding distributed solutions for the VNE. This is likely due to the fact that distributed algorithms are significantly harder to implement. Moreover, it will be more difficult to reach near-optimal solutions with a distributed algorithm. As such, there are only a few approaches to list here. The lower parts of Table 2.4 and Table 2.5 show the existing approaches in the **D/S/C**, **D/S/R**, **D/D/C** and **D/D/R** categories.

The taxonomy of existing VNE proposals shows a lack of research in distributed VNE solutions

2.2.4 *Emerging research directions*

This section highlights the future research directions of VNE. Three main fields that may propel the VNE research in the near future are identified: Approaches inside the categories of *distributed* proposals, VNE proposals looking for the optimization of *new objectives* and the application of VNE in specific network *environments*.

2.2.4.1 *Distributed VNE*

Since the VNE problem is NP-hard, finding an optimal embedding is computationally complex. Therefore, most approaches tend to relax the original problem, focusing on near-optimal solutions. However, even these heuristic approaches do not scale well for large networks [92]. Instead of relying on a single, central node that computes all the embeddings, virtual network requests can be distributed to multiple substrate nodes. In this way, load can be spread, possibly increasing scalability. Table 2.4 and Table 2.5 show that, currently, there is a lack of distributed solutions for the VNE.

The only static and concise approach that performs intraInP VNE in a distributed way was proposed by Houidi et al. in [92]. Although it solves the weaknesses of centralized approaches, it suffers from two main problems: Excessive number of messages and suboptimal embedding cost.

The comprehensive survey of VNE allowed to identify three main fields for future research: distributed VNE, VNE of new objectives and VNE in specific networking environments

This thesis tackles the aforementioned insufficiency, Chapter 4 introduces a distributed VNE framework that reduces the message overhead by using clustering techniques that, in turn, confine the embedding tasks to a reduced subset of the SN, thereby reducing the embedding cost.

Until now, there is also a lack of VNE distributed solutions providing resiliency. The main interest of the research community has focused on offering resiliency by setting up back up resources at the provisioning time in a centralized way. However, the field of reactive and distributed embedding reconfiguration that acts in real time upon a failure has been only treated in one paper so far [118]. Unfortunately, it does not evaluate the optimality of the solution after reconfiguration and results in a considerable overhead of interchanged messages for big SNs.

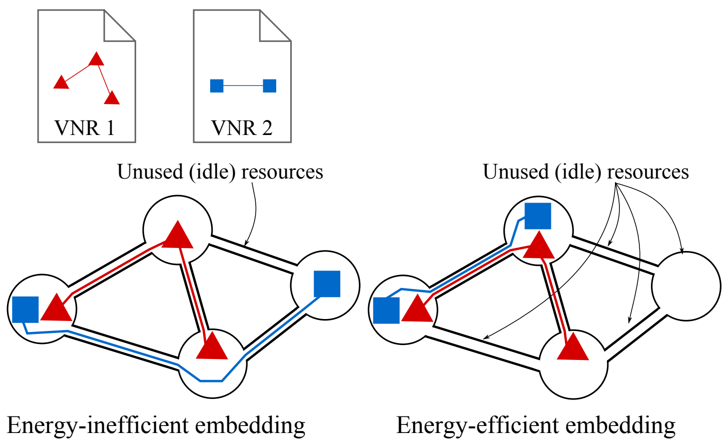


Figure 2.10: Green VNE

2.2.4.2 *Emerging VNE objectives*

Recently, VNE has acquired importance in two fields: Green Networking and Security.

GREEN NETWORKING Resource consolidation (several virtual instances in one physical resource), will be an enabler for energy savings in future infrastructure networks. Currently, some green strategies to allocate resources in cloud computing environments have been proposed. In [119], Chang and Wu propose an heuristic approach that seeks for the minimization of the computing and communications power of applications instantiated in a cloud substrate. In [120], Yang et al. propose a Green Power Management (GPM) to perform the virtual machine migration in cloud environments thanks to a dynamic resource allocation that seeks the ideal load balancing amongst virtual machines.

In network virtualization environments, if the VNE main goal is switched to the minimization of the power consumption, the SN can be dynamically dimensioned for current traffic demand rather than for peak demand. Due to power consumption insensitiveness of current network equipment to traffic load [121], the best approach to minimize the power consumption is to switch off or hibernate as many networking resources (nodes and links) as possible without compromising the network performance. The energy-aware dynamic VNE should be performed during low traffic demand periods, when some routers and interfaces can be switched off by rerouting the traffic to a smaller set of consolidated network equipment, as shown in Figure 2.10. Due to limited resource capacities, this sometimes leads to longer substrate paths between entities, which not only has side-effects to communication delays but also increases embedding costs. So in general, there is a trade-off between these objectives which should be taken into account.

One of the main contributions of this thesis is the energy-aware VNE approach called Energy-Aware Virtual Network Embedding (EA-VNE) [BHD+12] (see Chapter 5). EA-VNE's main goal is to minimize the switched on nodes and links after a VNR is mapped. However, EA-VNE uses exact ILP and just provides an optimal bound to evaluate future heuristic based algorithms. Furthermore, networking resources are assumed to be homogeneous with regard to their power consumption which is not a realistic assumption. The study also concludes that, as VN traffic load increases, the solution saves energy but at high costs in terms of acceptance ratio. To solve these issues, the second energy-aware VNE contribution presented in this thesis, published in [BH13], modifies and improves EA-VNE by introducing embedding cost and load balancing to its ILP formulation. Besides, as MIP-based energy-efficient VNE approaches are hard to solve for large network sizes and have an adverse effect in the number of successful embeddings, an heuristic approach to reconfigure, minimizing the energy consumption, the allocation of already embedded virtual networks, is proposed.

Future research could be focused in the provision of heuristic energy-aware strategies that find near-optimal solutions in reasonable running times. Besides, topology and load dependence are non-negligible aspects in power consumption. Current ISPs topologies are hierarchical with many network elements consuming low power amounts at the bottom and few network elements consuming higher amounts in the top. The recent development of green Information and Communications Technology (ICT) equipment [122] - where the power consumption depends on the load - can also be subject of future research in energy-aware VNE.

SECURITY In a virtualized future Internet, virtual network operators are envisioned to rent infrastructure from physical infrastructure providers, as indicated in Section 2.1. Virtual networks can be hosted on the hardware of multiple different InPs. At the same time, one InP can host networks from multiple VNPs/VNOs. As such, there will be security requirements on the one hand by the VNOs, and on the other hand by the InPs. Both parties have an interest to protect their respective assets (i.e. nodes and links – either physical or virtual).

In such an environment, the different stakeholders each have their own security requirements. A VNO is interested in having his network hosted on hardware that can offer a sufficient level of security. On the other hand, an InP will want to ensure that virtual networks are properly secured and do not run havoc on his equipment. Finally, different VNOs may distrust one another and require that their virtual infrastructure is not co-hosted on the same physical equipment in order to minimize the risk of cross-virtualization attacks.

This issue can not simply be solved by installing additional software in either the virtual or the physical nodes. Instead, it is necessary to avoid mappings that will increase risk for one of the stakeholders. Thus, there is a need for security-aware VNE algorithms. A security-aware VNE algorithm will have to take these requirements into account, trying to minimize risk exposure for all involved stakeholders. As such, it is necessary to allow virtual network operators and infrastructure providers to express their security needs. A VNE algorithm should then try to match those requirements as closely as possible [123].

2.2.4.3 *VNE environments*

Network virtualization maturity is motivating its application in specific network environments. For this reason, virtual network embedding strategies are migrating from theoretical to real scenarios trying to deal with their specific constraints.

WIRELESS NETWORKS Wireless networks have become one of the main type of access technologies nowadays, and virtualization is expected to be applied in wireless scenarios as well. Wireless links have broadcast nature and, consequently, the main distinctive feature of wireless network virtualization is how to virtualize links. Up to now, there have been some work on link virtualization based on Time-Division Multiplexing (TDM) [124, 125], Frequency-Division Multiplexing (FDM) [126] and space division multiplexing [127, 128].

The main challenge to overcome in VNE for wireless network comes from the broadcast nature of wireless links that may cause interference with other wireless links [129].

Recently, some approaches dealing with VNE in wireless environments have been proposed. An approach to solve VNE in TDM-based wireless virtualization environments was proposed in [70, 71]. Here, the authors introduce the *feasibility checking* to examine whether an embedding solution is feasible (not easy in wireless environments due to interference) and *embedding performance* to provide a measure on how good an embedding solution is. The approach, however, does not consider the time-varying conditions that could strongly affect the embedding performance.

VNE solutions for FDM-based virtualization have been recently provided in [79, 114, 105]. The approach in [79] introduces VNE for wireless network testbeds based on FDM link virtualization, while in [114, 105], the authors introduce VNE for the Wireless Mesh Network (WMN).

As can be noted, until now, VNE in space division multiplexing-based wireless virtualization environments remains unexplored. Besides, existing approaches miss some paramount characteristics of wireless environments that should be subject of future work, namely:

- Mobility: One of the main features of wireless network environments, is the mobility of their nodes. As a consequence, VNE dynamic approaches that consider nodes' mobility as a trigger for mapping reconfiguration could be incorporated to current proposals.
- Distribution: Some Wireless network environments are characterized by the lack of a centralized management entity, e.g. adhoc networks. Therefore, distributed VNE approaches in wireless networks deserve more interest in the near future.

OPTICAL NETWORKS The concept of Virtual Optical Network (VON) is introduced in [67], in turn, virtual optical network mapping is proposed. The VNE problem in optical environments is defined as the maximization of the number of mapped VONs from the demand set given the limited capacity of the optical SN. Two different variants of the problem are proposed:

- *Transparent optical mapping*: In this variant of the problem, optically transparent end-to-end services are provisioned over the VON. Transparent services mean that the optical VN is not assumed to have electronic termination capabilities in nodes. Therefore, the VON must allocate the same set of wavelengths for every virtual link.
- *Opaque optical mapping*: This variant assumes electronic capabilities in the optical VN, so that it is not necessary to allocate the same set of wavelengths for each virtual link thanks to the Optical-Electrical-Optical conversion.

The problem is formulated and solved with ILP techniques. To reach feasible running times for larger scenarios, a metaheuristic based on Greedy Randomized Adaptive Search Procedures (GRASP) [130] is also proposed.

Another VNE approach in the optical domain is presented in [84]. Here, Abedifar and Eshghi propose a VLiM approach that routes the virtual links on the physical infrastructure and assigns the available wavelengths to the resulting light paths trying to minimize the used wavelenghts per physical link.

These are the first approaches in the optical networks field. However, they open some questions, e.g. how to include the effect of physical layer impairment degradations in the optical VNE feasibility, especially in larger networks scenarios.

2.3 CONCLUSION

Virtual Network Embedding is a central problem to be solved when networks are virtualized. Optimizing the embedding of multiple virtual networks on one substrate network is computationally difficult for a number of important metrics. Multiple algorithms approaching this problem have been discussed in the literature, so far.

This chapter has presented a survey of current work in this area. A formal description of the VNE problem was provided. A categorization of VNE algorithms along three distinct dimensions (static vs. dynamic, centralized vs. distributed, concise vs. redundant) was developed. A list of optimization metrics was presented. A number of algorithmical approaches to the VNE problem was discussed. Finally, this information was used to create a taxonomy of VNE algorithms proposed in the literature.

There are a number of opportunities for future work in this area. It is to be noted, that the category of distributed VNE algorithms has received only sparse attention, so far. This thesis proposes an approach to alleviate this deficit in Chapter 4, in particular since centralized algorithms will always be prone to the criticism of having a single-point of failure. Moreover, there are also novel directions of VNE research like energy efficiency or security which have also been largely neglected by the scientific community up to now. New work in this area will have to define appropriate new metrics for the VNE problem and develop algorithms that will optimize according to power-saving or security-enhancing goals. Chapter 5 presents the first approximation to the energy-aware VNE and proposes exact and heuristic algorithms to solve it. The application of virtualization to real networking environments, e.g. wireless networks, is currently being studied by the scientific community. The definition of algorithms that study the VNE problem with a focus on environment-based constraints is also an exciting branch for future research.

Part II

CONTRIBUTIONS TO THE VNE PROBLEM

3 DISCOVERING PATHS FOR CENTRALIZED VNE

"If you do not change direction, you may end up where you are heading"
Gautama Buddha

This chapter is devoted to present two main contributions of this thesis to the VLiM: The introduction of *hidden hops* demands and the utilization of a mathematical multi-constraint routing framework called *paths algebra* to solve the virtual link mapping stage.

Up to now, existing VNE approaches did not include the virtual link demands of the hidden hops (an intermediate node in the SN path performing the VLiM of a determined virtual link). Also, they considered a limited number of single linear constraints: CPU in nodes and BW in links. Paths algebra provides the flexibility to introduce an unlimited number of linear and non-linear constraints and metrics to the problem while finding all the eligible paths in the SN to perform the virtual link mapping resulting in better and more flexible embeddings.

The introduction of the hidden hop demand concept is published in [BHFDM13, BHD+11] with the collaboration of Andreas Fischer and Hermann De Meer from the Chair of Computer Networks and Communications of the University of Passau (Germany). The virtual link mapping based on paths algebra approach is published in [BMHSA13]. This work was developed as a collaboration between Juan Felipe Botero and Xavier Hesselbach from the Telematics Department of the Universitat Politècnica de Catalunya and Miguel Molina and Jose Roberto Amazonas from the Department of telecommunications and control engineering of the University of São Paulo (Brazil).

In this chapter, two contributions dealing with the VLiM are presented: The addition of hidden hops demands and the addition of a flexible VLiM solution based on the paths algebra framework

3.1 INTRODUCTION

The virtual link mapping consists on how to route the virtual link demands between the pair of substrate nodes hosting the extremes of this virtual link (see Section 2.2.1.1). In this chapter, two contributions that deal with VLiM are presented: The addition of new demands caused by virtual link demands into *hidden hops* inside the SN paths and the addition of flexibility to the single-path VLiM solution thanks to the use of the *paths algebra* mathematical framework.

Hidden hops make reference to the intermediate nodes of a directed path in the SN that is mapping a specific virtual link of a VNR. The first contribution presented in this chapter is the introduction of hidden hop demands. This thesis claims that a hidden hop entails a resource demand because it has to perform packet forwarding of the traffic that will pass through this virtual link.

Hidden hops are the intermediate nodes of directed SN paths mapping virtual links demands. A hidden hop entails resource demand needed to perform forwarding of the virtual link traffic

Several implementation ways of QoS control (either by the network, directly by the application or via a mixed solution [131, 132, 133, 134, 135]) and the singular nature of each QoS parameters (availability, distance, flow, etc.) allow several network routing solutions to be proposed such as: Exact and approximated routing algorithms, algorithms based on backward-forward heuristics, linear composition, hybrid, random, routing computation from the origin or destination, reservation and resource allocation protocols etc. [136, 137, 138, 139].

In practice however, the multi-constraint routing problem is NP-complete [140, 141], and as such, it is possible to verify that many of the aforementioned solutions are restricted either by the size of the networks or by their topology, and their intuitive portability does not work for other applications. Normally, current solutions are focused on a specific type of metric (additive, multiplicative or concave). For instance, the use of an heuristic originally designed for distance metrics such as Dijkstra does not converge with another kind of metrics such as flow [142, 143, 144, 145, 146, 147].

Analyzing this problem under the perspective of protocols design, it has been necessary to conceive an heuristic or an algorithm to ensure the routing convergence for different types of QoS metrics or QoS metrics composition, in which this problem could be addressed from an integrated and generic manner by means of a mathematical framework that allows to validate the proposed solutions independently of network topology or implementation details [148].

Therefore, besides establishing a homogeneous mathematical basis, the concepts of *paths algebra* used in this work [149, 150] provide a guideline for developing a traffic engineering adaptive tool in which users can define their own path searching policy that can be closer to the existing traffic profile of their networks. The paths algebra is a mathematical framework that allows the validation and convergence analysis of the mono and multi-constraint routing problems independently of the network topology or size.

In addition, such mathematical framework allows for systematically comparing different mono-constraint and multi-constraint routing heuristics concerning their convergence guarantees, best path convergence and loop avoidance, and validating a generic and homogeneous solution that can be integrated into a single mathematical outline, flexible enough to be used in the development of new routing protocols that can be used either inside of a single administrative domain -Autonomous System (AS)- or across different ASs.

To accommodate a demand between two virtual nodes inside a virtual network, in this chapter, only one path is taken into account. This is often a realistic restriction due to the routing protocol used, or simply because it is an explicit management requirement. However, multi-path approaches have been proposed. To avoid the NP-completeness of the virtual link mapping, some VNE proposals [41, 45, 36, 51] split the virtual link demand and transport it through multiple SN paths (multi-path mapping). To do this, the virtual link mapping is reduced to the Multicommodity Flow Problem [43] that provides a multi-path routing solution for each virtual link using optimal linear programming algorithms. It is worth noting that optimal linear solutions are not available when the considered constraints are not linear (e.g. packet loss ratio or availability) and that, although the support of path splitting in the SN entails optimal link mapping solutions, the difficulty of its implementation is higher due to packet re-sequencing [151].

Most of the existing VNE proposals treat the single-path VLiM as a mono-constraint problem, that is, their objective is to map the virtual link in substrate paths that minimize/maximize the usage of one resource (typically bandwidth). This chapter introduces a virtual link mapping approach supporting multiple constraints thanks to the paths algebra routing framework.

The paths algebra is a mathematical framework that allows to compare different routing heuristics. Using paths algebra, VLiM can be performed in a multi-constraint basis, i.e. virtual links can be mapped to SN paths characterized by an unlimited number of constraints

The virtual link mapping stage of the VNE problem may be seen as a set of multiple multi-constraint routing problems. VLiM corresponds to finding the best route(s) on the substrate network for each virtual link in the VN, where best implies the adoption of some optimization criteria.

The *paths algebra* is an adequate mathematical framework to explore the design space. It solves the multi-constraint problem using linear metrics as bandwidth, number of hops and delay, or non-linear metrics as availability and package loss rate. It can also use a combination of metrics as, for example, QoS taken as:

$$\text{QoS} = f(\text{THRU}, \text{PDT}, \text{PDV}, \text{PLR}),$$

where:

- Throughput (THRU);
- Packet Delay Transfer (PDT);
- Packet Delay Variation (PDV) $\rightarrow$ delay jitter;
- Packet Loss Rate (PLR).

In this case, QoS is a function of physical parameters associated to the network performance.

However, considering that network virtualization is mainly a business strategy, the important metrics may be cost and revenue. The paths algebra may also use these metrics and, for example, associate them with the QoS.

Such flexibility associated to its computing efficiency makes the paths algebra a suitable tool to perform the VLiM stage inside the VNE.

The second contribution presented in this chapter is then the introduction of a flexible approach, based on the paths algebra framework, that provides a methodology to solve the link mapping stage of the VNE. Using paths algebra, VLiM can be performed in a multi-constraint basis, that is, virtual links can be mapped to substrate paths that are characterized by an unlimited number of constraints (or combination of constraints). For instance, if the instantiated VN runs an application that demands extremely low delay and also low packet loss rate, the proposed approach is able to rapidly provide all the SN paths ordered in first place by delay and then by PLR, so that, the virtual link mapping can choose the best suited path for each virtual link. VNE strategies using the proposed VLiM strategy can be categorized as uncoordinated **C/S/C** approaches (see Chapter 2).

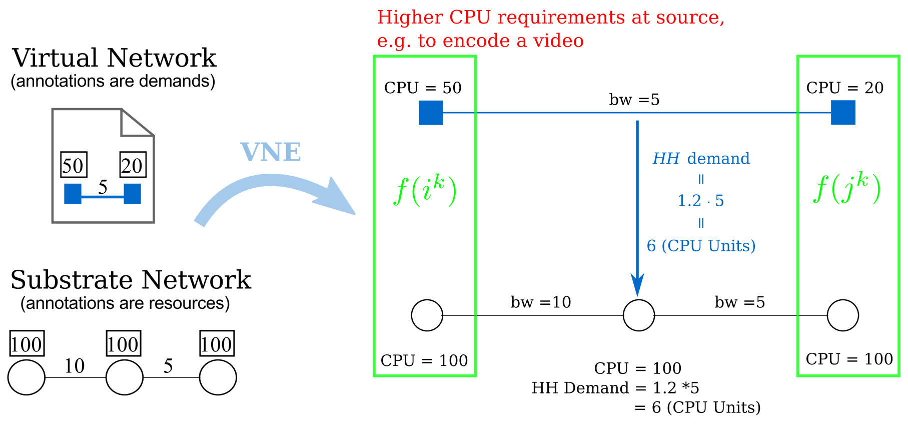
After this Introduction, Section 3.2 provides an insight of the possible demands that hidden hops entail and present the evaluation of several well known VNE algorithms in presence of hidden hop demands. To motivate the need for the paths algebra-based VLiM solution, Section 3.3 identifies the limitations of the existing VLiM strategies. Section 3.4 summarizes the paths algebra framework, while Section 3.5 introduces the novel paths algebra-based strategy to solve the VNE problem. Section 3.6 presents the simulations scenarios and experimental results. At last, Section 3.7 concludes the chapter and indicates the future work.

3.2 HIDDEN HOP DEMAND AND ITS IMPLICATIONS

Mapping a virtual link to a path in the SN obviously uses resources of the physical links on the path. However, there are also physical nodes on the path that will be traversed by the virtual link. These are called *hidden hops* here. The virtual link will also consume resources of all hidden hops on the paths. A hidden hop entails a resource demand because it has to perform packet forwarding of the traffic that is passing through this virtual link.

3.2.1 *Hidden hops: An example*

Hidden hop demand depends on the demand in the virtual links and the node type. Therefore, for each hidden hop $l \in N$ in a SN path mapping a virtual link (i^k, j^k), the demand is a function $HH(i^k, j^k, l)$, $l \in N, (i^k, j^k) \in L^k$ calculated from a combination of the virtual link demand and the node characteristics. It closely depends on the specific mapping scenario and equipment type.



Virtual Network
(annotations are demands)
50
20
5
VNE
Substrate Network
(annotations are resources)
100
100
100
10
5
Higher CPU requirements at source, e.g. to encode a video
CPU = 50
bw =5
CPU = 20
$f(i^k)$
HH demand = 1.2 · 5 = 6 (CPU Units)
$f(j^k)$
bw =10
bw =5
CPU = 100
CPU = 100
CPU = 100
HH Demand = 1.2 *5 = 6 (CPU Units)

Figure 3.1: Illustration of parameter categories and their application

Table 3.1: Evaluated algorithms

Notation		Algorithm Description
DViNE [45, 36]	SP →	Coordinated node and link mapping with k-shortest paths
	PS →	Coordinated node and link mapping with Path Splitting
GAR [41]	SP →	Greedy Available Resources with k-shortest paths
	PS →	Greedy Available Resources with Path Splitting

As an example of a hidden hop demand calculation, let's consider a SN transporting packets with size of 1500 bytes among substrate nodes as illustrated in Figure 3.1 (node type extracted from [152]). Assuming that CPU and bandwidth capacity units are scaled in a way that 100 CPU units = 2.66 GHz and 100 BW units = 1Gbps, the hidden hops CPU demand, due to the virtual link bandwidth demand, is calculated based on the number of cycles used to process a packet in the node when it is totally loaded (40000, in this case). One bandwidth demand unit is equivalent to ~ 833 packets/sec, therefore, the amount of CPU units needed to process it is 1.2. That is, the hidden hops belonging to the SN path mapped to the virtual link (i^k, j^k) CPU demand will be 1.2 multiplied by the units of BW demand of the virtual link (i^k, j^k).

3.2.2 *Evaluation of hidden hops Incidence*

To evaluate the incidence of hidden hop demands, several VNE algorithms are compared in different scenarios with different SNs, as well as different VNs, which cause a certain average resource load (the generation of load-based scenarios is detailed in Section 3.6.3.1).

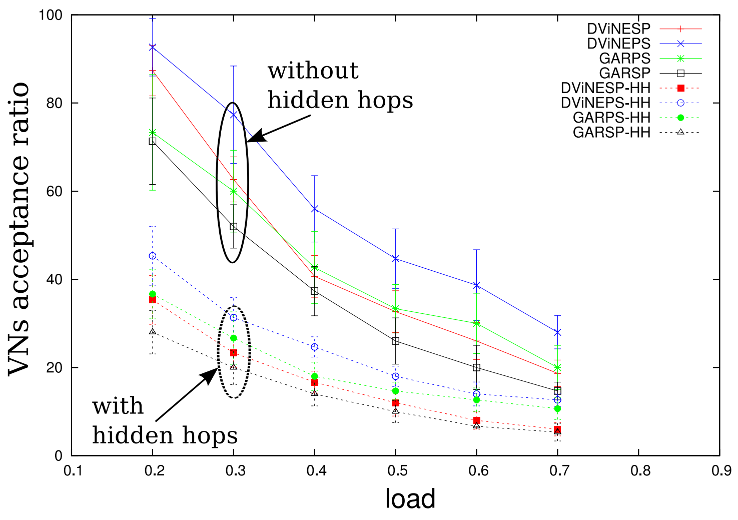


Figure 3.2: Evaluation of VNRs acceptance ratio with and without hidden hops

Including a hidden hop demand equivalent to the 50% of the realized virtual link's demand, results show a noticeable decrease of the VNs acceptance ratio

In this work, CPU cycles, denoted by cap_{CPU}, and bandwidth, denoted by cap_{BW} are considered as node and link resources, respectively, in the substrate network. Resource values are uniformly distributed with a maximum of $cap_{CPU}^{max} = 100$ and $cap_{BW}^{max} = 100$. As a trade-off between runtime of some algorithms and realistic scenarios, the number of substrate nodes and the number of virtual nodes per virtual network are chosen to be 50 and 20, respectively. Significant results were obtained by choosing 15 VNs to be embedded and performing 10 runs for each set of scenario parameters with a confidence level of 95%. Table 3.1 lists the evaluated VNE algorithms. For details of these algorithms, please refer to Chapter 2.

The evaluated algorithms were challenged by including a hidden hop demand factor of 0.5, i.e. each hidden hop on a substrate path will have a CPU demand equivalent to the 50% of the realized virtual link's demand. Figure 3.2 shows the VN acceptance ratio of the evaluated algorithms with the hidden hop factor and the behavior of the algorithms without considering hidden hops. The decrease of the VNs acceptance ratio is very noticeable (up to 50% in the worst case).

These results show the importance of hidden hop constraints. If VNE approaches do not take them into consideration, the obtained results neglect the effect of the extra demand in SN nodes, producing non-reliable results with regard to acceptance ratio and embedding cost.

3.3 SHORTCOMINGS OF EXISTING VLIM STRATEGIES

Depending on the assumption taken for the SN, the VLiM is solved in two different ways in the existing VNE proposals (see Chapter 2): Single and multi-path mapping. In single-path mapping, each virtual link must be mapped just to a single path in the SN whereas in multi-path mapping each virtual link demand can be carried out by several paths in the SN.

Existing single-path VLiM strategies are solved using K shortest path algorithms in a mono-constraint basis

To consider a unique substrate path to transport a virtual link (single-path mapping) is often a realistic restriction due to the used routing protocol, or simply an explicit management requirement stipulating avoidance of packet re-sequencing in receiving nodes. However, the problem of mapping multiple virtual link demands to a set of single paths in the SN is NP-Complete [BHFDM13]. Consequently, existing VNE proposals resolve single-path link mapping using heuristics based on K shortest path routing algorithms for increasing k [80, 52, 41, 46]. This greedy solution allows to find the best path working in a mono-constraint basis, optimizing just one constraint or a function combining a set of constraints characterized by a specific type of metric (additive, multiplicative or concave).

In this chapter, a flexible strategy to solve the single-path VLiM based on paths algebra is proposed. Contrary to K shortest path-based mono-constraint approaches, the multi-constraint routing algorithm based on paths algebra Loop Avoidance by the DestiNation (LADN) [150] is able to *find* all the possible paths between each pair of nodes in the SN and *organize* them based on an unlimited number of constraints (or combination of constraints). LADN is composed of four stages: SEARCHPATH (discovers all paths between each pair of SN nodes), SORTPATH (selects only cycle-free paths), EVALUATEPATH (characterizes each path based on the defined link parameters) and ORDERPATH (orders the paths according to the defined metrics and priorities). The proposed strategy tries to map each virtual link in the best compliant path (previously sorted by LADN). Also, the proposed strategy supports linear and non-linear constraints, or even a combination of them.

Here, a flexible strategy to solve the VLiM using the multi-constraint routing algorithm based on paths algebra called LADN

Summarizing, the contributions of the paths algebra-based VLiM strategy are threefold:

- It introduces constraint flexibility. Besides CPU and BW, an unlimited number of constraints can be introduced. The multi-constraint routing problem can be solved using any set of metrics resulting of the combination of different constraints.
- As well as linear, non-linear constraints can be easily introduced in paths algebra.
- Paths algebra finds all eligible paths between each pair of nodes in the SN. This feature can be used for virtual network resiliency or online topology changes in mapped VNs.

The paths algebra framework allows to consider multi-constraint routing with linear and not linear constraints providing a more flexible and extensible solution for the VLiM stage of the VNE.

3.4 THE PATHS ALGEBRA FRAMEWORK

In 1979, trying to analyze and solve path searching problems in directed graphs, Carré [153] proposed a paths algebra that allowed to validate the mono-constraint routing algorithms and analyze their convergence under a single paradigm independently of the network topology.

Taking this first approach as a reference, 20 years later, in [142], Gouda and Schneider included the multi-constraint routing problem and proposed a formal definition of routing metrics based on three basic metrics types: Flow, distance and reliability; and two metrics compositions: Additive composition and lexical composition. The main contribution of Gouda and Schneider is the identification and validation of two properties: Boundedness[1] and isotonicity[2], as the necessary and sufficient conditions to maximize a metric or, in other words, to build a tree using only the optimized paths that connect a starting node to all other nodes in a network.

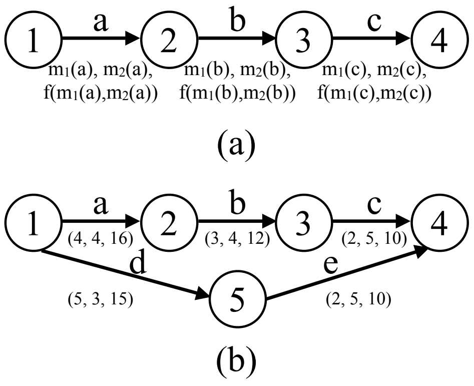
Using the previous ideas, in [154], Sobrinho redefined the concept of paths algebra presented by Carré and the previously described properties defined by Gouda and Schneider. Sobrinho also indicated that isotonicity is a necessary and sufficient condition to enable the hop-by-hop routing and to ensure the best path computation through a generalized Dijkstra algorithm. In [154], Sobrinho also proposed the strict isotonicity property as the necessary and sufficient condition to be followed by the path weight function in a hop-by-hop routing for loop avoidance. Subsequently, in [155], Sobrinho expands the algebra presented before incorporating the concepts of labels and signatures without analyzing the multi-constraint problem.

In 2008, in [150], Herman and Amazonas integrated the aforementioned approaches and proposed a new generic mathematical framework that can be applied without modifications to different applications. This framework facilitates the inclusion of new metrics, new ways of computing the resulting values of such metrics along a path and new ways of comparing these paths while keeping the same analysis criteria. This new framework also introduces a new property, called coherence[3], as the necessary and sufficient condition for a hop-by-hop routing algorithm to converge. The new framework introduced new tie-break criteria that allows to distinguish paths considered equivalent by the previous approaches. Its basic principle is the multidimensional lexical ordering.

1 *Boundedness* can be defined as a generalization of the concept of positive distance cycles, for which a cycle is said free if at least one of its nodes forwards the information to a node out of the cycle, avoiding the loop

2 The *isotonicity* property allows keeping the order relation between two different paths having the same starting node that are extended through the same node [5]

3 *Coherence* is the existence of at least one direct path between any pairs of source-destination nodes in a network



1
a
2
b
3
c
4
m1(a), m2(a), f(m1(a),m2(a))
m1(b), m2(b), f(m1(b),m2(b))
m1(c), m2(c), f(m1(c),m2(c))
(a)
1
a
2
b
3
c
4
(4, 4, 16)
(3, 4, 12)
(2, 5, 10)
d
(5, 3, 15)
5
e
(2, 5, 10)
(b)

Figure 3.3: (a) Example of a simple path (b) Example of two paths to be ordered

The main aim of this section is to show how to apply the new paths algebra framework into the solution of the VLiM stage of the VNE. Therefore, the mathematical formalism, already discussed in [150, 5], is not included here. Instead, to make it easy to understand, the paths algebra framework is presented in a simplified way by means of examples.

3.4.1 *Paths characterization*

A network is represented by a directed graph $G = (N, L)$, where N is the set of vertices and L the set of arcs. Consider the simple path represented in Figure 3.3.a. The set of vertices is given by $N = \{1, 2, 3, 4\}$ and the set of arcs is given by $L = \{a, b, c\}$. The source and destination nodes are $(s, d) = (1, 4)$. This path can be represented either as a succession of vertices $p_{1,4}$ or as a succession of arcs $p_{a,c}$.

In this example, each arc is characterized by a triple $(m_1(x), m_2(x), f[m_1(x), m_2(x)])$, where: $m_1(x)$ and $m_2(x)$ are the values of metrics m_1 and m_2 on the arc $x \in L$; $f[m_1(x), m_2(x)]$ is a function of combination of metrics applied to $m_1(x)$ and $m_2(x)$.

In general, the paths algebra uses **M** as the set of m adopted routing metrics and **F** as the set of k metrics combination functions.

The set of combined-metrics of all edges is given by:

$$\overline{C}(p_{a,c}) = \begin{bmatrix} \overline{C}_a \\ \overline{C}_b \\ \overline{C}_c \end{bmatrix} = \begin{bmatrix} m_1(a) & m_2(a) & f[m_1(a), m_2(a)] \\ m_1(b) & m_2(b) & f[m_1(b), m_2(b)] \\ m_1(c) & m_2(c) & f[m_1(c), m_2(c)] \end{bmatrix}$$

A synthesis $\overline{S}[.]$ is a set of binary operations applied on the values of the links combined-metrics along a path to obtain a resulting value that characterizes this path as far as the constraint imposed by the combined-metric is concerned. So far, the syntheses are restricted to the following set: $\{add(), mult(), max(), min()\}$.

Table 3.2: Synthesis result of the network given in Figure 3.3.b

PATH	S_1 MIN	S_2 MAX	S_3 ADD
γ	2; 3; 4	5; 4; 4	38; 28; 16
θ	2; 5	5; 3	25; 15

If the routing algorithm is mono-constraint, only one value is obtained as the synthesis result and it is called weight-word. If the routing algorithm is multi-constraint, with k constraints, then k values are obtained. In this example, $\overline{S}[.] = [S_1 S_2 S_3]^t$. The weight-word has as many letters as the path's number of arcs. The first letter corresponds to the resulting value of the synthesis applied to the whole path; the second letter corresponds to the resulting value of the synthesis applied to the sub-path obtained by dropping out the last arc; the last letter corresponds to the resulting value of the synthesis applied to the sub-path made of only the first arc. Any number of letters can be retained as the synthesis result and this is called an abbreviation: $\overline{b}_j\left(\overline{S}[.]\right)$ represents a j-letters abbreviation; $\overline{b}_\infty\left(\overline{S}[.]\right)$ represents no abbreviation, i.e., all letters are taken into account.

3.4.2 *Paths ordering*

Consider the network represented in Figure 3.3.b where two paths connect the source node 1 to the destination node 4. These paths are $\gamma = (1,2,3,4) = (a,b,c)$ and $\theta = (1,5,4) = (d,e)$. Each paths' arc is characterized by a triple $(m_1(x), m_2(x), f[m_1(x), m_2(x)])$, where $f[m_1(x), m_2(x)] = m_1(x) \times m_2(x)$. The syntheses to be used in this example are given by $\overline{S}[.] = [\min()\max()\text{add}()]^t$.

The result of the synthesis is shown in Table 3.2. A path γ is less optimized than a path θ, if $\overline{S}[\gamma] \preceq_{ML} \overline{S}[\theta]$, where $\preceq_{ML}$ stands for multidimensional lexical ordering. In the example $\preceq_{ML} = \{\geqslant, \leqslant, \geqslant\}$, that is translated by the following ordering relations:

- $S_1[\gamma] \preceq S_1[\theta] \Rightarrow S_1[\gamma] \geqslant S_1[\theta]$;
- $S_2[\gamma] \preceq S_2[\theta] \Rightarrow S_2[\gamma] \leqslant S_2[\theta]$;
- $S_3[\gamma] \preceq S_3[\theta] \Rightarrow S_3[\gamma] \geqslant S_3[\theta]$;

Different syntheses also have different priorities. In the example, S_1, S_2 and S_3 priorities go from the highest to the lowest.

Table 3.3 summarizes the results obtained for three different ordering criteria. It is important to realize that the syntheses letters are examined from the highest priority to the lowest priority synthesis. The next step, when the paths are considered equivalent, is then to

Table 3.3: Paths ordering of the network given in Figure 3.3.b

ABBREVIATION $\bar{b}_j$ $(\bar{S}[.])$	RESULT
$\bar{b}_1[S_1]\ \bar{b}_1[S_2]\ \bar{b}_1[S_3]$	$S_1 \Rightarrow \gamma \equiv \theta$
	$S_2 \Rightarrow \gamma \equiv \theta$
	$S_3 \Rightarrow \gamma \prec \theta$
$\bar{b}_\infty[S_1]\ \bar{b}_\infty[S_2]\ \bar{b}_\infty[S_3]$	$S_1 \Rightarrow$ 1st letters are equal
	$\Rightarrow \gamma \equiv \theta$
	$S_1 \Rightarrow$ 2nd letters $\Rightarrow$
	$3 < 5 \Rightarrow \theta \prec \gamma$
$\bar{b}_1[S_1]\ \bar{b}_\infty[S_2]\ \bar{b}_1[S_3]$	$S_1 \Rightarrow \gamma \equiv \theta$
	$S_2 \Rightarrow$ 1st letters are equal
	$\Rightarrow \gamma \equiv \theta$
	$S_2 \Rightarrow$ 2nd letters $\Rightarrow$
	$4 > 3 \Rightarrow \theta \prec \gamma$

examine either the next letter of the same synthesis or will move to the next synthesis. This is determined by the adopted abbreviation.

3.5 PATHS ALGEBRA-BASED VIRTUAL LINK MAPPING

In paths algebra, any set of metrics and combination of metrics can be employed, different optimization criteria can be examined simultaneously and the quality of different solutions can be evaluated. This is very useful to solve the VLiM

The example presented in Section 3.4.2 has shown that the paths algebra provides a great flexibility to the mapping of transport network requirements. Any set of metrics and combination of metrics can be employed, different optimization criteria can be examined simultaneously and the quality of different solutions can be evaluated. It is important to realize that lexical ordering will always provide a total ordering of the paths, while a cartesian product ordering accounts only for partial ordering.

As far as network virtualization is concerned, once the requirements of the virtual networks requests have been annotated in a weighted digraph, the paths algebra is a useful tool to find a suitable assignment of the substrate network. It is very powerful to consider different metrics as spare bandwidth, CPU use, reliability, power consumption, etc. Besides finding the best path between a source and destination pair of nodes, the paths algebra ranks all eligible paths from best to worst, along the cost functions values. This is a valuable information for restoration problems, where alternative paths can be stored in the database.

Paths algebra-based VLiM approach is uncoordinated, i.e. the substrate node assigned to each virtual node, is known in advance. Node

Table 3.4: Metrics for virtual link mapping

METRIC	DEFINITION
Spare CPU	Remaining CPU load available in a SN node after virtual link mapping
Stress	Number of virtual instances running on top of a specific substrate node or link
Spare memory	The remaining memory capacity of a node after mapping a virtual link
Delay	The individual delay of a link belonging to the SN
Link Jitter	The jitter of a SN link used in the mapping of a virtual link
Losses	The percentage of packet loss in a link belonging to the SN
Reliability	The individual reliability of a link belonging to the SN
Energy	The energy consumed by each SN node to process the packets
Cost	The cost incurred by each SN node to process the packets
Revenue	Revenue produced the SN to carry out the VN demands

mapping has been discussed in several previous works (see Chapter 2).

Table 3.4 shows some (but not limited to) possible metrics that can be applied to the virtual link mapping in the VNE. A complete set of VNE metrics can be found in Section 2.2.2.5.

All metrics are associated as arc weights to the SN, even when they represent a node characteristic as, for example, the spare memory capacity. This association will be detailed in Section 3.5.1.

As already mentioned in Section 3.4.1, individual metrics can be combined into a cost function. There is no restriction neither about how the metrics are combined nor about the number of metrics plus cost functions to be used in a specific VNE problem.

The value of a metric (or cost function) along a path and its corresponding sub-paths is evaluated by means of the paths algebra *Synthesis* operation. As already mentioned, four syntheses may be used: i) minimization - min(); ii) maximization - max(); iii) addition - add() and iv) multiplication - mult(). The synthesis to be used is metric dependent. For example, to evaluate the path delay, the links delay must be added, while to evaluate the path spare bandwidth capacity, the minimum of the links spare bandwidth must be found.

The paths algebra has been implemented by the LADN algorithm [150]. In hop-by-hop routing algorithms, each node decides about the best next hop to forward a packet independently from the decision by any other node. This may create loops when, for example, a node x to reach a destination d decides that the packet has to be sent to node y and node y to send the packet to the same destination d decides that the best next hop is node x. The packet will be traveling forever between x and y. The LADN algorithm takes care of this situation and enforces the *coherence* property [149, 150] that avoids loops.

3.5.1 *Solving the VNE*

To solve the mapping of virtual links, a paths algebra-based approach composed of four stages was used in this thesis.

Here, a flexible strategy to solve the single-path virtual link mapping based on paths algebra is introduced. Contrary to K shortest path-based mono-constraint approaches, this proposal uses the paths algebra-based multi-constraint routing algorithm LADN [150] to exploit its capability of *finding* all the possible paths between each pair of nodes in the SN and *organizing* them based on an unlimited number of constraints (or combination of constraints). LADN is composed of four stages:

- SEARCHPATH discovers all paths between each pair of SN nodes;
- SORTPATH selects only cycle-free paths;
- EVALUATEPATH characterizes each path based on the defined link parameters and
- ORDERPATH orders the paths according to the defined metrics and priorities.

The proposed strategy tries to map each virtual link in the best compliant path (previously sorted by LADN). Also, it supports linear and non-linear constraints, or even a combination of them.

3.5.1.1 *Dealing with node metrics*

The proposed VLiM strategy maps each virtual link in the best compliant path, previously sorted by LADN. It supports linear and non-linear constraints

The paths algebra has been developed associating weights to the arcs of the digraph that represents the network. In order to deal with the metrics associated to the digraph's nodes it is necessary to perform the following transformation. Consider, for example, the part of a digraph shown in Figure 3.4.a. In this figure, the weights are associated both to the nodes and arcs. Nodes A and B spare CPU capacity are given by CPU(A) and CPU(B), respectively. The spare bandwidth of the link connecting nodes A and B is the arc weight BW(A-B). When

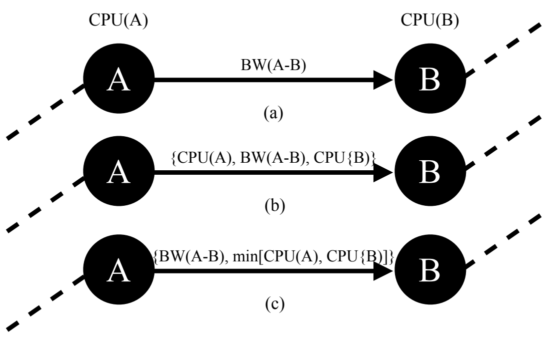


Figure 3.4: (a) Weights associated both to nodes and arcs; (b) Nodes' weights associated to the arc connecting the nodes; (c) Final metrics associated to the arc that is to be used by the paths algebra

a packet traverses the network going from node A to node B it will be processed by both CPUs and flow through the link connecting both nodes. This being so, all metrics can be artificially assigned to the arc connecting both nodes as depicted in Figure 3.4.b.

As one possible objective could be to maximize the spare CPU capacity, the most restricting condition is given by min[CPU(A), CPU(B)]. So, the metrics combination function provided by the paths algebra can be used and the arc's metrics reduced to the pair (BW(A-B), min[CPU(A), CPU(B)]) as shown in Figure Figure 3.4.c.

It is important to realize that independently of a specific objective, any VNE problem has to assign the VNRs to SN nodes and links, under the constraint of available physical resources. The possibility of the paths algebra to deal with both node and CPU metrics as well, makes it a complete framework to solve the VNE problem. It can be even envisaged the development of new algorithms in which nodes and link assignments can be done in a coordinated way instead of in separate stages as is presently the case.

3.5.1.2 *Paths enumeration*

The first step taken by the paths algebra algorithm is to *find all paths connecting all pairs of nodes in the SN*. It may seem that this is a too expensive procedure. This is true for highly connected networks but it is not true for small and sparse networks, what is usually the case. It is a procedure to be done only once and the computing time can be neglected as the procedure can be performed offline (before the embedding process starts). In addition, the procedure pays off in terms of the quality of solution. However, as it will be shown later, the processing time can be significantly reduced by limiting the number of paths to be discovered.

3.5.1.3 *VN requests ordering*

In existing VNE approaches, mapping optimization must be performed assigning the greatest number of virtual networks requests instead of the greatest number of virtual links. This means that each VN request is considered as a unit independently of how many virtual links it is made of.

It is important to realize that virtual requests consume physical resources, i.e., CPU processing power and BW. Such physical resources are finite and represent the limit of VN assignments. The way they are allocated affects the number of VN requests that can be satisfied. There are different ways to allocate the virtual requests:

i) allocate them in the order they arrive as if it were a First In First Out (FIFO) procedure;

ii) allocate them in a non-decreasing, or non-increasing, BW order;

iii) allocate them in a non-decreasing, or non-increasing, CPU order;

iv) allocate them in a non-decreasing, or non-increasing, (BW+CPU) order.

Suppose, for example, a substrate network that has a total BW of 10 and there are virtual requests of BW = 1, 2, 3, 4, 5. In principle, using the non-decreasing order, 1, 2, 3 and 4 can be allocated. On the contrary, in a non-increasing order only 5, 4 and 1 can be allocated. So, the order in which the VN requests are processed may affect the final result. *A priori* it is not possible to say what is the best policy. If, for example, the number of satisfied VNRs is the optimization criterion, then the non-decreasing order allocation is better. However, if the total amount of assigned BW represents revenue, both policies are equivalent.

Within a virtual request, there exists also the choice of ordering how the virtual links should be allocated.

It is out of the scope of this work a deep investigation of the effect of the VN requests and VN links ordering on the quality of the obtained results. Here, a FIFO ordering that emulates an online operation and non-increasing (BW+CPU) for off-line operations have been chosen to order the VN requests. This will decrease the chance to have to perform backtrack: i.e., to re-assign a previous virtual link in order to accommodate another one.

As far as the VN links ordering is concerned no specific ordering has been done and the VNR matrix is read by rows.

3.5.1.4 *VN links assignment*

Having performed the aforementioned preparatory steps, a virtual link is picked up to be assigned to the SN. Any assignment will con-

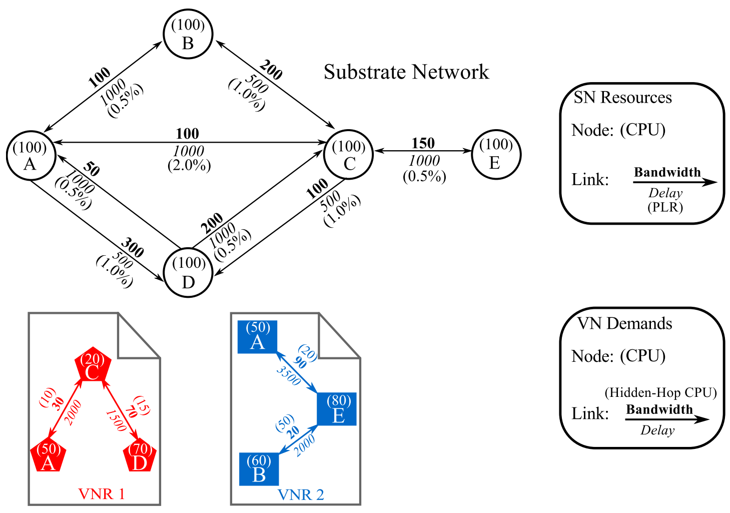


Figure 3.5: SN and VN requests used in the example

sume SN's resources and the packets will undergo the effects of the path it traverses: Delay, jitter etc. The SN's weight arcs are adjusted to the values they would have if the SN's resources were consumed according to the corresponding VN request. Using the paths algebra, the paths are ordered taking into account all specified metrics. The virtual link is assigned to the best path and the SN resources are updated according to the real values consumed by the chosen path. This procedure is repeated until all virtual links have been assigned to the SN.

In case a virtual assignment is not found because there aren't enough resources to accommodate the request, an analysis can be made to find out if a re-assignment (backtrack) of a previously assigned virtual link would free enough resources to accommodate the current virtual link. This analysis is left for future work. Currently, if there are no available resources to map a virtual link it is necessary to define if the VN request will be simply dropped out or if it will be accommodated with less resources than originally demanded. It is important to point out that the decision is related to the adopted business model and it is not affected by the paths algebra.

3.5.2 *Proof of concept examples*

In this section, two examples are introduced as proofs of concept. The first example deals with only physical resources: CPU and BW. It is a very simple example to show the basics of the paths algebra procedures. The second example introduces PLR and C/R to show how the paths algebra deals with non-linear metrics and non-physical resources.

3.5.2.1 *Example 1*

Consider the SN and VN requests represented in Figure 3.5. For SN, weights (in bold) over links indicate links' bandwidth and the nodes' weight (in parenthesis) are the nodes' capacity, link delay is shown in italics. For the VNs, the arcs weights indicate the requested bandwidth, and the requested virtual link delay refers to the maximum delay allowable when the virtual link is mapped, nodes' weight are the total CPU capacity requested for terminal nodes, and the arcs weights shown between parentheses are the total CPU capacity requested for hidden hops. Total CPU requested is to be understood as the capacity demanded by the whole VN request, independently of the number of times the node is used (as terminal or hidden) by the SN assignment.

In this example, the objectives to be fulfilled are:

- assign the largest number of virtual networks to the substrate network;
- maximize the global spare resources in SN, i.e., the links spare bandwidth added to the nodes spare CPU capacity. This objective can be fulfilled by mapping each virtual link using the shortest available path;
- minimize the path delay.

The metrics, syntheses and ordering relations to be used to fulfill the listed objectives are $\mathbf{M} = \{\text{Hops}, \text{Delay}\}$; $\mathbf{S} = \{\text{add}, \text{add}\}$; $\preceq_{ML} = \{\geqslant, \geqslant\}$.

Additionally, the links spare bandwidth and the nodes spare CPU capacity will be used as lower priority metrics to verify if the SN has enough resources to accommodate the VN requests. The corresponding metrics, syntheses and ordering relations are $\mathbf{M} = \{\text{BW}, \text{CPU}\}$, $\mathbf{S} = \{\text{min}, \text{min}\}$; $\preceq_{ML} = \{\leqslant, \leqslant\}$.

All the above criteria can be summarized as:

- $\mathbf{M} = \{\text{Hops}, \text{Delay}, \text{BW}, \text{CPU}\}$
- $\mathbf{F} = \mathbf{M}$
- $\mathbf{S} = \{\text{add}, \text{add}, \text{min}, \text{min}\}$
- $\preceq_{ML} = \{\geqslant, \geqslant, \leqslant, \leqslant\}$

It is important to note that there are different types of demands in the VN requests. Some resources lessen when virtual link requests are being assigned, and some others do not. Bandwidth and CPU demands are subtracted from the SN resources each time a virtual link is mapped, while delay remains unchanged after a virtual link mapping. Delay acts as a constraint: A SN path with a delay greater than the indicated one in the virtual link request cannot be used to map it.

The first step in solving the VNE problem is to enumerate all paths between pair of nodes of the SN. Such enumeration is completely independent of the VN requests. The pair of nodes of the SN are then used as terminal nodes by the VN requests. This sequence is also followed by the LADN algorithm. The obtained paths are:

- $(s, d) = (C, D)$: (C, D), (C, A, D), (C, B, A, D);
- $(s, d) = (A, C)$: (A, C), (A, B, C), (A, D, C);
- $(s, d) = (A, E)$: (A, C, E), (A, B, C, E), (A, D, C, E);
- $(s, d) = (B, E)$: (B, C, E), (B, A, C, E), (B, A, D, C, E).

Next, the VN requests have been sorted in an increasing order of total demand, and within a VN request the virtual links to be assigned to the SN have been sorted in a decreasing order. The total demand is calculated as the weighted sum of requested CPU and bandwidth; to simplify the example, equal CPU and bandwidth weights of value 1 are considered. CPU demand of hidden hops is not considered to calculate total demand. The final ordering is:

1. VN #1: Total demand = 240; links ordering: (C, D), (A, C);
2. VN #2: Total demand = 300; links ordering: (A, E), (B, E).

For each virtual network link request the digraph arcs' metrics are updated by subtracting the bandwidth and CPU capacity demands. The paths are then synthesized and lexically ordered. In this case, the whole weight-word was used. The synthesis results for VN #1 virtual link (C, D) are:

PATH	S1	S2	S3	S4
	#HOPS	DELAY	BW	CPU
(C, D)	1	500	30	30
(C, A, D)	2; 1	2000; 1000	30; 30	30; 80
(C, B, A, D)	3; 2; 1	2000; 1500; 500	30; 30; 130	30; 80; 80

The chosen path is (C, D): It is the shortest one; the delay is less than 1500; the spare BW and CPU are positive indicating that the SN has enough resources to accommodate the request. The assignment of the virtual link (A, C) does not present any difficulty and it is mapped to the SN's path (A, C). At the end of the VN #1 assignment the SN's resources are updated and the new values are:

- CPU capacity: (A) = 50; (C) = 80; (D) = 30
- links BW: (A, C) = 70; (C, D) = 30

The other values remain the same.

The synthesis results for VN #2 virtual link (A, E) are:

PATH	S1	S2	S3	S4
	#HOPS	DELAY	BW	CPU
(A, C, E)	2; 1	2000; 1000	−20; −20	0; 0
(A, B, C, E)	3; 2; 1	2500; 1500; 1000	10; 10; 10	0; 0; 0
(A, D, C, E)	3; 2; 1	2500; 1500; 500	60; 110; 210	0; 0; 0

It is seen that the shortest path does not have enough resources to fulfill the demand because the resulting value of S3 is negative. The chosen path is (A, D, C, E) because it shows a lower delay in the third letter of the weight word S2 than the path (A, B, C, E). It is important to realize that if only the first letter is used, both paths would be equivalent. After this assignment the SN's resources are updated and the new values are (note that CPU capacities are also updated subtracting hidden hops D and C demands):

- CPU capacity: (A) = 0; (C) = 60; (D) = 10; (E) = 20
- links BW: (A, D) = 210; (D, C) = 110; (C, E) = 60

The synthesis results for VN #2 virtual link (B, E) are:

PATH	S1	S2	S3	S4
	#HOPS	DELAY	BW	CPU
(B, C, E)	2; 1	1500; 500	40; 180	20; 40
(B, A, C, E)	3; 2; 1	3000; 2000; 1000	40; 50; 80	−5; −5; −5
(B, A, D, C, E)	4; 3; 2; 1	3500; 2500; 1500; 1000; 500	40; 80; 80; 80	−5; −5; −5; −5

The shortest path (B, C, E) is chosen because it fulfills all conditions. It is also worth noting that it is the only possible solution because the other alternatives don't have enough CPU capacity (S4 with a negative value).

All assignments were made without having to perform any backtrack.

3.5.2.2 *Example 2*

In this example, consider again the SN and VN requests represented in Figure 3.5. However, in this case, the following aspects are simultaneously considered:

- the use of non-linear metrics;
- a QoS-aware mapping;
- a business optimization criterion.

In Figure 3.5 each SN link is also characterized by its PLR, depicted by the value in parenthesis under the substrate links.

From the PLR, a function of metrics combination is applied and for each link the packet throughput, designated as pTHRU $= 1 - \text{PLR}$, is evaluated. The synthesis associated to pTHRU is mult() and the ordering relation is $\leqslant$.

It is important to realize that PLR is non-linear metrics but it introduces no difficulties to the paths algebra as it can be treated by one of the proposed syntheses.

Let's also consider that the demanded BW (in this case equal to BW(A-C) + BW(C-D) = 100) is a measure of revenue (R) and that the employed CPU processing power is a measure of cost (C).

Summarizing, the corresponding metrics and syntheses for pTHRU and C are $\mathbf{M} = \{\text{mult}, \text{add}\}$; $\mathbf{S} = \{\leqslant, \geqslant\}$.

Using only the first letter of weight word for pTHRU and C, the synthesis results for VN #1 are:

	LINK–(C-D)			
PATH	S5	S6	PLR	C/R
	pTHRU	C	(%)	
(C, D)	0.9999	90	1.0000	0.90
(C, A, D)	0.9702	105	2.9800	1.05
(C, B, A, D)	0.9752	120	2.4801	1.20
	LINK–(A-C)			
PATH	S5	S6	PLR	C/R
	pTHRU	C	(%)	
(A,C)	0.9800	70	2.0000	0.70
(A, B, C)	0.9850	80	1.4950	0.80
(A, D, C)	0.9850	80	1.4950	0.80

When only BW and CPU had been considered, for VN #1 virtual link $(C - D)$, the chosen path was (C, D): It is the shortest one; the delay is less than 1500; the spare BW and CPU are positive indicating that the SN has enough resources to accommodate the request. It is also the best one when PLR and C/R are taken into account. So, for the VN #1 virtual link $(C - D)$ the result does not change if either a QoS or a business criterion is employed.

It is also important to realize that lower PLR does not necessarily implies lower C/R. It is enough to observe the values obtained for paths (C, A, D) and (C, B, A, D).

Formerly, considering only the availability of physical resources, the assignment of the virtual link (A, C) was mapped to the SN's path (A, C). However, under QoS-aware PLR criterion it is the worst path. So, either (A, B, C) or (A, D, C) could be chosen. Under the PLR criterion they are equivalent. If PLR is not used as an ordering criterion but as a threshold, the path (A, C) could still be chosen if a 2.0 % loss rate is acceptable.

Table 3.5: Modeling of optimization strategies

Optimization criterion	Optimization metrics	Physical constraints	QoS thresholds	M F
Minimize cost	Hops	CPU, BW	-	$\mathbf{M} = \{Hops, BW, CPU\}$ $\mathbf{F} = \mathbf{M}$
Minimize cost under a maximum allowable delay	Hops	CPU, BW	Delay	$\mathbf{M}$ = {Hops,Delay,BW, CPU} $\mathbf{F} = \mathbf{M}$
Maximize the spare CPU	CPU	CPU, BW	-	$\mathbf{M} = \{CPU, BW\}$ $\mathbf{F} = \mathbf{M}$
Maximize the spare BW	BW	CPU, BW	-	$\mathbf{M} = \{BW, CPU\}$ $\mathbf{F} = \mathbf{M}$
Maximize the spare physical resources	CPU, BW	CPU, BW	-	$\mathbf{M} = \{BW, CPU\}$ $\mathbf{F} = \{CPU + BW\}$
Minimize cost and maximize throughput, under a maximum allowable delay	Hops, PLR	CPU, BW	Delay	$\mathbf{M}$ = {Hops,PLR,Delay, BW,CPU} pTHRU = 1 - PLR $\mathbf{F}$ = {Hops,pTHRU,Delay, BW,CPU}

If only the business criterion C/R is used the path (A, C) would be chosen. However, if a QoS-aware threshold of PLR equal to 1.5 % has to be achieved then the choice would be between paths (A, B, C) and (A, D, C).

The example is not developed further, but similar conclusions can also be derived for the VN #2.

Examples show how powerful and flexible the proposed approach is. It allows the InP to use different policies to achieve the SP goals

In this example, more important than the result itself is to realize how powerful and flexible the paths algebra approach is. It allows the InP to exercise different policies to achieve the SP goals. In this way, both technical and financial performances can be optimized. The access to the performance of all eligible paths also implies that recovery strategies can be implemented.

All the above observations were possible by just inspecting the results of the paths algebra syntheses. However, the paths algebra enables to explore different optimization strategies by just changing the metrics **M** and the function of combined metrics **F**.

Table 3.5 shows how some different strategies can be modeled using the paths algebra. In this table, the optimization metrics are primarily used to order the enumerated paths. The physical constraints indicate if the achieved mapping is feasible or not. A feasible mapping requires positive physical constraints. A QoS threshold is a lower priority metrics and represents a bound on the value of the corresponding metrics.

In Table 3.5 any strategy that minimizes cost (a business objective) also minimizes the cost over revenue because the demanded VNR's BW is constant and is a measure of the achievable revenue. For each strategy the number of mapped VNRs may be different and the mapped revenue as well. So, different strategies may be exploited to optimize business objectives and achieve different levels of QoS. The listed strategies also mix linear and non-linear metrics that are treated simultaneously by the paths algebra.

As mentioned before, the LADN is made of four stages: SEARCHPATH, SORTPATH, EVALUATEPATH and ORDERPATH. The SEARCHPATH and SORTHPATH are metrics independent. The paths enumeration depends only the SN topology. So, these two routines can be run offline and the results stored to be exploited by different optimization strategies. Different strategies imply running EVALUATEPATH and ORDERPATH, which can be done either online or offline.

3.5.3 *Preliminary conclusions*

The paths algebra-based strategy transforms the link mapping stage of the VNE problem into the simpler problem of mapping network virtualization policies into an appropriate set of metrics or combination of metrics. With paths algebra, the addition of new metrics will not go in detriment of the problem complexity. In addition, the order in which the metrics are considered introduces newer degrees of freedom. The paths algebra is a flexible and powerful tool to explore the design space to find the best strategy according to a specific business model.

The inclusion of paths algebra in the VLiM solution allows to introduce new degrees of freedom without causing a detriment of the problem complexity

It is important to realize that even if the discovery of all eligible paths is computing expensive, it has to be done only once, it can be performed off-line and is metrics independent.

3.6 EVALUATION SCENARIOS AND EXPERIMENTAL RESULTS

This section shows how to deal with the computational complexity of the multi-constraint routing algorithm, introduces the simulation scenarios and presents the experimental results.

3.6.1 *Computational complexity*

The paths algebra-based multi-constraint routing algorithm presented in [150] is made of four routines:

1. SEARCHPATH: Discovers all paths between all pair of nodes of a given network.
2. SORTPATH: Selects only the elementary and simple paths.[4]
3. EVALUATEPATH: Evaluates the syntheses along the paths.
4. ORDERPATH: Orders the paths according to the metrics and priorities of a given problem.

When the graph is fully connected with n nodes and $n \times (n-1)$ edges, the maximum number of simple and elementary paths can be obtained by induction and is given by $Np = 2n! + n!/2! + n!/3! + \cdots + n!/(n-2)!$ that can be written as $\sum_{i=0}^{n-2} C_{n,i} \times (n-i)!, n > 3$. Substituting the combination of (n) taken i at a time by $(n)!/[i! \times (n-i)!]$, $Np = (n)! \sum_{i=0}^{n-2} 1/i!$ is obtained. Therefore, for this case, the algorithm's execution time is $O(n^n)$, given that $\sum_{i=0}^{n-2} 1/i! < \sum_{i=0}^{\infty} 1/i! = e$, for $n \to \infty$, $e \times n! < e \times n^n \Rightarrow n! < n^n$, in which according to the usual definition of $O(g(n))$, for $n \geqslant n_0 = 2 : 0 \leqslant f(n) =! \leqslant cg(n) = n^n$.

This means that for fully connected networks, the paths algebra-based multi-constraint routing algorithm is computationally explosive. In practice, it has been verified that for real networks with a small number of nodes and sparse connectivity, paths algebra performs well. However, it is not guaranteed that it would perform adequately for any network but, as it will be shown in the sequence, the number of operations performed by the algorithm can be controlled without sacrificing the quality of results. By doing so, the algorithm can be applied to networks with hundreds of links.

In order to estimate the algorithm's performance, 100 random uniform and transit stub topologies with 10, 20, 50 and 100 nodes were generated, and total number of paths was evaluated. The results have shown that the number of paths do increase as n^n. Thus, it is necessary to introduce some modification in order to decrease the algorithm's complexity.

Consider a given network specified by its adjacency matrix A where the element $a[i,j]$ is equal to 1 when there is a link connecting the source node i to the destination node j. If $A^k = A \times A \times \cdots \times A$, k times, where $a_k[i,j] \neq 0$ indicates that there is at least a path of length k connecting the source node i to the destination node j, it is then possible, for each network, to evaluate the minimum length

4 Elementary and simple paths are those paths that traverse each graph's node and edge only once.

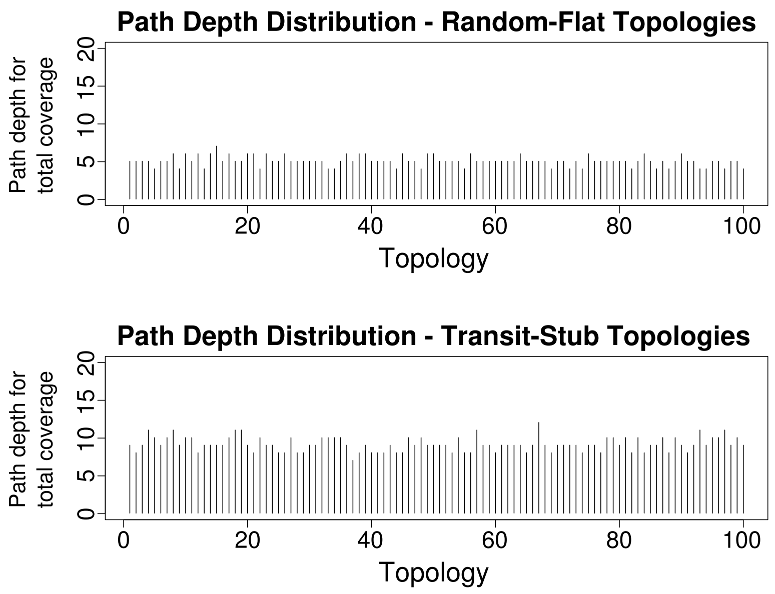


Figure 3.6: Path depth distribution for 50 nodes in random flat and transit stub topologies

Table 3.6: Minimum path depth for 20, 50 and 100 nodes topologies

Number of nodes	Transit stub topologies	Random flat topologies
20	10	12
50	12	7
100	13	5

of paths that guarantee full connectivity, i.e., that there exists at least one path between any pair of source and destination nodes [156].

Figure 3.6 shows the path depth which is the path length necessary to obtain full connectivity for 100 topologies both for 50 nodes random flat and transit stub cases. For n nodes topology the largest path length is $(n-1)$. So, for the results shown in Figure 3.6 the maximum length is $(n-1) = 49$. It can be seen that in both cases the necessary path length for obtaining 100% coverage is much less than the maximum value. Another way to appreciate this finding is observing Figure 3.7 that shows the connectivity increase as function of the path length. These results were obtained by evaluating the average coverage for each path length over all topologies. It can be seen that for both types of topologies, 100% connectivity is achieved for path lengths quite smaller than the maximum value. As the number of employed topologies is reasonably large, it is possible to state that the obtained results are representative for classes of topologies studied here.

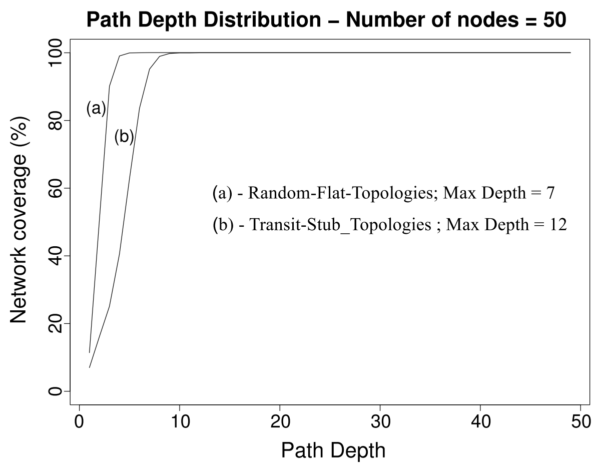


Figure 3.7: Connectivity increase as function of the path depth for 50 nodes random flat and transit stub topologies

Table 3.7: Comparison between the average maximum of paths with the number of paths up to the length equal to 6 links

Random flat topologies		
Number of nodes	Average Maximum number of paths	Number of paths up to length = 6
20	3,162	692
50	117,489	6,607
100	1,995,262	41,687
Transit stub topologies		
Number of nodes	Average Maximum number of paths	Number of paths up to length = 6
20	1,738	447
50	16,982	2,042
100	66,069	3,467

Table 3.6 summarizes the results for topologies with 20, 50 and 100 nodes.

Table 3.7 compares the average maximum number of paths with the number of paths up to the length equal to 6 links for random flat and transit stub topologies with 20, 50 and 100 nodes. It can be seen that limiting the maximum length of the paths produces an important reduction on the number of paths to be discovered and processed. This modification has been introduced into the multi-constraint routing algorithms, providing a control over the number of operations performed by the algorithm and enabling the possibility to work with large networks. It is important to realize that the path length limit introduced in the algorithms is not an arbitrary value. In fact, it is determined as function of the SN that is going to be processed. It is just a matter of finding the exponent k such that $\bigcup_{i=0}^{i=k} A^i = \mathbf{1}$, where $\mathbf{1}$ represents a matrix where all elements are equal to 1. On the other hand, it may be argued that limiting the search space may impact the possibility of finding a mapping for a VN request. However, the results obtained for the VNE mapping show that, if there is any impact, it seems to be of negligible practical importance.

3.6.2 *Simulations environments*

The simulation environment used for VNE is the Algorithms for Embedding of Virtual Networks [FBD+11] (see Chapter 6).

As already mentioned, the algorithm used to implement paths algebra, called LADN [150], was developed as a simulation tool for testing multi-constraint hop-by-hop routing algorithms in which the optimization strategy is user-defined. Considering that in hop-by-hop routing algorithms each node decides about the best path independently of any other nodes, loops may occur. The algorithm eliminates loops occurrences by imposing the property of coherence on a global basis to the next hop choice made by each node.

The LADN algorithm was modified to solve the VNE problem, i.e., to map virtual links to physical paths in a substrate network. Node mapping is performed by the algorithms implemented in ALEVIN and is out of the scope of this work.

Figure 3.8 shows the diagram of the VNE solution implementation using the association of paths algebra LADN and the ALEVIN environment. The ALEVIN environment is in charge of generating the VNRs and of performing VNoM. The other tasks are performed by the LADN. Presently the environments communicate by files exchange, what impacts the processing time. A complete integration will be implemented in a near future.

According to Figure 3.8, block 1 starts the LADN environment and block 2 enumerates the eligible paths and pre-orders them according to the number of hops metrics. Block 2 also implements the maximum length of the searched paths in order to limit the processing time.

Block 3 defines if mapping will be done on-line or off-line. In the case of on-line mapping, the VNRs must be processed in the order of their arrival as if they were stored in a FIFO queue. In the case of off-line mapping, the VNRs are ordered according to some policy, e.g. Most Consuming First (MCF), Least Consuming First (LCF), etc. In this work, the MCF policy has been selected.

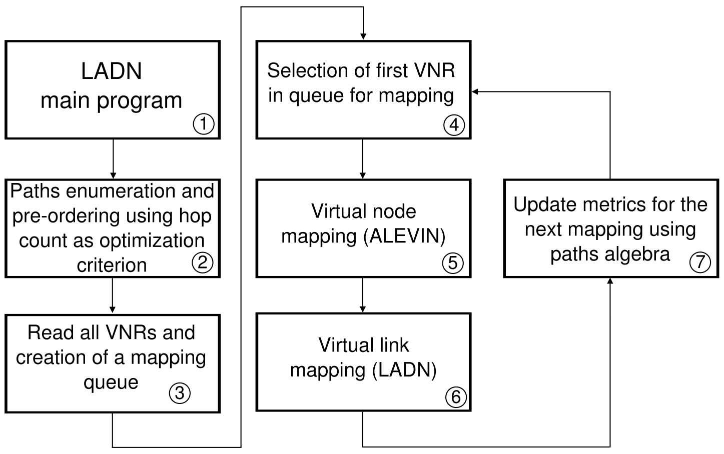

LADN main program ①
Paths enumeration and pre-ordering using hop count as optimization criterion ②
Read all VNRs and creation of a mapping queue ③
Selection of first VNR in queue for mapping ④
Virtual node mapping (ALEVIN) ⑤
Virtual link mapping (LADN) ⑥
Update metrics for the next mapping using paths algebra ⑦

Figure 3.8: LADN and ALEVIN integration for VNE

Block 4 selects the first VNR waiting to be mapped and sends it to the ALEVIN along the current status of the SN, i.e., the available resources in the SN.

Block 5, implemented in ALEVIN, produces a virtual nodes to physical nodes mapping and sends the results back to the LADN environment.

Block 6, maps the virtual links to the physical links according to the set of metrics and priorities selected by the user. If the mapping is successful the algorithm proceeds to block 7 to evaluate the new available resources in the SN and to update the metrics matrices. If there are still pending VNRs, the algorithm goes back to block 4 and repeats the procedure. If there are not pending VNRs, the algorithm stops and stores the results. If the mapping is not successful, the algorithm records the failure for further analysis and tests about pending VNRs.

3.6.3 *Simulation methodology*

In this section, a methodology to create load-targeted scenarios is introduced.

3.6.3.1 *Scenario Creation Methodology*

The aim of this methodology is to study the embeddings in a SN that is subject of different load demands in the virtual networks, consequently, a large set of offline evaluation scenarios with different substrate as well as different virtual networks is created, demanding a given average resource load ρ.

TOPOLOGY CREATION Substrate and virtual network topologies are generated using the *Waxman* algorithm [157]. The coordinates of the nodes inside the topologies are uniformly distributed.

Waxman generator takes two parameters α and β and determines the probability of an edge between every pair of vertices u and v as follows:

$$P(u,v) = \alpha \cdot e^{\frac{-d(u,v)}{\beta \cdot Le}} \tag{3.1}$$

where $0 < \alpha,\ \beta \leqslant 1$, d is the distance between node u and node v, and Le is the maximum distance between any two nodes. As α parameter grows, the probability of edges between any nodes in the graph is increased. As β grows, there will be a larger ratio of long edges to short edges. This topology generation is performed for the SN as well as for all k^{max} VNRs.

RESOURCE AND DEMAND DEPLOYMENT After substrate and virtual network topologies are created, resources and demands are created in two steps.

In first place, substrate network is filled with resources by uniformly distributing each node resource X in a given interval $(0, cap_X^{max}]$ on every SN node as well as each link resource Y in a given interval $(0, cap_Y^{max}]$ on every SN link. This uniform distribution has been chosen to provide comparability with previous approaches in the literature [41, 52, 45, 51].

In second place, the demands are generated in all virtual networks in order to achieve a certain average load of every resource. The creation of distributions for the node and link demands that fulfill these requirements can be calculated using the theorem of Little [158].

As the number of nodes is fixed in the Waxman topology generation, it is easy to calculate the mean resource on a SN node

$$E[cap_X] = \frac{0 + cap_X^{max}}{2} = \frac{cap_X^{max}}{2} \tag{3.2}$$

as well as the mean demand on a virtual node for a given overall load ρ as

$$E[dem_X] = \rho \cdot E[cap_X] \cdot \frac{E[|N|]}{E[|N^k|] \cdot k^{max}} \tag{3.3}$$

If a uniform demand distribution within $(0, dem_X^{max}]$ is considered, this results in a maximum demand of resource

$$dem_X^{max} = 2 \cdot E[dem_X] = \rho \cdot \frac{E[|N|]}{E[|N^k|] \cdot k^{max}} \cdot cap_X^{max} \tag{3.4}$$

It does not make sense to choose $dem_X^{max} > cap_X^{max}$, as it would lead to a situation where demands are generated, which can not be satisfied by any node in the network. Hence, $\rho \leqslant \frac{E[|N^k|] \cdot k^{max}}{E[|N|]}$ has to be chosen.

As the Waxman topology generation is probabilistic with regard to link creation, the number of links in a Waxman-network is not fixed, they can only be given by probabilities. To achieve the targeted load ρ of link resource as well, the average number of edges in a Waxman-network is considered. This is done by calculating the mean probability $E[P(u,v)]$ of creating an edge between any two nodes which depends on the basic parameters of the Waxman-network, i.e. the area, the number of nodes, and the distance metric used. Thus, an average number of edges $E[|L|]$ in a directed graph is yielded as:

$$E[|L|] = E[P(u,v)] \cdot |N| \cdot (|N| - 1) \tag{3.5}$$

and in the virtual network it is:

$$E[|L^k|] = E[P(u,v)] \cdot |N^k| \cdot (|N^k| - 1) \tag{3.6}$$

That way, the average link resource in a SN can be calculated as:

$$E[cap_Y] = \frac{0 + cap_Y^{max}}{2} = \frac{cap_Y^{max}}{2} \tag{3.7}$$

and the average link demand is given by:

$$E[dem_Y] = \rho \cdot E[cap_Y] \cdot \frac{E[|L|]}{E[|L^k|] \cdot k^{max}} \tag{3.8}$$

Again, it is necessary to be sure that $E[dem_Y] \leqslant E[cap_Y]$. Following Equation 3.8, and taking into account that $0 \leqslant \rho \leqslant 1$, this only holds when $E[|L|] < E[|L^k|] \cdot k^{max}$. To that end, substituting in Equation 3.5 and Equation 3.6, yields:

$$|N|^2 - |N| < (|N^k|^2 - |N^k|) \cdot k^{max} \tag{3.9}$$

that only holds if:

$$k^{max} \cdot E[|N^k|]^2 > |N|^2 \tag{3.10}$$

If Equation 3.10 is violated, the approximation provided above will achieve a higher load value than the targeted load ρ for the link resources and even result in $dem_Y^{max} > cap_Y^{max}$ which can never be fulfilled.

3.6.3.2 *Scenarios*

As previously mentioned, the Waxman topology generation is used. For evaluation, $\alpha = \beta = 0.5$ and the coordinates of the nodes are uniformly distributed in an 1×1 square area. Empirical studies for these parameters have provided an average distance of any two nodes $E[d] \approx 0.5$ and a maximum distance of $L = \sqrt{2}$. Thus, according to Equation 3.1 the average probability for creating a link between any two nodes is $E[p] \approx 1/4$.

Table 3.8: Parameters chosen for the simulation scenarios

Parameter description	Chosen values
Number of substrate nodes ($\|N\|$)	20
Number of VNRs (k)	10
Number of virtual nodes per VNR ($\|N^k\|$)	10
Range of loads (ρ)	{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8}

In this work, CPU cycles, denoted by R_{CPU}, and bandwidth, denoted by R_{BW}, are considered as node and link resources, respectively, in the substrate network. These values have been uniformly distributed chosen the upper bounds $R_{CPU}^{max} = 100$ and $R_{BW}^{max} = 100$.

Table 3.8 shows the parameters used in the simulations. The values chosen for the number of substrate nodes and the number of virtual nodes per virtual network seem to be a good trade-off between runtime of some algorithms and realistic scenarios. The loads' range from 0.2 until 0.8 also covers situations from lightly loaded to heavily loaded situation.

As the Waxman topology generation is probabilistic, 20 runs for each value of load are performed.

Paths algebra-based single-path VLiM produces similar results compared with existing approach but, in turn, provides the capability to include an unlimited number of linear or non-linear constraints

3.6.4 *Experimental results*

The proposed approach introduces a new strategy to face the network embedding optimization, permitting linear and non-linear parameters. According to the current state of the art (see Chapter 2), only results with linear parameters are published yet. Therefore, this section analyzes and compares results with the relevant linear parameters based strategies. These comparisons show that the proposed procedure in this work is more powerful, since it produces similar results and, in addition, provides the capacity to include non-linear parameters. Apart from the comparison with existing methods, an example with non-linear parameters is depicted in Section 3.5.2.2.

Optimal cost-based solutions for the VNE problem have been provided in the literature. In [51], VNE is formulated as a MIP and solved using exact algorithms [48]. However, MIPs are not scalable and the time needed to solve them in medium to large networks is not affordable. Besides, unlike the approach proposed here, the MIP solution in [51] provides multi-path VLiM. To make this cost-optimal solution comparable with single-path approaches, additional binary variables indicating that just one path can carry the virtual link demand should be added to the MIP model, resulting in further complexity that would increase the time to solve the MIP even for very small substrate and virtual networks.

Table 3.9: Algorithms used in the simulations

Algorithm's name	Algorithm's description
GARSP [41]	Greedy Available Resources with k-Shortest Paths
PathsAlgebra policy = M1	Greedy available resources for node mapping and paths algebra for link mapping using **M** = (Hops, BW, CPU), **S** = (add, min, min) and ordering relations $(\geqslant, \leqslant, \leqslant)$.
PathsAlgebra policy = M2	Greedy available resources for node mapping and paths algebra for link mapping using **M** = (Hops, Total BW), **S** = (add, add) and ordering relations $(\geqslant, \leqslant)$.

Hops means the number of hops in the path

BW means the links available bandwidth after mapping

Total BW is the total SN available BW after mapping

Consequently, to evaluate the paths algebra-based approach, it is compared against a well-known single-path VNE heuristic [41]. Simulations have been performed for all scenarios described in Section 3.6.3.2. Paths algebra-based approach considers just the single path-based VLiM stage of the VNE. As a consequence, to compare it against other VNE solutions, the VNoM made by them is reused and the focus is conducted to VLiM stage. As most of the current VNE approaches performing single path-based virtual link mapping use K shortest path approach (see Section 3.3), the performance of the paths algebra approach is evaluated against the GARSP algorithm (also called baseline algorithm in [41]) available in the ALEVIN environment.

Table 3.9 summarizes the characteristics of the algorithms used in the simulations. It is important to note that all algorithms produce single path solutions. Multi-path investigation is left for future work.

Note that two different policies for the Paths Algebra are being employed. The difference between them is the set **M** of metrics and the appropriate syntheses and ordering relations for each set. The maximum path length is evaluated according to the SN that is being processed.

Among the several parameters that can be used to compare the results obtained by the different algorithms, the following have been chosen:

- VNR acceptance ratio: The percentage of virtual networks successfully mapped;
- Mapped revenue ratio: The percentage of mapped revenue over the total revenue that could be mapped;

Table 3.10: Summary of the best results obtained for different parameters and policies

Parameter	M1 + M2	GARSP
Accepted VNRs	155	5
Ratio mapped revenue	144	16
C/R relationship	133	27

- Cost/Revenue (C/R) relationship: The lower this relationship is, the better is the result.

Out of 160 simulated cases, the paths algebra produced the best result 155 times, i.e., in 96.87% of the cases.

The discussion of the results shown in this section and the exploitation of numerical results is made in Section 3.6.5.

3.6.5 *Results compared to cost-optimal*

The total number of times that each algorithm has been exercised is given by $N \times k \times |\rho| = 20 \times 10 \times 8 = 1,600$, and as 3 different policies are being compared, the total number of VNR assignment attempts is 4,800.

The dimension of the sample space does not allow to derive quantitative conclusions with statistical significance but it is large enough to provide qualitative comparisons and obtain an understanding of the observed behaviors.

Table 3.10 summarizes the best results obtained for different parameters and policies.

It is clear that the paths algebra policies performance almost always exceeds that of the other strategy. It is not possible at this stage to indicate a specific reason why the paths algebra policies is not always the best. Looking at the whole set of results, it is possible to observe that there isn't any common pattern among the cases that the paths algebra has not produced the best result. Such cases need to be further investigated to identify the specific reason underlying a degraded performance of paths algebra. Possible explanations for such cases may be:

- different policies may employ different node mappings that may affect link mapping. If this is the case, a better coordination between node and link mapping has to be developed;
- the non-paths algebra strategies may achieve a better result by using longer paths. If this is the case an attempt may be made by relaxing the limit of paths length and trying to find a mapping;

- the order the VNRs are processed may impact the final result. A backtrack procedure may be needed.

It is also important to realize that there are cases in which M1 produces the best result and cases in which M2 produces the best result. The right interpretation of this observation is that the paths algebra is powerful and flexible to allow the identification of the adequate policy to optimize the performance of a given parameter. Moreover, it is not necessary to choose a policy beforehand. Several policies can be simultaneously implemented without any significant computational time increase. Remember that the enumeration of eligible paths is computationally explosive. However, the number os operations is under control by the limitation imposed on the paths length and the enumeration of eligible paths may be performed off-line and is independent of any optimization strategy.

Figure 3.9 shows part of (only 10 out of 20 runs) the obtained results of accepted VNRs for the M1 policy. The histogram like graphs represent the results of each evaluated scenario. In all cases a trend of a decrease can be observed in the number of accepted VNRs as the load increases. Some scenarios don't present a monotonic behavior. Such instabilities have to be investigated in a case by case basis and it is most likely that the explanations proposed above concerning the reasons why the paths algebra is not always the best one may apply.

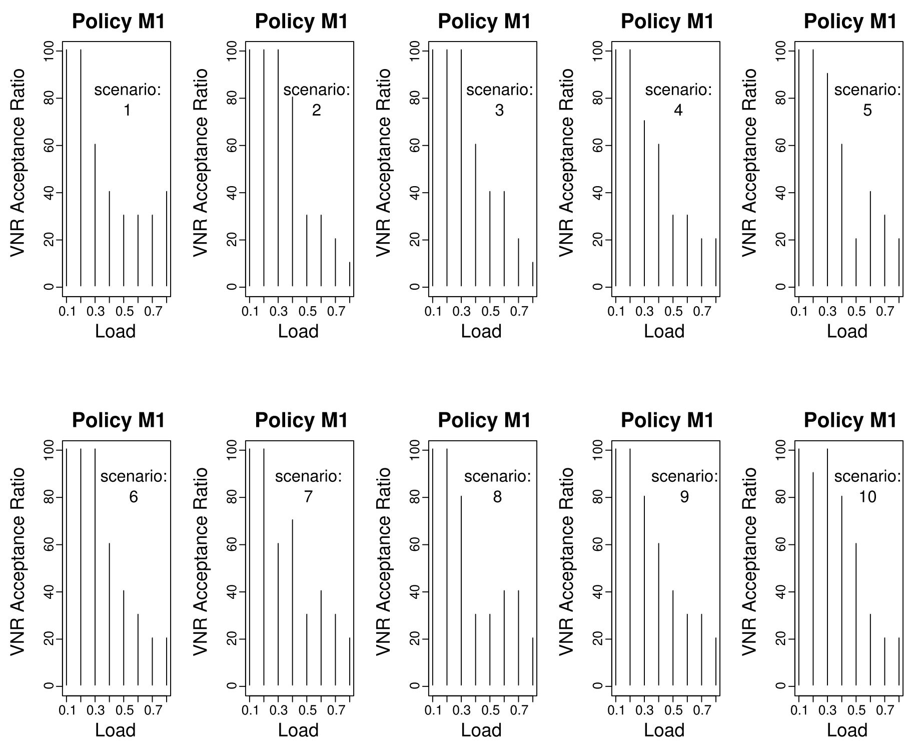
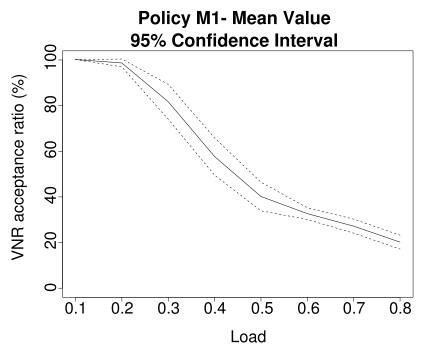
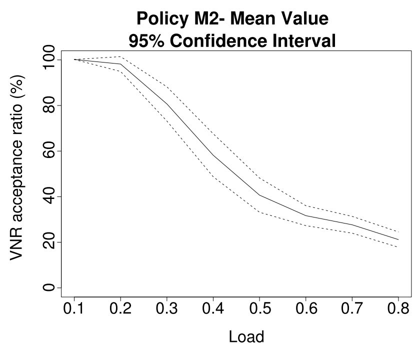


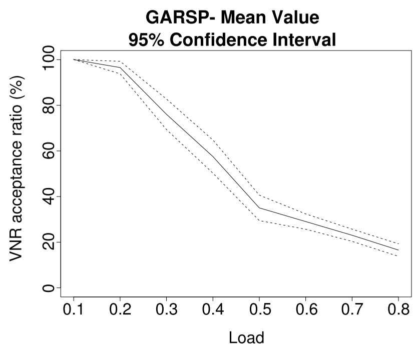
Figure 3.9: VNR acceptance ratio for M1 policy



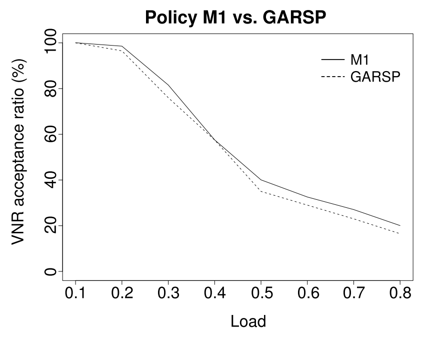
(a) M1



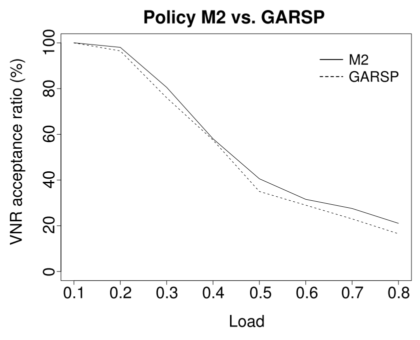
(b) M2



(c) GARSP



(d) M1 x GARSP



(e) M2 x GARSP

Figure 3.10: VNR acceptance ratio mean values behavior for M1, M2 and GARSP policies and comparisons: M1 vs. GARSP; M2 vs. GARSP

Figure 3.10 summarizes the results obtained for the VNR acceptance ratio. In top three figures, Figures (a), (b) and (c), are shown the mean values for all policies along the 95% confidence interval. The two bottom figures, Figures (d) and (e), show the comparison between the paths algebra policies (M1 and M2) and the GARSP strategy. The comparison is made based on the mean values and it is clearly seen that the paths algebra policies outperforms the GARSP strategy. In addition all policies present the same behavior and the 95% confidence interval is similar for all of them.

Table 3.11: Comparison of two possible mappings

	MAPPING 1	MAPPING 2
Node mapping	A → 1; B → 2	A → 4; B → 5
Path	1;2	4; 3; 5
Cost	CPU(1) + CPU(2) + BW(1-2)	CPU(4) + CPU(3) + CPU(5) +
		BW(4-3) + BW(3-5)
	= 20 + 30 + 100	= 20 + 0 + 30 + 100 + 100
	= 150	= 250
Revenue	CPU(A) + CPU(B) + BW(A-B)	CPU(A) + CPU(B) + BW(A-B)
	= 150	= 150
C/R	150/150 = 1	250/150 = 1.67

Note: CPU(3) = 0 because the cost of hidden hops is not being considered

The observed behavior can be understood considering that a SN offers a certain amount of resources to be used by the VNE procedure. The total resource that will be actually used is a fraction of the total CPU and BW available. When the load ρ offered to the SN is low, there are enough resources to absorb the demand and all VNRs are successfully mapped. Increasing the load offered to the SN a value is reached, ρ_0, that all VNRs are still successfully mapped but all SN resources are used. Above ρ_0, the excess load cannot be mapped and the VNR will be dropped. So, for a load $\rho > \rho_0$, the percentage of accepted VNRs will roughly follow a $\frac{\rho_0}{\rho}$ trend.

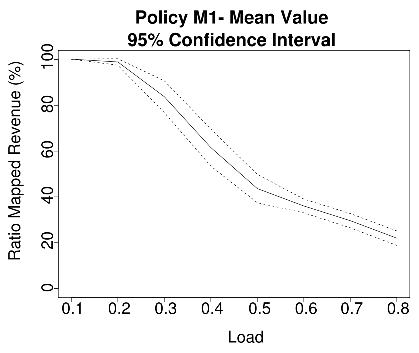
As the revenue is proportional to the number of accepted VNRs, the ratio mapped revenue has the same behavior as the percentage of accepted VNRs as it can be seen in Figure 3.11.

In order to understand the obtained results for C/R it is necessary to consider how Cost and Revenue are evaluated. Let's assume that there is a VNR characterized by:

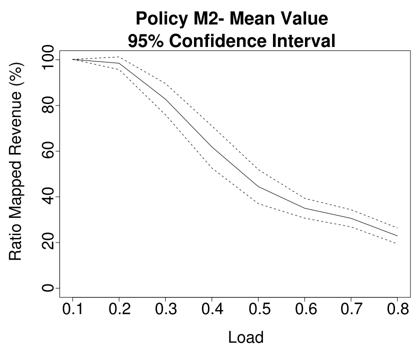
- virtual source node: A
- virtual destination node: B
- requested bandwidth: BW(A-B) = 100
- requested CPU: CPU(A) = 20; CPU(B) = 30

For this request the associated revenue is: R = CPU(A) + BW(A-B) + CPU(B) = 20 + 100 + 30 = 150.

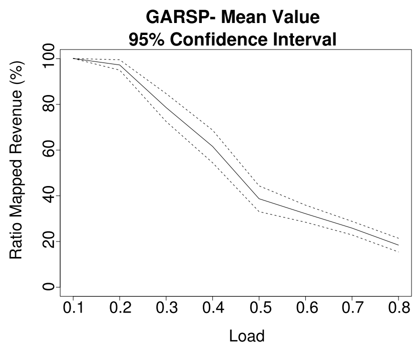
Let's assume that the paths algebra algorithm finds two possible mappings summarized in Table 3.11.



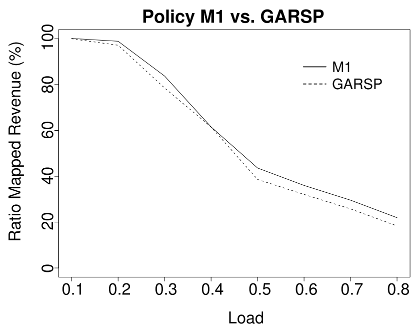
(a) M1



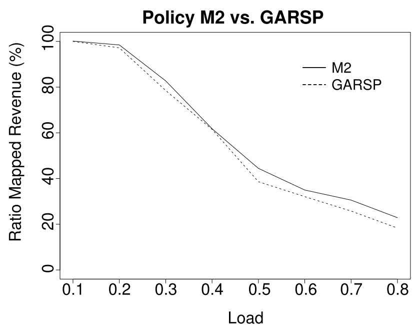
(b) M2



(c) GARSP



(d) M1 vs. GARSP



(e) M2 x GARSP

Figure 3.11: Ratio mapped revenue mean values behavior for M1, M2 and GARSP policies and comparisons: M1 vs. GARSP; M2 vs. GARSP

It can be readily seen that the best solution is achieved when a virtual link is directly mapped on a single substrate link. In this case, $\frac{C}{R} = 1.0$. Each time a hidden hop has to be used, the cost increases by the cost of the CPU plus the cost of the bandwidth. As in these examples the cost of the hidden hops' CPU is taken equal to zero then for a mapping that uses k hidden hops the cost is given by:

$$C = CPU(A) + CPU(B) + (1 + k)BW.$$

As the revenue for a given VNR is a fixed number given by $R = CPU(A) + CPU(B) + BW$, then C/R is given by:

$$C/R = 1 + k\frac{BW}{R},$$

i.e., C/R increases linearly as the number of hidden nodes increases.

All strategies evaluated in this section take the minimization of cost as a priority. This means that all strategies try to map a VNR using a shortest available path and the shortest paths will be consumed first. When the load increases, in order to satisfy the VNRs, longer paths have to be used and the cost will increase. It is then expected that C/R will show either a constant behavior or slowly increasing as the load increases.

Results show that the best strategy for all loads is always a paths algebra strategy. It allows to use different policies (combination of constraints) allowing a detailed exploration of the solutions space to identify the most suitable criteria

Figure 3.12 compares the C/R relationship performance of the adopted strategies. The obtained results are in agreement with what is expected and the paths algebra policies outperform the GASP strategy.

According to the whole set of results, it is important to realize that depending on the performance criterion that is chosen, the best strategy may not be the same for all loads but it is always a paths algebra strategy. The meaning of this result is twofold:

1. Even if the numerical results do not vary significantly, it is important to emphasize that, according to the best of our knowledge, the paths algebra sets the current limit of achievable performance.

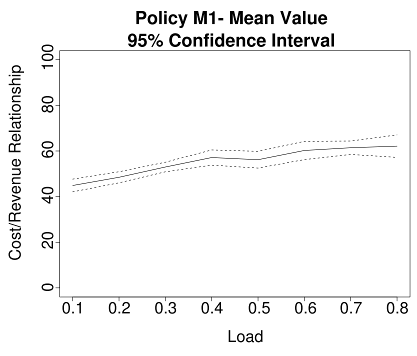
2. Different paths algebra strategies can be exploited to achieve the best performance according to the chosen criterion.

   In other words it may be stated that the paths algebra is a powerful and flexible strategy.

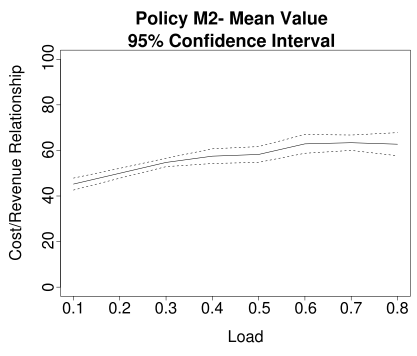
Such flexibility allows a detailed exploration of the solutions space and the identification of criteria that best suit the objectives of the InP, taking into account technical requirements to guarantee a given QoS. It is important to note that the exploration of the solutions space is feasible as far as processing time is concerned because the paths enumeration routine (that is the most time consuming) does not have to be processed. Paths enumeration can be made off-line and needs to be processed again only when there is a change in the physical network.

In this work, the number of hops, spare bandwidth, spare CPU and total spare bandwidth have been chosen as metrics. Spare bandwidth and spare CPU are synthesized using the min() function, that is a non-linear function. Other non-linear metrics as packet loss ratio and reliability could be also employed.

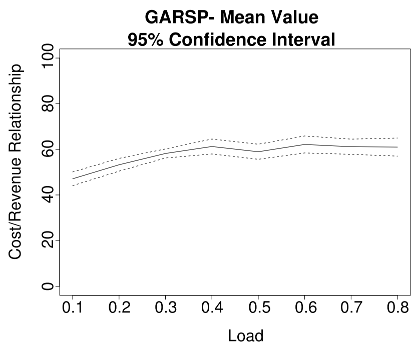
In summary, it is possible to state that:



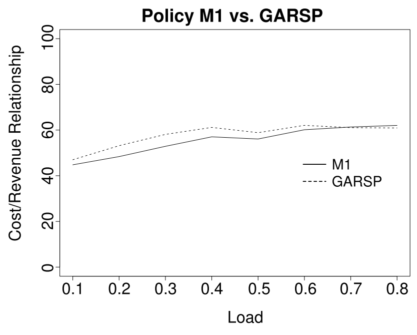
(a) M1



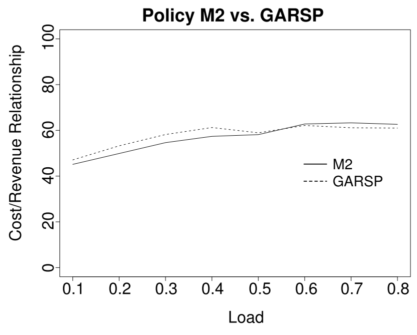
(b) M2



(c) GARSP



(d) M1 x GARSP



(e) M2 x GARSP

Figure 3.12: Cost revenue (C/R) mean values behavior for M1, M2 and GARSP policies and comparisons: M1 vs. GARSP; M2 vs. GARSP

the paths algebra does not impose any restriction on the type and number of metrics that are used. Right on the contrary, linear and non-linear metrics can be used together; any combination of metrics can also be employed.

The paths algebra empowers either the InP or the SP to look for the best strategy that may satisfy the needs of end customer and at the same time maximize the financial performance of the system. Strategies can be changed whenever necessary to adapt to new demands conditions to keep the system operation close to the best achievable performance.

In this chapter, the results have been only shown in terms of the VNR acceptance ratio, mapped revenue ratio and Cost/Revenue (C/R) relationship. The complete set of results obviously records the mapped paths. However, the paths algebra orders all eligible paths and, therefore, not only the best solution is available but all possible solutions can be accessed. This is a very valuable feature to implement recovery techniques in case of failure. Not only an alternate path for fast recovery is known by the system but also any performance degradation can be anticipated and dealt with.

The feature that all eligible paths are enumerated and ordered also shows the potential of the technique to implement multipath solutions. Different multipath strategies may be envisaged, as for example:

- always look for a multipath solution to promote load balancing;
- look for a multipath solution immediately after a single path solution has not been found for a given VNR;
- look for a multipath solution after the completion of a single path phase in order to map the set of discarded VNRs.

Multipath strategies are out of the scope of this work and will be investigated in the near future.

3.7 CONCLUSION

In this chapter, a novel strategy to solve the link mapping stage of the VNE problem based on the paths algebra framework have been presented.

The paths algebra technique has been originally developed to solve the multi-constraint routing problem. It is a powerful technique that can work simultaneously with any number of linear and non-linear parameters permitting the optimization of technical and business related criteria. Besides, it provides great flexibility allowing to introduce any number of network constraints to the virtual link mapping stage and to provide the facility to organize the complete set of substrate paths between any pair of nodes in mono and multi-constraint environments.

Several parameters can be used to compare the results obtained by the different algorithms. This work has chosen to use: VNR acceptance ratio, Mapped revenue ratio and Cost/Revenue (C/R) relationship. It has been shown that for all offered loads the Paths Algebra strategies produce similar or better results when compared with shortest path-based strategies both with the same node mapping approach.

The process of finding all the possible paths between any pair of nodes shows an increasing complexity for big networks. To overcome this limitation, the computational complexity has been kept low for all practical purposes by limiting the maximum length of the paths to be considered. This procedure produces an important reduction on the number of paths to be discovered and processed. This modification has been introduced into the multi-constraint routing algorithms, providing a control over the number of operations performed by the algorithm and enabling the possibility to work with large networks. The results obtained for the VNE mapping show that, if there is any impact in the embedding, it is of negligible practical importance.

It is possible to conclude that the paths algebra is a powerful and flexible strategy. Such flexibility allows a detailed exploration of the solutions space and the identification of criteria that best suit the objectives of the infrastructure provider, taking into account technical requirements to guarantee a given QoS. In addition, the paths algebra does not impose any restriction on the type and number of metrics that are used. Right on the contrary, linear and non-linear metrics can be used together; any combination of metrics can also be employed.

Besides, paths algebra orders all eligible paths and, therefore, not only the best solution is available but all possible solutions can be accessed. This is a very valuable feature to implement additional strategies such as survivability or planning.

New VNE proposals show that the performance of the embedding is improved when the node and link mapping stage are performed in a coordinated way, i.e. nodes are mapped in a way that optimizes the link mapping stage. Taking advantage of paths algebra, future work can be devoted to improve the proposed VLiM approach by including a coordinated node mapping stage that could provide a two stages-based VNE solution or even a solution performed in one stage.

In addition, future work can be also dedicated to investigate backtracking strategies to act when a VNR cannot be satisfied due to bottlenecks present in the substrate network. The investigation of multipath strategies based on paths algebra can also be an exciting branch for future research.

4 DISTRIBUTED, PARALLEL AND UNIVERSAL VNE

"Alone we can do so little; together we can do so much"
Helen Keller

Chapter 2 showed that there has been little interest in the research community in finding distributed VNE solutions. Only few approaches are part of the **D/S/C, D/S/R, D/D/C** and **D/D/R** categories.

The main contribution presented in this chapter is DPVNE, a *Distributed*, *Parallel* and *universal* Virtual Network Embedding framework. DPVNE can be used as a framework to run any existing embedding algorithm in a distributed way. Thereby, computational load for embedding multiple virtual networks is spread across the substrate network. This reduces work load of individual nodes and enables the network to embed multiple virtual networks in parallel. In contrast to state-of-the-art distributed algorithms, DPVNE achieves lower message overhead and, despite of being distributed, embedding costs are kept comparable to those of centralized approaches. This contribution was published in [BBF+13] as a collaboration with Andreas Fischer, Michael Till Beck and Hermann De Meer from the Chair of Computer Networks and Communications of the University of Passau.

Distributed VNE solutions have received little interest by the research community. This thesis has contributed in the elaboration of DPVNE: A Distributed, Parallel and Universal VNE framework

4.1 WHY DISTRIBUTED VNE

Taking the different constraints and the heterogeneity of the networks into account, the VNE problem typically becomes NP-hard (cf. Chapter 2). As such, the optimal embedding of virtual networks will be difficult to compute – one has to refer to near-optimal solutions instead.

However, although reducing the computational effort, even heuristic approaches do not scale well for large networks. Moreover, most of current proposals are centralized and therefore, load is not distributed among nodes. Instead, embeddings are computed by a single node, which goes against scalability [92]. In addition, centralized algorithms are not able to embed multiple VNRs in parallel.

Spreading load to several nodes introduces the problem of communication overhead. Current distributed approaches rely on the transmission of a significant amount of messages that keep all embedding nodes up-to-date. Increasing the size of the substrate network also boosts the number of exchanged messages.

So, both centralized and distributed approaches do either not scale in terms of computational or message overhead. However, *scalability* is a paramount requirement, especially for large networks.

In this chapter, DPVNE: An Universal, fully Distributed and Parallel VNE approach, is presented. Contrary to a centralized approach where a single node computes the entire embedding, DPVNE has the advantage of making better use of available computing resources, letting multiple nodes (*embedder nodes*) calculate embeddings in parallel. Multiple substrate nodes that are part of the SN perform the actual embedding, instead of having a central instance that might be overwhelmed by receiving a large amount of requests or by managing a highly dynamic substrate network.

Centralized VNE approaches show: 1) single point of failure, 2) poor scalability and 3) delay to embed simultaneous requests. These reasons motivate the need of distributed VNE approaches

DPVNE makes use of hierarchical partitioning that allows to perform the VNE in smaller, cooperating parts of the SN. To avoid inconsistency within the hierarchy, locking mechanisms are used. Some substrate nodes are defined to be the entry points of the embedding framework (*delegation nodes*), managing these partitions. The VNRs are initially randomly distributed to these nodes by a load balancer. The delegation nodes then try to embed the VNRs they received, either by delegating it to other, cooperating parts of the network or by performing the embedding themselves. Having several parts of the network that can operate independently not only distributes computational load to multiple nodes but also enables parallel embedding of multiple VNRs.

Another advantage of the hierarchical partitioning applied in the DPVNE framework is that message overhead is noticeably reduced when compared with currently available distributed algorithms. Although being distributed, the approach still achieves good embedding results in terms of embedding cost.

More precisely, the main contributions of DPVNE are:

- Distributed Embedding: The VNE is not performed by a centralized *embedder node*. Instead, the network is partitioned into several parts and computational overhead is spread among them.

- Parallel Embedding: Since parts of the network can operate independently, the SN is able to embed, in parallel, a set of VNRs, which increases performance of computation.

- Universal Embedding Framework: DPVNE describes a general framework that can integrate any existing VNE solution. The embedder nodes use one of the existing VNE algorithms to actually embed the VNR. As such, depending on the integrated VNE solution DPVNE can belong either to **D/S/C**, **D/S/R**, **D/D/C** or **D/D/R** category.

Since DPVNE is distributed and embeds multiple VNRs in parallel, it scales with respect to computational and messaging overhead – even for large network topologies.

Although DPVNE is universal and embedder nodes can use any arbitrary centralized algorithm to actually compute the embedding, this contribution is focused on algorithms that aim to minimize cost. For evaluation purposes, the centralized ASID algorithm is used [52] in conjunction with the proposed framework.

The rest of this chapter is structured as follows. Section 4.2 presents the shortcomings of existing distributed VNE proposals. In Section 4.3 VNE is formally described and DPVNE framework is introduced. Section 4.4 evaluates DPVNE and compares it against centralized and distributed VNE solutions. Finally, in Section 4.5 conclusions are drawn and future work is discussed.

4.2 SHORTCOMINGS OF EXISTING CENTRALIZED AND DISTRIBUTED VNE APPROACHES

As discussed in Chapter 2, due to the NP-hardness of the VNE problem, most of the current solutions are based on heuristics or meta-heuristics.

The majority of the existing VNE heuristics are centralized, i.e. they rely on a *centralized embedder node* to perform the VNE while they serve *just one VNR at a time*. Being centralized, they can benefit from full knowledge of the substrate network to compute a near-optimal VNE result without any communication overhead. However, they have the following shortcomings:

- *Single point of failure*: If the centralized entity fails, the whole system fails.
- *Poor scalability*: The centralized entity has to communicate the embeddings to the nodes of the SN, so they can adjust their parameters according to the results. Thus, the centralized entity requires efficient message-passing mechanisms which can impose a large overhead on it as the network grows.
- *Delay to embed simultaneous requests*: When multiple VNRs arrive, as the requests are not attended in parallel, the delay to map a VNR is increased.

In [92], Houidi et al. propose a distributed VNE approach that aims to guarantee load-balancing among all substrate nodes during the VN mapping. The proposal maps a VNR by subdividing it into a set of hub-and-spoke clusters where each cluster is handled by a root node in the SN. Although this proposal solves some of the weaknesses of centralized approaches, it suffers from three main problems:

- *Excessive number of messages*: Updated resources capacities are broadcasted among all SN nodes. Besides, as each SN node can be a root, all SN nodes receive the message containing the VNR to be mapped. The exchange of this huge amount of messages requires a separated signaling network.

Existing distributed VNE approaches either have an excessive number of messages, provide suboptimal cost, assume unlimited SN resources or are partly distributed

- *Suboptimal embedding cost*: The algorithm does not consider coordination to choose the root nodes for a set of VNR clusters. Therefore, it is possible that VNR clusters are mapped in separated locations in the SN. This increases the cost needed to map the virtual links among these clusters.
- *Assumes unlimited SN Resources*: The distributed embedding assumes unlimited resources. In real scenarios, this assumption is unrealistic and may lead to overloaded substrate nodes and links.

Another, partly distributed approach is presented in [69]: the substrate network is split into different, hierarchical partitions. However, this approach is not fully distributed as it relies on a general manager and a set of sub-managers dealing with each partition level.

In contrast, DPVNE is a fully *distributed* and *parallel* approach that partitions the SN into several isolated parts where the VNE is performed. DPVNE keeps message overhead low while, despite being distributed, also minimizes embedding cost. Besides, simulation results show that the message overhead is noticeable lower than the one in [92]. Furthermore, in contrast to [69], DPVNE can be used as an universal framework to run any arbitrary centralized embedding algorithm.

DPVNE is focused on cost-optimizing embedding strategies and therefore, for evaluation purposes, one cost-optimizing centralized algorithm in conjunction with the proposed framework is used: The Advanced Subgraph Isomorphism Detection (ASID) [52]. ASID tries to detect areas inside the SN that have the same or similar VNR structure and fulfills VNR's demands. By keeping the length of paths between communication substrate nodes low, cost is reduced. However, DPVNE is not limited to ASID but can also be used together with any VNE centralized approach.

DPVNE is: 1) a fully distributed approach, with 2) low messaging overhead, that provides 3) low embedding cost and is able to 4) embed VNRs in parallel

4.3 DISTRIBUTED, PARALLEL AND UNIVERSAL VNE FRAMEWORK

The Distributed, Parallel and Universal VNE framework presented here uses a hierarchical partitioning scheme to achieve an efficient distributed embedding. By partitioning the network, the partitions can be used to perform parallel embeddings in independent parts of the SN. In particular, DPVNE aims to achieve the following goals:

- *Distributed VNE*: VNRs are embedded in a distributed manner. This lays the foundations for an approach that really scales in respect to the size of the underlying substrate network.
- *Low messaging overhead*: As DPVNE is distributed, SN nodes have to exchange information by means of messages. The number of exchanged messages should be minimized to eliminate unnecessary overhead and increase scalability.
- *Low embedding cost*: Although only part of the SN is available for each VNR (i.e. the assigned partition), still the embedding should be near-optimal with regard to embedding cost. Since DPVNE prefers to cluster closely connected parts of the network, the embedding cost is expected to remain low.
- *Embed VNRs in parallel*: Taking advantage of the SN partitioning, different VNRs are embedded at the same time in isolated SN partitions. To achieve this, DPVNE keeps information about busy partitions, distributing VNRs to parts of the SN that are currently idle.

In order to accomplish these features, the network partition has to be performed in a preparatory phase, before the first VNR arrives. DPVNE is, thus, divided into two stages: The *SN initialization phase* and *embedding phase*.

4.3.1 *Substrate network initialization phase*

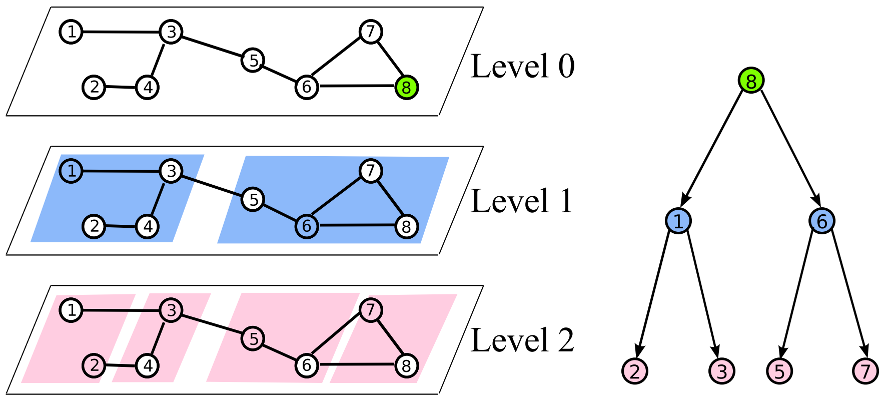
DPVNE is divided in two stages: SN initialization phase and embedding phase

Before performing the actual embeddings, the SN has to be initialized. The initialization phase consists of three steps: Partitioning the SN, assigning delegation nodes, and setting up distributed lock trees.

4.3.1.1 *Partitioning the SN*

The partitioning algorithm divides a substrate network (or a partition of the SN) into several smaller parts. In principle, partitioning can be done with respect to the nature of the expected VNRs that will be embedded later. Here, partitions are created with a preference to closely connected nodes, in order to keep embedding cost low. To this end, DPVNE looks for a hierarchical partitioning that provides, in each layer, a set of non-overlapping partitions. To facilitate embedding, partitions should be highly connected: This increases the likelihood that virtual entities are assigned to well-connected substrate resources. Figure 4.1a depicts an example, where a SN is partitioned two times, from Level 0 to Level 1 and from Level 1 to Level 2. It can be seen that partitions are not overlapping.

Here, the *Multilevel Recursive Bisection* partitioning algorithm proposed by Karypis et al. in [159] is chosen. This algorithm finds a



a) Hierarchical Network Partitioning b) Tree structure

Clusterhead	*Network subset in the cluster*	*Cluster Size*
8	1, 2, 3, 4, 5, 6, 7, 8 (whole network)	S_1
1	1, 2, 3, 4	$S_2 \approx S_1/2$
6	5, 6, 7, 8	$\sim S_2$
2	1, 2	$S_3 \approx S_2/2$
3	3, 4	$\sim S_3$
5	5, 6	$\sim S_3$
7	7, 8	$\sim S_3$

c) Coverage and size of each clusterhead

Figure 4.1: Hierarchical partitioning of a network

specified number of isolated and non-overlapping partitions inside the substrate network's graph that have nearly the same amount of available CPU and bandwidth resources. It also aims at minimizing the number of links between partitions. This means that the algorithm tends to group nodes that are highly interconnected. Later, these partitions of nodes are used to embed virtual networks. The frequent presence of direct links between nodes inside partitions makes it easier for the embedding algorithm to find a valid embedding result within the partition.

For each partition, the node with the highest available CPU resources is selected. This node is called a *clusterhead*. It runs the partitioning algorithm that builds isolated, non-overlapping partitions, and runs also the actual embedding algorithm that the DPVNE implementation is assembled with. After the partitioning algorithm has finished, the procedure is recursively repeated: Within each smaller partition, another *clusterhead* (that has not been chosen as a cluster-

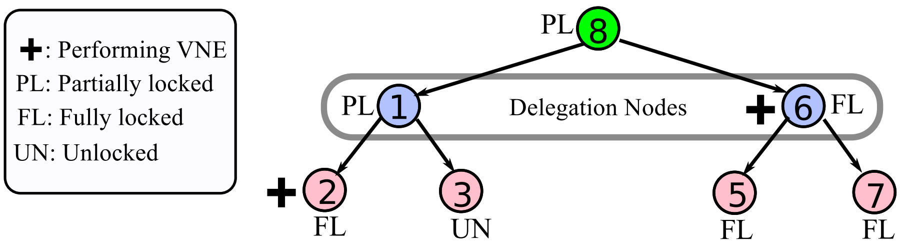


Figure 4.2: Locking of network areas

head before) is selected. Each new clusterhead then divides the partition it is located in. Figure 4.1b gives an example, depicting the chosen clusterheads that govern the partitions shown in Figure 4.1a.

For the sake of simplicity, partitioning is described as a centralized process here, however, it can also be done in a fully distributed way.

This way, a hierarchical partitioning structure of the substrate network is built. Network partitions are organized in a hierarchical tree that allows an easy distribution of the VNRs through the SN. Partitions located at the same hierarchy level do not intersect, and each partition encloses all nodes of its children. The hierachic partitioning stops when the size of the partitions is 1 node. On each level, DPVNE's partitioning approach tries to find partitions having a similar amount of resources. Figure 4.1c illustrates this for the hierarchy depicted in Figure 4.1b.

Each clusterhead is aware of the topology and available resources inside its partition and, as consequence, it is able to run an embedding algorithm to compute the actual VNE.

Partitions of the SN are organized in a hierarchical tree that allows an easy distribution of future VNRs through the SN

4.3.1.2 *Assigning delegation nodes*

The Network Operator (NO) sets up a load balancer that randomly distributes VNRs. However, these requests are not sent to any arbitrary substrate node, but only to special ones: Some clusterheads inside the hierarchy will be in charge of receiving the VNRs and delegating them to the partition that fits best to perform the embedding. Those nodes are called *delegation nodes*. To increase the number of virtual networks that can be embedded in parallel, more than just one single node inside the partition hierarchy are defined to be a delegation node.

If delegation nodes are situated at different levels in the hierarchy, problems can occur with overlapping areas of responsibility. For instance, if one delegation node is a child of another delegation node, both nodes will forward VNRs to clusterheads below both of them, leading to inconsistent information with regard to available resources. To avoid this kind of problem, it is assumed here that delegation nodes are all located on the same level in the hierarchy, taking care of mutually independent parts of the network (cf. Figure 4.2).

4.3.1.3 *Setting up distributed lock trees*

To perform parallel VNEs, a locking mechanism is implemented to prevent inconsistencies of clusterheads sharing parts of the same substrate topology. Therefore, partitions have to be locked temporarily.

For example, in the scenario presented in Figure 4.1a, a mapping algorithm may not be run on the left partition of the level 1, while at the same time, an embedding process is going on at the leftmost partition of the level 2. Without locking mechanisms, in a distributed environment, this implies inconsistencies. However, in the example, the embedding can continue at the right partition of the level 1 in parallel, since this area is independent.

For this reason, each clusterhead maintains a local locktree. This is a data structure where information about the state of partitions that are currently involved in a running embedding process is managed. Each local locktree only covers the set of clusterheads that the clusterhead itself communicates to (i.e., the subtree below the partition and the parent clusterhead). Each partition is in one of the following three states (cf. Figure 4.2):

- *Fully Locked:* The clusterhead that is responsible for the partition is currently embedding a VNR.
- *Partially Locked:* Although the clusterhead that is responsible for the partition is not currently performing a VNE, one or more of its children are. As a consequence, the partition can not embed a VNR at the moment, because it does not know the latest, up-to-date resource information on all its nodes and has to wait for it before starting new embeddings.
- *Unlocked:* Neither the partition nor its children are involved in a VNE. Therefore, they can accept any VNR.

The hierarchical structure of the clusters and the existence of the partially locked state avoid the possibility of deadlocks because two competing clusterheads will not have to wait the other to finish in order to start a new VNE.

4.3.2 *Distributed and parallel embedding*

After initializing the SN, the actual embedding can take place. To reduce the number of exchanged messages, the distribution of the VNR is confined to the hierarchical subtree starting at the delegation node that has received the request.

4.3.2.1 *Embedding within the scope of the delegation nodes*

Each delegation node controls an independent part of the network. To avoid interferences, a delegation node may only delegate embedding requests to clusterheads within its assigned partition scope (or hand it off to its parent node). So, when an embedding request is received, the delegation node tries to find another underlying clusterhead within its partition scope that might be able to embed the VN request.

The first requirement of a partition to perform a VNE is to have, at least, the same number of nodes as the VNR. After this is checked, the partition should also count on sufficient available resources to perform the embedding. To check that, a simple strategy is used. Its aim is to calculate the potential π of a partition p ($SN^p = (N^p, L^p)$, where N^p and L^p are the nodes and links of the partition p, respectively) to embed the virtual network VNR^k. The potential is calculated by the delegation nodes for every underlying partition they know. It is a combination of CPU potential and bandwidth potential. CPU potential is calculated by dividing available CPU resources by the CPU demands of a VNR:

$$\pi_{CPU}(N^p, N^k) = \frac{\sum\limits_{i \in N^p} cap_{CPU}(i)}{\sum\limits_{i^k \in N^k} dem_{CPU}(i^k)}$$

Likewise, the bandwidth potential is calculated by dividing the available bandwidth by the bandwidth demanded by a VNR. Since virtual links can span several physical links, the demanded bandwidth is weighted with an additional parameter ω, which estimates how much of the physical bandwidth will be used by a virtual link. A higher ω leads to a more pessimistic bandwidth potential, expecting that more substrate resources will have to be spent to fulfill the demands:

$$\pi_{BW}(L^p, L^k) = \frac{\sum\limits_{(i,j) \in L^p} cap_{BW}(i,j)}{\omega \cdot \sum\limits_{(i^k,j^k) \in L^k} dem_{BW}(i^k, j^k)}$$

The overall potential is then defined as:

$$\pi(SN^p, VNR^k) = \pi_{CPU}(N^p, N^k) \cdot \pi_{BW}(L^p, L^k) \tag{4.1}$$

The potential tries to estimate whether a specific virtual network can be embedded into a substrate partition. In this definition, only bandwidth demands are weighted because the VNE definition assumes that a virtual node consumes exactly as much substrate resources as it demands, and not more. If the partition's potential is greater than 1.0, it is considered as a candidate to embed the VNR. Note that Equation 4.1 can easily be extended to take various other constraints (like link delays and location based constraints) into account.

There might be more than just one single partition able to embed the request. The delegation nodes start delegating the embedding from the partition in the bottom hierarchy level having the smallest complying potential value. The embedding is then performed by the clusterhead of that partition. This behavior ensures both minimization of the embedding cost in the SN and maximization of the number of network areas that can be used for parallel VNRs: Only the smallest possible network partitions are busy, while other parts of the SN can be used in parallel. In the meantime, when a clusterhead of an underlying partition is still busy calculating the VNE, the delegation node that forwarded the VNR can continue receiving further VNRs and delegate them to other clusterheads that are not busy at the moment.

4.3.2.2 *Embedding outside the scope of the delegation nodes*

If the aforementioned strategy cannot find any suitable partition in the scope of the delegation node, the request is forwarded to the parent clusterhead. Clusterheads above the level of delegation nodes in general do not have up-to-date information (since some nodes within their scope are working in parallel so the availability of resources might have changed in the meantime) and therefore do not estimate to which of their children they forward the request to. They simply forward the request to all of their children (except the one that sent the request), one by one, and stop as soon as one of them can embed the VNE. The original child that failed to embed the request stops embedding further VNEs until its parent node sends a notification to continue (this is done as soon as the parent node receives the final embedding result). If none of the children is able to embed the VNE, the parent clusterhead tries to embed the VNE itself. In case of failure, it forwards the request to its parent clusterhead and waits for its answer. This parent clusterhead again operates in the same way. If there is no parent clusterhead to forward the request to, the highest level of the partition hierarchy has been reached and the VNE failed. In this case, the process is not able to perform the VNE and the VNR is finally rejected.

To deal with the communication among the different levels of the hierarchy, a VNE protocol is proposed

4.3.2.3 *Distributed VNE protocol*

To deal with the communication among the clusterheads of the hierarchy, a VNE protocol with the following messages is proposed.

- `DEREQ`: A DElegate REQuest is used to delegate a VNR to one clusterhead. It can be sent by delegation nodes or nodes above them in the hierachical tree. This message must always be answered with an `EMRES` message.

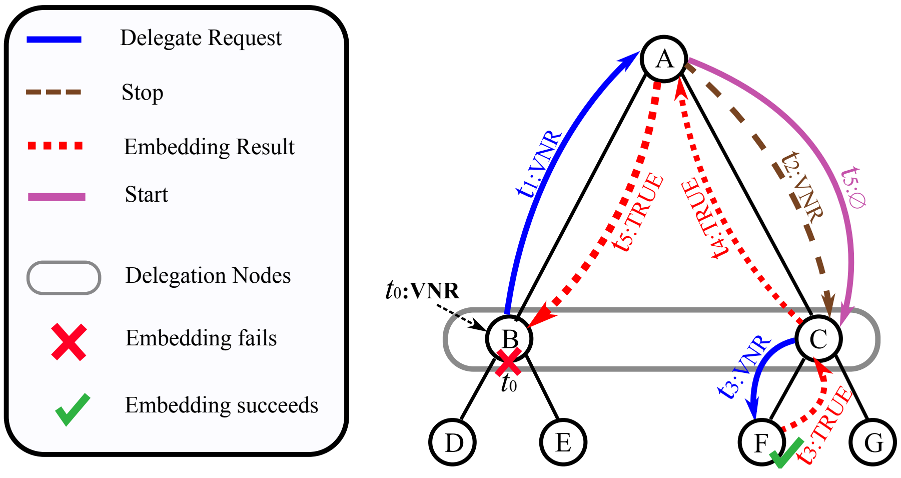


Figure 4.3: Distributed algorithm behavior

- `EMRES`: An EMbedding RESult is the reply following a VNE. If the embedding was successful, it also carries the updates in the SN due to this embedding.
- `STOP`: When a delegation node receives this message, it has to follow three steps: First, it stops accepting new requests. Second, it waits until all the VNRs being embedded in its scope finish. And finally it tries to embed the received VNR within its partition scope. This message is always answered with an `EMRES` message.
- `START`: When a delegation node receives this message, it can start receiving VNRs again.

To facilitate the understanding of DPVNE, its behavior is illustrated in Figure 4.3. The delegation nodes are chosen at level 1 of the partitions' hierarchy. The VNRs are processed for each time (t_x) as follows:

t_0: VNR arrives at delegation node B. The delegation node B tries to find a suitable partition inside its scope by using the partition potential (Equation 4.1). Let's assume the potential tells the delegation node that there are not enough resources available inside any of its subpartitions. In this case, the delegation node tries to run the embedding algorithm inside its own partition. However, let's also assume that it is not able to map the VNR.

t_1: Since the VNR can not be mapped inside the partitions of delegation node B, it delegates the request to its parent clusterhead A sending a `DEREQ` message. There might be unapplied updates generated by previous embeddings that only B and some underlying clusterheads are aware of, so B can also include some update information inside its `DEREQ` message if neccessary. Clusterhead B fully locks all of its own partition scope locally, including its parent clusterhead, so it queues any other VNRs that arrive in the mean time until the embedding of the current VNR is performed.

t_2: Upon receiving a `DEREQ` message, a clusterhead located above the delegation nodes (clusterhead A) knows that the sending clusterhead is unable to embed the VNR itself. So it fully locks the corresponding partition scope locally and randomly chooses one of its other, locally unlocked children and sends to it a `STOP` message. (Since this example is a binary tree, the only possible clusterhead is C). Parent node A sends a `STOP` message to the delegation node C asking for the embedding of the VNR.

t_3: Delegation node C receives a `STOP` message from its parent. As defined previously, upon receiving a `STOP` message, delegation node C stops receiving new requests, waits until all the VNRs being embedded inside its partition finish and, using the potential, chooses the clusterhead F to delegate the VNR and sends a `DEREQ` message to it. C fully locks node F locally; it also locks itself partially. In the next step, VNR is successfully mapped in clusterhead F and the result is notified to the clusterhead C by means of an `EMRES` message.

t_4: After the VNE finishes, clusterhead C unlocks the clusterheads again and notifies its parent node A that the VNE was successfully embedded and which resource allocations changed by means of an `EMRES` message. Clusterhead A receives the message, unlocks the locally locked clusterheads and applies the corresponding updates.

t_5: Clusterhead A forwards the `EMRES` message to delegation node B, indicating that the embedding was performed successfully. In turn, it sends a `START` message (with an empty set of updates, since no relevant resource allocations were done in the meantime that have to be notified) to the delegation node C that was previously stopped. This message tells node C that it can continue processing VNRs. B unlocks the resources that were locked locally.

DPVNE builds a hierarchy of SN partitions to embed the VNRs and coordinates the distribution of VNRs and the embedding process by means of a VNE protocol

In summary, the DPVNE framework performs embeddings in parallel, distributing the load among different nodes in the SN. It first partitions the SN to create a hierarchy of SN partitions within which VNRs can be embedded. Each partition is coordinated by a clusterhead. Each clusterhead maintains a local lock tree, containing information about parts of the SN currently busy with an embedding. Some clusterheads are assigned as Delegation Nodes, receiving VNRs from the outside world and distributing them within the hierachy. VNRs can be directed to higher levels in the hierarchy, if a clusterhead does not have sufficient resources to satisfy the VNR. To coordinate

Table 4.1: VNE approaches used for comparison with DPVNE

ALGORITHM	DESCRIPTION
DAVNM	Distributed approach proposed in [92]
Baseline	Centralized VNE based on available resources for node mapping and k-shortest paths for link mapping [41]
ASID	Centralized coordinated node and link mapping VNE using an heuristic based on subgraph isomorphism detection [52]
RW-MM-SP	Centralized VNE with topology-aware node mapping and k-shortest paths for link mapping [46]
RW-BFS	Coordinated node and link mapping topology-aware centralized VNE [46]

the distribution of VNRs, a protocol with four message types (`DEREQ`, `EMRES`, `STOP`, and `START`) is used.

In the following section, the efficiency of this approach is evaluated and compared to other VNE algorithms.

4.4 EVALUATION

In this section, various characteristics of DPVNE with regard to its distributed nature are evaluated. Therefore, it is compared to the Distributed Virtual Network Mapping (DAVNM) algorithm [92], which is, to the best of the knowledge of this thesis, the only other fully distributed VNE proposal up to now. Independent of its distributed nature, DPVNE is also analyzed concerning the quality of the embedding, in particular the embedding cost. To this end, DPVNE is compared to a set of well known VNE centralized algorithms. In contrast to a distributed approach, the single embedder node of a centralized approach benefits from full knowledge of the whole substrate topology.

4.4.1 *Evaluation methodology*

In order to evaluate DPVNE, the ALEVIN simulation framework [FBD+11, DSS+11] is used. This tool facilitates the development, comparison, and analysis of VNE algorithms. Since the proposed framework focuses on the optimization of cost, a set of appropriate VNE algorithms sharing the same goal were selected. The list and description of these algorithms is presented in Table 4.1. All of them were implemented in ALEVIN.

As already stated, DPVNE is a generic framework that can be used in conjunction with any arbitrary embedding algorithm. For the following evaluation, embedder nodes used the ASID algorithm that already was part of the ALEVIN tool.

As described in Section 4.3, the substrate network is initially partitioned into hierarchical sections. This partitioning is carried out by means of the *metis* tool[1], which provides a free implementation of Karypis et al. algorithm [159]. SN is split into two parts at each level, creating a binary tree hierarchy.

The set of evaluation scenarios were created over random network topologies. This is in line with evaluations done in related work [41, 45, 52]. To generate the random topologies, a Waxman generator [157], parametrized with edge probability 0.5 and long edge ratio 0.5 was employed. Substrate networks were generated with 100 nodes. In order to evaluate the effect of increased stress on the substrate network, requests of different sizes (5, 10, 15, or 20 nodes per VNR) were generated.

As in most related work, CPU power in nodes and bandwidth in links were chosen as constraints, but other kinds of constraints can also be used without constriction. Substrate nodes received a CPU capacity uniformly distributed between $1-100$, whereas virtual nodes received a CPU demand uniformly distributed between $1-50$. Likewise, substrate links received a bandwidth capacity uniformly distributed between $1-100$, whereas virtual links received a bandwidth demand uniformly distributed between $1-50$. These parameters were chosen similar to the ones presented in related work. In order to ensure stability of the results, 30 scenarios were generated for each VNR's size (5, 10, 15, and 20 nodes per VNR).

As stated in previous sections, DPVNE focuses on cost-optimization. Therefore, several existing algorithms towards embedding costs were evaluated and compared. As in previous approaches, embedding cost is defined as the sum of the SN resources used to allocate the VNR:

$$C(VNR^k) = \sum_{i^k \in N^k} dem_{CPU}(i^k) + \sum_{(i^k,j^k)\in L^k} \sum_{(i,j)\in L} BW\big((i^k,j^k),(i,j)\big) \tag{4.2}$$

where $BW\big((i^k,j^k),(i,j)\big)$ indicates the amount of bandwidth of the $dem_{BW}(i^k,j^k)$ that is allocated in the substrate link (i,j).

However, embedding cost heavily depend on the topology and the demands of VNRs. Since generated topologies are random, costs vary and thus, direct comparison can be unfair. A better strategy to compare results is to additionally take the demands of VNRs into account

1 http://glaros.dtc.umn.edu/gkhome/metis/metis/overview

by determining the revenue. As in previous work, revenue is defined as the sum of bandwidth and CPU of the VNRs:

$$R(VNR^k) = \sum_{i^k \in N^k} dem_{CPU}(i^k) + \sum_{(i^k,j^k) \in l^k} dem_{BW}(i^k, j^k) \quad (4.3)$$

So, to bring cost and revenue together for an appropriate comparison, the cost/revenue (C/R) ratio is defined as the division of cost and revenue:

$$\frac{C(VNR^k)}{R(VNR^k)}$$

A smaller value means that less resources are consumed by the embeddings, whereas a higher value means that less resources remain for embedding further VNRs. So, in general, embedding algorithms should try to keep this value small.

4.4.2 *Results*

As discussed in Section 4.1, the goal of the contribution presented in this chapter was to build a *distributed*, *scalable* and *parallel* framework. From the obtained results[2], the following are the key observations.

4.4.2.1 *Reduction of message overhead by DPVNE*

In general, there are two primary factors for the success of a *scalable* algorithm that operates in a *distributed* manner – the distribution of load (set of VNRs) and the message overhead generated by it: On the one hand it is preferable that load is spread across as many nodes as possible (parallel VNE). On the other hand, communication overhead should be low. Otherwise, the approach will not scale well.

The amount of coordination messages is significantly reduced with DPVNE when compared with existing distributed VNE approaches

This is a problem of the DAVNM algorithm: DAVNM shows a noticeable communication overhead in the performed simulations in order to complete the embeddings. Table 4.2 compares the message overhead of DAVNM to DPVNE with $\omega = 10$, using the root node as the only delegation node that distributes load. It can be seen, that DPVNE comes with significantly lower message overhead in this scenario. Later, the influence of different parameters in the number of exchanged messages is detailed.

4.4.2.2 *Influence of pessimistic estimations*

Of course, there are various parameters that influence the behavior of DPVNE and also have an impact to the number of messages that need to be sent to quickly find an appropriate embedding result. One important factor is a reasonable estimation of the available resources inside SN partitions. DPVNE introduces the potential π of a determined partition that can be fine-tuned by its ω parameter: A higher

2 All figures are presented with a confidence level of 95%

Table 4.2: Average number of coordination messages for *DAVNM* vs. *DPVNE*

#NODES (PER VNR)	DAVNM	DPVNE
5	53554	18
10	61591	13
15	74918	13
20	93470	14

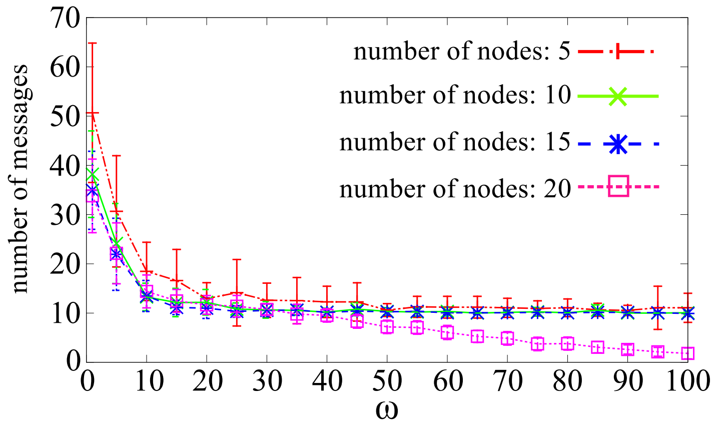


Figure 4.4: Coordination messages vs ω

value results in a more pessimistic evaluation of available substrate resources, a smaller value in a more optimistic one.

Figure 4.4 shows the number of coordination messages that are exchanged between substrate nodes. This is evaluated for varying ω. Additionally, the size of the virtual networks was varied between 5 nodes and 20 nodes per VN, and the root node was chosen as the only delegation node. It can be seen that, regardless of the size of the VN, there is a sharp drop up to $\omega = 10$ with the amount of messages staying at or below 20 messages for larger ω. This is because for lower ω, DPVNE acts in an optimistic way, trying to embed virtual networks at levels too deep in the tree. This leads to an increased messaging overhead, when oversized virtual networks have to be delegated to larger partitions. On the other hand, if ω is too large, eventually everything will be embedded by the root node, leading to lower message overhead (but increased centralization). In the figure, this is the case for 20 virtual nodes per network.

4.4.2.3 *Influence of delegation node placement*

For Figure 4.4, to have a clear and understandable presentation of results, the root node (level 0) has been chosen as the only delegation node. However, the number of messages also depends on where the

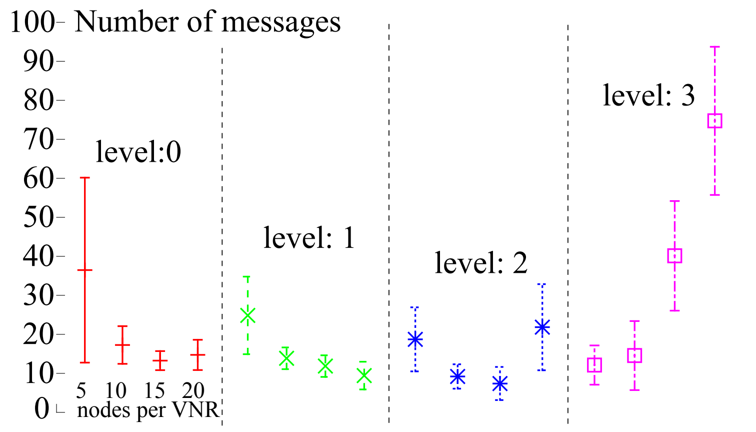


Figure 4.5: Coordination messages vs delegation level

delegation nodes are placed, because this influences which level of the hierarchy the VNRs are initially sent to and how they are delegated. In Figure 4.5, the number of messages in dependence of the level where delegation nodes are placed is shown. Each level is evaluated with varying load, having virtual networks with 5, 10, 15, or 20 nodes each. It can be seen, that message overhead does not significantly change when delegation nodes were set to a lower level.

The advantage of choosing multiple delegation nodes that delegate and spread computation to other clusterheads is the reduction of message passing overhead and the dependability on a single node. However, delegation nodes should not be placed too deep in the hierarchy. Otherwise, this has a negative effect to message overhead: As depicted in the figure, for level 3, message overhead starts to grow again, especially for bigger networks. The average number of nodes within one of the 2^3 partitions at level 3 is $\frac{100}{2^3} \approx 13$. Since a partition of this size will, on average, not be able to embed a VNR that has 15 or 20 nodes, these VNRs have to be delegated to a higher level inside the hierarchy – which results in additional message overhead. As shown in the figure, for these values there is a significant increase of communication.

4.4.2.4 *Trade-off between number of messages and level of parallelism*

Keeping the number of coordination messages at a low level is not the only aim of a distributed VNE algorithm. Otherwise, an algorithm that only uses one single node to compute all the embeddings (like centralized algorithms do) would be ideal, already. For a distributed approach, however, VNRs should be appropriately distributed among substrate nodes, so that they can be embedded in parallel. To spread load equally, DPVNE organizes substrate nodes in a tree structure. As such, it is interesting to see which percentage of the requested virtual networks is mapped at which level in the tree. The deeper in the hierarchy, the more nodes will participate in the set of embeddings.

There is a trade-off between the number of messages and the level of parallelism. The higher the number of messages, the higher the number of nodes performing parallel VNEs

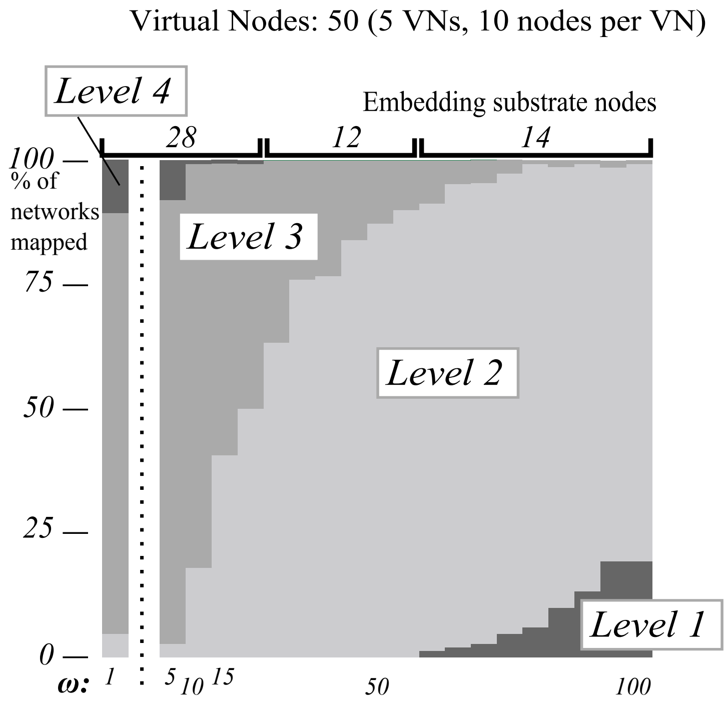


Figure 4.6: Embedding involvement vs ω

E.g. if 10% of all networks are mapped at level 3, then $2^3 = 8$ nodes take 10% of the load. This is depicted in Figure 4.6. In this figure, the load distribution between levels in the binary tree is shown in relation to ω, which is the main regulating factor in this approach. A higher ω means that estimates for embedding costs are more pessimistic, leading to networks being embedded at a higher level in the tree and, as a consequence, to a lower level of parallelism. The number of nodes participating in the embedding is shown at the top of the figure as "embedding substrate nodes". It can be seen that for higher ω there are less nodes participating in the embedding.

Taking Figure 4.4 and Figure 4.6 into account, one can conclude that $\omega = 10$ provides a good tradeoff between message overhead and load distribution. Taking the example of five virtual networks with 10 nodes per virtual network and a substrate network size of 100 nodes, it can be seen that in this scenario, for $\omega = 10$ more than one quarter of the substrate network (28 nodes) participates in the embedding, while only about 20 messages are necessary on average to coordinate the embeddings. Comparing this to the more than 60000 messages generated by the *DAVNM* algorithm, it becomes clear that DPVNE is a significant improvement in terms of message overhead.

4.4.2.5 *Economical resource usage*

Figure 4.7 shows the cost/revenue ratio for several VNE algorithms with a varying load of 5, 10, 15, and 20 virtual nodes per VNR respectively. It can be seen that the C/R ratio of DPVNE is comparable to the

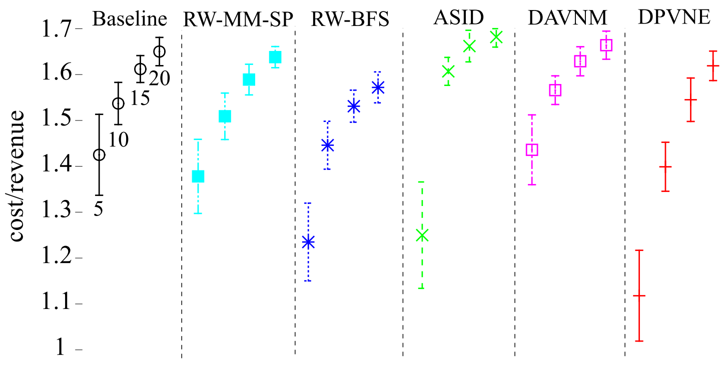


Figure 4.7: C/R in different VNE algorithms

Table 4.3: Node stress of ASID compared to DPVNE

#NODES (PER VNR)	ASID	DPVNE
5	0.4	0.3
10	1.9	0.5
15	4.0	0.8
20	6.9	1.0

ratio of other approaches. On average, results are better than for the distributed DAVNM algorithm. In some cases, DPVNE even outperforms centralized approaches, especially for smaller virtual networks, although embedder nodes do not have full knowledge of the overall network topology.

It is even slightly better than the pure ASID approach, because embeddings are confined to a partition with just enough resources. Since Karypi's graph partitioning algorithm is used to detect highly connected partitions inside the substrate network, the possibility to find appropriate embedding candidates within the immediate surroundings of a node is higher than when just choosing any arbitrary, random node inside the whole network topology (as ASID does). In contrast, the ASID algorithm also tries to embed parts of a virtual network into areas of the SN that are already crowded and therefore has to choose further links and nodes that are only reachable through longer paths. By measuring the number of virtual nodes that are mapped to a substrate node on average, this behavior can be demonstrated (see Table 4.3).

Due to the implemented partitioning algorithm, C/R ratios of DPVNE remain comparable (and even sometimes outperform) the ones of centralized VNE approaches

Thus, by distributing the workload to various parts of the network, the number of virtual nodes mapped to a substrate node decreases. More importantly, this also has a positive effect of reducing embedding costs.

Summarizing, simulation results have shown, on the one hand, that DPVNE drastically reduces the number of exchanged messages when compared with the other distributed alternative, DAVNM. Even though substantially reducing the exchanged messages, the performance of DPVNE depends on two main factors: *potential estimation* – pessimistic estimations decrease the number of messages; and *delegation nodes location* – placing delegation nodes in deep hierarchy levels leads to a growth of exchanged messages. However, making pessimistic estimations and placing delegation nodes up in the hierarchy reduces the level of parallelism of DPVNE (number of different SN nodes performing VNEs). Therefore, there is a tradeoff between the amount of exchanged messages and the number of VNRs that can be embedded in parallel. On the other hand, results have shown that embedding costs reached by DPVNE are comparable with the ones of centralized approaches. With VNRs of low sizes, DPVNE even achieves better cost/revenue ratio, due to the restriction of the VNEs to smaller and strongly interconnected SN partitions.

4.5 CONCLUSION

This chapter presented DPVNE a generic, distributed and parallel framework for embedding virtual networks (DPVNE) that can be used in conjunction with any arbitrary embedding algorithm. Simulation results showed that, using this framework with one well-known embedding algorithm, message overhead can be kept significantly smaller than other fully distributed VNE solutions. This is achieved by hierarchic partitioning of the SN instead of using the complete SN topology to perform the embedding. Simulation results also showed that, despite its distributed nature, embedding costs of DPVNE remain comparable to those of centralized algorithms.

Also, the influence of the location of SN nodes in charge of delegating VNE and the available resources of SN partitions in the number of exchanged messages was illustrated. Results allowed to indicate a tradeoff between the number of messages and the level of parallelism that can be achieved with DPVNE.

DPVNE used ASID to perform VNE within each partition of the SN. An interesting branch for future investigation can be devoted to the evaluation of DPVNE with different VNE algorithms. Besides, more sophisticated heuristics for estimating the potential of different partitions could be investigated. Another subject of future research is the placement of delegation nodes and its effect on message overhead. Optimal delegation nodes placement is strongly related to the expected size of the VNRs. Furthermore, the effects of different partitioning methods could be investigated.

5 ENERGY-EFFICIENT VNE

"Efficiency is doing things right; effectiveness is doing the right things"
Peter Drucker

Waste of energy due to over-provisioning and over-dimensioning of network infrastructures has recently stimulated the interest on energy consumption reduction by Internet Service Providers. By means of resource consolidation, network virtualization based architectures will enable energy saving. The main contribution presented in this chapter is the introduction of the Energy-Aware Virtual Network Embedding Problem, defined as the VNE where the main objective is to switch off as many network nodes and interfaces as possible by allocating the virtual demands to a consolidated subset of active physical networking equipment.

The first contribution to solve EA-VNE consists on a **C/S/R** approach based on a mixed integer program that looks for the minimization of active substrate nodes and links after performing the embedding. EA-VNE shows great power savings over existing cost-based VNE approaches. This contribution was published in [BHD+12] as a collaboration with Andreas Fischer and Hermann De Meer from the Chair of Computer Networks and Communications of the University of Passau and Michael Duelli and Daniel Schlosser from the Institute of Computer Science of University of Würzburg.

This thesis introduced the Energy-Aware VNE. It is the first approach to allocate the virtual demands trying to minimize the power consumption of the physical network

The second contribution to solve EA-VNE was published in [BH13]. This **C/S/R** proposal modifies and improves the existing energy-aware VNE solution proposed in [BHD+12] by introducing embedding cost and load balancing to solve ties in optimal embedding solutions that consume the same amount of power. Besides, as MIP-based energy efficient VNE approaches are hard to solve for large network sizes and have an adverse effect in the acceptance ratio, a **C/D/R** VNE heuristic approach to reconfigure, minimizing the power consumption, the allocation of already embedded virtual networks, is also proposed.

5.1 ENERGY EFFICIENCY AND NETWORK VIRTUALIZATION

Network operators are becoming aware of the pressing necessity of energy saving. According to [160], the ICT has the potential to reduce from 20 to 30% of its greenhouse gas emissions. This growing interest of the NOs is mainly due to the rise of energy costs and the international community call to act against the global warming.

In developed countries, telecom and NOs are alarmingly increasing their energy consumption [161]. A recent study [162] shows that the impact of energy-aware strategies over the data and control planes of wired network infrastructures would approximately be of 50% energy savings in the short-term and 80% in the long-term, resulting in an unexpected potential for NOs to cut their energy bills in the near future.

Nowadays, networks are designed for traffic loads during the busy hour, making heavy use of over-provisioning to fulfill QoS constraints under high loads. However, this leads to resource under-utilization and, along with it, unnecessary power consumption [161]. This situation has substantially increased interest in the research field of green networking. In particular, four different research areas have been recently identified [122]: Adaptive Link Rate (ALR), interface proxying, energy-aware applications and energy-aware infrastructure.

Most of the advances in energy-efficient networking are concentrated in the former area. As the energy consumption of a link depends on its negotiated capacity, *ALR* goal is to adapt link power consumption to its current load rather than to its negotiated rate. There are several proposals to *sleep* a link for idle times or to dynamically renegotiate its rate to fit the current load and save energy. *Interface proxying* techniques are applied to idle end devices dealing with control traffic that can be dropped or just delegated to be processed in different entities, leaving the device free to switch to sleep mode. *Energy-aware software and applications* is a research field where both user-level applications and kernel-level network stack have to be redesigned to save energy.

Nowadays, networks are designed for peak traffic loads and not for current loads. This leads to unnecessary energy consumption

The remaining research area is concentrated on energy savings, not just in individual networking equipment, but in the whole network infrastructure. One key element of future network architectures will be their energy awareness; future Internet will rely on architectures that guarantee high performance and tight QoS while making an efficient use of the energy.

One of the main characteristics of network virtualization is the possibility of concentrating several virtual instances in one physical resource, commonly known as consolidation. This feature will be one of the main enablers of power savings in future Internet architectures. These savings can be reached by the concentration of several virtual networks in a reduced subset of strongly consolidated physical equipment, leaving the rest of the infrastructure to be switched off or to enter in sleep mode during low traffic demand periods (e.g. night traffic).

Applying virtualization of network resources leads to the problem of allocating virtual network demands to physical network resources, known as the Virtual Network Embedding problem. Conducting the objective of the VNE to the minimization of the power consumption, the physical network can be dynamically dimensioned for current traffic demand rather than for peak demand.

The power spent by today's routers and interfaces is almost insensitive to the traffic load (according to [121] an idle router consumes 90% of the power it requires when working at maximum capacity). This affirmation is also supported by the study realized in [163]. There, the authors performed a detailed benchmark of the power consumption of current networking devices, taking into account the components of current network switches (Base chassis power, line cards, active ports, ternary content addressable memory and firmware). Inside the study, the authors introduce the Energy Proportionality Index (EPI) that measures the relationship between load and power consumption; EPI is expressed in percentage, with 100 implying that the device has perfect energy proportionality and 0 implying that the energy consumed by the device is completely agnostic to the offered load. The tests performed in the benchmark, for different networking equipment, show EPI values ranging from 8.6% to 25%. These results allow to conclude that the power equipment of current networking equipment is almost not influenced by the traffic load.

Taking this into account, the first contribution presented in this chapter is to lead the objective of the VNE to the minimization of the power consumption, so that the physical network can be dynamically dimensioned, leaving free as much networking equipment as possible, to accomplish the VNRs demands. Free networking equipment can be then turn to sleep (low power consumption) mode or switched off. The right side of Figure 5.1 shows how two different virtual network requests are mapped to just two nodes and one link in the substrate network avoiding to switch on two more nodes and links when compared with a cost-based embedding, that shows energy inefficient results, of the same requests (on the left side of the figure).

Here, the objective of the VNE is shifted to the minimization of the energy consumption by dynamically dimensioning the SN, leaving free as much equipment as possible

By means of resource consolidation, network virtualization enables the energy-efficient use of network infrastructure.

In this chapter, the energy-aware virtual network embedding problem, where the goal is to allocate the set of virtual network requests in a reduced group of physical network equipment is introduced, and a MIP to optimally solve it is proposed. As a second step, an Energy-Aware Virtual Network Embedding with Load Balancing (EA-VNE-LB) and embedding cost is introduced, allowing to improve the acceptance ratio and power savings after performing the embedding.

Integer Linear Programs are in many practical situations NP-complete, therefore, the implementation of the proposed EA-VNE and EA-VNE-LB models is not scalable for large scenarios (big network sizes). However, exact formulations represent an optimal bound for future energy-aware VNE heuristics. To the best of our knowledge this is the first attempt to formulate VNE for energy awareness.

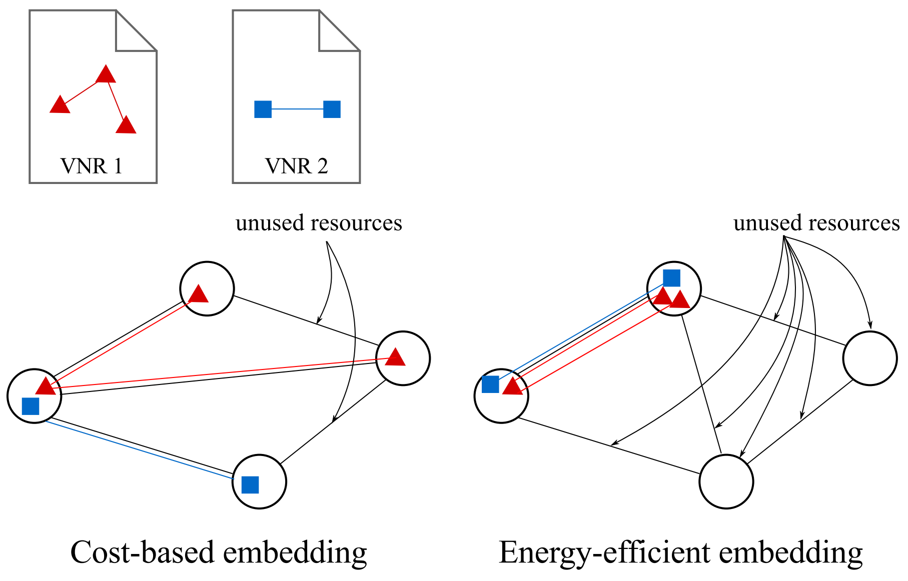


Figure 5.1: VNE consolidating network resources

Most of the algorithms proposed to solve the VNE look for the minimization of the resources spent by the SN to map a VNR, also known as *embedding cost*. In this way, as more SN resources are available, the next VNR is more likely to be embedded. This results in an increment of the number of mapped VNRs (acceptance ratio). If the VNE objective is totally shifted from cost to energy efficiency, there is a trade-off between power savings and acceptance ratio; above all, in energy-aware solutions, the acceptance ratio is lower (even including embedding cost and load balancing) than in cost-based solutions, in general, over equal load conditions.

Therefore, EA-VNE and EA-VNE-LB suffer from two main shortcomings: 1) due to MIP's inherent complexity, the running time needed to solve them is not affordable for big problem instances (big SNs and VNRs sizes) and 2) they result in lower acceptance ratios than cost-based VNE approaches (especially for high network loads). To overcome these shortcomings, a scalable Energy-Aware Reconfiguration Heuristic (EA-RH) is also introduced. EA-RH takes a set of already embedded VNRs as input to perform an energy-efficient relocation of resources while maintaining the VN acceptance ratio for small and large network sizes.

The remainder of this chapter is organized as follows. Section 5.2 provides a review of the related work. Section 5.3 formulates and evaluates the EA-VNE and EA-VNE-LB problems using a MIP formulation. Section 5.4 presents and evaluates the heuristic algorithm (EA-RH) to relocate existing embeddings. Finally, Section 5.5 presents the conclusions of this chapter.

5.2 RELATED WORK

The aim of this contribution is to minimize the power consumption in the network infrastructure by including it as the main objective of the VNE. There have been considerable research effort dealing with energy efficiency in the network architecture as well as numerous proposals to solve the VNE. But, until now, they have been investigated separately. Accordingly, this section separates the background on energy efficiency proposals in the network architecture from the proposals to solve the VNE.

5.2.1 *Energy efficiency in network architecture*

Most of the mechanisms to save energy in ICT technologies have been devoted to the reduction of power consumption in a single networking device or even in individual parts of it (network interfaces, packet processing engines, etc.). However, the interest in energy-aware infrastructures has been growing recently.

Virtualization has been used in backbone networks to decouple physical elements (e.g. line cards), which can be put in standby, from their virtual functionalities, so these functionalities can be migrated towards active physical elements of the same device [164].

An energy-aware network and a power-aware protocol design are proposed in [121]. The objective is to replace power-hungry systems in the core with energy-efficient systems that still provide the required reliability and performance. By means of a benchmark over two different router equipment (access and core routers), the authors conclude that, to save power, the network should be dimensioned by minimizing the number of powered router chassis at a given point of presence, and maximizing the number of line cards attached to each chassis. With this objective, they formulate a mixed integer program constrained by the network traffic matrix. The solution of the problem indicates how the nodes must be provisioned and which are the best routes to minimize the power consumption. The obtained results show power savings up to 65% for certain topologies.

From a point of view of the Internet Protocol (IP) layer, several energy-aware routing approaches have been proposed. Their main aim is to aggregate traffic flows in a reduced set of network equipment using the consolidation principle, so that the unused links and nodes can be switched off or slept. The proposal in [165] formulates the routing problem as an ILP where the goal is to minimize the power consumed by physical nodes and links while fulfilling the demands of the traffic matrix and maintaining a certain level of QoS. The authors propose heuristic algorithms to solve the problem and reach power savings up to 30% in links and 50% in nodes. The approach in [166] follows the same goal, but taking care of the fact that in backbone networks, pairs of routers are typically connected by multiple physical cables that form one logical bundled link. This fact is introduced in the ILP formulation where the goal is to turn off as much cables of the bundled links as possible. The proposed heuristics reach up to 73% of power savings when tested over the Abilene topology. The authors in [167] propose to solve numerically the ILP formulation of the energy-aware routing problem with the introduction of three different models for the network device energy consumption: idle energy (difference in power consumption between an idle and a full loaded device is not too noticeable), fully proportional (power consumption is directly proportional to device load) and energy-agnostic (power consumption is equal independently of the device load). Recently, a similar approach was proposed in [168] where different energy-profiles are proposed and a linear programming problem looking for the minimization of the power consumption is solved for each profile. Finally, Cianfrani et al. propose an OSPF-compliant approach called Energy Saving IP Routing (ESIR) [169] where the main goal is to share the Shortest Path Trees (SPTs) between neighbor routers so that the overall set of active network links is minimized. ESIR uses heuristic algorithms to solve the NP-complete problem of sharing SPTs and reduces the number of active links in a percentage of 40%.

As it has been briefly reviewed, current research in energy-aware infrastructures is devoted to finding the best routing solution minimizing the power consumption in the network while maintaining the QoS. Commonly, this can be modeled with ILP having as input the network topology, the power consumption of the devices and the traffic matrix. Although VNE can be also modeled with ILP and can share the power consumption minimization of the whole network as objective, the problem is not the same. VNE tries to allocate the demands of one virtual network in links and also in *nodes*. The complexity of VNE is higher as it has also to find, inside the SN, the most adequate nodes according to the optimization objective. Besides, virtual network demands do not necessarily mean the same as current traffic demands; VN demands are the capacity that the SP expects its virtual network to have, they are the upper bound of the traffic flow demands generated and transported by the VN. Even inside the VN, when these demands have been allocated, further energy optimization can be carried out by applying any of the previously mentioned routing techniques.

Energy-aware VNE is different to energy-aware IP routing. Besides finding the routes to map virtual links, also suitable nodes have to be find to map the virtual nodes demands

5.2.2 *Virtual network embedding in energy efficiency*

As it is already stated in Chapter 2, all existing VNE approaches have different goals, e.g. maximize the acceptance ratio or provide resilient VNEs. However, to the best of our knowledge, the first contribution of this chapter [BHD+12] is the first approach tackling the power consumption in VNE. There, an exact **C/S/R** solution, called EA-VNE, based on mixed integer programming, that consolidates the network resources in a subset of the SN, allowing the remaining resources to turn to an energy-efficient mode (switched off, hibernated, slept, etc.) is proposed. It shows that noticeable power savings can be reached when the load of traffic demands is low. However, it presents two shortcomings:

- EA-VNE does not consider the embedding cost and does not balance the load demanded by the VNRs. It does not have a mechanism to choose among different optimal embeddings that consume the same amount of power; it chooses arbitrarily one of them. This behavior may quickly deplete some substrate resources leading to low acceptance ratios for high traffic loads.
- EA-VNE is not scalable. The complexity of the exact MIP is high when the size of the problem grows resulting in unaffordable execution times to solve problems with large network sizes.

Two contributions are presented in this chapter: 1) An MIP VNE formulation concentrating the virtual demands in a small number of active equipment and 2) a dynamic heuristic to relocate the mapped VNRs in an energy-efficient way

To overcome these inconveniences, two new contributions are presented in [BH13]:

1. An exact **C/S/R** solution (EA-VNE-LB) that, instead of making an arbitrary decision among optimal embeddings with equal power consumption, minimizes the embedding cost and, besides, balances the load demanded by the VNRs (i.e. it chooses the substrate elements with higher remaining resources). The resulting embedding maximizes the spare capacity of the SN and reduces bottlenecks so the likelihood to succeed in the embedding of the next VNR is increased. Through intensive simulations, EA-VNE-LB is shown to reach greater power savings when compared against the approach proposed in [BHD+12] and the optimal cost-based approach proposed in [51].
2. An **C/D/R** heuristic VNE approach (EA-RH) that relocates current embedded VNRs in an energy-efficient manner while maintaining the acceptance ratio. EA-RH reconfigures the embeddings performed by cost-based VNE approaches in reasonable execution time. For evaluation purposes, EA-RH relocates the embeddings made by exact [51] and heuristic [45] cost-based approaches.

5.3 MODELING GREENER NETWORKS

Most of the current proposals to solve VNE considers that two virtual nodes of the same virtual network can not be mapped to the same substrate node, that is, $f_k(i^k) = f_k(j^k) \iff i^k = j^k, \forall i^k, j^k \in N^k$.

Table 5.1: MIP inputs and variables

	Term	Description
Inputs	$SN = (N, L)$	Directed Graph representing the SN
	$VNR^k = (N^k, L^k)$	Directed Graph representing the VNR k
	$dem^{CPU}_k(i^k)$	CPU demand of virtual node i^k
	$dem^{BW}_k(i^k, j^k)$	Bandwidth demand of virtual link (i^k, j^k)
	$cap_{CPU}(i)$	Available CPU resource of SN node i
	$cap_{BW}(i, j)$	Available bandwidth resource of SN link (i, j)
	w_e	Energy weight in the objective function
	w_c	Load-balanced embedding cost weight in the objective function
	NO_i	Binary parameter, takes '1' value if SN node i is active before the mapping, '0' otherwise
	$LO_{(i,j)}$	Binary parameter, takes '1' value if SN link (i, j) is active before the mapping, '0' otherwise
	$PowerNode_i$	Power consumed by the SN node i
	$PowerLink_{(i,j)}$	Power consumed by the SN link (i, j)
	$match(i^k)$	Set of substrate nodes available to map the virtual node i^k
	$MaxDegree$	Maximum degree of any $i \in N$
Variables	$f^{(i,j)}_{i',j'}$	Bandwidth mapped from SN node i' to substrate node j' passing through substrate link (i, j)
	$x^i_{i^k}$	Binary variable indicating whether the virtual node i^k is allocated in the substrate node i
	$LD^{i,j}_{BW}$	Amount of BW allocated from SN node i to SN node j supporting the BW demand of one or more virtual links (i^k, j^k). $LD^{i,j}_{BW} = \sum_{(i^k,j^k)\in L^k} dem^{BW}_k(i^k, j^k) x^i_{i^k} x^j_{j^k}, \forall i, j \in N$
	$z^{ij}_{(i^k,j^k)}$	Auxiliary binary variable equal to $x^i_{i^k} \cdot x^j_{j^k}$ introduced to avoid the non-linearity of the formulation [51]
	$\delta_{(i,j)}$	Binary variable indicating whether the substrate link (i, j) is activated after the mapping
	γ_i	Binary variable indicating whether the substrate node i is activated after the mapping

However, in [51] the cost minimization of the embedding is improved by avoiding this constraint. Since there are mechanisms to create links between virtual machines that are hosted in the same physical machines [170], EA-VNE, EA-VNE-LB and EA-RH rely in the fact that to reach better levels of power consumption, it is necessary to allow the hosting of more than one virtual node of the same VNR in the same physical node as long as it fulfills all the virtual node demands.

5.3.1 *VNE energy-efficient problem*

As previously discussed, the power consumption of current networking equipment is insensitive to the traffic load. Therefore, to reduce it in the SN, the best approach is to use the minimum amount of equipment to fulfill the VNRs demands. In this section, a power efficient

MIP formulation of the VNE where the objective is to realize the mapping of a VNR in a small set of substrate nodes and links (here called active equipment) introducing embedding cost and load balancing in the embedding is proposed.

This formulation considers that all the SN equipment is inactive (switched off) if there are not virtual network demands mapped on top of it. Its main goal is to embed the arriving VNRs in the smallest consolidated SN subset able to fulfill the demand constraints. Therefore, it produces a VNE where virtual demands are allocated in the smallest feasible set of substrate links and nodes trying to minimize the SN resources to activate. In this way, the unused network interfaces and nodes (inactive equipment) remain switched off or slept to minimize the overall power consumption of the SN. To reduce the complexity of the problem, network resources are considered to be homogeneous with regard to their power consumption, this can be the case of SNs confined to just one InP segment (access, transport or core) where network equipment shares similar characteristics. Embedding cost and load balancing are integrated to the EA-VNE MIP formulation by including the load balancing-based cost definition [45, 36] in the objective function of EA-VNE-LB.

EA-VNE AND EA-VNE-LB FORMULATION

Parameters and variables of the MIP formulation are shown in Table 5.1.

Objective Function

Minimize

$$w_e \cdot \left(\sum_{i \in N : NO_i = 0} (\gamma_i \cdot \mathrm{PowerNode}_i) + \sum_{(i,j) \in L : LO_{(i,j)} = 0} (\delta_{(i,j)} \cdot \mathrm{PowerLink}_{(i,j)}) \right) + w_c \cdot \mathrm{costLB}(VNR^k) \tag{5.1}$$

The objective function (Equation 5.1) seeks to minimize the power consumption of the SN after embedding the VNR k. It is composed of two main terms: The left side of the expression represents the minimization of the power consumption, i.e. the sum of the power consumption of the substrate nodes and links that are activated after performing the VNE. It improves the objective in [BHD+12] that considered the number of substrate nodes and links activated without considering the power consumption attached to each of them.

To avoid ties in optimal solutions with the same power consumption, embedding cost is included in the objective function using the formal definition in [45] that, unlike [51], introduces load balancing in the embedding ensuring that the resources with more residual capacities are more likely chosen over the resources with less residual capacities. This feature is added in the objective function by including the embedding cost needed to map the VNR k in the right side of Equation 5.1: $\mathrm{costLB}(VNR^k) = \sum_{(i,j)\in L} \frac{1}{cap_{BW}(i,j)+\delta} \sum_{i',j'\in N} f_{i',j'}^{(i,j)} + \sum_{i\in N} \frac{1}{cap_{CPU}(i)+\delta} \sum_{i^k\in N^k} x_{i^k}^{i} dem_k^{CPU}(i^k)$, where $\delta \to 0$ is a small value introduced to avoid dividing by zero. $\mathrm{cost}(VNR^k)$ represents the amount of resources spent in the SN to map the VNR k.

The main objective of EA-VNE-LB [BH13] is then to minimize the power consumption in the SN after the mapping is performed. The inclusion of $\mathrm{costLB}(VNR^k)$ in the objective function help us to evaluate the pure cost based approach (with $w_e = 0$ and $w_c = 1$) and to introduce load balancing in the energy-aware solution, i.e. to solve the draws of energy-efficient embeddings promoting the solution with more residual resources in the SN and resulting in better embeddings with regard to power savings and acceptance ratio. To do this, it is necessary to ensure that the right side of Equation 5.1 has a very small value with respect to the value of the left side (power consumption), therefore $w_e = 1$ and $w_c \to 0$. To evaluate EA-VNE [BHD+12], cost is disregarded ($w_e = 1$ and $w_c = 0$).

Constraints

Transformation constraint:

$$\sum_{(i^k,j^k)\in L^k} dem_k^{BW}(i^k,j^k) \cdot z_{(i^k,j^k)}^{i,j} = LD_{BW}^{i,j}, \forall i,j \in N \tag{5.2}$$

Constraint 5.2 introduces the z variable to avoid the non-linearity on the formulation. Variable $LD_{BW}^{i,j}$ is also introduced in this constraint.

Flow related constraints:

- Source flow constraints:

$$\sum_{(i,h)\in L} f_{i,j}^{(i,h)} - \sum_{(h,i)\in L} f_{i,j}^{(h,i)} = LD_{BW}^{i,j}, \forall i,j \in N \tag{5.3}$$

- Destination flow constraints:

$$\sum_{(h,j)\in L} f_{i,j}^{(h,j)} - \sum_{(j,h)\in L} f_{i,j}^{(j,h)} = LD_{BW}^{i,j}, \forall i,j \in N \tag{5.4}$$

- Input-Output flow constraints:

$$\sum_{(i,l)\in L} f_{i',j'}^{(i,l)} = \sum_{(l,j)\in L} f_{i',j'}^{(l,j)}, \forall i',j' \in N, l \in N\backslash\{i',j'\} \tag{5.5}$$

Equation 5.3, Equation 5.4 and Equation 5.5 are flow conservation constraints.

Constraints to ensure that $z_{(i^k,j^k)}^{ij} = x_{i^k}^{i} \cdot x_{j^k}^{j}$:

$$\sum_{i\in N} z_{(i^k,j^k)}^{i,j} = x_{j^k}^{j}, \forall (i^k,j^k) \in L^k, \forall j \in N \tag{5.6}$$

$$\sum_{j \in N} z^{i,j}_{(i^k,j^k)} = x^i_{i^k}, \forall (i^k, j^k) \in L^k, \forall i \in N \quad (5.7)$$

$$x^i_{i^k} + x^j_{j^k} - z^{i,j}_{(i^k,j^k)} \leqslant 1, \forall (i^k, j^k) \in L^k, \forall i, j \in N \quad (5.8)$$

The correlation between variables x and z is introduced in constraints 5.6- 5.8. $z^{i,j}_{(i^k,j^k)}$ will be 1 only if $x^i_{i^k}$ and $x^j_{j^k}$ are 1.

Capacity Constraints:

$$\sum_{i',j' \in N} f^{(i,j)}_{i',j'} \leqslant cap_{BW}(i,j) \cdot \delta_{(i,j)}, \forall (i,j) \in L \quad (5.9)$$

$$\sum_{i^k \in N^k} x^l_{i^k} dem^{CPU}_k(i^k) \leqslant cap_{CPU}(l), \forall l \in N \quad (5.10)$$

Node and link capacity constraints are indicated in Equation 5.9 and Equation 5.10. Equation 5.9 assures that $\delta_{(i,j)}$ will be set to '1' when $\sum_{i',j' \in N} f^{(i,j)}_{i',j'} > 0$ (i.e. when substrate link (i,j) is active).

One substrate node per virtual node constraint:

$$\sum_{i \in N} x^i_{i^k} = 1, \forall i^k \in N^k \quad (5.11)$$

$$x^i_{i^k} = 0, \forall i \notin match(i^k) \quad (5.12)$$

Equation 5.11 assures that just one substrate node is assigned to each virtual node while Equation 5.12 makes sure that virtual node i^k is mapped to one SN node of its candidate set.

Active substrate node constraints:

$$\sum_{(i,j) \in L} \delta_{(i,j)} + \sum_{(j,i) \in L} \delta_{(j,i)} \geqslant \gamma_i \quad (5.13)$$

$$\sum_{(i,j) \in L} \delta_{(i,j)} + \sum_{(j,i) \in L} \delta_{(j,i)} \leqslant \gamma_i * MaxDegree \quad (5.14)$$

Finally, Equation 5.13 and Equation 5.14 show the correlation between variables δ and γ, setting a node as active when at least one of its incoming or outgoing links is also active.

5.3.2 *Performance evaluation*

In this section, a methodology to create load-targeted scenarios is used and the results of the EA-VNE-LB solution are presented and discussed.

5.3.2.1 *Scenario creation*

The load-targeted scenarios used for the evaluation are parameterized by:

- topology creation performed just as in Section 3.6.3.1;
- resource and demand deployment performed just as in Section 3.6.3.1.

5.3.2.2 *Evaluation*

SIMULATION SETTINGS In this work, CPU cycles are considered as node resource, denoted by cap_{CPU}, and bandwidth as link resource, denoted by cap_{BW} in the substrate network. For the uniform distribution of these values $cap_{CPU}^{max} = 100$ and $cap_{BW}^{max} = 100$ are chosen following related work [45, 41].

Due to the complexity and unaffordable execution time of MIP solutions for large network sizes, a small number of substrate and virtual nodes $|N| = 15$ and $|N^k| = 5$ were chosen. To explore the impact of consolidation of virtual networks, and the possibilities of power saving, network scenarios with $k^{max} = 80$ VNRs were considered. To study the VNs acceptance ratio and its implications in power saving, all scenarios were considered for different loads $\rho \in \{0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. The average load depicts the ratio of demands generated by the set of VNRs to the SN capacity. As Waxman topology generation is probabilistic, 50 runs for each set of scenario parameters were performed.

According to Equation 3.5 with $E[P(u,v)] \approx 0.25$, $E[|L|] \approx 53$, and therefore there are $\frac{E[|L|]}{|N|} \approx 4$ links for each node.

As mentioned before, network resources are considered to be homogeneous with respect to their power consumption. Hence, to calculate the power consumed by each network element, the SN is supposed to be confined to the transport segment of InPs with links of maximum 1000 Mbps that, according to [167], consume approximately 4 Watt (2 Watt for each end interface), thus $PowerLink_{(i,j)} = 4, \forall (i,j) \in A$. In turn, the power consumption of a node is given by $P = C^{2/3}$ [171], where P [Watt] is power consumption and C is switching capacity. For this case, C =4000 Mbps to switch the link traffic of each node, therefore, $PowerNode_i = 4000^{2/3} \approx 252$ Watt, $\forall i \in N$. When all the elements of the SN are active, the maximum average power consumption accounts for $PowerNode_i \cdot |N| + PowerLink_{(i,j)} \cdot |L| \to 252 \cdot 15 + 4 \cdot 53 = 3992$ Watt.

COMPARISON METHOD AND METRICS VNE approaches evaluated in this work are described in Table 5.2. These algorithms were implemented in the ALEVIN framework [FBD+11, DSS+11] which uses GLPK [49] to solve the formulated MIPs.

Table 5.2: Compared approaches

NOTATION	APPROACH DESCRIPTION
OCA	Optimal cost approach that map each VNRs by using minimum resources in the SN while balancing the load in it. This approach is similar to the one proposed in [51] but improved by introducing load balancing ($w_e = 0$ and $w_c = 1$ in the MIP formulation)
EA-VNE	Proposed energy-efficient VNE approach without considering load balancing [BHD+12] (using $w_e = 1$ and $w_c = 0$)
EA-VNE-LB	Proposed load balancing based energy-efficient approach [BH13] by using $w_e = 1$ and $w_c = 10^{-5}$ that embeds VNRs in the smallest feasible set of SN equipment, trying at the same time to use the resources with greater remaining capacity

The difference among those VNE approaches is better illustrated in Figure 5.2 where three VNRs, each composed of two nodes and one link, have to be embedded in a SN with four nodes and four links. The Optimal Cost Approach (OCA) is able to map all VNRs but it does not produce power savings because all substrate resources are occupied after the embedding (Figure 5.2a). EA-VNE produces power savings but, as it does not perform load balancing, embeds VNR 1 in substrate nodes A,B leaving substrate link A-B depleted, preventing the successful mapping of VNR 2 (Figure 5.2b). On the contrary, thanks to load balancing, EA-VNE-LB embeds VNR 1 to nodes A, D which leaves room for the embedding of VNR 2 and VNR 3 (Figure 5.2c), while saving the same amount of power than EA-VNE.

To evaluate the performance of OCA, EA-VNE and EA-VNE-LB, eight metrics are used:

- VN average acceptance percentage: This metric measures the percentage of successfully mapped VNRs, also called acceptance ratio.
- Total Power Consumption: This metric shows the power consumed by the substrate network after the set of VNRs is embedded.
- Power Consumed by nodes: This metric presents the power consumed by all SN nodes after embedding the set of VNRs.
- Power Consumed by links: This metric presents the power consumed by all SN links after embedding the set of VNRs.

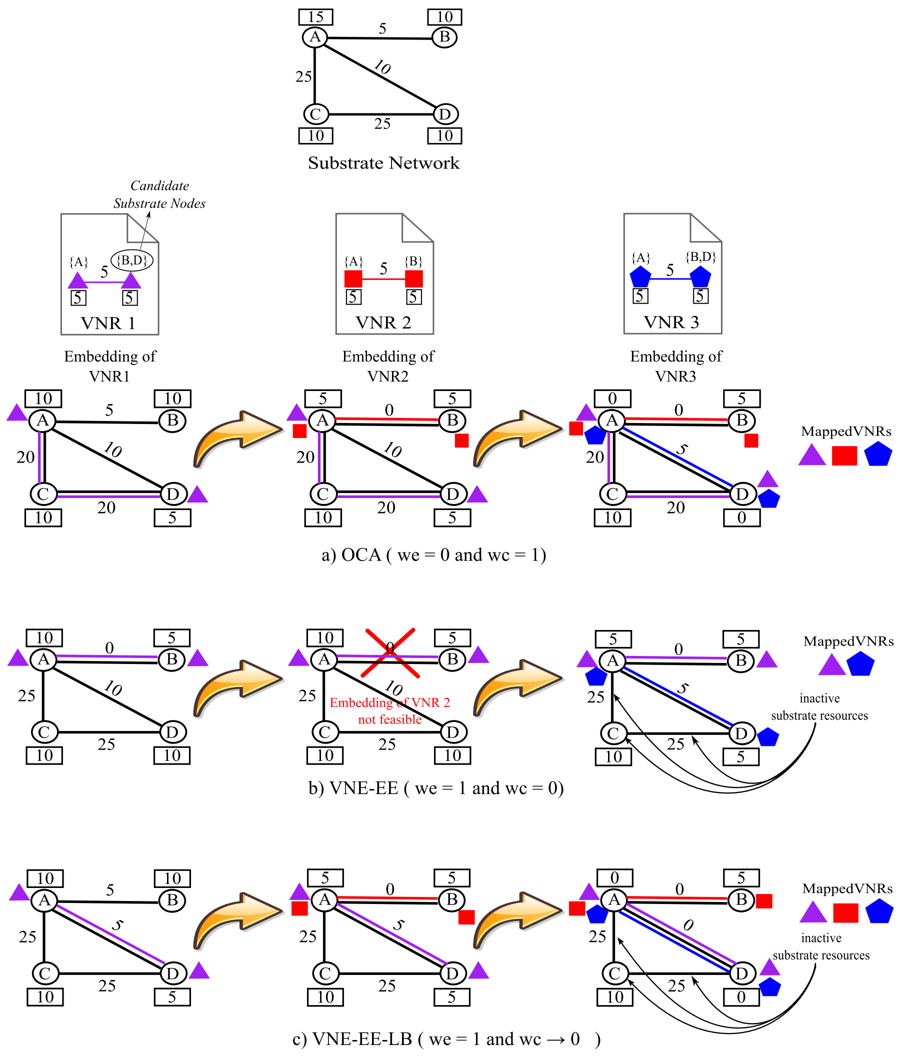

Substrate Network
Candidate Substrate Nodes
VNR 1
VNR 2
VNR 3
Embedding of VNR1
Embedding of VNR2
Embedding of VNR3
MappedVNRs
a) OCA (we = 0 and wc = 1)
Embedding of VNR 2 not feasible
inactive substrate resources
b) VNE-EE (we = 1 and wc = 0)
c) VNE-EE-LB (we = 1 and wc → 0)

Figure 5.2: Load balancing in different VNE approaches

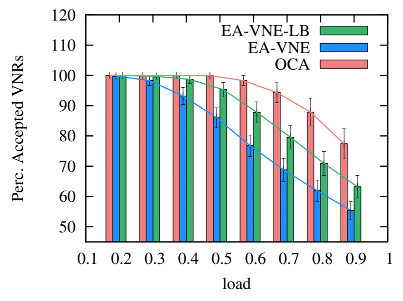
- Average embedding cost per mapped VN: The embedding cost is calculated as the amount of resources spent in the SN to map a VNR k.

$$\text{cost}(\text{VNR}^k) = \sum_{(i,j)\in L} \sum_{i',j'\in N} f_{i',j'}^{(i,j)} + \sum_{i\in N} \sum_{i^k\in N^k} x_{i^k}^{i} \text{dem}_k^{CPU}(i^k)$$

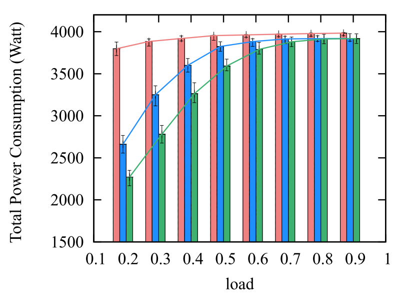
- Utilization in SN nodes: It measures the utilization of active SN nodes.

- Utilization in SN links: It measures the utilization of active SN links.

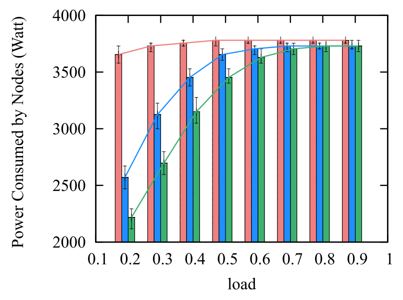
- Runtime per mapped scenario average: It measures the time needed to map a load-based scenario.



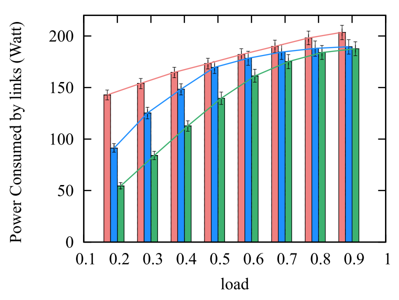
(a) VNRs Acceptance Ratio



(b) Power Consumption



(c) Power Consumption in Nodes



(d) Power Consumption in Links

Figure 5.3: OCA,EA-VNE and EA-VNE-LB performance

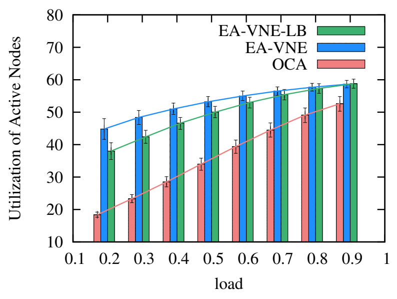
RESULTS Evaluation results are shown in Figure 5.3 and Figure 5.4. These results are displayed with a confidence level of 95%. The key observations can be summarized as follows.

1. *Great power savings with EA-VNE-LB*: Figure 5.3b shows that, after performing the embedding, EA-VNE-LB solution obtains greater power savings with respect to OCA, especially for low loads. For instance, when the load is 0.2 EA-VNE-LB consumes 40% less power than OCA. In fact, the power consumption in OCA stays almost constant as the load grows and its value is very close to 3992 Watt, indicating that nearly all SN elements are activated due to the VNE performed by OCA.

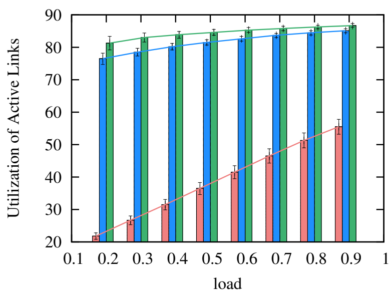
   When compared with cost-based VNE solutions, the energy-aware VNE MIP approach reaches up to 40% power savings

   In addition, EA-VNE-LB noticeably decreases the power consumption when compared with EA-VNE for all loads. For instance, when the load is 0.2, EA-VNE-LB saves 10% more power than EA-VNE. Power savings come with a slight increase of the embedding cost of the energy-aware solutions (see Figure 5.4c). However, as EA-VNE-LB includes the embedding cost in the objective function to resolve the ties when several optimal solutions consume the same amount of power, it reduces the cost needed to embed a VNR compared to EA-VNE.

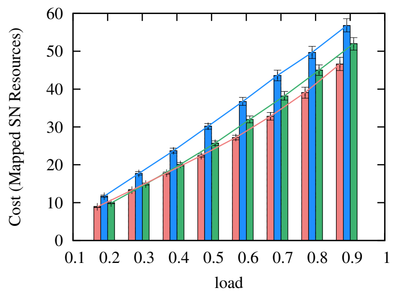
2. *Tiny contribution of SN links to the total power consumption*: Figure 5.3c and Figure 5.3d show that almost 100% of the power consumption in the SN is generated by nodes. This is



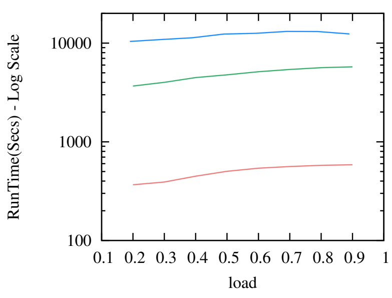
(a) Utilization of SN Nodes



(b) Utilization of SN Links



(c) Embedding Cost per Accepted VNR



(d) Runtime per Scenario

Figure 5.4: OCA,EA-VNE and EA-VNE-LB embedding cost, utilization and runtime

highly predictable with the values taken by $PowerNode_i$ and $PowerLink_{(i,j)}$. In the best case (when the load is 0.9), the power consumed by SN links is around the 5% of the total power consumption in all the evaluated approaches. However, as shown in the figures, the energy-aware solutions save power both in links and nodes. It is worth noting that although EA-VNE-LB consume less power in links and nodes, for all loads, than EA-VNE; their consumption tends to match for very high loads (0.8 and 0.9) because the room for power saving is almost nonexistent in such loaded SNs.

There is a trade-off between acceptance ratio and power savings. When the network is highly loaded, power savings come at the cost of lower acceptance ratios

3. *Trade-off between acceptance ratio and power savings*: EA-VNE-LB improves the percentage of accepted VNRs when compared against EA-VNE (see Figure 5.3a). Even with these improvements, EA-VNE-LB acceptance ratio is lower than the one of OCA for high loads. This situation creates a trade-off between power savings and acceptance ratio because, while for low loads EA-VNE-LB obtains great power savings maintaining the same acceptance ratios of OCA, it becomes hard to allocate VNRs with greater demands in a lower number of highly loaded SN equipment that is more prone to reject future VNRs. Figure 5.3b shows that, under loads from 0.6 to 0.9, EA-VNE-LB can consume up to 2% less power than OCA, while the percentage of accepted VNRs can be up to 17% lower than OCA's. This trade-

off should lead the ISP to choose the most convenient VNE approach depending on its goals and on the SN load.

4. *Better resource utilization in the SN for energy-aware approaches*: Figure 5.4a and Figure 5.4b show how, due to the concentration of the virtual demands, the energy-aware approaches have a significantly better utilization of nodes and links than OCA. In nodes, EA-VNE has an utilization until 25% (when $\rho = 0.2$) higher than OCA. Due to load balancing, EA-VNE-LB has a better distribution of the load demand and, as a consequence, a lower utilization than EA-VNE. In links, the differences between energy-aware strategies and OCA can be of almost 60% (when $\rho = 0.2$).

5. *EA-VNE-LB is more scalable than EA-VNE*: As previously mentioned, even for small networks (as the ones used in the performed experiments), a remarkable difference in runtime of EA-VNE and EA-VNE-LB when compared with OCA can be observed (cf. Figure 5.4d[1]). Even noting that EA-VNE-LB improves the execution time with respect to EA-VNE, none of these exact solutions (including OCA) scale for big networks but can be used as a baseline to evaluate new heuristic-based energy-aware VNE approaches.

Due to the inherent complexity of MIPs, the energy-aware VNE MIP approach is not scalable for medium and big networks

5.4 ENERGY-EFFICIENT EMBEDDING RECONFIGURATION

Previous section showed that EA-VNE-LB improves the power consumption in the network but, as traffic load grows, the VN acceptance ratio decreases. The main economic objective of the InP is to reject the minimum number of VNRs, therefore, a long-term embedding strategy based on EA-VNE-LB will not be profitable for high loads. A desirable approach would be to reduce power consumption while minimizing the number of rejected VNRs. Another shortcoming of EA-VNE-LB is its excessive execution time that could deteriorate the algorithm effectiveness for medium to large problem sizes.

The best strategy to minimize the VNRs rejection rate is the optimal cost embedding. Embedding cost is defined as the amount of resources spent by the substrate network to realize the mapping of a VNR. When the cost of mapping a VNR is minimized, the successful mapping of the next VNR is more likely to occur as the SN counts with more resources.

EA-VNE is not scalable and, for high loads, reduces the acceptance ratio. These shortcomings are solved by a VNE heuristic that relocates already mapped VNRs to minimize the power consumption

Over time, as new VNRs arrive and are embedded and others expire and release their resources from the SN, the embedding becomes fragmented and the ratio of accepted VNRs diminishes, resulting in a long-term revenue abatement. Current reconfiguration

1 Confidence intervals are not shown for time figures as they are not representative in logarithmic scales

VNE approaches perform a periodic reconfiguration of the mapped VNRs in order to optimize and reorganize the resource allocation [30]. One of the contributions of this thesis, presented in this section, is a **C/D/R** Energy-Aware VNE Reconfiguration Heuristic to relocate already mapped VNs trying to minimize the power consumption. This reconfiguration approach is not motivated by the fragmentation of the embedding but for its energy inefficiency. EA-RH tries to remap, in short runtime, current embeddings to maximize power savings without rejecting any mapped VN. The idea is to find the SN nodes and links that are susceptible to be left free and to relocate their demands in different regions of the SN. In this context, the concept of stress is used. As introduced by Zhu and Ammar [80], the stress of a SN node or link is defined as the number of virtual instances mapped on top of it.

The relocation implies the migration of virtual nodes and links from one place to another in the SN network. The undesirable effects of virtual routers and links migration in the QoS of working transmissions can be mitigated using strategies such as the ones proposed in [10, 172]:

- In [10], Wang et al. propose an approach to perform live migration of virtual routers that: 1) is transparent to routing protocols and 2) does not impact the performance of the data traffic. The proposed approach is able to migrate a virtual router to a different physical router without disrupting the flow of traffic or changing the virtual topology. This is done by enabling the separation of the control and data plane of the routers; the approach only migrates the control plane while forwarding data in the old data plane. When the migrated control plane can start at the new location, the new data plane is populated in parallel.
- In [172], Lo et al. propose the live migration of a complete VN using the aforementioned approach as a subroutine of the complete VN migration. It tries to determine the best schedule (sequence) of virtual router migrations to move a complete VN to a new physical location. The proposed approach introduce heuristic algorithms to minimize the cost and duration of the VN migration.

5.4.1 *Energy-aware relocation heuristic (EA-RH)*

The proposed greedy heuristic is divided into two steps: node and link relocation.

5.4.1.1 *Node relocation*

The main idea behind node relocation is to reconfigure current embeddings to leave the SN with as many inactive substrate nodes as

Procedure 1 Node Relocation Heuristic

```
Input: SN, set of embedded VNRs, relocating percentage
 1: Find MaxNodeStress, numberStressedNodes
 2: for all i=1 to MaxNodeStress do
 3:     Find ordered set SNODES of substrate nodes with stress = i
 4:     for all sNode ∈ SNODES do
 5:         Unmap the set VNODES of virtual nodes and its incident virtual links
            mapped on top of sNode
 6:         for all vNode ∈ VNODES do
 7:             Find the best candidate SN node bestSNode based on its SUITABILITY
                with regard to vNode
 8:             if bestSNode found then
 9:                 Remap vNode to bestNode and its incident virtual links to the
                    shortest energy paths
10:             else
11:                 Unmap previous mappings of VNODES, map them to their
                    previous location in sNode and GOTO 4
12:             end if
13:         end for
14:         if remapped SN nodes ⩾ numberStressedNodes * relocating percentage then
15:             Node relocation finished, FINISH PROCEDURE
16:         end if
17:     end for
18: end for
```

possible. To do so, substrate nodes with low stresses are identified, virtual nodes on top of them are unmapped and suitable substrate nodes in the SN to relocate those virtual nodes and their incident virtual links are looked for.

Node relocation algorithm is detailed in Procedure 1. The algorithm tries to relocate a certain percentage of the current lowly stressed nodes. The idea is to migrate the embeddings of low stressed SN nodes to higher stressed ones. To do that, the algorithm starts a loop from the substrate nodes with the lowest stress (stress = 1) to the ones with the maximum node stress. The set of SN nodes with the same stress are ordered from lowest to highest by the value of the virtual demand mapped on top of them, that is, the first will have the lowest mapped demand and, consequently, its relocation will be easier. For each SN node, the proposed approach unmaps the virtual nodes on top of it and their respective virtual links. Each virtual node is then relocated in the substrate node with the highest SUITABILITY value. When the number of deactivated (or attempted to be deactivated) substrate nodes is greater or equal to the relocating percentage of total stressed nodes (an input parameter), the algorithm finishes its execution.

The suitability is calculated in all the available substrate nodes for each virtual node that has to be relocated. It is defined as the sum of the suitability of the substrate node itself (NS) and the suitability of the substrate paths (PS), starting or ending in this substrate node, that map the virtual links incident to the virtual node being relocated.

$$suit(snode \in V, vnode \in V^k) = NS(snode) + PS(snode, vnode)$$

$NS(snode) = stress_{snode} * NR_{CPU}(snode)$

$PS(snode, vnode) = \sum_{vlink \in incVLinks_{vnode}} \frac{PathRemBW(SEP(snode, vlink))}{InactiveLinks(SEP(snode, vlink))+1}$

where:

Node Suitability (NS) depends on the stress and the available CPU resource of the node. The main goal of the proposed heuristic is to relocate the virtual nodes in high stressed nodes that have the maximum amount of available resources, so they will be able to relocate more virtual nodes for future reconfigurations.

Paths Suitability (PS) considers, for each virtual link incident to the virtual node being relocated, the Shortest Energy Path (SEP) in the SN that maps it. SEP is calculated by weighting each substrate link considering whether it is active or not.

$$weight_{(i,j)} \begin{cases} 1 \iff (i,j) \in ActLinks \\ 100 \iff (i,j) \in InactLinks \end{cases}, \forall (i,j) \in V$$

In this way the heuristic promotes the relocation of virtual links in substrate paths with active links. $PathsSuit$ also considers the remaining BW resource and the number of inactive links in the path, where $PathRemBW(path) = \min_{(i,j) \in path} cap_{BW}(i,j)$, choosing paths with greater remaining resources and less number of inactive links.

Figure 5.5 depicts a simple example of a virtual node relocation. Figure 5.5a shows the embedding of two virtual networks on top of the SN. Virtual node relocation starts in the substrate nodes with the lowest stress, for this example, suppose that the virtual node $\triangle^*$ hosted by a substrate node with $stress = 1$ has the lowest demand and, therefore, is chosen for relocation. To relocate it, the algorithm unmaps $\triangle^*$ belonging to the VNR 1 and its incident links as shown in Figure 5.5b. After calculating the suitability of all substrate nodes to remap virtual node $\triangle^*$, the rightmost substrate node is chosen due to its high suitability. The relocation of the virtual node implies the relocation of its incident virtual links as shown in Figure 5.5c. In this case, the virtual links are mapped on top of substrate active links. Due to the relocation of the virtual node, one node and two links of the SN are deactivated.

5.4.1.2 *Link relocation*

Even when the node relocation phase has been performed, there could be substrate links with low stress that can be deactivated by relocating the virtual links mapped on top of them to more stressed areas in the SN. The left side of Figure 5.6 shows the embedding of a VNR and the substrate link chosen to be left inactive (red link). The right side of the figure shows how the virtual link is remapped to a

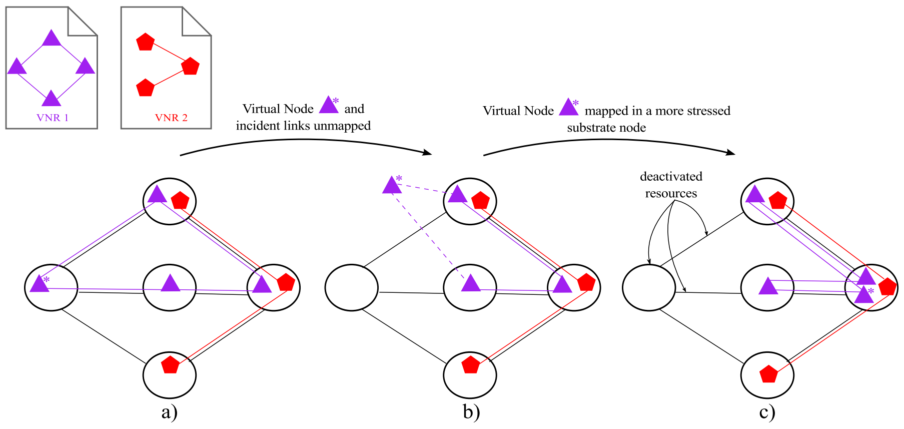

VNR 1
VNR 2
Virtual Node * and incident links unmapped
Virtual Node * mapped in a more stressed substrate node
deactivated resources
a)
b)
c)

Figure 5.5: Relocation of an already embedded virtual node

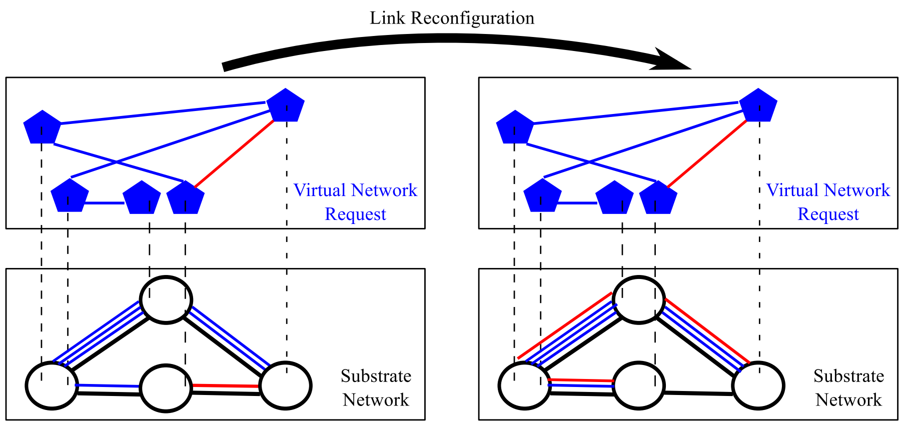


Figure 5.6: Relocation of an already embedded virtual node

new substrate path that is worse in terms of hops number, but better in terms of energy because it just contains active substrate links.

Link relocation, detailed in Procedure 2, is similar to the algorithm used for node relocation. In first place, it starts a loop looking for the lowest stressed substrate links (stress = 1) to the ones with the maximum stress and also order them by the value of their virtual demands. For each substrate link with low stress, the algorithm unmaps its mapped virtual links and subsequently starts a loop to remap each of those virtual links to the shortest energy path in the SN. When the number of deactivated (or attempted to be deactivated) SN links is greater or equal to the predefined relocating percentage, the link relocation phase finishes.

5.4.2 *Performance evaluation*

EA-RH is evaluated in the same set of load-based scenarios described in Section 3.6.3.1. To show that EA-RH is scalable and can run over networks of short, medium and large sizes, EA-RH is run to reconfigure the embeddings made by the following VNE algorithms:

Procedure 2 Link Relocation Heuristic

```
Input: SN, set of embedded VNRs, relocating percentage
 1: Find MaxLinkStress, numberStressedLinks
 2: for all i=1 to MaxLinkStress do
 3:     Find set SLINKS of substrate links with stress = i
 4:     for all sLink ∈ SLINKS do
 5:         Unmap the set VLINKS of virtual links mapped on top of sLink
 6:         for all vLink ∈ VLINKS do
 7:             MAP vLink to the shortest energy path in the SN
 8:         end for
 9:         if remapped SN links ⩾ numberStressedLinks * relocating percentage then
10:             Link relocation finished, FINISH PROCEDURE
11:         end if
12:     end for
13: end for
```

- OCA: Exact optimal cost approach used in Section 5.3.2. As OCA solves a MIP formulation, it does not scale well and short network sizes must be used in the simulations (see Section 5.3.2.2).

- RViNEPS: Cost-based VNE heuristic, introduced in [45, 36], based on Randomized VNE with Path Splitting. RViNEPS is scalable and can run in networks of medium/large size.

To show the impact of the number of nodes and links to relocate, the reconfiguration is performed with four different relocation percentages (30%, 45%, 60%, 90%) called EA-RH-30%, EA-RH-45%, EA-RH-60% and EA-RH-90%.

5.4.2.1 *Simulation settings*

EA-RH relocates the embeddings performed by OCA and RViNEPS. For OCA, the embedding results of Section 5.3 were taken. For the execution of RViNEPS, the scenario and topology creation methodology introduced in Section 3.6.3.1 were used. RViNEPS implementation needs to include location constraints in nodes. These constraints are introduced by means of a distance parameter which indicates the maximum distance a substrate node can be away from a virtual node to be considered its available candidate. Although the value of the maximum distance is not specified in [45, 36], a significant value, not too short that a virtual node is left without substrate candidate nodes, but not too long to avoid that all virtual nodes have the same candidates is considered. Section 3.6.3.1 already mentions that the chosen area size for the generation of SNs and VNRs is 1^2. Therefore, to choose an appropriate value of maximum distance, RViNEPS is run with different distance values (proportional to the maximum length): 0.25, 0.3, 0.35, 0.4, 0.45 and 0.5, and the results are compared based on the acceptance ratio. Table 5.3 shows that RViNEPS achieves better acceptance ratios for all loads when the maximum distance takes a value of 0.35.

Table 5.3: Average acceptance ratio for different maximum distances of RViNEPS

Max. Distance	Load 0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.25	90.0	77.3	62.9	51.0	43.1	36.4	32.4	28.2
0.3	94.8	80.0	63.0	51.5	43.2	36.5	32.4	28.2
0.35	**95.3**	**80.4**	**63.1**	**51.7**	**43.5**	**36.6**	**32.5**	**28.5**
0.4	95.1	78.9	60.2	49.5	41.5	35.3	30.7	26.6
0.45	94.7	78.0	59.3	48.5	41.2	34.3	29.4	25.7
0.5	93.8	78.0	59.3	48.5	41.2	34.3	29.4	25.7
	Average Acceptance Ratio (percentage)							

It is worth noting that, as RViNEPS is an heuristic algorithm, the runtime to embed a VNR in large topologies is affordable. Therefore, substrate and virtual network sizes are larger than the ones used for OCA. Consequently, the load-based scenarios for RViNEPS were generated with $|N^k| = 15$, all with the same SN size $|N| = 50$ and $k^{max} = 80$ VNRs, these values fulfill Equation 3.10. The power consumed by network resources in this topology is calculated as before (see Section 5.3.2.2). With $E[P(u,v)] \approx 0.25$, the average number of substrate links is $E[|L|] \approx 613$ (Equation 3.5), and $\frac{E[|L|]}{|N|} \approx 12$ links per node. The links are also assumed to have a maximum 1000 Mbps of capacity that consume approximately 4 Watt. Remember that the power consumption of a node is given by $P = C^{2/3}$. In this case, $C =$12000 Mbps to switch the traffic of node's links resulting in a power consumption of each active node equal to $12000^{2/3} \approx 524$ Watt. When all the elements of the SN are active, the maximum average power consumption accounts for $\mathit{PowerNode}_i \cdot |N| + \mathit{PowerLink}_{(i,j)} \cdot |L| \rightarrow 524 \cdot 50 + 4 \cdot 613 = 28,652$ KWatt.

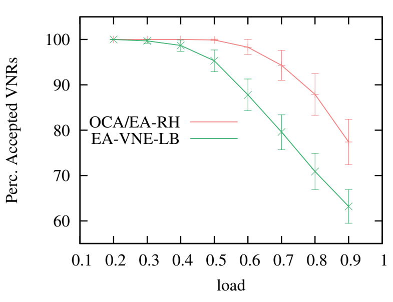
Values of cap_{CPU}, bandwidth cap_{BW} were uniformly distributed with $cap_{CPU}^{max} = 100$ and $cap_{BW}^{max} = 100$ following the settings in [45, 36]. As Waxman topology generation is probabilistic, $N = 50$ runs for each set of scenario parameters were performed.

5.4.2.2 *Comparison method and metrics*

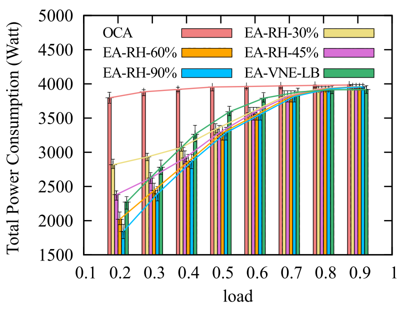
Here, the same set of metrics (see Section 5.3.2.2) were used to evaluate the EA-RH behavior. RViNEPS was also implemented in the ALEVIN software.

5.4.2.3 *Results*

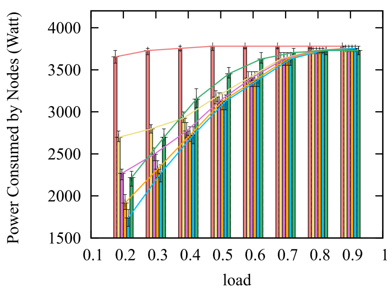
Evaluation results are shown in Figure 5.7, Figure 5.8 Figure 5.9 and Figure 5.10 with a confidence level of 95%. They are divided into two



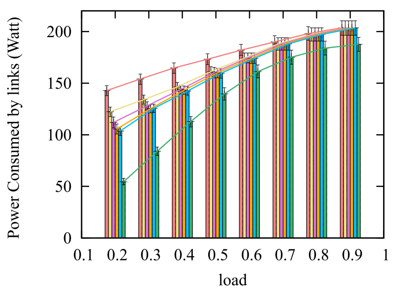
(a) VNRs Acceptance Ratio



(b) Power Consumption



(c) Power Consumption in Nodes



(d) Power Consumption in Links

Figure 5.7: Performance evaluation OCA and EA-RH

separated subsections. In first place, EA-RH is run over the embedding results of the exact OCA solution and then it is run over the embedding results of the RViNEPS heuristic algorithm.

EA-RH OVER OCA In the figures to evaluate the performance of EA-RH over OCA, the results of EA-VNE-LB are also included, so the power savings of EA-RH can be compared against the ones of the exact ILP formulation. Intuitively, one can deduce that, on the one hand, for high loads; as the acceptance ratio does not change when EA-RH is applied, the relocation of VNRs becomes harder and the power savings are shorter than the ones obtained for low loads. On the other hand, power savings obtained by EA-VNE-LB tend to be noticeable even during high loads due to the lower acceptance ratio in these situations which is not the situation in EA-RH.

The results for the OCA algorithm obtained in Section 5.3.2, relocated using EA-RH are shown in Figure 5.7. From these results, the following are the key observations.

1. *Great power savings maintaining acceptance ratio*: EA-RH achieves considerable power savings. When the network is lightly loaded, EA-RH-90% consumes 50% less power than OCA (see Figure 5.7b).

   It is worth noting that the power consumption is lower with EA-RH-60% and EA-RH-90% than the one obtained by EA-VNE-LB for loads up to 0.9, with EA-RH-45% for loads between 0.3 and

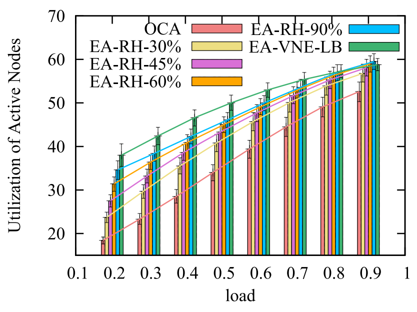
0.8 and with EA-RH-30% for loads between 0.4 and 0.8. This is an excellent outcome taking into account that the acceptance ratio of the EA-RH solutions is always equal or better than the one of EA-VNE-LB. The explanation of this behavior comes from the fact that offline approaches, like EA-RH, act over the set of embedded VNRs at the same time while online approaches, like RViNEPS embed the arriving VNR without considering the characteristics of the next request and, consequently, perform a mapping that may not optimally adjust the SN for the subsequent VNR.

When compared with cost-based VNE solutions, EA-RH reaches up to 50% power savings. It even outperforms the energy-aware MIP VNE solution

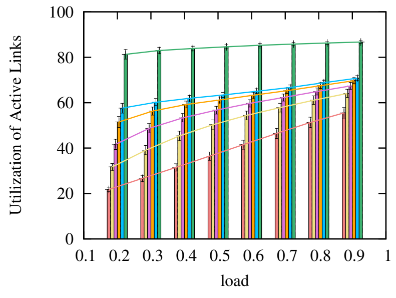
At the highest loads (0.8, 0.9), EA-VNE-LB's power consumption is slightly lower or equal than EA-RHs'. EA-VNE-LB obtains this positive result at the cost of a worse EA-VNE-LB acceptance ratio (see Figure 5.7a), this is also reflected in the embedding cost of EA-VNE-LB that starts to be higher than the EA-RH's from loads of 0.6 (see Figure 5.8c).

2. *Greater power savings in nodes than in links with EA-RH*: As analyzed in Section 5.3.2, the contribution to the total power consumption made by SN links is almost negligible. EA-RH takes advantage of this situation by deactivating as many nodes as possible in its first step and subsequently, when the node mapping is concentrated in a short node-set, deactivating as many links as possible. This division in the heuristic plays in favor of nodes deactivation. Figure 5.7c and Figure 5.7d show how the difference in power consumption between EA-RH approaches and OCA is greater in nodes than in links. In fact, the power consumption in nodes for some EA-RH relocation percentages (60% and 90%) is lower than the one in EA-VNE-LB for almost all loads, while the power consumption in links for EA-VNE-LB is lower than the one of EA-RH for all loads and all relocation percentages.

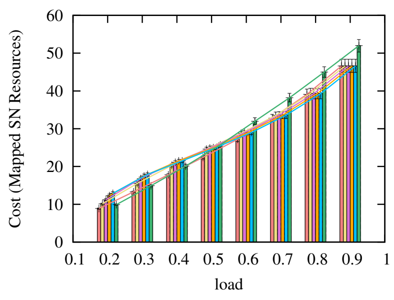
3. *High impact of relocating percentage in EA-RH*: Figure 5.7b shows that different relocating percentages affect the amount of power savings. The major differences can be seen in the intervals between 30% and 45% and between 45% and 60%. For low loads, EA-RH-45% saves up to 12% more power than EA-RH-30%. Similar differences, thought slightly lower, can be observed between EA-RH-60% and EA-RH-45%. However, the difference is not the same between EA-RH-60% and EA-RH-90%, here the savings obtained by fixing the relocating percentage in 90% are short for low loads (up to 5%) and almost nonexistent when the load grows. This is caused by the increased difficulty to perform new reconfigurations after relocating a big percentage (60%) of nodes and links.



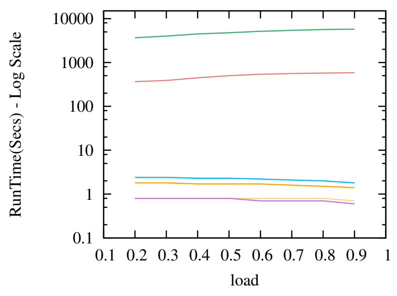
(a) Utilization of SN Nodes



(b) Utilization of SN Links



(c) Embedding Cost per Accepted VNR

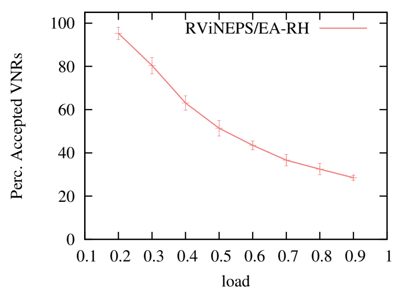


(d) Runtime per Scenario

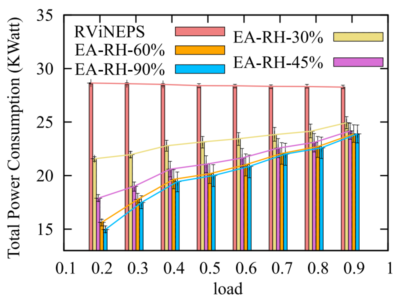
Figure 5.8: OCA, EA-RH utilization, embedding cost and runtime

4. *Utilization is increased in EA-RH compared with OCA*: Figure 5.8a and Figure 5.8b show how, due to the concentration of the virtual demands after reconfiguration, EA-RH has a better utilization of nodes and links than OCA, specially for high relocation percentages. In nodes, EA-RH-90% has an utilization until 20% (when $\rho = 0.2$) higher than OCA. The relocation tries to concentrate the demands in a few set of nodes and links. However, the utilization of active equipment is higher in EA-VNE-LB both in nodes and links than in EA-RH. This is due to the acceptance ratio, for high loads, EA-VNE-LB rejects more VNRs due to the high concentration of VNRs and therefore has a high utilization both on nodes and links.

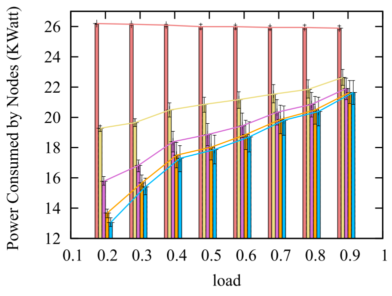
5. *Shorter runtime of EA-RH*: Figure 5.8d shows the enormous difference between the runtime used by OCA and EA-VNE-LB to map a set of VNRs compared with the time used by EA-RH to relocate them. It is worth clarifying that EA-RH is performed after the embedding (in this case OCA) strategy has been executed, therefore the total time to map and relocate the VNRs is the sum of OCA's and EA-RH's runtime. Reconfiguration runtime of EA-RH is almost insignificant and, as a consequence, periodic energy-aware reconfigurations are affordable. Although the runtime of EA-RH-30% and EA-RH-45% is lower than the one spent by EA-RH-60% and EA-RH-90%, one can conclude



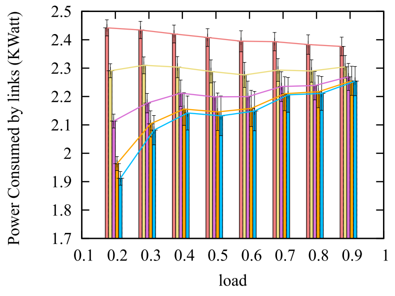
(a) VNRs Acceptance Ratio



(b) Power Consumption



(c) Power Consumption in Nodes



(d) Power Consumption in Links

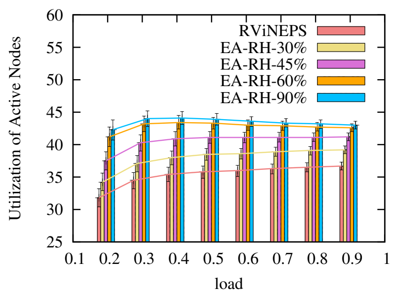
Figure 5.9: Performance Evaluation RViNEPS and EA-RH

that the difference in power savings justifies the adoption of high relocating percentages.

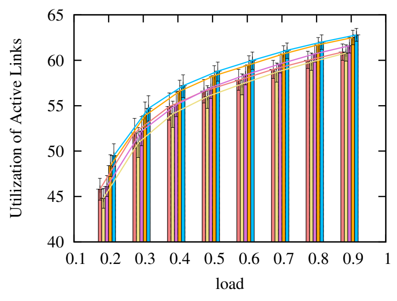
EA-RH OVER RVINEPS From the results of EA-RH reconfiguration over RViNEPS, shown in Figure 5.9, the following observations can be made.

1. *Low acceptance ratios when load increases* (Figure 5.9a): Until now, there had not been research attempts to evaluate the performance of VNE heuristics under increasing load produced by the VNRs. In this context, RViNEPS shows very low acceptance ratios when the load increases. Even with low loads (0.2), the acceptance ratio does not arrive to 100%. When the load is high (0.9) RViNEPS just embeds the 29% of the VNRs. Even with such low acceptance ratios, the power consumption in RViNEPS remains almost constant for all loads (see Figure 5.9b) and very close to 28,652 KWatt. This situation indicates that, even rejecting a big percentage of VNRs, RViNEPS activates nearly all SN resources.

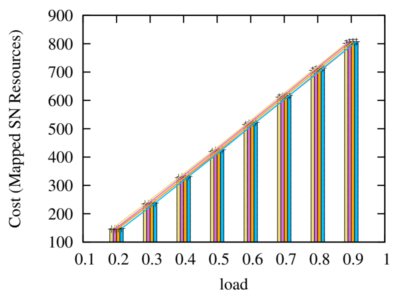
2. *Great power savings with EA-RH* (Figure 5.9b): Significant power savings are achieved when EA-RH is applied over RViNEPS. For low loads the results are similar to the ones obtained over OCA. When the load is low (0.2), EA-RH-90% consumes 49% and EA-RH-60% 46% less power than RViNEPS.



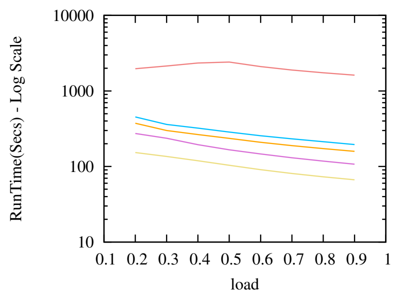
(a) Utilization of SN Nodes



(b) Utilization of SN Links



(c) Embedding Cost per Accepted VNR



(d) Runtime per Scenario

Figure 5.10: RViNEPS, EA-RH utilization, embedding cost and runtime

However, unlike EA-RH over OCA,the application of EA-RH over RViNEPS achieves noticeable power savings in situations of high load. At the highest load (0.9), EA-RH-90% and EA-RH-60% consume 15% less power than RViNEPS. The difference between power savings, under high loads, obtained by EA-RH over OCA and RViNEPS can be explained by the poor behavior of RViNEPS under high loads. Figure 5.9a shows that under very high loads, the acceptance ratio of RViNEPS is very poor (71% of the VNRs are rejected when the load is 0.9). Therefore, RViNEPS leave some regions of the SN under low utilization and EA-RH relocates the virtual demands mapped on top of those regions into highly utilized ones. Figure 5.10c confirms that RViNEPS leaves SN regions with low utilization because the execution of EA-RH does not increase the embedding cost of the relocated VNRs.

3. *Reconfiguration of nodes produces greater power savings that reconfiguration of links*: The results presented in Figure 5.9c and Figure 5.9d allow us to obtain the same conclusion for EA-RH over RViNEPS than over OCA. EA-RH's first step is focused in the deactivation of as many nodes as possible and then, when the node mapping is concentrated in a short node-set, it tries to deactivate as many links as possible. This division in the heuristic is explained by the fact that nodes consumes more power than

links. Similar to EA-RH over OCA, it is easy to note that the difference in power consumption between EA-RH and OCA is greater in nodes than in links.

4. *Noticeable differences when the relocation percentage changes for low loads*: Figure 5.9b confirms that the relocation percentage is a parameter of paramount importance when running EA-RH.

   When the load is 0.2, EA-RH-90% consumes 24% and 11% less power than EA-RH-30% and EA-RH-45% respectively. However, power savings in EA-RH-90% with respect to EA-RH-60% are very short, almost nonexistent for high loads. As in EA-RH over OCA, it is possible to conclude that after trying to relocate a certain amount of nodes and links (60% in this case), saving power gets harder because active elements (nodes and links) become more stressed and, as a consequence, unable to be relocated.

5. *Utilization is increased in EA-RH with respect to RViNEPS*: Figure 5.10a and Figure 5.10b show how, due to the concentration of the virtual demands after reconfiguration, EA-RH has a better utilization of nodes and links than RViNEPS, specially for high relocation percentages. As EA-RH reconfigures nodes in first place and links subsequently, the difference in utilization of links between EA-RH and RViNEPS is lower than the difference observed in the utilization of SN nodes.

6. *EA-RH is scalable* (Figure 5.10d): It is worth noting that the runtime of EA-RH decreases when the load grows. This is due to the fact that with more load, RViNEPS accepts less VNRs and there are less mapped instances to reconfigure.

   EA-RH is scalable. It was applied over exact and heuristic cost-based VNE approaches. Results show an almost negligible relocation runtime

   When compared with the runtime needed to relocate small networks (see Figure 5.8d), the one needed for larger networks is not negligible. However, it is acceptable to perform periodic reconfigurations triggered by the detection of inefficient power consumption in the SN. The highest execution time to reconfigure the mapped VNRs (0.2 load for EA-RH-90%), is a quarter of the time spent by RViNEPS to map all the VNRs. For high loads (0.8 or 0.9), as the results of EA-RH-60% are almost the same than the ones of EA-RH-90% and the time required to perform the relocation is very small when compared with the one spent by RViNEPS (for a load of 0.9 is one eleventh), it is a good idea to use a relocation percentage of 60%.

From these results one can conclude that EA-RH reaches considerable power savings both in small/medium (embedded with exact VNE approaches) and in large SNs (embedded with heuristic VNE approaches). Bigger power savings are reached when the network load is low because there is more room in the stressed network elements to admit the relocation of virtual elements hosted in low stressed areas of the SN. Finally, it is also important to highlight that EA-RH is a scalable strategy to reduce power consumption. Unlike EA-VNE-LB, EA-RH can run in affordable times in networks with short, medium or large sizes.

5.5 CONCLUSION

One of the main outcomes of this thesis, is the introduction of the energy-aware virtual network embedding problem and its optimal formulation as a mixed integer program.

Besides, the EA-VNE power efficient MIP model was improved by introducing embedding cost and load balancing to the formulation. By means of intensive simulations, the new proposed approach, called EA-VNE-LB, is shown to outperform EA-VNE in terms of power savings and acceptance ratio. Simulations also shown that EA-VNE-LB reaches big power savings when compared with an exact cost-based VNE approach; power consumption can be reduced up to a 40% for low loads.

It was also demonstrated that, under large networks, EA-VNE-LB reduces its effectiveness, and also, that the power consumption is minimized at the cost of reducing the VNRs acceptance ratio creating a trade-off between acceptance ratio and power savings. For those reasons, an energy-aware reconfiguration heuristic trying to minimize the power consumption by relocating the virtual nodes and links of already mapped VNRs was also introduced. Through intensive simulations EA-RH showed shorter running times and noticeable power gains (up to 50%), maintaining the VNRs acceptance ratio, when compared with exact and heuristic cost-based VNE algorithms.

To reduce the complexity of the problem, this work considered homogeneous power consumption for nodes and interfaces. In future scenarios, the power consumption of the networking devices will not be constant when they are active, but will highly depend on their load. Additionally, the current EA-VNE-LB model is not aware of the high modularity of current network devices. In terms of power, the main goal should not be just minimize the number of active links, but to try to deactivate all the links terminated on the same line-card, so that it can be shut down or slept.

Consequently, future work can be devoted to extend EA-VNE-LB and EA-RH, so that so that the modularity of network devices and the load dependent power consumption are included.

The evaluation environment used in this evaluation was off-line and allowed to use a certain load and different VN sizes for evaluation. However, future work can be also devoted to explore online evaluation scenarios to see how EA-RH behaves throughout the time and which will be the optimal interval to execute periodic energy-aware VN relocations.

6 SOFTWARE TO DEVELOP, COMPARE, AND ANALYZE VNE ALGORITHMS

"Initiative is doing the right thing without being told"
Victor Hugo

This chapter is devoted to present ALEVIN: The Software Framework to Develop, Compare, and Analyze VNE Algorithms. ALEVIN was developed in the framework of the VNREAL project in collaboration with the universities of Passau and Würzburg.

ALEVIN was presented in [FBD+11] and its source code is available online [DSS+11]. Some of the results (shown in the previous chapters) of the research presented throughout this thesis were developed and evaluated in ALEVIN.

6.1 INTRODUCTION

Virtualization of network resources has been identified as key technology for Future Internet research and is actively used in current research testbeds [13, 173]. Network virtualization is recognized as an enabling technology for the Future Internet. Applying virtualization of network resources leads to the problem of mapping virtual resources to physical resources, known as "Virtual Network Embedding". Several algorithms attempting to solve this problem have been discussed in the literature, so far.

The optimality of VNE algorithms can be computed with regard to different parameters, ranging from load distribution over energy-efficiency to security of the networks. Comparison of VNE algorithms is difficult, as typically each algorithm optimizes only for a subset of all possible network parameters. In order to compare the efficiency of these algorithms, a set of appropriate metrics has to be found. Here, a framework is presented that allows to compare VNE algorithms according to different metrics.

This chapter introduces ALEVIN, a software framework to implement, analyze and compare VNE algorithms

The remainder of this chapter is organized as follows: Section 6.2 discusses the VNE problem in detail. Section 6.3 gives an overview of the different parameters that can be taken into account for the embedding. Section 6.4 presents the framework and metrics proposed in ALEVIN to compare different VNE algorithms. Finally, Section 6.5 concludes this chapter.

6.2 VIRTUAL NETWORK EMBEDDING ALGORITHMS

Algorithms solving the VNE problem come in two forms: offline algorithms and online algorithms. Offline algorithms take a given set of VNRs together with the description of a SN and compute a near optimal embedding for these requests. While this approach achieves good results with regard to optimality, it does not consider a dynamic arrival process of the VNRs (i.e. each VNR arrives at a different time and must be mapped in real time). Online algorithms, on the other hand, take VNRs on a FIFO-basis, redistributing virtual resources as the requests arrive. While this approach is better suited to deal with high dynamicity, it tends to come at the cost of less optimal solutions. Things might be further complicated by looking at heterogeneous resources in the SN (i.e. resources that are described by different sets of parameters or that have widely varying attributes). Adapting algorithms to support such an assumption therefore remains an open issue for now.

6.3 PARAMETERS

This section introduces a categorization of parameters for substrate as well as virtual nodes and links. In addition, a mapping of primary to secondary parameters is introduced. This was the first step in order to provide a categorization allowed us to present a detailed distinction of parameters presented in Section 2.2.2.

6.3.1 *Categorization*

Parameters for nodes and links are divided into three categories regarding their mutability, accessibility, and interdependency with other parameters:

The parameters for substrate and virtual nodes and links belong to three categories: Primary, Secondary and Indirect parameters

- *Primary parameters* are fixed properties of the node or link itself. These parameters – e.g. CPU or memory of a node, capacity and delay of a link – are capabilities that do not depend on another node's or link's state, but only on its utilization. Primary parameters can be directly requested by a virtual network. The set of primary parameters is denoted by PRI.
- *Secondary parameters* are derived attributes of a node or link. These parameters depend on the state of its node or link, i.e. other primary and/or secondary parameters. For instance, the loss probability of a node depends on its CPU load, which can, in turn, indicate the queue size at a router, and the transmission delay of a virtual link is the sum of the transmission and propagation delay of all traversed substrate links and the processing and forward delay of all traversed substrate nodes.

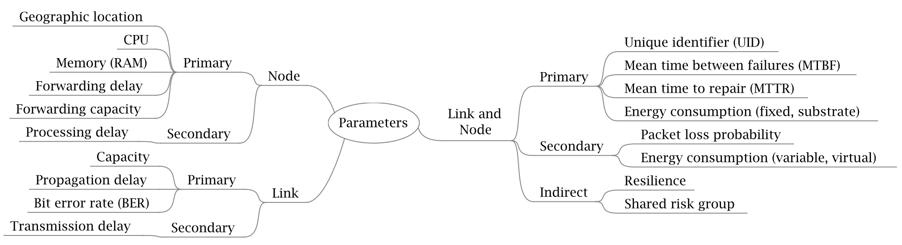


Figure 6.1: A categorization of parameters that can be considered in the VNE problem

To prevent cyclic dependencies, only the dependency on primary parameters is considered. Secondary parameters can also be directly requested by a virtual network. The set of secondary parameters is denoted by SEC.

- *Indirect parameters* are requested for a whole substrate path including traversed links and nodes. For instance, the demand for a resilient virtual link comprises that traversed substrate nodes and links fulfill certain failure probabilities and that the primary path is disjoint from the backup path in the substrate to avoid a *shared risk group*.
  These parameters are not specific to nodes and links. Hence, indirect parameters are not proportional to the number of virtual networks embedded. Indirect parameters can only be *indirectly* requested by a VNR.

Figure 6.1 illustrates some common parameters that can be considered in the VNE problem.

It is important to note, that some of the parameters have to be handled differently for substrate and virtual network entities. For instance, CPU resource is a primary (i.e. fixed) parameter for a substrate node while it is a secondary parameter on a virtual link which comprises the CPU resource on the nodes that are traversed in the substrate to realize this virtual link. As a consequence, it is important to define a mapping of virtual to substrate parameters including requests to *hidden hops*.

6.3.2 *Parameter mapping*

To declare secondary parameters, it is necessary to define how the mapping of primary parameters to secondary parameters is calculated.

6.3.2.1 *Primary to secondary*

PRI and SEC are defined as the set of primary and secondary parameters respectively. Secondary parameters are calculated from a set of surjective functions $map_{sec} : PRI^n \to SEC, sec \in SEC, 1 \leqslant n \leqslant |PRI|$.

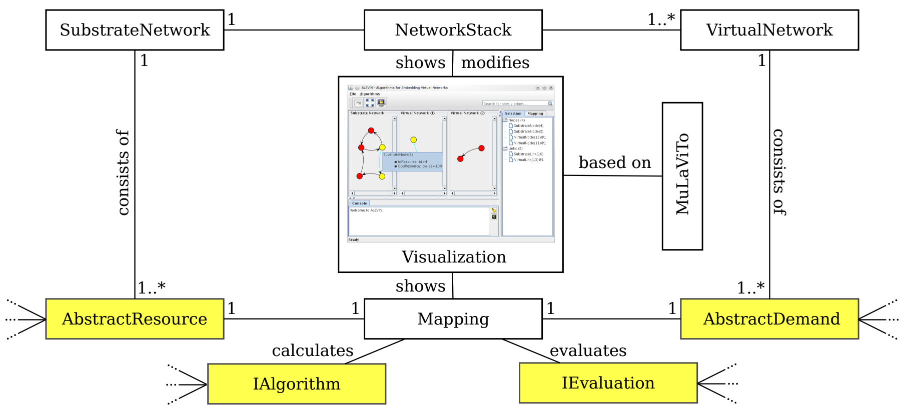


Figure 6.2: The framework ALEVIN and its modularity in parameters, algorithms, and evaluations

Once all primary substrate node and link parameters have been defined, a planning algorithm would compute each secondary parameter $sec \in SEC$, based on the node characteristics and using the mapping function map_{sec}.

6.4 SOFTWARE FRAMEWORK AND EVALUATION METRICS

One of the contributions of this thesis is the development a software framework to support the development of new VNE algorithms, ease their comparison and analysis, and apply arbitrary evaluation metrics. In this section, this framework – named *ALEVIN* – is introduced and its capabilities regarding parameters, algorithms, and evaluations are outlined. After implementing new algorithms and defining their optimization parameters, this framework can be used to test and compare those algorithms.

6.4.1 *ALEVIN – algorithms for embedding virtual networks*

The focus in the development of ALEVIN is on modularity and efficient handling of arbitrary parameters for resources and demands, as described in Section 6.3, and the support of integration of new and existing algorithms and evaluation metrics (cf. Chapter 2). For platform independency, ALEVIN is written in Java. ALEVIN's Graphical User Interface (GUI) and multi-layer visualization component is based on the MuLaViTo project [174] which enables us to visualize and handle the substrate and arbitrary virtual networks as directed graphs. Figure 6.2 depicts the architecture of ALEVIN and highlights the modular interaction of parameters for substrate as well as virtual networks, algorithms, and evaluation.

ALEVIN provides the ability to illustrate the deployment of resources in the SN and demands in an arbitrary number of VNs as well as the mapping of demands on resources calculated by a VNE algorithm. Moreover, ALEVIN can be used to create VNE scenarios as well as import and export them using an XML-based exchange format. ALEVIN is completely modular regarding the addition of new parameters to the VNE model. The use of the *visitor design pattern* in a sophisticated way makes it possible to avoid any casts to concrete demand/resource classes. Java's `instanceof` statement within the implemented algorithms. Thus, the number of parameters is not performance-relevant and a convenient implementation of arbitrary parameters is possible. To increase ALEVIN's modularity and to make it a flexible and extensible platform to compare existing and upcoming algorithms, the implementation of algorithms is kept independent of the resource/demand implementation. To that end, a simple interface is provided defining the rough structure of an algorithm and connecting its output to the GUI as illustrated in Figure 6.2. process demands sequentially with or without the ability to revoke an already mapped demand for later re-processing.

6.4.2 *Evaluation and comparison of VNE algorithms*

ALEVIN also is the basis for the evaluation and comparison of VNE algorithms. Therefore, ALEVIN provides a simple interface to add evaluation metrics which are independent of the implemented parameters and algorithms. This further emphasizes the modularity of ALEVIN.

During and after the calculation of a mapping of either one or several virtual network demands, different evaluation metrics can be applied. Table 2.2 list several evaluation metrics. These evaluation metrics can be combined in an optimization objective/strategy for a whole VNE scenario. For instance, a optimization objective for a VNE scenario might be to minimize the maximum link utilization, maximize the residual bandwidth capacity, minimize the total cost for the substrate provider, maximize the acceptance ratio, avoiding fragmentation or using load balancing minimize the overall energy consumption, or minimize run time. Due to the ability to combine arbitrary evaluation metrics and optimization objectives in combination with arbitrary parameters and a simple interface for VNE algorithms, ALEVIN is a powerful framework for the comparison and analysis of VNE algorithms regarding different evaluation metrics and optimization objectives.

6.5 CONCLUSION

With the expected rise of network virtualization as enabler for flexible and autonomic network management, the decision on how to map virtual resources to physical resources is gaining importance. Achieving optimal VNEs, therefore, is an important problem to solve for

Future Internet infrastructures. In this chapter, the VNE problem has been described in detail. Several algorithms solving the VNE problem have been presented. It was argued that it is necessary to compare these algorithms according to different metrics. This will allow to rank different algorithms against each other and choose the best solution for a given embedding problem. ALEVIN - a framework that allows to implement these algorithms and compute solutions for different networks - was presented and discussed in this chapter. ALEVIN is developed as *open source* and has been released under *GNU General Public License (GPL)* and *GNU Lesser General Public License (LGPL)*. ALEVIN as well as the VNE exchange format will be made publicly available on [DSS+11].

Part III

CONCLUSION AND FUTURE WORK

7 CONCLUSION

"If one begins all deeds well, it is likely that they will end well too"
Sophocles

This thesis has exhaustively explored the resource allocation problem in network virtualization, commonly known as Virtual Network Embedding. Thanks to this exploration, several contributions to the existing state of the art were introduced in this thesis. In particular, a new taxonomy to classify existing VNE approaches, a general formulation of the problem, its parameters-metrics, VNE's existing coordination variants, optimization objectives, algorithm strategies, software tools and emerging research directions were identified and delved in detail. In addition, this thesis introduced two contributions to the virtual link mapping stage of the VNE: The inclusion of new *hidden hop* demands and the flexible multi-constraint solution based on the *paths algebra* framework. A clever approach to solve the VNE in a distributed fashion is also a contribution of this thesis. Also, the first approach tackling the energy-efficient VNE is a contribution of this thesis.

It would not have been possible to implement and evaluate the aforementioned contributions had it not been for ALEVIN: a software framework to develop, compare and analyze VNE algorithms. The author of this thesis actively participated in the development of ALEVIN.

This chapter sums up the main contributions and results of the thesis that were mainly directed to the exploitation of the identified research opportunities. It will be split in three sections. Section 7.1 details the main results obtained throughout the development of the thesis. Section 7.2 presents the publications achieved in this thesis and their quality assessment. Finally, Section 7.3 shows the participation of the author of this thesis in national and international research projects and how the contributions of this thesis were developed following the main objectives of these projects.

7.1 MAIN CONTRIBUTIONS AND RESULTS

This thesis has been devoted to the study of the main resource allocation challenge in network virtualization: the VNE problem. This thesis introduces a novel taxonomy that allows any variant of the

VNE problem to be classified (see Section 2.2.1.2). Existing solutions to the VNE problem are described with the following syntax:

[C | D]/[S | D]/[C | R]

The first character denotes, whether the algorithm is **C**entralized or **D**istributed. Likewise, the second character denotes whether the algorithm is **S**tatic or **D**ynamic. Finally, the third character denotes whether the algorithm is **C**oncise or **R**edundant. So, an algorithm denoted as **C/D/R** is a centralized, dynamic, redundant algorithm. This allows to quickly categorize any given algorithm and properly compare it with similar approaches. Thanks to this taxonomy, this thesis classified the main existing VNE approaches (see Section 2.2.3).

Besides, the thesis has deeply surveyed VNE; the following concepts are presented (see Chapter 2): a formal and generic mathematical *formulation* of the VNE problem, the different *parameters* that can be considered in the embedding for substrate and virtual networks, the main *embedding objectives* which relate to VNE, the possible alternatives to *decompose* the VNE problem and to solve the *coordination* between virtual node and link mapping, the set of possible *optimization strategies* to solve VNE and the different *metrics* used to evaluate the performance of the solutions, a number of VNE *simulators*.

Two main contributions in the virtual link mapping stage are presented in Chapter 3. In first place, a new "hidden hops" demand for each virtual link is introduced: Mapping a virtual link to a path in the SN obviously uses resources of the physical links on the path. However, there are also physical nodes on the path that will be traversed by the virtual link. These are called *hidden hops* here. The thesis claims that a hidden hop entails a resource demand because it has to perform packet forwarding of the traffic that will pass through this virtual link (see Section 3.2). In second place, this thesis introduces a flexible **C/S/C** strategy to solve the single-path virtual link mapping based on the paths algebra mathematical framework (see Section 3.5). Unlike current single-path VLiM approaches that are mono-constrained and use K shortest paths to solve the VLiM stage, paths algebra-based multi-constraint routing algorithm is able to FIND all the possible paths between each pair of nodes in the SN and ORGANIZE them based on an unlimited number of constraints (or combination of constraints). It allows to consider multi-constraint VLiM with linear and nonlinear constraints providing a more flexible and extensible solution for the VLiM stage of the VNE. Simulation results show that the paths algebra-based VLiM approach perform equal or better than existing single-path heuristics and, besides, it does not impose any restriction on the type and number of metrics that are used (see Section 3.6). Right on the contrary, linear and non-linear metrics can be used together; any combination of metrics can also be employed.

This thesis identified that, until now, there has been little interest of the research community in finding distributed solutions for the

VNE (see Section 2.2.3). To remedy this situation, this thesis proposed DPVNE: an Universal, fully Distributed and Parallel VNE approach. Contrary to centralized approaches where a single node computes the entire embedding, DPVNE has the advantage of making better use of available computing resources, letting multiple nodes calculate embeddings in parallel (see Chapter 4). Simulation results showed that, using DPVNE, message overhead can be kept significantly smaller than other fully distributed VNE solutions. These results also showed that, despite its distributed nature, embedding costs of DPVNE remain comparable to those of centralized algorithms.

One of the emerging research branches introduced by this thesis is the energy-aware VNE (see Chapter 5). Two different **C/S/R** energy-aware VNE approaches based on Mixed Integer Programming, called EA-VNE and EA-VNE-LB, are proposed in this thesis (see Section 5.3). The main goal of these approaches is then to minimize the power consumption in the SN after the mapping is performed; this is done by dynamically dimensioning the substrate network leaving free as much networking equipment as possible to accomplish the VNRs demands. Free networking equipment can be left in sleep (low energy consumption) mode or switched off. Simulation results show that, under lightly loaded substrate networks, the proposed energy-aware approaches consume up to 40% less power than cost-based VNE approaches.

It was also demonstrated that, under large networks, EA-VNE and EA-VNE-LB reduce their effectiveness, and also, that, under highly loaded SNs, the power consumption is minimized at the cost of reducing the VNRs acceptance ratio creating a trade-off between acceptance ratio and power savings. To overcome these shortcomings, this thesis also introduced an **C/D/R** energy-aware VNE reconfiguration heuristic trying to minimize the power consumption by relocating the virtual nodes and links of already mapped VNRs (see Section 5.4). Simulation results showed that, also under lightly loaded substrate networks, EA-RH consumes up to 50% less power that cost-based VNE solutions.

One outcome of this thesis is the collaboration in the development of ALEVIN: The Software Framework to Develop, Compare, and Analyze VNE Algorithms. ALEVIN was used to implement and evaluate the aforementioned contributions to solve the VNE problem.

7.2 PUBLICATIONS

The research work presented in this PhD. Thesis has been internationally validated in different networking peer reviewed journals and conferences. Several experts have provided their comments in the peer reviews allowing us to improve our investigations and to guide the direction of our research. The articles published during the develop-

Table 7.1: Publications in journals

Journals			
Year	Paper Title	Journal	Quality Indicator
2011	ALEVIN - a framework to develop, compare, and analyze virtual network embedding algorithms [FBD+11]	Electronic Communications of the EASST	Open Access Journal Indexed in DBLP
2012	Energy Efficient Virtual Network Embedding [BHD+12]	IEEE Communications Letters	Impact Factor: 0.982 Quartile Q2
2013	Optimal Mapping of Virtual Networks with Hidden Hops [BHFDM13]	Telecommunication Systems	Impact Factor: 0.689 Quartile Q3
2013	Virtual network embedding: A survey [FBB+13]	IEEE Communications Surveys & Tutorials	Impact Factor: 6.311 Quartile Q1 ***First journal** in telecommunications and computer science categories according to JCR*
2013	A novel paths algebra-based strategy to flexibly solve the link mapping stage of VNE problems [BMHSA13]	Journal of Network and Computer Applications	Impact Factor: 1.065 Quartile Q2
2013	Greener networking in a network virtualization environment [BH13]	Computer Networks	Impact Factor: 1.2 Quartile Q2

ment of this thesis are detailed in Table 7.1 and Table 7.2 with a quality indicator that highlight the quality level of the journal/conference.

7.3 RESEARCH PROJECTS

The research contribution made by this thesis to the VNE problem was made in the context of several national and international research projects. The author has actively participated in the set of projects summarized in Table 7.3. The interest in network virtualization started with the Manticore Project[1] which developed a proof of the IP Network as a Service concept, the project allowed to create a service for the Network Operations Centre and end users which allows them to customize the configuration of their own dedicated IP physical and/or logical network. As, in Manticore, the allocation of physical resources to logical entities was performed manually and without any optimization criteria, the need to find algorithms that optimally allocate the physical resources to virtual demands emerged. The Manticore project evolved to Manticore II project[2] and finally to the EU FP7 Mantychore project[3] that is currently running and test the

1 http://www.i2cat.net/en/projecte/manticore-1
2 http://www.terena.org/activities/ngn-ws/ws1/061107-reijs-TERENA-NGN-WS-01.pdf
3 http://www.mantychore.eu/

Table 7.2: Publications in conferences and produced software

INTERNATIONAL CONFERENCES			
YEAR	PAPER TITLE	CONFERENCE	QUALITY INDICATOR
2009	The Bottlenecked Virtual Network Problem in Bandwidth Allocation for Network Virtualization [BH09]	IEEE Latin-American Conference on Communications	CORE Ranking: **C**, Indexed in IEEE Xplore
2010	MANTICORE II: IP Network as a Service Pilots at Heanet, Nordunet and Rediris [GFB+10]	TERENA Networking Conference	-
2010	Generalized Cognitive Networks of Virtual Resources [MMM+10]	IX Workshop in G/MPLS Networks	-
2011	Virtual Network Resource Embedding Algorithms in the Framework ALEVIN [DSB+11]	7th EURO-NGI Conference on Next Generation Internet	Indexed in DBLP and IEEE Xplore
2011	Flexible VNE Algorithms Analysis using ALEVIN [BHD+11]	11th Würzburg Workshop on IP: Joint ITG and Euro-NF Workshop "Visions of Future Generation Networks"	-
2013	A Distributed, Parallel, and generic Virtual Network Embedding Framework [BBF+13]	IEEE International Conference on Communications	CORE Ranking: **B**, Indexed in DBLP and IEEE Xplore
NATIONAL CONFERENCES			
2008	Estrategia de asignación de recursos basada en criterios de justicia para las interfaces de encaminadores lógicos IP [BH08]	Jornadas de Ingeniería Telemática (JITEL)	-
2011	Comparación de Algoritmos para el Mapeo de Redes Virtuales [BH11]	Jornadas de Ingeniería Telemática (JITEL)	-
SOFTWARE			
YEAR	SOFTWARE NAME	URL	QUALITY INDICATOR
2011	ALEVIN – ALgorithms for Embedding VIrtual Networks [DSS+11]	http://alevin.sf.net	*Software published in Sourceforge and used in several research papers*

IP Network as a Service in grid and e-health virtual research communities.

The author of this thesis was actively involved in the Euro-NF network of excellence. Specially, participating in VNREAL[4], a Specific Joint Research Project funded by Euro-NF which extended research on network virtualization by creating a framework for VNE algo-

4 http://www.fim.uni-passau.de/fileadmin/files/lehrstuhl/meer/publications/pdf/Duelli2011a.pdf

Table 7.3: National and international research projects

Project Name	Duration	State	Partners Involved
The Manticore Project	01/11/2007 10/05/2008	F	i2cat (Spain), Heanet (Ireland), Nordunet (Sweden), Department of Telematics Engineering UPC (Spain), Juniper Networks - Europe and Red Iris (Spain)
Manticore II	15/01/2009 30/10/2010	F	i2cat (Spain), Heanet(Ireland), Nordunet (Sweden), Department of Telematics Engineering UPC (Spain), Juniper Networks - Europe and RedIris (Spain)
EU FP7 Mantychore Project	10/01/2009 01/02/2013	E	i2cat (Spain), Heanet(Ireland), Nordunet(Sweden), Department of Telematics Engineering UPC (Spain) and University of Essex (England)
National Spanish Project CICYT TSI2007-6637-C02	10/10/2007 03/08/2010	F	Department of Telematics Engineering UPC and Technical University of Valencia
National Spanish Project TIN2010-20136-C03	2010 2013	E	i2cat, Department of Telematics Engineering UPC (Spain) and Carlos III University
Euro-NF. European Network of Excellence in the 7 framework program (`http://euronf.enst.fr/`)	01/03/2008 01/06/2012	F	36 highly qualified european partners (`http://euronf.enst.fr/p_en_menu1_NFcommunit_396.html`)
EU FP7 All4Green Project	01/11/2011 01/05/2014	E	University of Passau (Germany), Hewlett Packard Italy, University of Manheim (Germany), UPC (Spain), WIND (Italy), Almende (Holland), AKT (Germany) and SWP (Germany)

Project State E: Executing, F:Finished

rithms, allowing researchers to evaluate and compare novel solutions to the VNE problem according to a wide set of criteria (see Chapter 6).

Finally, the author currently participates in the All4Green project[5] which aims to save up to 20% of energy in data centers thanks to the collaboration of the energy provider, the data center and the Information Technology (IT) customer. Thanks to All4Green, this thesis has introduced the energy-efficient VNE problem that looks to use the consolidation of virtual machines to dynamically dimension the network trying to minimize the energy consumption(see Chapter 5).

At national level, the author of the thesis participated in the CICYT projects TSI2007-6637-C02 and TIN2010-20136-C03 in the study of Future Internet architectures, specifically in the VNE problem within the network virtualization context.

5 `http://www.all4green-project.eu/`

8 FUTURE WORK

"I dread the events of the future, not in themselves but in their results"
Edgar Allan Poe

The contributions presented in this thesis can be used in several existing network virtualization projects and testbeds with management systems that use VNE algorithms to automatically allocate VN demands to SN resources. The contributions introduced throughout this thesis can still be improved and, as network virtualization is still in its infancy, several exciting VNE branches can be subject of further investigation.

This chapter is divided into three sections: Section 8.1 makes a summary of the possible projects/testbeds where the contributions introduced in this thesis can be applied. Section 8.2 details the possible enhancements that can be made in order to evolve and improve the current state of the presented contributions. Finally, in Section 8.3, new VNE branches subject of future research are highlighted.

8.1 THESIS CONTRIBUTIONS, WHERE TO APPLY THEM?

Within the European scope, the GÉANT[1] network is the fast and reliable pan-European communications infrastructure serving Europe's research and education community. Funded by the European National Research & Education Networks (NRENs) and the European Community, the GÉANT network and project is in its third generation, along with associated development activities. One of the joint research activities promoted by GÉANT is the Future Network Research. Specifically, inside this activity, GÉANT identifies the current and potential uses of virtualization[2] as one of the main research areas. Inside this area, GÉANT recognizes network virtualization as one key technological enabler to provide IaaS in order to build a new network architecture. The VNE mechanisms proposed in this thesis can be used by GÉANT to allocate virtual resources either in a centralized o distributed way helping to accomplish one of the main GÉANT's objectives: *to build GÉaNt virtUalization Service (GENUS) [175] based on the developments and achievements of European Union (EU) projects.*

1 http://www.geant.net/
2 http://www.geant.net/Research/Future_Network_Research/Pages/CurrentandPotentialUsesofVirtualisation.aspx

GÉANT plans to build on top of the results of different EU projects. Mantychore is one of these projects. It provides a NV solution at IP level. One of the Mantychore's objectives is to implement a *market-place* [176] when networking resources can be offered by InPs and acquired by VNPs. The contributions to solve the VNE problem introduced in this thesis would be of high value in such a marketplace. The output of optimized VNE algorithms provides an automatic way for the VNPs to choose the more suitable substrate resources offered by InPs.

One of the first EU project dealing with NV was 4WARD[3]. This project clearly defined a network architecture based on NV. The follow-up of 4WARD called Scalable and Adaptive Internet Solutions (SAIL)[4] uses self-managed virtual networks to test its Information Centric Networking (ICN) approach [177]. 4WARD and SAIL are clear examples of where the VNE solutions proposed by this thesis can be applied. Either in 4WARD in the definition of the resource allocation tasks needed in a NV based architecture or in the testbeds used in SAIL.

There are several, either finished or in-progress, EU projects based on network virtualization that could make use of the contributions presented in this thesis to optimize their virtual resource allocation mechanisms: Generalized Architecture for Dynamic Infrastructure Services (GEYSERS)[5] is a project that provides a NV-based business model at layer 1. FEDERICA[6] is a testbed infrastructure based upon Gigabit Ethernet circuits, transmission equipment and computing nodes capable of virtualization, to host experimental activities on new Internet architectures and protocols. PHOSPHORUS[7] is a project addressing some of the key technical challenges to enable on-demand end to end network services across multiple domains. The Phosphorus network concept and test-bed will make applications aware of their complete Grid resources (computational and networking) environment and capabilities, and able to make dynamic, adaptive and optimized use of heterogeneous network infrastructures connecting various high-end resources.

Within the scope of United States (US), several NV-based initiatives have been proposed: Global Environment for Network Innovation (GENI)[8] is an experimental facility designed to form a robust, federated environment to allow computer networks' researchers to experiment on a wide variety of problems in communications, networking, etc. GENI is funded by the National Science Foundation (NSF). Virtualization in GENI is based on OpenFlow [23], an open standard

3 http://www.4ward-project.eu/
4 http://www.sail-project.eu/
5 http://www.geysers.eu/
6 http://www.fp7-federica.eu/
7 http://www.ist-phosphorus.eu/
8 http://www.geni.net/

that allows researchers to directly control the way packets are routed in the network. Basically, network virtualization is enabled by FlowVisor [178], an OpenFlow-based approach to switch virtualization in which the same hardware forwarding plane can be shared among multiple logical networks, each with distinct forwarding logic. To choose the proper resources in the hardware forwarding plane to be instantiated by FlowVisor, the contributions presented in this thesis can be very valuable.

Also in the US span, different testbeds for research experiments like PlanetLab[9] [179] or VINI [21] (that runs on PlanetLab) provide network virtualization capabilities. These public testbeds can take advantage of the VNE mechanisms developed throughout this thesis to provide an optimal provision of VNs to the researchers that act their users.

NV can be an useful tool to test clean-slate Future Internet architectures proposed by Internet pluralists (see Section 2.1). Research initiatives such as: The Atomic Redesign of the Internet Future Architecture (TARIFA)[10] [180], the Recursive InterNetwork Architecture (RINA)[11] [181] and Content Centric Networking (CCN) [182] may use NV-based infrastructures to test their proposed approaches in realistic scenarios. In that context, the contributions introduced in this thesis can be used to provide efficient and optimized resource allocation of virtual demands onto physical resources.

8.2 FURTHER IMPROVEMENT OF PROPOSED CONTRIBUTIONS

Some of the aspects of the VNE contributions presented in this monograph have still room for further improvement, namely:

In Chapter 3, a single-path flexible VLiM strategy based on the paths algebra framework was proposed. The paths algebra framework is able to obtain all the paths between each pair of nodes in the SN, besides, they can be ordered by any metric. To take advantage of the possibilities provided by the paths algebra, future work can be devoted to improve the current approach by including a coordinated node mapping stage that could provide a two stages-based VNE solution or even a solution performed in one stage. In addition, future work can be devoted to investigate backtracking strategies to act when a VNR cannot be satisfied due to bottlenecks present in the SN.

DPVNE was introduced in Chapter 4. It makes use of hierarchical partitioning that allows to perform the VNE in smaller, cooperating parts of the SN. Up to now, DPVNE uses the Multilevel Recursive Bisection partitioning algorithm proposed in [159] . Future work can

9 http://www.planet-lab.org/
10 http://www.i2cat.net/en/projecte/tarifa-1
11 http://csr.bu.edu/rina/index.html

be devoted to the study of the the effects of different partitioning methods in the results of DPVNE.

DPVNE just used one centralized algorithm (ASID [52]) to perform VNE within each partition of the SN. An interesting branch for future investigation can be devoted to the evaluation of DPVNE with different centralized VNE algorithms. Another subject of future research is the placement of delegation nodes and its effect on message overhead.

The VNE approach presented in Chapter 5 is, to the best of our knowledge, the first approximation to energy-aware VNE solutions. Consequently, there is substantial room for future improvement. In particular, this first approximation considered homogeneous power consumption for nodes and interfaces, and omit the high modularity of current network devices: In future scenarios, the power consumption of the networking devices will not be constant when they are active, but will highly depend on their load. Additionally, in terms of power, the main goal should not be just minimize the number of active links, but to try to deactivate all the links terminated on the same line-card, so that it can be shut down or slept.

Consequently, future work can be dedicated to extend the current static MIP-based VNE approaches (EA-VNE and EA-VNE-LB) and the dynamic relocation heuristic EA-RH, so that so that the modularity of network devices and the load dependent power consumption are included. Also, as the evaluation environment used in Chapter 5 is offline, another unexplored research branch may be to explore online evaluation scenarios to see how EA-RH behaves throughout the time and which will be the optimal interval to execute periodic energy-aware VN relocations.

8.3 FUTURE RESEARCH VNE BRANCHES

Apart from the aforementioned improvements to the contributions presented in this thesis, several key challenges of the VNE problem remain unexplored. For instance, Section 2.2.4.2 mentions the security requirements that VNOs and InPs while performing a VNE and Section 2.2.4.3 introduces the network environments where VNE is currently being investigated.

Also, recent research [183] has introduced the concept of data center network virtualization where a *Virtualized data center* is "a data center where some or all of the hardware (e.g. servers, routers, switches, and links) are virtualized" and a *Virtual data center* is a collection of virtual resources (VMs, virtual switches, and virtual routers) connected via *virtual links*".

In this environment, the virtual network embedding problem becomes the Virtual Datacenter Embedding (VDE) problem where more resources can be virtualized (they include routers, switches, storage devices, servers, security systems and links). Existing VNE solutions, commonly considering just bandwidth and processing requirements, will not be sufficient to solve the VDE as the requirements for the new resources should be considered as well. Specifically, future investigation can be dedicated to the study of energy-aware approaches to solve VDE as energy consumption is one of the main concerns in data centers since it accounts for a significant amount of their operational costs.

CONTRIBUTIONS

[BBF$^+$13] Michael T. Beck, J.F. Botero, Andreas Fischer, Xavier Hesselbach, and Hermann De Meer. Dpvne: A distributed, parallel, and generic virtual network embedding framework. In *IEEE International Conference on Communications*, ICC 2013, 2013. to appear.

[BH08] J.F. Botero and X. Hesselbach. Estrategia de asignación de recursos basada en criterios de justicia para las interfaces de encaminadores lógicos ip. In *Jornadas de Ingeniería Telemática (JITEL)*. Jornadas de Ingeniería Telemática (JITEL), September 2008.

[BH09] J.F. Botero and X. Hesselbach. The bottlenecked virtual network problem in bandwidth allocation for network virtualization. In *Communications, 2009. LATINCOM'09. IEEE Latin-American Conference on*, pages 1–5. IEEE, 2009.

[BH11] J.F. Botero and X. Hesselbach. Comparación de algoritmos para el mapeo de redes virtuales. In *Jornadas de Ingeniería Telemática (JITEL)*. Jornadas de Ingeniería Telemática (JITEL), September 2011.

[BH13] J.F. Botero and X. Hesselbach. Greener networking in a network virtualization environment. *Computer Networks*, (0):–, 2013.

[BHD$^+$11] J.F. Botero, Xavier Hesselbach, Michael Duelli, Daniel Schlosser, Andreas Fischer, and Hermann de Meer. Flexible VNE Algorithms Analysis using ALEVIN. In *Euroview 2011*, August 2011.

[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, may 2012.

[BHFDM13] J.F. Botero, Xavier Hesselbach, Andreas Fischer, and Hermann De Meer. Optimal mapping of virtual networks with hidden hops. *Telecommunication Systems*, 52(3):1–10, 2013.

[BMHSA13] Juan Felipe Botero, Miguel Molina, Xavier Hesselbach-Serra, and José Roberto Amazonas. A novel paths algebra-based strategy to flexibly solve the link mapping stage of vne problems. *Journal of Network and Computer Applications*, (0):–, 2013.

[DSB+11] M. Duelli, D. Schlosser, J.F. Botero, X. Hesselbach, A. Fischer, and H. de Meer. Vnreal: Virtual network resource embedding algorithms in the framework alevin. In *Next Generation Internet (NGI), 2011 7th EURO-NGI Conference on*, pages 1 –2, june 2011.

[DSS+11] Michael Duelli, Daniel Schlosser, Vlad Singeorzan, Juan Felipe Botero, Xavier Hesselbach, Lisset Diaz Cervantes, Andreas Fischer, and Michael Till Beck. ALEVIN: ALgorithms for Embedding VIrtual Networks, May 2011. http://alevin.sf.net.

[FBB+13] A. Fischer, J.F. Botero, M. Beck, H. de Meer, and X. Hesselbach. Virtual network embedding: A survey. *Communications Surveys Tutorials, IEEE*, PP(99):1 –19, 2013.

[FBD+11] Andreas Fischer, Juan F. Botero, Michael Duelli, Daniel Schlosser, Xavier Hesselbach, and Hermann De Meer. ALEVIN - a framework to develop, compare, and analyze virtual network embedding algorithms. *Electronic Communications of the EASST*, 37:1–12, 2011.

[GFB+10] E. Grasa, S. Figuerola, X. Barrera, L. Ferrao, C. Baez, P. Minoves, X. Hesselbach, J.F. Botero, V. Rejs, D. Wilson, L. Fischer, T. de Miguel, J.M. Uzé, C. Lonvich, D. Simeonidou, and I. Cabello. Manticore ii: Ip network as a service pilots at heanet, nordunet and rediris. In *TERENA Networking Conference*, June 2010.

[MMM+10] A. Manzalini, C. Moiso, R. Minerva, X. Hesselbach, J.S. Pareta, and J.F. Botero. Generalized cognitive networks of virtual resources. In *IX Workshop in G/MPLS (WGN9)*, 2010.

BIBLIOGRAPHY

[1] J. Turner, T. Anderson, L. Peterson, and S. Shenker. Virtualizing the net: A strategy for network de-ossification, 2004. Available: http://www.arl.wustl.edu/~jst/talks/hotI-9-04.pdf.

[2] L. Peterson, S. Shenker, J. Turner, et al. Overcoming the internet impasse through virtualization. *IEEE Computer*, 38(4):34–41, April 2005.

[3] N. M. M. K. Chowdhury. Network virtualization: State of the art and research challenges. *IEEE Communications Magazine*, 47(7): 20–26, July 2009.

[4] N.M. Mosharaf Kabir Chowdhury and Raouf Boutaba. A survey of network virtualization. *Computer Networks*, 54(5):862 – 876, 2010. ISSN 1389-1286.

[5] W. P. Herman and J. R. A. Amazonas. Hop-by-hop routing convergence analysis based on paths algebra. In *Proceedings of the 2007 Electronics, Robotics and Automotive Mechanics Conference - CERMA 2007*, pages 9–14, 2007.

[6] S. Kent and K. Seo. Security architecture for the internet protocol, December 2005. RFC 4301.

[7] Ralf Steinmetz and Klaus Wehrle. *Peer-to-Peer Systems and Applications (Lecture Notes in Computer Science)*. Springer-Verlag New York, Secaucus, NJ, USA, 2005. ISBN 354029192X.

[8] K. Tutschku, T. Zinner, A. Nakao, and P. Tran-Gia. Network virtualization: Implementation steps towards the future internet. In *Proc. of the Workshop on Overlay and Network Virtualization at KiVS*, Kassel, Germany, mar 2009.

[9] A. Berl, A. Fischer, and H. de Meer. Using system virtualization to create virtualized networks. In *Workshops der Wissenschaftlichen Konferenz Kommunikation in Verteilten Systemen (WowKiVS2009)*, Kassel, Germany, March 2009. vol. 17, EASST.

[10] 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, 2008. ISSN 0146-4833.

[11] S. Bhardwaj, L. Jain, and S. Jain. Cloud computing: A study of infrastructure as a service (iaas). *Int J. Eng. and Information Technology*, 2:60–63, 2010.

[12] Nick Feamster, Lixin Gao, and Jennifer Rexford. How to lease the internet in your spare time. *Computer Communication Review*, 37(1), 2007.

[13] Marcus Brunner, Henrik Abramowicz, Norbert Niebert, and Luis M. Correia. 4ward: A european perspective towards the future internet. *IEICE Transactions*, 93-B(3):442–445, 2010.

[14] N. Niebert, S. Baucke, I. El-Khayat, M. Johnsson, B. Ohlman, H. Abramowicz, K. Wuenstel, H. Woesner, J. Quittek, and L.M. Correia. The way 4ward to the creation of a future internet. In *Personal, Indoor and Mobile Radio Communications, 2008. PIMRC 2008. IEEE 19th International Symposium on*, pages 1 –5, sept. 2008.

[15] Gregor Schaffrath, Christoph Werle, Panagiotis Papadimitriou, Anja Feldmann, Roland Bless, Adam Greenhalgh, Andreas Wundsam, Mario Kind, Olaf Maennel, and Laurent Mathy. Network virtualization architecture: proposal and initial prototype. In *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, VISA '09, pages 63–72, New York, NY, USA, 2009. ACM.

[16] Andrew T. Campbell, Herman G. De Meer, Michael E. Kounavis, Kazuho Miki, John B. Vicente, and Daniel Villela. A survey of programmable networks. *SIGCOMM Comput. Commun. Rev.*, 29 (2):7–23, 1999. ISSN 0146-4833.

[17] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek. The click modular router. *ACM Trans. Comput. Syst.*, 18(3):263–297, 2000.

[18] Jonathan Turner. A proposed architecture for the geni backbone platform. In *In Proc. Architecture for Networking and Communications Systems*, pages 12–2006, 2006.

[19] C.-S. Li and W. Liao. Software defined networks [guest editorial]. *Communications Magazine, IEEE*, 51(2):113, february 2013.

[20] Dennis Schwerdel, Daniel Günther, Robert Henjes, Bernd Reuther, and Paul Müller. German-lab experimental facility. *Future Internet Symposium (FIS) 2010*, 9 2010.

[21] Andy Bavier, Nick Feamster, Mark Huang, Larry Peterson, and Jennifer Rexford. In vini veritas: realistic and controlled network experimentation. *SIGCOMM Comput. Commun. Rev.*, 36(4):3–14, August 2006. ISSN 0146-4833.

[22] P.T. Endo, A.V. de Almeida Palhares, N.N. Pereira, G.E. Goncalves, D. Sadok, J. Kelner, B. Melander, and J. Mangs. Resource allocation for distributed cloud: concepts and research challenges. *Network, IEEE*, 25(4):42 –46, july-august 2011.

[23] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, March 2008. ISSN 0146-4833.

[24] ONF. Open networking foundation. https://www.opennetworking.org/, 2011.

[25] A. Haider, R. Potter, and A. Nakao. Challenges in resource allocation in network virtualization. In *20th ITC Specialist Seminar*, volume 18, page 20, 2009.

[26] A. Belbekkouche, M. Hasan, and A. Karmouch. Resource discovery and allocation in network virtualization. *Communications Surveys Tutorials, IEEE*, PP(99):1–15, 2012. ISSN 1553-877X.

[27] David G. Andersen. Theoretical approaches to node assignment. Unpublished Manuscript, December 2002.

[28] J.M. Kleinberg. *Aproximation Algorithms for Disjoint Paths Problems*. PhD thesis, Massachusetts Institute of Technology, 1996.

[29] S.G. Kolliopoulos and C. Stein. Improved approximation algorithms for unsplittable flow problems. In *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on*, pages 426 –436, oct 1997.

[30] 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 –5, dec. 2011.

[31] David Stezenbach, Matthias Hartmann, and Kurt Tutschku. Parameters and challenges for virtual network embedding in the future internet. In *First IEEE Workshop on Algorithms and Operating Procedures for Federated Virtualized Networks (FEDNET2012)*, 2012.

[32] Gang Sun, Hongfang Yu, Lemin Li, Vishal Anand, Yanyang Cai, and Hao Di. Exploring online virtual networks mapping with stochastic bandwidth demand in multi-datacenter. *Photonic Network Communications*, pages 1–14, 2011.

[33] Sheng Zhang, Zhuzhong Qian, Bin Tang, Jie Wu, and Sanglu Lu. Opportunistic bandwidth sharing for virtual network mapping. In *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pages 1 –5, dec. 2011.

[34] Sheng Zhang, Zhuzhong Qian, Bin Tang, Jie Wu, and Sanglu Lu. An opportunistic resource sharing and topology-aware mapping

framework for virtual networks. In *INFOCOM 2012. 31th IEEE International Conference on Computer Communications*, 2012. To appear.

[35] Johannes Inführ and Günther R. Raidl. Introducing the virtual network mapping problem with delay, routing and location constraints. In *Proceedings of the 5th international conference on Network optimization*, INOC'11, pages 105–117, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-21526-1.

[36] M. Chowdhury, M. R. Rahman, and R. Boutaba. Vineyard: Virtual network embedding algorithms with coordinated node and link mapping. *Networking, IEEE/ACM Transactions on*, PP(99):1, 2011.

[37] 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, june 2011.

[38] Wai-Leong Yeow, Cédric Westphal, and Ulaş Kozat. Designing and embedding reliable virtual infrastructures. In *Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures*, VISA '10, pages 33–40, New York, NY, USA, 2010. ACM.

[39] Hongfang Yu, V. Anand, Chunming Qiao, and Gang Sun. Cost efficient design of survivable virtual infrastructure to recover from facility node failures. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1 –6, june 2011.

[40] Muntasir Raihan Rahman, Issam Aib, and Raouf Boutaba. Survivable virtual network embedding. In *Networking*, pages 40–52, 2010.

[41] Minlan Yu, Yung Yi, Jennifer Rexford, and Mung Chiang. Rethinking virtual network embedding: Substrate support for path splitting and migration. *ACM SIGCOMM CCR*, 38(2):17–29, April 2008.

[42] David Eppstein. Finding the k shortest paths. *SIAM J. Comput.*, 28:652–673, February 1999. ISSN 0097-5397.

[43] Michal Pióro and Deepankar Medhi. *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004. ISBN 0125571895.

[44] Dimitris Bertsimas and John Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1997. ISBN 1886529191.

[45] N. M. M. K. Chowdhury, Muntasir Raihnan Rahman, and Raouf Boutaba. Virtual network embedding with coordinated node and link mapping. In *Proc. IEEE INFOCOM*. IEEE Infocom, April 2009.

[46] Xiang Cheng, Sen Su, Zhongbao Zhang, Hanchi Wang, Fangchun Yang, Yan Luo, and Jie Wang. Virtual network embedding through topology-aware node ranking. *SIGCOMM Comput. Commun. Rev.*, 41:38–47, April 2011. ISSN 0146-4833.

[47] 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*, VISA '10, pages 49–56, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0199-2.

[48] L. A. Wolsey. *Integer Programming*. John Wiley & Sons, New York, 1998.

[49] GLPK. GNU Linear Programming Kit. http://www.gnu.org/software/glpk, 2008.

[50] ILOG, Inc. ILOG CPLEX: High-performance software for mathematical programming and optimization. http://www.ilog.com/products/cplex/, 2012.

[51] Ines Houidi, Wajdi Louati, Walid Ben Ameur, and Djamal Zeghlache. Virtual network provisioning across multiple substrate networks. *Computer Networks*, 55(4):1011 – 1023, 2011. Special Issue on Architectures and Protocols for the Future Internet.

[52] 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. New York, USA, Aug. 2009.

[53] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[54] John H. Holland. *Adaptation in natural and artificial systems*. MIT Press, Cambridge, MA, USA, 1992. ISBN 0-262-58111-6.

[55] DORIGO M. Optimization, learning and natural algorithms. *Ph.D. Thesis, Politecnico di Milano, Italy*, 1992.

[56] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942 –1948 vol.4, nov/dec 1995.

[57] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.*, 13:533–549, May 1986. ISSN 0305-0548.

[58] Thomas Stützle and Holger H. Hoos. Max-min ant system. *Future Gener. Comput. Syst.*, 16:889–914, June 2000. ISSN 0167-739X.

[59] 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, june 2011.

[60] 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*, 2012. ISSN 1099-1131.

[61] Nabeel Farooq Butt, N. M. Mosharaf Kabir Chowdhury, and Raouf Boutaba. Topology-awareness and reoptimization mechanism for virtual network embedding. In *Networking*, pages 27–39, 2010.

[62] S.B. Masti and S.V. Raghavan. Vna: An enhanced algorithm for virtual network embedding. In *Computer Communications and Networks (ICCCN), 2012 21st International Conference on*, pages 1 –9, August 2012.

[63] Bo LU, 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. ISSN 1005-8885.

[64] 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, jan. 2012.

[65] Wenzhi Liu, Yang Xiang, Shaowu Ma, and Xiongyan Tang. Completing virtual network embedding all in one mathematical programming. In *Electronics, Communications and Control (ICECC), 2011 International Conference on*, pages 183 –185, sept. 2011.

[66] T. Trinh, H. Esaki, and C. Aswakul. Quality of service using careful overbooking for optimal virtual network resource allocation. In *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2011 8th International Conference on*, pages 296 –299, may 2011.

[67] A. Pages, J. Perello, S. Spadaro, and G. Junyent. Strategies for virtual optical network allocation. *Communications Letters, IEEE*, 16(2):268 –271, february 2012. ISSN 1089-7798.

[68] Hao Di, Lemin Li, V. Anand, Hongfang Yu, and Gang Sun. Cost efficient virtual infrastructure mapping using subgraph isomorphism. In *Communications and Photonics Conference and Exhibition (ACP), 2010 Asia*, pages 533 –534, dec. 2010.

[69] T. Ghazar and N. Samaan. A hierarchical approach for efficient virtual network embedding based on exact subgraph matching. In *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pages 1 –6, dec. 2011.

[70] 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. ISBN 978-1-4503-0821-2.

[71] Donggyu Yun, Jungseul Ok, Bongjhin Shin, Soobum Park, and Yung Yi. Embedding of virtual network requests over static wireless multihop networks. *CoRR*, abs/1207.1878, 2012.

[72] Hongfang Yu, Vishal Anand, Chunming Qiao, Hao Di, and Xuetao Wei. A cost efficient design of virtual infrastructures with joint node and link mapping. *Journal of Network and Systems Management*, 20:97–115, 2012. ISSN 1064-7570.

[73] Jiang Liu, Tao Huang, Jian ya Chen, and Yunjie 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.

[74] Xiao-ling Li, Huai-min Wang, Chang-guo Guo, Bo Ding, Xiaoyong Li, Wen-qi Bi, and Shuang Tan. Topology awareness algorithm for virtual network mapping. *Journal of Zhejiang University - Science C*, 13:178–186, 2012. ISSN 1869-1951.

[75] J. Lu and J. Turner. Efficient mapping of virtual networks onto a shared substrate. Technical Report WUCSE-2006-35, Washington University in St. Louis, 2006.

[76] 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, sept. 2010.

[77] 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, nov. 2011.

[78] 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, july 2011.

[79] Aris Leivadeas, Chrysa Papagianni, Evripidis Paraskevas, Georgios Androulidakis, and Symeon Papavassiliou. An architecture for virtual network embedding in wireless systems. In *Proceedings of the 2011 First International Symposium on Network Cloud Computing and Applications*, pages 62–68. IEEE Computer Society, 2011.

[80] Y. 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.

[81] 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. ISSN 1389-1286.

[82] S. Zhang, J. Wu, and S. Lu. Virtual network embedding with substrate support for parallelization. In *To appear in Proc. of the IEEE Global Telecommunications Conference*, 2012.

[83] Hao Di, Hongfang Yu, Vishal Anand, Lemin Li, Gang Sun, and Binhong Dong. Efficient online virtual network mapping using resource evaluation. *Journal of Network and Systems Management*, 20:468–488, 2012. ISSN 1064-7570.

[84] V. Abedifar and M. Eshghi. A novel routing and wavelength assignment in virtual network mapping based on the minimum path algorithm. In *Ubiquitous and Future Networks (ICUFN), 2012 Fourth International Conference on*, pages 204 –208, july 2012.

[85] Aris Leivadeas, Chrysa A. Papagianni, and Symeon Papavassiliou. Socio-aware virtual network embedding. *IEEE Network*, 26 (5):35–43, 2012.

[86] Tae-Ho Lee, Shahnaza Tursunova, and Tae-Sang Choi. Graph clustering based provisioning algorithm for virtual network embedding. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pages 1175 –1178, april 2012.

[87] Marcin Bienkowski, Anja Feldmann, Dan Jurca, Wolfang Kellerer, Gregor Schaffrath, Stefan Schmid, and Joerg Widmer. Competitive analysis for service migration in vnets. In *Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures*, VISA '10, pages 17–24, New York, NY, USA, 2010. ACM.

[88] 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, april 2006.

[89] 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, dec. 2010.

[90] Zhang Shun-li and Qiu Xue-song. A novel virtual network mapping algorithm for cost minimizing. *Cyber Journals: Journal of Selected Areas in Telecommunications (JSAT)*, 02(01):1–9, January 2011.

[91] Gang Sun, Hongfang Yu, Vishal Anand, and Lemin Li. A cost efficient framework and algorithm for embedding dynamic virtual network requests. *Future Generation Computer Systems*, (0):–, 2012. Available online 25 August 2012.

[92] 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.

[93] I. Houidi, W. Louati, and D. Zeghlache. A distributed and autonomic virtual network mapping framework. In *Autonomic and Autonomous Systems, 2008. ICAS 2008. Fourth International Conference on*, pages 241 –247, march 2008.

[94] Yufeng Xin, Ilia Baldine, Anirban Mandal, Chris Heermann, Jeff Chase, and Aydan Yumerefendi. Embedding virtual topologies in networked clouds. In *Proceedings of the 6th International Conference on Future Internet Technologies*, CFI '11, pages 26–29, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0821-2.

[95] 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, nov. 2010.

[96] A. Leivadeas, C. Papagianni, and S. Papavassiliou. Efficient resource mapping framework over networked clouds via iterated local search based request partitioning. *Parallel and Distributed Systems, IEEE Transactions on*, PP(99):1, 2012.

[97] 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, april 2010.

[98] Xian Zhang, C. Phillips, and Xiuzhong Chen. An overlay mapping model for achieving enhanced qos and resilience performance. In *Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2011 3rd International Congress on*, pages 1 –7, oct. 2011.

[99] Cong Wang and T. Wolf. Virtual network mapping with traffic matrices. In *Architectures for Networking and Communications Systems (ANCS), 2011 Seventh ACM/IEEE Symposium on*, pages 225 –226, oct. 2011.

[100] J. Shamsi and M. Brockmeyer. Qosmap: Qos aware mapping of virtual networks for resiliency and efficiency. In *Globecom Workshops, 2007 IEEE*, pages 1 –6, nov. 2007.

[101] J. Shamsi and M. Brockmeyer. Efficient and dependable overlay networks. In *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pages 1 –8, april 2008.

[102] J. Shamsi and M. Brockmeyer. Qosmap: Achieving quality and resilience through overlay construction. In *Internet and Web Applications and Services, 2009. ICIW '09. Fourth International Conference on*, pages 58 –67, may 2009.

[103] G. Koslovski, Wai-Leong Yeow, C. Westphal, Tram Truong Huu, J. Montagnat, and P. Vicat-Blanc. Reliability support in virtual infrastructures. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 49 –58, dec. 2010.

[104] Hongfang Yu, Chunming Qiao, V. 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, dec. 2010.

[105] Pin Lv, Zhiping Cai, Jia Xu, and Ming Xu. Multicast service-oriented virtual network embedding in wireless mesh networks. *Communications Letters, IEEE*, 16(3):375 –377, march 2012. ISSN 1089-7798.

[106] Gang. Sun, Hao. Di, Hongfang. Yu, Lemin. Li, and Vishal. Anand. The Framework and Algorithms for the Survivable Mapping of Virtual Network onto a Substrate Network. *IETE Technical Review*, 28(5):381–391, 2011.

[107] Xiujiao Gao, Hongfang Yu, V. Anand, Gang Sun, and Hao Di. A new algorithm with coordinated node and link mapping for virtual network embedding based on lp relaxation. In *Communications and Photonics Conference and Exhibition (ACP), 2010 Asia*, pages 152 –153, dec. 2010.

[108] Fan Yang, Zhen kai Wang, Jian ya Chen, and Yun jie Liu. Vlb-vne: A regionalized valiant load-balancing algorithm in virtual network mapping. In *Wireless Communications, Networking and Information Security (WCNIS), 2010 IEEE International Conference on*, pages 432 –436, june 2010.

[109] 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, nov. 2010.

[110] 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, dec. 2010.

[111] Yang Yu, Chen Shan-zhi, Li Xin, and Wang Yan. Rmap: An algorithm of virtual network resilience mapping. In *Wireless Communications, Networking and Mobile Computing (WiCOM), 2011 7th International Conference on*, pages 1 –4, sept. 2011.

[112] Sheng Zhang, Zhuzhong Qian, Song Guo, and Sanglu Lu. Fell: A flexible virtual network embedding algorithm with guaranteed load balancing. In *2011 IEEE International Conference on Communications (ICC),*, pages 1 –5, june 2011.

[113] Gang Sun, Hongfang Yu, Vishal Anand, Lemin Li, and Hao Di. Optimal provisioning for virtual network request in cloud-based data centers. *Photonic Network Communications*, pages 1–14, 2012. ISSN 1387-974X.

[114] Pin Lv, Xudong Wang, and Ming Xu. Virtual access network embedding in wireless mesh networks. *Ad Hoc Networks*, 10(7): 1362 – 1378, 2012. ISSN 1570-8705.

[115] Xian Zhang, Xiuzhong Chen, and Chris Phillips. Achieving effective resilience for qos-aware application mapping. *Computer Networks*, 56(14):3179 – 3191, 2012. ISSN 1389-1286.

[116] Gregor Schaffrath, Stefan Schmid, and Anja Feldmann. Generalized and resource-efficient vnet embeddings with migrations. *Arxiv preprint arXiv10124066*, 2010.

[117] Dongdong Chen, Xuesong Qiu, Zhaowei Qu, Shunli Zhang, and Wenjing Li. Algorithm for virtual nodes reconfiguration on network virtualization. In *Advanced Intelligence and Awareness Internet (AIAI 2011), 2011 International Conference on*, pages 333 –337, oct. 2011.

[118] 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. ISBN 978-1-4503-0199-2.

[119] Ruay-Shiung Chang and Chia-Ming Wu. Green virtual networks for cloud computing. In *Communications and Networking in China (CHINACOM), 2010 5th International ICST Conference on*, pages 1 –7, aug. 2010.

[120] Chao-Tung Yang, Kuan-Chieh Wang, Hsiang-Yao Cheng, Cheng-Ta Kuo, and William Cheng C. Chu. Green power management with dynamic resource allocation for cloud virtual machines. In *Proceedings of the 2011 IEEE International Conference on High Performance Computing and Communications*, HPCC '11, pages 726 –733, sept. 2011.

[121] 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, april 2008.

[122] A.P. Bianzino, C. Chaudet, D. Rossi, and J.-L. Rougier. A survey of green networking research. *Communications Surveys Tutorials, IEEE*, 14(1):3 –20, quarter 2012. ISSN 1553-877X.

[123] Andreas Fischer and Hermann De Meer. Position paper: Secure virtual network embedding. *Praxis der Informationsverarbeitung und Kommunikation*, 34(4):190–193, 2011.

[124] Gregory Smith, Anmol Chaturvedi, Arunesh Mishra, and Suman Banerjee. Wireless virtualization on commodity 802.11 hardware. In *Proceedings of the second ACM international workshop on Wireless network testbeds, experimental evaluation and characterization*, WinTECH '07, pages 75–82, 2007.

[125] S. Perez, J.M. Cabero, and E. Miguel. Virtualization of the wireless medium: A simulation-based study. In *Vehicular Technology Conference, 2009. VTC Spring 2009. IEEE 69th*, pages 1 –5, april 2009.

[126] S. Singhal, G. Hadjichristofi, I. Seskar, and D. Raychaudhri. Evaluation of uml based wireless network virtualization. In *Next Generation Internet Networks, 2008. NGI 2008*, pages 223 –230, april 2008.

[127] Anura P. Jayasumana, Qi Han, and Tissa H. Illangasekare. Virtual sensor networks - a resource efficient approach for concurrent applications. In *Proceedings of the International Conference*

on Information Technology, ITNG '07, pages 111–115, Washington, DC, USA, 2007. IEEE Computer Society.

[128] R. Mahindra, G.D. Bhanage, G. Hadjichristofi, I. Seskar, D. Raychaudhuri, and Y.Y. Zhang. Space versus time separation for wireless virtualization on an indoor grid. In *Next Generation Internet Networks, 2008. NGI 2008*, pages 215 –222, april 2008.

[129] Keun-mo Park and Chong-kwon Kim. A framework for virtual network embedding in wireless networks. In *Proceedings of the 4th International Conference on Future Internet Technologies*, CFI '09, pages 5–7, New York, NY, USA, 2009. ACM.

[130] Mauricio G.C. Resende and Celso C. Ribeiro. Greedy randomized adaptive search procedures: Advances, hybridizations, and applications. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of Metaheuristics*, volume 146, pages 283–319. Springer US, 2010. ISBN 978-1-4419-1665-5.

[131] M. Furini and D. F. Towsley. Real time traffic transmissions over the internet. *IEEE Transactions on Multimedia*, 3:33–40, March 2001.

[132] A. Jukan and G. Franzl. Path selection methods with multiple constraints in service-guaranteed WDM networks. *IEEE/ACM Transactions on Networking*, 12(1):59–72, February 2004.

[133] B. Quoitin, C. Pelsser, L. Swinnen, O. Bonaventure, and S. Uhlig. Interdomain traffic engineering with bgp. *Communications Magazine, IEEE*, 41(5):122 – 128, may 2003. ISSN 0163-6804.

[134] P. Su and M. Gellman. Using adaptive routing to achieve quality of service. *Performance Evaluation*, 57:105–119, June 2004.

[135] Xipeng Xiao. *Providing quality of service in the internet*. PhD thesis, East Lansing, MI, USA, 2000. AAI9972020.

[136] Y. Bejerano, Y. Breitbart, A. Orda, R. Rastogi, and A. Sprintson. Algorithms for computing qos paths with restoration. *Networking, IEEE/ACM Transactions on*, 13:648–661, june 2005. ISSN 1063-6692.

[137] N. Fujita and A. Iwata. Adaptive and efficient multiple path pre-computation for qos routing protocols. In *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*, USA, 2001.

[138] F. Kuipers, P. Van Mieghem, T. Korkmaz, and M. Krunz. An overview of constraint-based path selection algorithms for qos routing. *Communications Magazine, IEEE*, 40:50–55, dec 2002.

[139] Lin Shen, Mingwei Xu, Ke Xu, Yong Cui, and Youjian Zhao. Simple quality-of-service path first protocol and modeling analysis. In *Communications, IEEE International Conference on*, China, june 2004.

[140] P. V. Mieghem and F. A. Kuipers. Concepts of exact QoS routing algorithms. *IEEE/ACM Transactions on Networking*, 12:415–428, June 2004.

[141] P. V. Mieghem and F. A. Kuipers. Conditions that impact the complexity of QoS routing. *IEEE/ACM Transactions on Networking*, 13:717–730, August 2005.

[142] M. G. Gouda and M. Schneider. Maximizable routing metrics. *IEEE/ACM Trans. Networking*, 11(4):663–675, August 2003.

[143] C.M. Lagoa, Hao Che, and B.A. Movsichoff. Adaptive control algorithms for decentralized optimal traffic engineering in the internet. *Networking, IEEE/ACM Transactions on*, 12:415–428, june 2004. ISSN 1063-6692.

[144] T. Miyamura, T. Kurimoto, and M. Aoki. Enhancing the network scalability of link-state routing protocols by reducing their flooding overhead. In *High Performance Switching and Routing, 2003, HPSR. Workshop on*, Italy, june 2003.

[145] D. Pei, X. Zhao, D. Massey, and L. Zhang. A study of bgp path vector route looping behavior. In *Distributed Computing Systems, 2004. Proceedings. 24th International Conference on*, Japan, 2004.

[146] Joao Luis Sobrinho. Network routing with path vector protocols: theory and applications. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '03, Germany, 2003.

[147] R. Wattenhofer and S. Venkatachary. The impact of internet policy and topology on delayed routing convergence. In *INFOCOM 2001 - 20th Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 537–546, Anchorage, Alaska, EUA, April 2001. IEEE Computer and Communications Societies.

[148] A. D. Jaggard and V. Ramachandran. Relating two formal models of path-vector routing. In *INFOCOM 2005 - 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 1, pages 619–630, Miami, FL, EUA, March 2005. IEEE Computer and Communications Societies.

[149] W. P. Herman. *Formulação algébrica para a modelagem de algoritmos de roteamento multi-restritivo hop-by-hop*. Phd thesis, Escola Politécnica da USP, São Paulo, 2008.

[150] W. P. Herman and J. R. Amazonas. Hop-by-hop routing convergence analysis based on paths algebra. In *Proceedings of the 2007 Electronics, Robotics and Automotive Mechanics Conference - CERMA 2007*, Mexico, 2007.

[151] Xavier Hesselbach, Christos Kolias, Ramón Fabregat, Mónica Huerta, and Yezid Donoso. Problems in dynamic bandwidth allocation in connection oriented networks. In Arie Koster and Xavier Muñoz, editors, *Graphs and Algorithms in Communication Networks*, Texts in Theoretical Computer Science. An EATCS Series, pages 179–197. Springer Berlin Heidelberg, 2010.

[152] Yong Liao, Dong Yin, and Lixin Gao. Europa: efficient user mode packet forwarding in network virtualization. In *INM/WREN'10: Proceedings of the 2010 internet network management conference on Research on enterprise networking*, pages 6–6, Berkeley, CA, USA, 2010. USENIX Association.

[153] B. Carré. *Graphs and Networks*. Oxford Applied Mathematics and Computing Science Series. Oxford University Press, Walton Street, Oxford, 1979.

[154] J.L. Sobrinho. On the convergence of path vector routing protocols. In *High Performance Switching and Routing, 2001 IEEE Workshop on*, pages 292 –296, USA, 2001.

[155] J. L. Sobrinho. An algebraic theory of dynaminc network routing. *IEEE/ACM Trans. Networking*, 13(5):1160–1173, October 2005.

[156] G. Chartrand. *Introductory Graph Theory*. Dover Publications, Inc, New York, 1985.

[157] Bernard M. Waxman. Routing of multipoint connections. *IEEE Journal of Selected Areas in Communication*, 6(9):1617–1622, December 1988.

[158] John DC Little. A proof for the queuing formula: L= λw. *Operations research*, 9(3):383–387, 1961.

[159] G. Karypis, V. Kumar, and Vipin Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, 48:96–129, 1998.

[160] M Webb. Smart 2020: Enabling the low carbon economy in the information age. *The Climate Group London*, 2008.

[161] R. Bolla, R. Bruschi, F. Davoli, and F. Cucchietti. Energy efficiency in the future internet: A survey of existing approaches and trends in energy-aware fixed network infrastructures. *Communications Surveys Tutorials, IEEE*, 13(2):223 –244, quarter 2011.

[162] Raffaele Bolla, Roberto Bruschi, Alessandro Carrega, Franco Davoli, Diego Suino, Constantinos Vassilakis, and Anastasios Zafeiropoulos. Cutting the energy bills of internet service providers and telecoms through power management: An impact analysis. *Computer Networks*, 56(10):2320 – 2342, 2012. ISSN 1389-1286.

[163] Priya Mahadevan, Puneet Sharma, Sujata Banerjee, and Parthasarathy Ranganathan. A power benchmarking framework for network devices. In *NETWORKING 2009*, volume 5550 of *Lecture Notes in Computer Science*, pages 795–808. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-01398-0.

[164] R. Bolla, R. Bruschi, A. Cianfrani, and M. Listanti. Enabling backbone networks to sleep. *Network, IEEE*, 25(2):26 –31, march-april 2011.

[165] L. Chiaraviglio, M. Mellia, and F. Neri. Reducing power consumption in backbone networks. In *Communications, 2009. ICC '09. IEEE International Conference on*, pages 1 –6, june 2009.

[166] Will Fisher, Martin Suchara, and Jennifer Rexford. Greening backbone networks: reducing energy consumption by shutting off cables in bundled links. In *Proceedings of the first ACM SIGCOMM workshop on Green networking*, Green Networking '10, pages 29–34, 2010. ISBN 978-1-4503-0196-1.

[167] A.P. Bianzino, C. Chaudet, F. Larroca, D. Rossi, and J. Rougier. Energy-aware routing: A reality check. In *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, pages 1422 –1427, dec. 2010.

[168] J.C.C. Restrepo, C.G. Gruber, and C.M. Machuca. Energy profile aware routing. In *Communications Workshops, 2009. ICC Workshops 2009. IEEE International Conference on*, pages 1 –5, june 2009.

[169] A. Cianfrani, V. Eramo, M. Listanti, M. Polverini, and A.V. Vasilakos. An ospf-integrated routing strategy for qos-aware energy saving in ip backbone networks. *Network and Service Management, IEEE Transactions on*, 9(3):254 –267, september 2012. ISSN 1932-4537.

[170] Juniper Networks. Configuring logical tunnel interfaces. http://www.juniper.net/techpubs/en_US/junos9.6/information-products/topic-collections/config-guide-services/services-configuring-logical-tunnel-interfaces.html, 2010.

[171] Hidetoshi Takeshita, Daisuke Ishii, Satoru Okamoto, Eiji Oki, and Naoaki Yamanaka. Highly energy efficient layer-3 network

architecture based on service cloud and optical aggregation network. *IEICE Transactions*, 94-B(4):894–903, 2011.

[172] S. Lo, M. Ammar, and E. Zegura. Design and analysis of schedules for virtual network migration. Technical Report GT-CS-12-05, Georgia Institute of Technology, 2012. available at https://smartech.gatech.edu/bitstream/handle/1853/44245/GT-CS-12-05.pdf.

[173] Dennis Schwerdel, Daniel Günther, Robert Henjes, Bernd Reuther, and Paul Müller. German-lab experimental facility. *Future Internet Symposium (FIS) 2010*, 9 2010.

[174] Michael Duelli and Julian Ott. Mulavito – multi-layer visualization tool, November 2010. URL http://mulavito.sf.net.

[175] R. Nejabati, D. Simeonidou, V. Reijs, D. Wilson, M. Campanella, B. Belter, J. Jofre, S. Figuerola, C. Tziouvaras, F. Loui, P. Kaufman, T. Breach, D. Salmon, and J. Sharp. Genus: Virtualisation service for géant and european nrens. In *TERENA Networking Conference*, May 2011.

[176] Bo Peng, A. Hammad, R. Nejabati, S. Azodolmolky, D. Simeonidou, and V. Reijs. A network virtualization framework for ip infrastructure provisioning. In *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, pages 679–684, Dec 2011.

[177] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman. A survey of information-centric networking. *Communications Magazine, IEEE*, 50(7):26–36, 2012. ISSN 0163-6804.

[178] Rob Sherwood, Glen Gibb, Kok-kiong Yap, Guido Appenzeller, Martin Casado, Nick Mckeown, and Guru Parulkar. Flowvisor : A network virtualization layer. Technical Report Openflow-tr-2009-1, Stanford University, 2009. available at http://sb.tmit.bme.hu/mediawiki/images/c/c0/FlowVisor.pdf.

[179] Larry Peterson, Steve Muir, Timothy Roscoe, and Aaron Klingaman. PlanetLab Architecture: An Overview. Technical Report PDN–06–031, PlanetLab Consortium, May 2006.

[180] Xavier Sanchez-Loro, José Luis Ferrer, Carles Gomez, Jordi Casademont, and Josep Paradells. Can future internet be based on constrained networks design principles? *Computer Networks*, 55(4):893 – 909, 2011. ISSN 1389-1286.

[181] John Day, Ibrahim Matta, and Karim Mattar. Networking is ipc: a guiding principle to a better internet. In *Proceedings of the 2008 ACM CoNEXT Conference*, CoNEXT '08, pages 1–6, 2008.

[182] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael Plass, Nick Briggs, and Rebecca Braynard. Networking named content. *Commun. ACM*, 55(1):117–124, January 2012. ISSN 0001-0782.

[183] M. Bari, R. Boutaba, R. Esteves, L. Granville, M. Podlesny, M. Rabbani, Q. Zhang, and M. Zhani. Data center network virtualization: A survey. *Communications Surveys Tutorials, IEEE*, PP (99):1 –20, 2012. ISSN 1553-877X.