Multi-Provider Network Service Embedding

Von der Fakultät für Elektrotechnik und Informatik
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des akademischen Grades

Doktor-Ingenieur

genehmigte

Dissertation

von

Dipl.-Ing. David Dietrich
geboren am 22. August 1977 in Lauchhammer

2016

Referent	:	Prof. Dr. Panagiotis Papadimitriou
Korreferent	:	Prof. Dr. Symeon Papavassiliou
Tag der Promotion	:	10. März 2016

Dipl.-Ing. David Dietrich: *Multi-Provider Network Service Embedding,*
Dissertation, © 2016

ABSTRACT

Existing technologies that fulfill the requirements of the steadily emerging network services experience difficulties in transcending organizations or enterprise boundaries which in turn hinders the deployment of network services in wide areas. Network virtualization technologies can overcome this barrier by enabling the concurrent deployment and operation of service-tailored virtual networks (VNs) on top of shared physical infrastructures. In this respect, service providers benefit from high performance and reliability without the need to acquire and deploy physical network equipment. The substrate providers essentially benefit from improved resource efficiency which in turn translates into lower operational and technology investment costs. Similar to VNs, the emerging concept of network function virtualization aims at mitigating limitations of hardware middleboxes in terms of customization, resource efficiency, and manageability. The migration of network functions (NFs) into virtualized infrastructure brings significant benefits to enterprise networks while creating new cloud service models such as NF-as-a-service enabling individual and wide-spanning service chains, i. e., network services.

The thesis at hand focuses on the embedding of VNs and service chains across multiple substrate providers. Wide-area deployment of VNs requires the ability to embed and operate VNs across multiple substrate providers due to their limited geographic footprint. Multi-provider VN embedding raises the need for a layer of indirection, interposed between the service provider and the substrate providers. Essentially, this layer represents a third party that discovers, selects, and allocates resources from multiple substrate providers that are assigned VN segments which are eventually stitched together creating a wide-area VN that can be configured and operated by the service provider. In this respect, the unwillingness of the substrate providers to disclose details of their network topology and resource information exacerbates the selection of substrate providers and therewith the assignment of the virtual resources to the physical resources.

In a similar manner, service chains are embedded into virtualized software middleboxes located in data centers operated by different NF providers. The limited information disclosure policy of the NF providers again aggravates the embedding of service chains. Furthermore, middlebox policies prescribed by network operators, implications of NFs on network traffic, and location dependencies for certain NFs complicate network service embedding. In addition

to that, multi-provider embedding architectures foster competition among the substrate providers that in turn need to attract customers and to maximize their revenue.

We study the challenging problem of multi-provider VN embedding with limited information disclosure and investigate the visibility of substrate network resources. Based on this study, we designed an abstraction model containing non-confidential substrate information and an embedding framework. This framework embeds VNs as traffic matrices in two stages. First, a centralized coordinator partitions the VN request across the substrate providers by using the abstract substrate model as mentioned beforehand. Subsequently, all the involved substrate providers map their respective VN segment to their substrate network by exploiting the fully available substrate information. Evaluation results show that multi-provider VN embedding under limited information disclosure is feasible at the price of moderate extra cost.

Moreover, we tackle the service chain deployment and devise (i) a new service model that simplifies the specification of network service requests and (ii) a topology abstraction for service chain partitioning. In this respect, we present a two-stage framework for the embedding of service chains where a network service composition layer is interposed between the clients and the NF providers. By using this framework, we evaluate the impact of different business objectives for network service embedding.

We further show that the embedding of a VN request yields a different profitability from the perspective of the substrate provider depending on the substrate network and the VN request demands. We present an algorithm that takes into account a policy dimension at which substrate providers enforce profitability in VN embedding. A comprehensive implementation work, predominantly relying on linear programming methods, is used for the evaluation of VN and service chain embedding.

Keywords: Network Function Virtualization, Service Chaining, Resource Allocation, Topology Abstractions, Combinatorial Optimization

ZUSAMMENFASSUNG

Die ständig wachsenden Anforderungen von Netzwerkdiensten gebieten den Einsatz neuartiger Technologien in einem Umfeld in dem mehrere administrative Systeme involviert sind, also Unternehmensgrenzen überschritten werden. Dies birgt jedoch Schwierigkeiten die eine breite Zurverfügungstellung von Netzwerkdiensten in der Fläche behindern. Netzwerk-Virtualisierung ist ein möglicher Ansatz zur Umgehung dieser Schwierigkeiten, indem der gleichzeitige Betrieb von virtuellen Netzen auf derselben physischen Infrastruktur ermöglicht wird. Dabei kann den Netzwerkdienst-Anbietern ein bestimmtes Maß an Performanz und Zuverlässigkeit zugesichert werden ohne dass sie selbst technisches Equipment anschaffen und betreiben müssen. Gleichzeitig profitieren die Infrastruktur-Betreiber von effizienterer Ressourcennutzung, was sich in niedrigeren Betriebs- und Anschaffungskosten für die Infrastruktur niederschlägt. Darüber hinaus können die Nachteile von Middleboxen, in Bezug auf Anpassbarkeit, Ressourcennutzung und Administrierbarkeit, durch Virtualisierung eliminiert werden. Dies ermöglicht beispielsweise die geografisch unabhängige Nutzung von virtuellen Netzfuntionen. Entsprechend des Anwendungsfalls können Serviceketten erstellt werden, in denen festgelegt ist, welche Netzfunktionen in welcher Reihenfolge auf einen bestimmten Datenfluss angewandt werden.

Die vorliegende Dissertation beschäftigt sich mit der Substrat-Netzwerke übergreifenden Einbettung von Netzwerkdiensten mit dem Ziel, ihre geografische Reichweite beliebig zu erweitern. Das Einbetten von virtuellen Netzen erfordert eine zusätzliche Instanz zwischen dem Netzwerkdienst-Anbieter und den Substrat-Netzwerk-Betreibern. Diese dient im Wesentlichen der Identifikation von potentiell nutzbaren Ressourcen verschiedener Substrat-Netzwerk-Anbieter sowie deren Zuordnung zu den angefragten Ressourcen. Die Betreiber der Substrat-Netzwerke übernehmen dann die finale Zuteilung ihrer physischen Ressourcen zu dem ihm zugeordneten Segment. Zusammen formen diese Segmente dann das virtuelle Netz welches dann vom Netzwerkdienst-Anbieter individuell konfiguriert und betrieben wird. Die Auswahl der Substrat-Netzwerke und somit die Zuordnung der virtuellen zu den physischen Ressourcen wird allerdings erschwert durch den Umstand, dass Substrat-Netzwerk-Anbieter nicht bereit sind, Details über ihre Netzwerktopologie und Ressourcen preiszugeben.

Serviceketten werden in virtualisierte Software-basierte Middlexboxen in Rechenzentren eingebettet, die durch verschiedene Netzfunktions-Anbieter be-

trieben werden. Diese versuchen ebenso jegliche Substrat-Information für Außenstehende zu verschleiern. Außerdem müssen bei der Virtualisierung von Middlexboxen weitere Bedingungen, wie existierende Middlebox-Regeln, Einflüsse der Netzfunktionen auf die Datenrate sowie örtliche Abhängigkeiten, berücksichtigt werden. Allgemein ist anzunehmen und deshalb zu berücksichtigen, dass in Umgebungen mit konkurierenden Substrat- und Netzfunktions-Anbietern jeweils das Ziel verfolgt wird, möglichst viele Kunden zu gewinnen und den Umsatz zu maximieren.

Wir analysieren die Sichtbarkeit der physischen Ressourcen und untersuchen Möglichkeiten des Einbettens von virtuellen Netzen über mehrere Substrat-Netzwerke. Zu diesem Zweck entwerfen wir erstens ein Substrat-Modell welches nur nicht-vertrauliche Informationen zulässt und zweitens ein Framework zum Einbetten von virtuellen Netzen. Dieses verarbeitet in zwei Schritten Anfragen über virtuelle Netze mit Verkehrsmatrizen. Zuerst partitioniert ein zentraler Koordinator das virtuelle Netz auf Basis des oben genannten Substrat-Modells in Segmente, die je einem Substrat-Netzwerk-Anbieter zugeordnet sind. Im Anschluss stellen die betreffenden Substrat-Netzwerk-Anbieter physische Ressourcen für diese Segmente zur Verfügung. Wir zeigen, dass das Einbetten von Netzwerkdiensten über mehrere Substrat-Netzwerk-Anbieter trotz begrenzter Ressourcen-Sichtbarkeit möglich ist, wenn auch zu moderaten Zusatzkosten.

Darüber hinaus gehen wir auf das Einbetten von Serviceketten ein. Dazu entwickeln wir ein neues Netzwerkdienst-Modell, welches die Spezifizierung von Netzwerkdienst-Anfragen vereinfacht. Außerdem entwerfen wir ein abstraktes Topologie-Modell, welches wir in unserem zweistufigen Framework zur Partitionierung von Serviceketten und der Zuordnung von Netzfunktions-Anbietern nutzen. Mithilfe dieses Frameworks untersuchen wir unter anderem die Anwendung verschiedener Unternehmensziele.

Weiterhin zeigen wir, dass die Profitabilität eines einzubettenden virtuellen Netzes variieren kann. Wir stellen einen Algorithmus vor, der es dem Substrat-Netzwerk-Betreiber ermöglicht, ein bestimmtes Maß an Profitabilität sicherzustellen. Umfangreiche Implementierungsarbeiten, welche im Kern auf linearer Programmierung basieren, dienen als Basis für die Evaluierung des Einbettens von Netzwerkdiensten über mehrere Substrat-Netzwerke.

Schlagwörter: Netzfunktionen-Virtualisierung, Serviceketten, Ressourcen-Allokation, Abstrakte Topologien, Kombinatorische Optimierung

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

API	Application Programming Interface
AS	Autonomous System
BGP	Border Gateway Protocol
CAPEX	Capital Expenditure
CDF	Cumulative Distribution Function
CPU	Central Processing Unit
CRR	Cost to Revenue Ratio
DC	Data Center
FID	Full Information Disclosure
GLPK	GNU Linear Programming Kit
GRE	Generic Routing Encapsulation
GUI	Graphical User Interface
IDS	Intrusion Detection System
ILP	Integer Linear Programming
ILS	Iterated Local Search
InP	Infrastructure Provider
I/O	Input/Output
IP	Internet Protocol
IPS	Intrusion Prevention System
IPTV	Internet Protocol Television
ISP	Internet Service Provider
IXP	Internet Exchange Point
LAN	Local Area Network
LID	Limited Information Disclosure
LP	Linear Programming
Mbps	Megabits per second
MCF	Multi-Commodity Flow

MED	Multi-Exit Discriminator
MILP	Mixed-Integer Linear Programming
MIP	Mixed-Integer Programming
MIQP	Mixed-Integer Quadratic Programming
MPLS	Multiprotocol Label Switching
NF	Network Function
NFaaS	Network Function as a Service
NFP	Network Function Provider
NFV	Network Function Virtualization
NSCL	Network Service Composition Layer
NSE	Network Service Embedding
OPEX	Operational Expenditure
OS	Operating System
OSPF	Open Shortest Path First
PoP	Point of Presence
QoS	Quality of Service
RE	Redundancy Eliminator
SDN	Software-Defined Networking
SP	Service Provider
ToR	Top of Rack
VGW	Virtual Gateway
VN	Virtual Network
VNE	Virtual Network Embedding
VPN	Virtual Private Network
WAN	Wide Area Network
XML	Extensible Markup Language

Part I

DISSERTATION

1 INTRODUCTION

The Internet has evolved into a mass medium where an enormous number of network applications became an integral part of the people's lives. Video conferencing, IPTV, and online gaming are examples of network applications that have been established over the last years while a steady emergence of new network applications can be expected in the future. This, in fact, leads to a continuous development of requirements for data delivery over the Internet in terms of performance, reliability, and security. The Internet architecture and the core protocols cannot cope with these challenging requirements of today and beyond. Current Internet users typically experience best-effort data delivery at which traffic paths span multiple substrate networks (autonomous systems, ASs) without any service level guarantee. Existing technologies for quality of service (QoS), robust routing, and security experience difficulties in transcending organization or enterprise boundaries. This in turn hinders the wide-area deployment of network services, specifically at the presence of service level requirements in terms of throughput, delay, packet loss, or security.

Today, service level agreements are not or only partially supported by major Internet core protocols such as (i) the dominating exterior gateway protocol BGP [1] for inter-AS routing, (ii) intra-AS routing protocols (e. g., OSPF [2]), and (iii) the Internet protocol (IP) itself. We have been waiting decades for changes in the Internet core to overcome such limitations while the operators lack the incentives to deploy new disruptive technologies in their production networks. The risk potential finally obstructs innovative changes to the Internet core. Instead, innovation has predominantly taken place at the Internet edges and within the applications themselves. Fundamental changes, for example, debated by Shenker [3], Clark et al. [4], or Feldmann [5], have been postponed to the future. The Internet still works, and the incentives for the ISPs to become active seem still to be not strong enough [6].

However, network virtualization is seen as a potential means to overcome the Internet ossification towards a diversified Internet as it enables the concurrent deployment and operation of service-tailored virtual networks (VNs) on top of shared physical infrastructures. The respective requirements (e. g., high packet forwarding rates, performance isolation among virtual components, packet classification offload) are covered by recent advances on the server, router, and network interface virtualization (e. g., [7–11]). As such, network virtualization

significantly lowers the barrier to introducing changes to the Internet core and, therefore, fosters innovation [12, 13]. Network virtualization further allows a paradigm shift towards more specific business roles. The owner of the physical substrate acts as an *infrastructure provider (InP)* that operates and leases his physical resources (CPU, bandwidth, etc.) while a *service provider (SP)* represents another new business role running services on top of leased infrastructures. In this respect, both the operators of the VNs (the SPs) and the owners of the infrastructure (the InPs) benefit from network virtualization. SPs can deploy network services within customized VNs that provide high performance and reliability, without the need to acquire and deploy physical network equipment. For InPs, network virtualization improves resource efficiency, reduces the capital expenditure (CAPEX) as well as the operational expenditure (OPEX).

The network virtualization approach takes into account the links of a substrate topology and servers that host multi-purpose virtual machines. Beyond that, middleboxes moved in the focus of network virtualization. Middleboxes embed a broad range of flow processing functions, i. e., *network functions (NFs)*, into the network infrastructure, satisfying the increasing needs of network operators and end-users. Prominent examples for currently deployed middleboxes are IP and application firewalls, WAN optimizers, proxies, application gateways, virtual private networks (VPNs), load balancers, intrusion detection and prevention systems (IDS/IPS) [14, 15]. Despite their widespread adoption, middleboxes exhibit significant limitations in terms of customization, resource efficiency, and manageability. More precisely, middleboxes are typically built of specialized hardware and offer single-purpose functionality, leading to appliance sprawl and increased CAPEX and OPEX for enterprise networks [14, 16]. *Network function virtualization (NFV)* is an emerging concept that aims at mitigating some of these problems by enabling the consolidation of NFs on platforms built of commodity components [17, 18]. Similar to the aforementioned cost savings by VN deployment, NFV can reduce the expenses for NF deployment either by deploying consolidated software middleboxes in enterprise networks or by outsourcing NFs to virtualized network infrastructures. The latter, in particular, is very appealing to enterprises, since NF as a service (NFaaS) obviates the need to acquire, deploy, and operate additional network appliances on clients' premises, leading to significant savings in CAPEX and OPEX. Furthermore, the recent trend of micro-data center deployment by large ISPs leads to a more considerable number of NFV points of presence (PoPs) [19]. The clients in turn can benefit from service chains in which flows individually traverse NFs fulfilling the network service requirements without geographic dependencies incurred by hardware middleboxes.

Wide-area network service deployment typically requires coverage of wide geographic areas that exceed the limited footprint of a single resource provider. To overcome this limitation, we embed VNs and service chains across multiple InPs and NF providers each fulfilling certain geographic and resource requirements. The multi-provider *VN embedding (VNE)* and the multi-provider service chain embedding (*network service embedding*, NSE) raises the need for a layer of indirection, interposed between the SPs (or clients) and the InPs (or NF providers) [12, 20]. Essentially, this layer represents a third party that discovers, selects, and allocates from multiple InPs and NF providers resources in terms of bandwidth and computing capacity. The network service can then finally be deployed by using these resources.

1.1 CHALLENGES

Multi-provider VN and service chain embedding is mainly associated with the following challenges.

- VNE across multiple substrate networks entails significant challenges, primarily due to InPs' policies that restrict resource information disclosure and hinder interoperability with other parties. For example, considering the InPs' policies, the disclosure of router-level topologies is prohibitive. This problem can be rectified by partitioning VN requests into VN segments, which are subsequently mapped onto the substrate networks by the InPs. Nevertheless, partitioning with the knowledge that is based merely on **limited resource information** can still cause suboptimality in VNE. Similar to VNs, the embedding of service chains is affected by the limited knowledge of the substrate topologies.

- Compared to VNs, the embedding of **service chains** is further exacerbated by additional requirements, such as the effect of **traffic rate modification** throughout the chain caused by certain NFs that amplify or reduce traffic. In addition to that, NFs can be associated with **location dependencies**. For example, proxies and caches should be placed in proximity to the enterprise network while packet filters should be deployed close to traffic sources for increased bandwidth conservation in the event of denial-of-service attacks.

- Multi-provider VNE/NSE fosters competition among the providers for resources (i. e., virtual machines, link bandwidth, and NFs). Thus, resource providers need policies for their market participation. Distributed and auction-based embedding architectures particularly raise the requirement

for policy-compliant VNE/NSE at which providers only attract the **profitable VNs or service chains**. In this respect, a metric for profitability needs to be defined and later on applied, in a form of threshold, to the potential VN and service chain embeddings.

- VNE/NSE seeks for each request a feasible assignment of physical resources according to the demand. In particular, a request contains nodes and links that are associated with demands in terms of virtual machine CPU cycles, bandwidth demands, and additional constraints such as geographic dependency. Such an assignment should be near to the optimal solution with respect to a given objective such as cost minimization. The **complexity of the combinatorial optimization problems** formulated in this work can be compared to related problems such as the multiway-separator problem where finding an optimal solution is known to be NP-hard [21]. Moreover, the multi-commodity flow problem, at which multiple flow demands between different sources and sinks are concurrently assigned to a substrate network, is known to be NP-complete [22]. Such problems are generally known to be computationally intractable.

1.2 THESIS CONTRIBUTION

In this work, we tackle the above-mentioned challenges as follows.

- We conduct a feasibility study of multi-provider VNE with limited information disclosure (LID) at which we first discuss the visibility of a third party on substrate network resources. We define a realistic level of information disclosure based on the composition of resource and network topology information that is not treated as confidential by InPs. Based on that, we formulate an abstraction model that can be used by a centralized coordinator (i. e., VN provider) to assess the resource availability of an InP. Our study additionally discusses limitations of VNE with topology-based VN requests. Instead of topologies, we formulate VN requests as traffic matrices which yield higher acceptance rates. We further present a framework that decomposes multi-provider VNE into a set of operations allowing VN providers and InPs to process incoming VN requests based on their visibility on the substrate resources. In particular, the VN provider matches requested resources to offered resources and partitions VN requests across multiple InPs. Subsequently, the InPs map the VN segments to their substrate topologies exploiting their complete knowledge of their substrate network. We apply linear programming methods to the VN request partitioning and the VN segment mapping problem. In this

respect, we introduce integer linear programming formulations and their transformations to linear programming equivalents using relaxation and rounding techniques.

- We further contribute with our holistic approach to multi-provider NSE taking into account the additional requirements of service chains. In particular, we propose an NSE orchestrator, which generates efficient embeddings via network graph rendering, request partitioning among data centers (DCs), and NF subgraph mappings onto the DC networks. In this respect, we introduce a new service model, tailored to NSE, that simplifies (i) the specification of network service requests and (ii) the estimation of computing and bandwidth demands for each request. We further define a topology abstraction that facilitates request partitioning while obscuring any information that is deemed confidential by NF providers. Our NSE orchestrator provides the rendering of such topology abstractions from detailed topology graphs accessed only by NF providers. We decouple network service composition from NF providers by interposing a network service composition layer between the clients and the NF providers. Similar to our VNE framework, we rely on (integer) linear programming methods for NSE.

- We introduce a policy dimension to the VNE problem, allowing only profitable VNEs according to the InP's policy. Such policy can express the balance between the generated revenue and resource efficiency as well as specific constraints that the InP wants to apply. In particular, we use the *embedding cost to revenue ratio (CRR)* in order to express the profit that a provider generates from the embedding of a VN request. In this respect, an InP can set an upper bound to CRR, adjusting the trade-off between revenue generation rate and resource efficiency. This CRR threshold can be adjusted dynamically based on the substrate network resource utilization and the VN request arrival rate. Relying on CRR bounds for expressing VNE policies, we present a VNE algorithm that embeds the most profitable subset of a VN request according to the InP's policy. Our simulation results show that our algorithm can increase the generated revenue by a large margin depending on the policy adjustment. Our algorithm allows an InP to trade short-term revenue gains for higher revenue in the long term and cope better with evolving demands. The proposed algorithm can comprise an essential component of any multi-provider VNE architecture that facilitates the relaying of VN requests across the InPs or auction mechanisms.

- Our comprehensive implementation work constitutes a substantial framework for the realization and evaluation of different VNE/NSE approaches. The core of the C/C++ based implementation is formed by a set of libraries basically providing functionality in terms of topology and request generation, request processing, logging, and statistics. Theses libraries have been used for the proposed VNE/NSE frameworks and also for a visualized and distributed evaluation of VNE/NSE. In this respect, our implementation forms a basis from which future work in the field of network service embedding can benefit.

1.3 THESIS OUTLINE

The remainder of this thesis is structured as follows.

- Chapter 2 provides the background for this work including an overview of the current Internet structure, enabling technologies and concepts for the wide-area deployment of network services and embedding methods.
- Chapter 3 describes our framework for multi-provider VNE in which we conduct a feasibility study of VNE with restricted view on resource information.
- Chapter 4 introduces a new policy dimension to the VNE problem and shows that InPs benefit from restricting the embedding of unprofitable VN requests.
- Chapter 5 presents our NSE orchestrator, a holistic approach to multi-provider NSE dealing with the additional requirements of service chains.
- Chapter 6 shows details related to the implementation work including an automated evaluation tool with GUI for multi-provider VNE/NSE.
- Chapter 7 highlights our conclusions.

2

BACKGROUND

In today's Internet "best-effort" data delivery is common practice, especially across the ISPs' domain borders [23]. This Internet represents the underlying substrate over which network services are deployed. As a consequence, network services are hindered with regard to service quality and new feature development. Nevertheless, a great bunch of technologies and concepts exists to overcome these limitations. Specifically, virtualization technologies and management concepts allow the embedding of (i) virtual networks (VNs) and (ii) service chains into multiple substrate networks and DCs under consideration of service guarantees. Multi-provider environments raise the need for architectures that facilitate coordination among the requester (i. e., SP) and the different substrate providers. In this context, coordination particularly refers to the identification of potential resource providers and the assignment of physical resources to all the requested resources followed by the service instantiation. However, these steps imply the partitioning of the requests across multiple substrate providers and the resource mapping for each VN segment and NF subgraph. Both, request partitioning and resource mapping represent combinatorial optimization problems on which we focus in this thesis.

This chapter is structured as follows. Section 2.1 provides an overview of the Internet structure, i. e., the substrate over which we deploy network services. Section 2.2 discusses the enabling technologies and concepts for wide-area deployment of network services. Section 2.3 specifies the two different request types: VNs and service chains. Section 2.4 gives an overview of multi-provider architectures for the embedding of VNs and service chains, followed by Section 2.5 that describes the needed embedding steps using a centralized embedding architecture. Section 2.6 introduces linear programming optimization methods for VN and service chain embedding.

2.1 BASIC INTERNET STRUCTURE

The Internet represents the interconnected substrate networks into which we embed network services. In the following, we distinguish between (i) the ISP networks which are basically used for connectivity and transit; and (ii) data

centers (DCs) which are mainly used for cloud services and virtual machine hosting.[1]

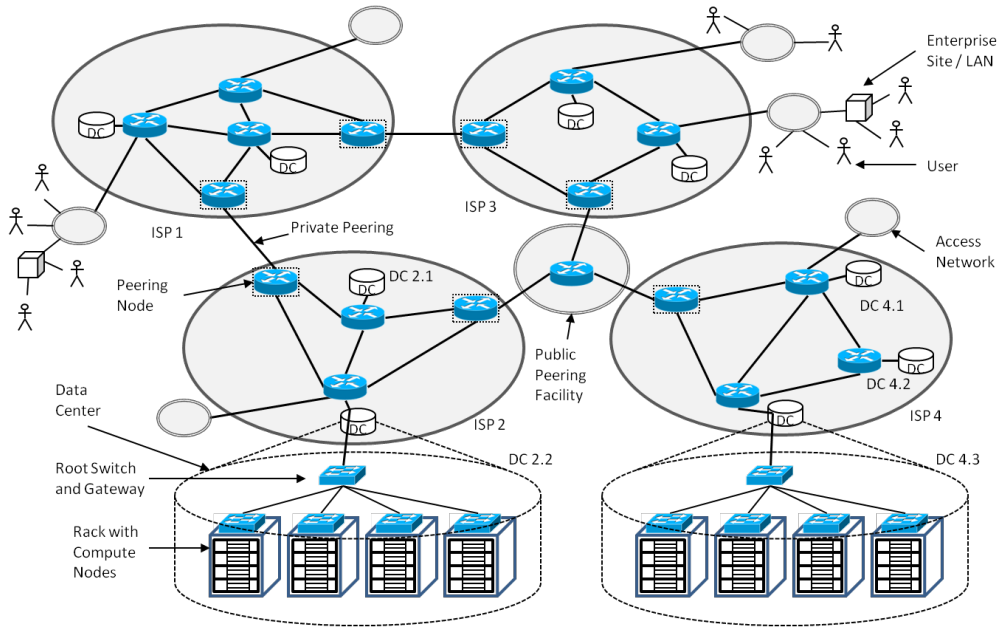


Figure 1: The Internet is composed of numerous autonomous systems, including access and backbone networks. Data centers attached to the Internet provide compute capacity.

2.1.1 *ISP Substrate Networks*

Figure 1 exemplifies the basic Internet structure. The Internet consists of numerous autonomous systems (ASs), i. e., individually administered ISP substrate networks. The subscribers of each ISP are via access networks connected to the ISP's core network. Institutions such as enterprises or universities operate LANs or even own ASs. The different ASs are interconnected in order to achieve reachability of all users in any AS. Interconnections between ASs are realized over peerings that are associated with certain agreements (e. g., with respect to IP transit). Nevertheless, the actual AS path to be traversed between the endpoints is determined in a decentralized manner by the dominating exterior gateway protocol BGP [1, 25]. In this context, the lack of coordination hinders wider deployment of domain-spanning service guarantees although technically available [26, 27]. It can be argued that each provider should be able to reserve a specific resource, e. g., bandwidth, for a requester and between a pair of end-

1 Armbrust et al. provide an overview of cloud computing [24].

points or points of presence (PoPs), as long as he is willing to pay for this extra service. At present, the majority of network services, especially the ones spanning wider areas, is merely based on "best-effort" data delivery.

2.1.2 *Data Centers*

Cloud providers lease compute capacity in form of virtual machines hosted on top of the physical machines, i. e., servers, of a DC. A network topology within the DCs interconnects a large number of servers among each other. The DCs' gateway further provides connectivity to other DCs over an inter-DC network. In the following, we discuss two common DC topologies based on (i) the simple fat-tree and (ii) the k-ary fat-tree. Figure 2 exemplifies a simple fat-tree-based DC topology. In this topology, ToR and aggregation switches steer all traffic over a single powerful core switch whose attached links need to be overprovisioned accordingly. This in turn could cause service degradation due to overutilized links. Furthermore, the single core switch is a single point of failure.

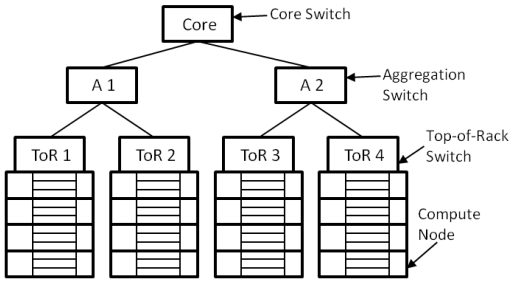


Figure 2: In a simple fat-tree data center topology, all traffic traverses a single powerful core switch.

In contrast to that, the k-ary fat-tree-based DC topology as exemplified by Figure 3 consists of multiple core switches and an aggregation layer with multiple pods.[2] Such topologies eliminate the drawbacks of the before-mentioned simple fat-tree topology at the cost of a higher number of switches and links for the same number of servers. [3]

2 This example uses k=4.
3 Al-Fares et al. provide a study of common practice for DC networks [28].

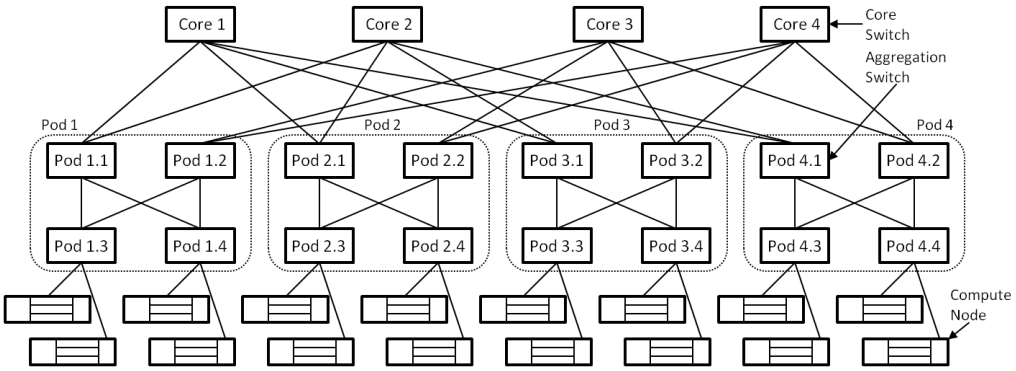


Figure 3: The k-ary fat-tree is an alternative data center topology comprising multiple pods and core switches.

2.2 ENABLING TECHNOLOGIES AND CONCEPTS

The network virtualization paradigm entails an abstraction layer at which physical resources are viewed as logical instances. This concept enables the concurrent deployment and operation of service-tailored VNs on top of shared physical infrastructures – even across multiple substrate networks [29, 30]. As such, network virtualization is seen as a viable path towards wide-area network service deployment. VNs are particularly enabled by host and link virtualization technologies as described in the Sections 2.2.1 and 2.2.2. In addition to that, the emerging concept of *network function virtualization (NFV)* aims at the replacement of inflexible hardware middleboxes by virtual instances that are run flexibly on commodity servers. In this respect, NFV extends the network virtualization concept and enables flexible deployment of service chains. Section 2.2.3 provides an overview of recent works related to NFV. Moreover, Section 2.2.4 introduces the *software-defined networking* (SDN) concept that enables centralized control over network resources with respect to traffic forwarding.

2.2.1 *Host Virtualization*

Host virtualization refers to the virtualization of computer hardware such as servers. Efficient resource sharing and isolation, as well as the concurrent use of different operating systems, are the main goals of host virtualization. In this regard, we distinguish between three major virtualization techniques:

- Full virtualization (e. g., KVM [31], VMware [32]) requires a hypervisor between the guest operating systems (OSs) and the actual hardware, i. e., host OS. Thus, the guest OSs view only virtualized components exclusively assigned to them. Any privileged command can be executed from within

the guests since each guest runs its own kernel, and further no restrictions apply to the selection of the guest OS. Full virtualization provides full flexibility and a high level of isolation at the drawback of performance overhead.

- Paravirtualization (e. g., Xen [7] [4]) uses, similar to full virtualization, a hypervisor to isolate different guests from the hardware. The difference lies in the modified kernel of the guest OS that uses hypercalls to pass through privileged commands to the host OS. Paravirtualization achieves an adequate level of isolation and performance at the price of the limited availability of kernel updates for the guest OSs.

- Container-based virtualization (also referred to as OS-based virtualization, e. g., OpenVZ [8], Docker [34]) refers to the creation of multiple containers (i. e., guest environments) in the user space of the host OS. In this case, guests use the same kernel of the host OS simultaneously. Hence, container-based virtualization does not provide flexibility with respect to the guest OSs and achieves merely a low level of isolation, specifically between the user-spaces of the different guests. However, this approach achieves high performance due to small overhead.

2.2.2 *Link Virtualization*

Link virtualization commonly refers to the creation of tunnels (i. e., virtual links) that interconnect endpoints even across the substrate network borders. Tunnels are typically created by using one of the various existing encapsulation protocols, such as IP-in-IP [35] which embeds the original IP packets with header into new IP packets whose outer IP header is used for the forwarding over a IP carrier network. Solutions for encapsulation of protocols other than IP exist, e. g., EtherIP [36] or generic routing encapsulation (GRE, [37]). In addition to the above-discussed tunneling approaches, multiprotocol label switching (MPLS) [38, 39] is a method that distinguishes between traffic flow classes. IP packets of the same class are assigned a label. This label in turn is considered in the forwarding tables of the MPLS-capable routers (i. e., label-switched routers) to identify the traffic class. In this respect, substrate providers are given the means to assign a certain path to a particular traffic flow class. Furthermore, signaling protocols have been proposed that allow virtual link setup across multiple substrate networks with QoS guarantees [40].

4 Fayyad-Kazan et al. provide a performance comparison of Xen with full virtualization versus Xen with paravirtualization [33].

2.2.3 *Network Function Virtualization*

A rich variety of proprietary hardware appliances (i. e., middleboxes) is deployed across the Internet today.[5] Those purpose-specific middleboxes fulfill certain tasks (i. e., network functions, NFs) such as network address translation or load balancing. Hardware middleboxes introduce limitations with respect to customization, resource efficiency, and manageability. Network function virtualization (NFV) is an emerging concept that aims at mitigating some of these problems by enabling the consolidation of NFs on platforms built of commodity components [17, 18, 41]. This can reduce the expenses for NF deployment either by deploying consolidated software middleboxes in enterprise networks or by outsourcing NFs to virtualized network infrastructures [41–46]. In addition to that, NFaaS allows elastic resource provisioning and thereby the scaling of the network services in response to evolving demands. Furthermore, the recent trend of micro-data center deployment by large ISPs (e. g., AT&T, Deutsche Telekom, Telefonica) leads to a larger number of NFV PoPs and a wide geographic coverage [19]. Thus, the deployment of NFs even with hard geographic constraints is feasible and also the clients potentially benefit from better NFaaS offerings. Finally, NFV facilitates service chaining, i. e., the composition of multiple NFs and endpoints (Section 2.3).

2.2.4 *Software-Defined Networking*

The SDN concept represents an approach towards programmable network equipment [47]. In principle, SDN introduces the ability to make traffic forwarding decisions over abstracted interfaces (control plane) and apart from any forwarding device (data plane). In more detail, centralized controllers instruct the network devices to forward packets based on rules at the granularity of flows. These controllers build an abstraction layer that provides higher-level functionality to the network applications. The data plane can be realized by any SDN-capable forwarding device. The SDN concept has, for example, been materialized in the widespread *OpenFlow* protocol [48, 49]. OpenFlow is supported by more than one dozen of OpenFlow controllers such as NOX [50], Beacon [51], or OpenDayLight [52]. OpenFlow switches, serving as data plane, are available from many vendors but also commodity servers could be used. In the latter case, a software-based OpenFlow switch such as *Open vSwitch* needs to be used [53].

SDN is also used in the context of NFV. The works on NFV as discussed in Section 2.2.3 rely on the SDN paradigm. They use, for example, SDN controllers to steer traffic through a sequence of NFs.

5 Sherry et al. provide a survey of enterprise middlebox deployments [16].

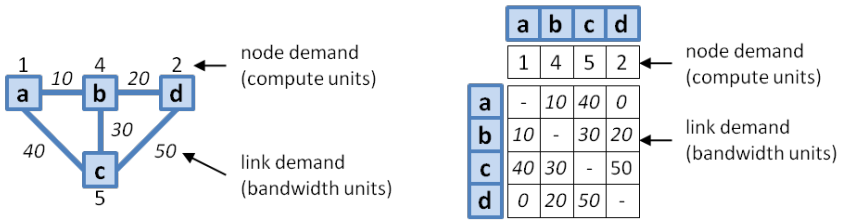
Furthermore, virtualized SDN combines the benefits of virtualization and SDN [54]. This concept introduces SDN hypervisors that abstract the physical SDN components so that each tenant views himself as the exclusive user of an SDN controller. The shared resources are then managed and isolated by the hypervisor. FlowVisor [55], AutoSlice [56], and HyperFlex [57] are examples for SDN hypervisor approaches.

2.3 SPECIFICATION OF VIRTUAL NETWORKS AND SERVICE CHAINS

The previous sections explain the enabling technologies for wide-area deployment of network services and the substrate that is used to host those services. In the following, we discuss (i) virtual network and (ii) service chain specifications.

Virtual Networks

The SPs benefit from service-tailored VNs that fulfill requirements in terms of processing capability, link quality, and geographic location. From a technical point of view, a VN consists of (i) geographically diverse virtual machines (i. e., virtual nodes) that take over certain processing tasks (e. g., packet inspection, filtering, transcoding, caching) and (ii) virtual links that are setup between the virtual nodes. A common metric for processing capability is the number of compute cycles over time (GHz). Other attributes such as hard disk storage or main memory size could also be considered. Agreements related to data transfers are often associated with a data volume that can be consumed during a certain time interval or with a peak traffic rate. Links can be further associated with other performance metrics such as latency to support additional service guarantees to the end-users. In practice, no consensus has been found so far on how to define and account for network performance, from both the requesters' side and the providers' side [58]. For this reason, we specify a basic VN request model associated with virtual node demand (GHz) and bandwidth demand (Mbps). Furthermore, VNs can be represented in different forms, either as topology or as traffic matrix, as exemplified in Figure 4. In the chapters 3 and 4, we discuss the embedding of such VNs into substrate networks.



(a) topology-based VN (b) traffic matrix-based VN

Figure 4: The bandwidth demands of the virtual links of a VN can be represented in different ways. Non existing virtual links can be expressed as zero bandwidth demand.

Service Chains

Service chaining is a common abstraction for the expression of network service requirements [46, 59]. A service chain represents the exact sequence of NFs traversed by one or multiple flows. Figure 5 illustrates an example of service chaining. In this illustration, two different groups of enterprise network users at one site (e. g., front-desk and sales) access a web server cluster and a database server residing in another site. Traffic from both groups traverses a cache, a firewall, and a redundancy elimination (RE) appliance, whereas the traffic of "Group A" is sent through a load balancer and a web application firewall. In contrast to VNs, such service chain requests do not specify processing demands and link bandwidth demands among the chain nodes. Instead, those demands are estimated, during the embedding phase, based on NF resource profiles and the traffic rate at the sources. The embedding of service chains is subject of Chapter 5.

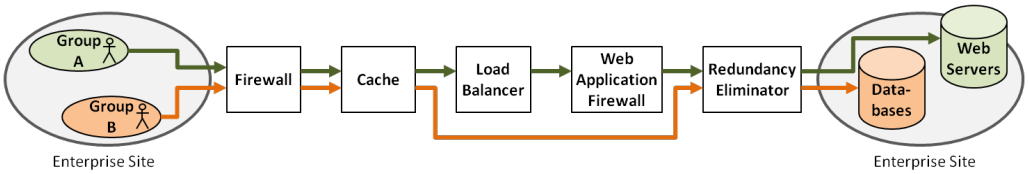
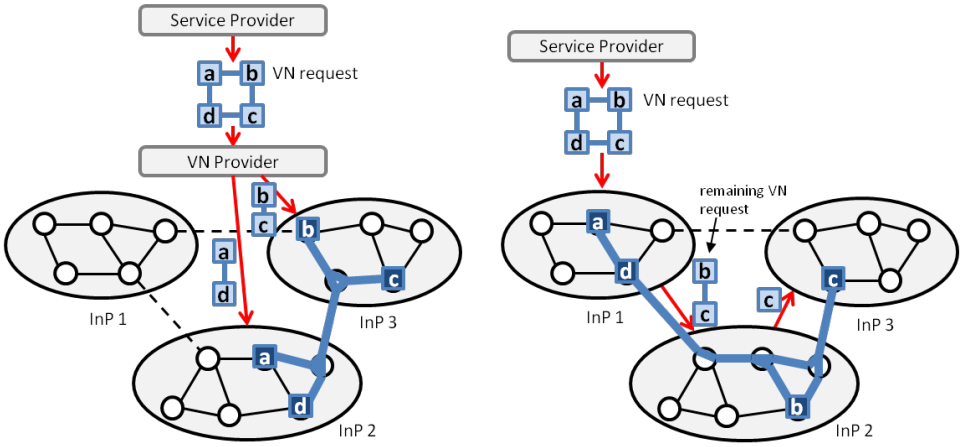


Figure 5: Service chains are composed of virtualized network functions to be traversed for a specific purpose and client. In the example, two groups of enterprise network users require different network functions applied to their traffic from the one to the other site.

2.4 MULTI-PROVIDER EMBEDDING ARCHITECTURES

The embedding of such VNs and service chains into multiple substrate networks requires coordination among the different actors (i. e., SPs, InPs). In this context, coordination comprises several embedding mechanisms (e. g., resource discovery, resource assignment, service instantiation) that are facilitated by multi-provider VN embedding architectures that we discuss in the following. All these architectures consider the SP, that formulates the VN requests and the infrastructure providers (InPs), i. e., the operators of the substrate resources. We further distinguish between centralized and distributed approaches as illustrated in Figure 6.

- Centralized architectures rely on a coordinator that partitions the original VN request into segments that are subsequently relayed to the InPs.
- Distributed architectures let an InP first embed any subset of the arrived VN request and then relay any remaining VN request to a neighboring InP.



(a) VNE with centralized coordinator (b) Distributed VNE

Figure 6: We distinguish between VNE architectures that either facilitate a layer of indirection between SP and InPs (coordinator), or that allow relaying of requests among the InPs.

Schaffrath et al. propose an architecture as shown in Figure 6a where a VN provider acts as a layer of indirection between SP and InPs [12]. The VN provider maintains a repository containing resource information from InPs, such as virtual machine attributes or transit costs. Based on this information, he then partitions the requested VNs according to the SP's objective, e. g., minimizing the VNE cost. A similar architecture called Cabernet is also proposed by Zhu et al. [20]. Such centralized architectures can also be used with dynamic pricing

at which InPs bid for profitable VN subsets. For example, *V-Mart* [60] uses a two-stage Vickrey auction model where an auctioneer undertakes the partitioning of the VN requests.[6]

In contrast to the centralized approaches, *PolyViNE* [62] relies on a distributed VNE architecture, as illustrated in Figure 6b. In this case, no layer of indirection exists. Instead, the SP sends his VN request directly to one or multiple InPs. Each InP selects a subset which can be efficiently mapped to his substrate network, and the remaining part of the VN will be again forwarded to one or multiple InPs until completion of the VN mapping. All InPs propagate back the VNE cost and the SP will eventually trigger the final VNE over those InPs with the least aggregated VNE cost.

The architectures presented in this section have been particularly proposed for the embedding of VNs. Nevertheless, the embedding of service chains is, similar to VNE, exacerbated by limited information disclosure in the multi-provider context. Hence, multi-provider VNE architectures can also be considered for multi-provider NSE after an adaptation of the request and network models. In this case, the NF provider replaces the InP business role. We embrace the centralized approach for the VN and service chain embedding in the Chapters 3 and 5.

2.5 EMBEDDING STEPS

The embedding of VNs and service chains consists of a sequence of embedding steps. According to the VNE frameworks proposed by Houidi et al. [63] and Papadimitriou et al. [64], multi-provider VNE with centralized coordinator comprises the following embedding steps:

- resource advertisement,
- resource matching,
- VN request partitioning,
- resource mapping,
- VN instantiation.

More specifically, VNE takes place as follows. Participating InPs advertise the resources, that can potentially be used for hosting new VNs, to the VN provider that in turn stores this information in a local repository. The resource advertisement phase is a continuous process. Thus, updates announced by the

6 Fundamentals of auction theory are, for example, provided by Klemperer [61].

InPs are immediately stored in the VN provider's repository. Regardless of the resource advertisement phase, SPs direct VN requests to the VN provider at any time. He in turn matches the requested resources of the VN against his local repository that contains the resources advertised by the InPs. The outcome of the matching is the identification of potential InPs for the embedding of the requested VN. Next, the VN provider splits the VN request into segments that can each be embedded by a single InP. The demanded resources of the VN segments are then assigned to the physical resources by the corresponding InPs. Finally, the VN is instantiated after all resources are assigned.

This thesis focuses on (i) the request partitioning and (ii) the resource mapping phase. According to that, the following sections provide an overview of existing methods used in related works.

2.5.1 *Request Partitioning Methods*

Request partitioning is challenging due to the limited knowledge of the substrate resources. Existing works either ignore this issue or they consider only very abstract views. In this respect, Houidi et al. [63] carry out VN request partitioning based on an AS-level substrate network view. In particular, virtual nodes are assigned to ASs while the link mapping takes place subsequently within the ASs. In this case, all the intra-domain link costs that impact VNE efficiency are not taken into account during the partitioning phase. The authors provide a heuristic algorithm based on the max-flow min-cut algorithm and an integer linear programming solution. Leivadeas et al. [65] rely on a cloud broker and propose for the request partitioning a heuristic algorithm which is based on iterated local search (ILS). ILS yields substantially lower runtime compared to integer linear programming, but may still require a large number of iterations and considerable communication overhead between the broker and the InPs before an approximate solution has been found. This work does not discuss the visibility of the broker on the substrate resources. Additional related works on VN and service chain partitioning are discussed in detail in the Sections 3.8 and 5.7.

2.5.2 *Resource Mapping Methods*

Resource mapping in single substrate networks has already been studied comprehensively [66]. The vast majority of these works focus on resource efficiency with the goal of achieving maximum revenue and request acceptance. In this respect, algorithms commonly optimize cost while some strive to achieve load balancing [67, 68]. There exist various approaches based on

- integer linear programming and mathematical modeling:
  - integer linear programming is used widely (e. g., by [63, 69–72]).
  - Quadratic programming is also used in one of the earlier works by Lu and Turner [73].
  - Cheng et al. [74] rank nodes based on its resource and topological attributes using the Markov random walk model.
- heuristic algorithms:
  - Fajjari et al. [75] deal with the hardness of the mapping problem and employ the ant colony metaheuristic while Zhang et al. [68] use the simulated annealing metaheuristic.
  - Mijumbi et al. [76] propose a multiagent learning algorithm for dynamic resource allocation.
  - Chen et al. [77] use a border matching strategy that reduces the fragmentation effect on the substrate network.
- existing solutions to well-known problems in graph theory:
  - Lischka and Karl [78] devised an algorithm based on subgraph isomorphism detection.
  - Some works consider the link mapping a separate phase and solve the corresponding multi-commodity flow problem (e. g., [79]).

Such resource mapping algorithms commonly aim at achieving optimality and low run time, which can be seen as dominating algorithm efficiency metrics. Scalability issues with integer linear programming-based solutions are addressed by a transformation to real linear programming equivalents and rounding algorithms. Chowdhury et al. [70] take such an approach for coordinated node and link mapping and apply deterministic and randomized rounding techniques. Another goal of VNE works is to introduce new features to existing solutions. For example, Fuerst et al. [80] propose a pre-clustering algorithm for optimization of link resources. Furthermore, an initial embedding can be continuously optimized by reprovisioning of the virtual resources, for example, to cover evolving substrate networks [81] or to react to link failures [82]. Yet other works question the suitability of request attributes or the assumptions made in previous work. The former can be seen in the case where requests could also have a flexible instead of a fixed starting time. This is considered by Rost et al. [83] who propose a continuous-time mathematical programming approach where resource allocations are rescheduled for improving overall system performance. The rich variety of VNE works further incur variations in the assumptions. For example, Yu et al. [79] take into account path splitting

and migrations while VN reconfiguration is considered in [84]. Furthermore, assumptions are rethought in order to make VNE appear more realistic. In this respect, Botero et al. [85] propose a model taking additionally into account the processing effort of the forwarding nodes (hidden hops) in terms of CPU expenditure. Apart from this general overview of related work on resource mapping, we discuss them in the context of VN segment and VN subset mapping in the Sections 3.8 and 4.5; and in the context of NF subgraph mapping in Section 5.7.

2.6 LINEAR PROGRAMMING

The previous section discusses the required embedding steps, particularly the request partitioning and the resource mapping step. These steps can be viewed as combinatorial optimization problems which can be tackled by linear programming methods. This section provides a general introduction to these methods.

An optimization problem can formally be expressed as follows.

$$\text{Minimize (Maximize)} \qquad f_0(x) \tag{1}$$

$$\text{subject to:} \qquad f_i(x) \leqslant b_i \qquad i = 1 \dots m \tag{2}$$

x represents the set of optimization variables – the solution of the optimization problem. The objective statement (Equation 1) contains $f_0(x)$, the objective function, which is to be either minimized or maximized. The solution space is restricted by m constraint functions (Equation 2) that need to be less or equal than the constants in b representing bounds for those functions. If $x \in \mathbb{R}$ and $f_0(x), f_i(x)$ are linear functions then these linear problems can be solved by linear programming (LP) algorithms based on the simplex, the interior point, or the ellipsoid method. Highly optimized solvers rely, depending on the problem, on such algorithms and solve LP models efficiently within polynomial time.

However, optimization problems in computer networking are often combinatorial problems at which, for example, nodes in a substrate topology are to be selected jointly with links forming a traffic path. In this case, the decision variables are of type binary, i. e., $x \in \{0, 1\}$, and cannot be solved directly with the above-mentioned LP algorithms. Instead, those variables are treated as integers, i. e., $x \in \mathbb{Z}$, thus integer linear programming (ILP) algorithms, for example, based on the branch and cut method, need to be applied here [7]. Mixed integer linear programming (MILP)[8] is a less restrictive ILP subclass considering models

7 ILP covers binary variables as they represent an integer subset. In this case, the terms "binary linear programming" or "0-1 linear programming" can be used to put emphasize on the hardness of the optimization problem.

8 The term MIP is often used as a synonym for MILP and MIQP.

with both integer and real variables. Analogously, MIQP is an MILP variant with additional quadratic terms in the objective function caused by combination, i. e., multiplication, of two linear variables.

MILP, ILP, and MIQP are known to be computational intensive – Such models do not scale well and, in many cases, exact solutions cannot be obtained in polynomial time. A common way to prevent from exorbitant solver run time is to instruct the solver to terminate after a near-optimal solution was found. The level of suboptimality, quantified in the MIP gap, is expressed in the objective function's value difference between the best solution found and the optimal solution. This approach reduces solver run time dramatically, even at marginal suboptimality, i. e., in the magnitude of 0.1% MIP gap. However, this does not solve the scalability problem. Hence, approximation algorithms are often employed that use ILP models which have been transformed to LP models by relaxation of the integer and binary variables.

Existing constraints in a relaxed LP model may be without effect after relaxation which in turn yields suboptimal or infeasible results. Suboptimality is an expected effect caused by complexity reduction of the original ILP/MILP formulation together with the required rounding of the relaxed LP variables. Infeasible solutions due to rounding can finally be prevented by re-running the LP solver with fixing selected variables according to the rounding results. This iterative reduction of the solution space will finally exclude all non-binary solutions which wrongly appear feasible in the LP. LP relaxation and rounding typically achieves near optimal results but much faster processing time. Such approaches are taken in the Sections 3.5.2, 3.6.2, 5.4.2, and 5.5.2.

Optimization problems can alternatively be solved with the aid of expert knowledge. Heuristics are such an example at which an algorithm is based on a practical methodology as applied by human experts, e. g., network operators. On the one hand, heuristics are often optimized for a very specific environment, and extreme cases of suboptimality cannot be precluded. On the other hand, heuristic-based algorithms can typically be implemented with limited time complexity and can in most cases be further optimized for run time.

Furthermore, greedy algorithms represent the lower bound of heuristic algorithms in terms of intelligence. They are extremely simple, therefore even faster, and for some problems sufficient or useful as a baseline.

Fundamentals in combinatorial optimization are provided, for example, by Papadimitriou and Steiglitz [86].

3

MULTI-PROVIDER VIRTUAL NETWORK EMBEDDING

This chapter aims at addressing the issue of *limited information disclosure (LID)* of the InPs' substrate topologies and resources [87, 88]. Specifically, we investigate the level of resource and network topology information that can be divulged by InPs, taking into account the confidentiality of this information (Section 3.1). Furthermore, we question the suitability of network topologies for VN request specifications (Section 3.2). We further present a framework for multi-provider VN embedding (VNE) in which a centralized coordinator relies on abstracted information which is not considered confidential by the InPs. In particular, this centralized coordinator represents a VN provider that partitions a requested VN into multiple segments that are subsequently mapped to the infrastructure by the corresponding InP (Sections 3.3–3.6). We evaluate VNE with limited information disclosure against a "best-case" scenario with *full information disclosure (FID)* (Section 3.7).

3.1 INFORMATION DISCLOSURE OF THE INFRASTRUCTURE PROVIDERS

We depart from the example in Figure 7, where the embedding of a VN consisting of virtual nodes and links associated with certain requirements (e. g., CPU and bandwidth) is requested at a VN provider. VN requests are formulated based on the requirements of a particular service and are typically defined at a high level of abstraction. VNE consists in mapping the VN request onto multiple InPs, such that virtual node and link requirements are satisfied. Furthermore, the assigned InPs should offer the geographic footprint required by the SP. We rely on the centralized control plane architecture of Schaffrath et al., which interposes a connectivity layer (i. e., VN provider) between SPs and InPs, responsible for the partitioning of VN requests into segments assigned to InPs [12][1].

The VN provider's visibility on the substrate networks is critical for the efficiency of VN partitioning. In this respect, a well-defined and realistic level of information disclosure comprises a prerequisite for any VN partitioning problem formulation. To this end, we consider the information disclosure policies of InPs and cloud data center network operators with respect to (i) (virtual) resource availability and (ii) substrate network topology.

1 VNE architectures are discussed in Section 2.4

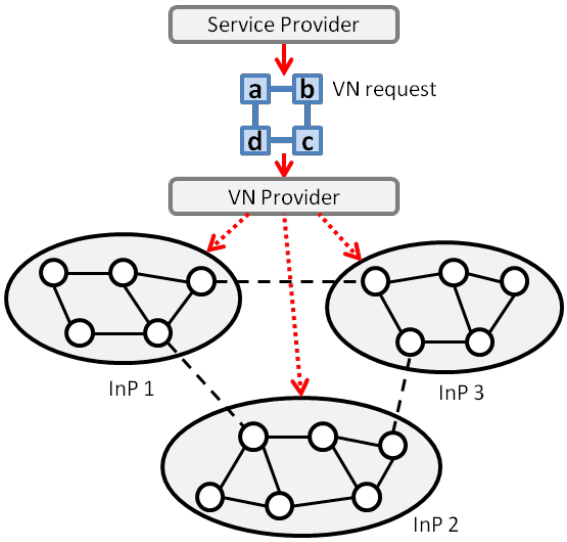


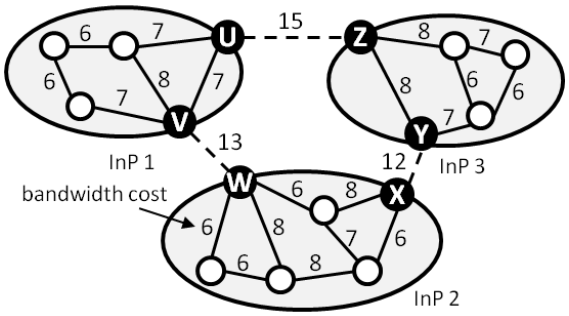
Figure 7: The VN provider represents a layer of indirections between the requester (SP) and the InPs. It partitions the incoming VN requests across the potential InPs.

Virtual Resources. Amazon EC2, a prominent example of a cloud datacenter, classifies its resources (i. e., virtual machines) into certain types, each one having a common set of attributes (e. g., operating system, main memory, storage, I/O performance) [89]. Each resource type is advertised along with the associated cost. Amazon EC2 does not disclose the number of available instances for each offered resource type, concealing the resource utilization.

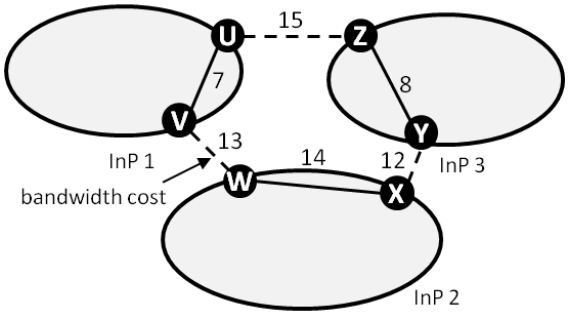
Network Topology. For InPs, revealing detailed topology information, such as router-level topologies is deemed prohibitive. Instead, some InPs publish topologies with PoPs. However, most of these topologies are oversimplified lacking not only router-level connectivity but also PoP structure [90–93]. In contrast to InP topologies, there is publicly available information on InP peerings [94]. Specifically, certain Internet exchange points (e. g., DE-CIX [95], AMS-IX [96])[2] advertise information about peerings and traffic statistics. Additional information on peering locations and participants is collected and published by peering databases (e. g., PeeringDB [98]).

Based on these observations, detailed topology information cannot be assumed to be accessible by VN providers. However, VN providers can enrich their limited substrate network view with certain aspects of the substrate network topologies which are not treated as confidential, such as InP peerings, including the locations of peering nodes. We further consider InPs are advertising the costs of the links between the disclosed peering nodes (i. e., cost per bandwidth unit). The peering link costs will comprise the transit fees to the provider (in the case of paid peering) or the cost for the operation and maintenance of

2 Ager et al. studied the anatomy of a large European IXP [97].



(a) Substrate network topologies



(b) VN provider's view on the substrate network topologies

Figure 8: The VN provider has only a limited view on the substrate network topologies. Router-level topologies are usually obscured by the InPs.

the peering link (in the case of settlement-free peering) plus the profit for the InP. Since all these costs will be accumulated in the advertised peering link costs, any potential confidentiality of peering agreements will not constitute a limiting factor for VN request partitioning.

Along these lines, Figure 8b illustrates the view of the VN provider on the substrate network topologies of Figure 8a, with $U, V, W, ..., Z$ representing the peering nodes. Furthermore, Figure 9 combines this topology view with the set of offered virtual node types, represented by $\{a, b, c, ..., g\}$, and their associated costs. This essentially constitutes the information used by the VN provider for VN request partitioning.

We handle VN partitioning as a cost minimization problem, i. e., the VN provider will seek to minimize the VN embedding cost, which accumulates all virtual node and link costs. Since the VN provider lacks detailed knowledge of the substrate topologies, he will not be in position to account for all intra-provider link costs. However, the disclosure of link costs between peering nodes within each InP allows the VN provider to have a rough estimation of the total virtual link cost (Figure 8b). We quantify the impact of such limited information disclosure on VNE efficiency in Section 3.7.

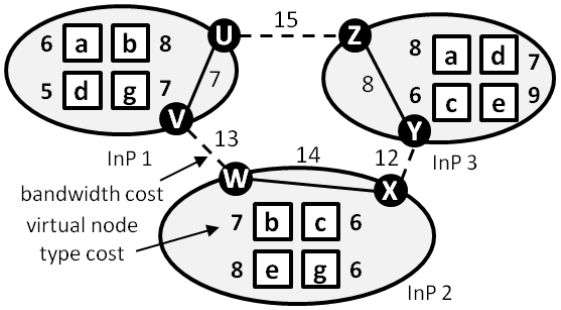


Figure 9: VN request partitioning uses abstract resource and network topology information. InPs will advertise costs for available virtual node types and for transit between peering PoPs.

3.2 VN REQUEST SPECIFICATION

Nearly all existing VNE algorithms (e. g., [63, 67, 70, 78, 79, 84])[3] process topology-based VN requests, which are commonly specified as undirected weighted graphs. To provide more flexibility in VNE, we consider VN requests in which the bandwidth demands are expressed with a traffic matrix.

Such a VN request specification entails significant benefits to all actors. First, a traffic matrix simplifies the specification of bandwidth demands between a set of virtual nodes. Using traffic matrices, an SP can benefit from a higher level of abstraction, which obviates the need for any VN topology specifications. On the other hand, traffic matrix-based VN requests can also be beneficial to InPs since they yield high flexibility for VN segment mapping.

For example, Figure 10 illustrates a traffic matrix that specifies the bandwidth demands between three virtual nodes. This traffic matrix essentially abstracts all four topology-based requests shown in this figure, while it further represents the bandwidth demands of additional VN topologies that include intermediate nodes. As such, a traffic matrix gives directly the bandwidth demands, irrespective of the virtual node mapping. In contrast, a VN graph may require transformations for the estimation of bandwidth demands, so that the feasibility of a certain mapping can be evaluated. Otherwise, the VN graph may unnecessarily restrict the VNE problem space excluding efficient solutions [10].

Another aspect of VN topologies is redundancy at which operators consider backup paths in their requested topologies. Traffic rerouting across the physical network domains is hard to predict over time and the case that primary and backup path partially share the same physical medium cannot be precluded, e. g., due to cascaded embedding into non-physical resources. Instead, we argue that infrastructure providers are finally in charge to fulfill the respective

3 Section 2.5.2 discusses such algorithms accordingly.

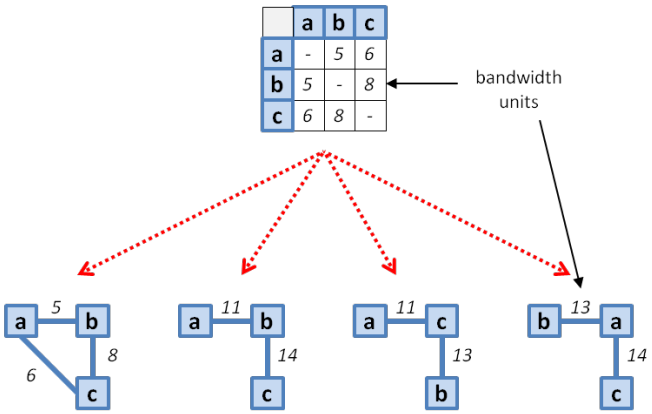


Figure 10: A VN as traffic matrix abstracts multiple topology-based VNs.

service level agreements. In this context, we investigate the impact of topology-based VNE by repeating the mapping of the same requested VN topology but with a stepwise reduction of the maximum number of virtual links sharing a single physical link as shown in Table 1. Each step enforces more and more the exact mapping as VN topology while no restriction means the VN can be handled as traffic matrix. Our tests include 250 VN topologies with each 5 to 15 virtual nodes to be mapped onto a substrate network with 125 nodes.[4] Table 1 shows further acceptance rate and generated revenue relative to the embedding with traffic matrices and further the penalty cost incurred by additional links. These results show that a threshold of 10 and less overlapping virtual links significantly impact the efficiency of VN embedding. At the first view, penalty in terms of VN embedding cost for topologies seems low. Together with the observation that revenue decreases faster than the acceptance rate, we conclude that larger VNs are affected exceedingly.

Based on these observations, our multi-provider VNE embedding framework is tailored to traffic matrix-based requests. We use traffic matrices for the expression of bandwidth demands in the initial VN request formulated by the SP as well as in the VN segments generated upon VN partitioning.

3.3 VN EMBEDDING FRAMEWORK

Hereby, we present our multi-provider VNE framework. Similar to [63], we decompose VNE into a set of operations that allow the VN provider and InPs to process VN requests, depending on their level of access and visibility on the substrate networks. In the following, we discuss these VNE steps.

4 Evaluation environment and parameters similar to Section 3.7.

Table 1: The restriction of the shared use of substrate links emulates for the resource mapping similarity with a topology-based representation of the same VN. The results indicate that an increasing similarity with the VN graph comes along with a higher impact on mapping efficiency.

Request Type	Max. # virtual links per substrate link	VN request acceptance rate (%)	Generated revenue (%)	Penalty cost (%)
VN traffic matrix	∞	100.0	100.00	0.00
VN topology	20	100.0	100.00	0.89
	10	68.8	52.02	3.54
	8	54.8	36.77	4.56
	6	41.2	24.40	5.48
	5	34.0	18.51	5.53
	4	27.6	14.28	7.98

Resource Information Disclosure. The disclosure of network topology and resource availability information can facilitate resource discovery and VN request partitioning. As discussed in Section 3.1, we consider each InP advertising (i) his geographic footprint, i. e., the list of peering nodes, (ii) the offered virtual node types along with the associated cost, and (iii) the bandwidth cost of the links between the peering nodes. Virtual node types comprise the resource attributes, for example, its availability. All disclosed information is collected and registered by the VN provider into a local repository, which is updated after the arrival of an InP advertisement and subsequently used for resource matching and VN partitioning.

Resource Matching. The VN provider relies on the information disclosed by InPs to match requested to offered resources. To this end, the VN provider identifies a set of candidate resources that fulfill the requirements of each requested virtual node (e. g., with respect to location, main memory, network configuration). Figure 11 depicts how four different virtual node types (i. e., a, b, c, d) of a VN request are matched against the resources advertised by the InPs. In this particular example, the requested and advertised resources have specifications of the same level (i. e., the same set of attributes) which simplifies their matching. It is possible that the attributes of the requested resources comprise a subset of the attributes of the advertised resources. In this case, the identification of matches across the set of disclosed resources can be computed by employing virtual resource clustering techniques (e. g., [99, 100]).

VN Request Partitioning. VN requests are partitioned across multiple InPs when there is not a single InP that satisfies all the resource requirements in the request. For example, VN partitioning is required for any requested VN exceeding the

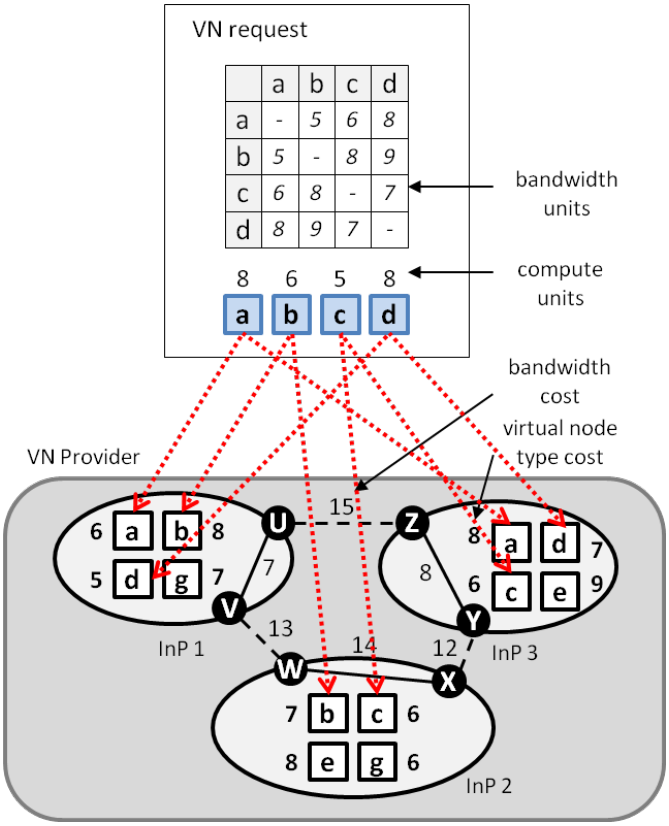


Figure 11: The VN provider matches requested node types to the potential InPs. For example, node a is only available at InP 1 and InP 3 at a different cost. InP 2 does not offer this particular node a.

geographic footprint of each InP. Any requested virtual node may be available from more than one InP and possibly at different costs, as shown in Figure 11. Virtual node assignment affects the virtual link costs, resulting in trade-offs between the virtual node and link costs. We thereby consider VN partitioning as a cost minimization problem, taking into account all disclosed resource costs as well as the inter-dependencies between virtual node and link costs. In Section 3.5, we present an ILP/LP-based solution to the VN request partitioning problem. The partitioning results in a set of VN segments, where each virtual node is mapped onto one of the disclosed peering nodes. Similar to the initial VN request, each VN segment consists of virtual node specifications, and a traffic matrix that represents the bandwidth demands between all pairs of virtual and peering nodes (Figure 12).

VN Segment Mapping. Upon VN request partitioning, InPs map their assigned VN segments onto their substrate networks while satisfying certain virtual node and bandwidth requirements. The segment mapping complies with the virtual node to peering node bindings designated in the VN segment specification. This step corresponds to VNE with a single InP, a problem that has been investigated

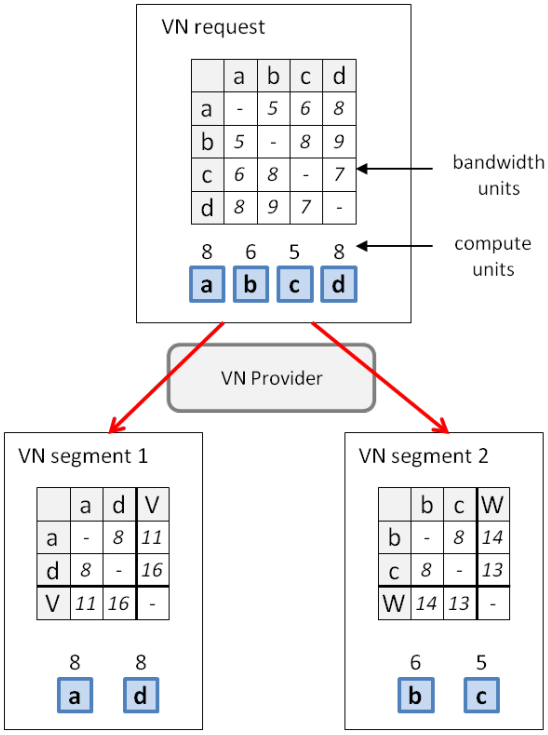


Figure 12: The VN provider partitions the requested VNs across the potential InPs. The virtual nodes a, d are mapped onto the peering node V (InP 1), and the virtual nodes b, c are mapped onto the peering node W (InP 2). Bandwidth demands for each VN segment are expressed with additional traffic matrices.

in [63, 67, 70, 78, 79, 84]. However, these methods are tailored to topology-based VN requests, instead of traffic matrix-based VN requests used by our VNE framework. As such, in Section 3.6 we present an MILP/LP-based solution to the mapping problem with traffic matrix-based VN requests.

3.4 NETWORK MODEL

In this section, we introduce the substrate and virtual network models used in the VN request partitioning and VN segment mapping problem formulations (Sections 3.5 and 3.6).

Substrate Network Model. The substrate network is represented as a weighted directed graph $G_S = (V_S, E_S)$, where V_S is the set of substrate nodes and E_S is the set of substrate links between the nodes within the set V_S. Each substrate node $u \in V_S$ is associated with a set of attributes, such as the location and the residual capacity denoted by r_u. Each substrate link $(u, v) \in E_S$ between two substrate nodes u and v is associated with the residual capacity denoted by r_{uv}.

The set of peering nodes disclosed by all participating InPs is further denoted by P.

VN Request Model. A VN request consists of the set of virtual nodes V_V and the bandwidth demands d^{ij} between any pair of virtual nodes $i, j \in V_V$. Each virtual node i is associated with a set of attributes (e. g., location) and a capacity demand denoted by d^i. For a VN request partitioned into k segments, V_V^k denotes the set of nodes that comprise the VN segment specification, including the virtual nodes plus the assigned peering nodes (e. g., VN request segment 1 and 2 in Figure 12). For any pair of nodes, $i, j \in V_V^k$, d^{ij} represents the respective bandwidth demands for the k^{th} VN segment.

Table 2: Notations used in the Sections 3.4–3.6.

Symbol	Description
c_p^i	cost of virtual node i when assigned to peering node p
$c_{pp'}^{ij}$	accumulated link costs between virtual nodes i, j when assigned to the peering nodes p, p', respectively
d^i	computing demand for virtual node i
d^{ij}	bandwidth demand from virtual node i to virtual node j
f_{uv}^{ij}	variable expressing the amount of bandwidth from virtual node i to j assigned to substrate link (u, v)
E_S	set of substrate links
V_S	set of substrate nodes
V_V	set of virtual nodes
V_V^k	subset of virtual nodes belonging to VN segment k
P	set of peering nodes
r_u	residual capacity of substrate node u
r_{uv}	residual capacity of substrate link (u, v)
w_p^i	constant set to one if the mapping of virtual node i to peering node p is feasible; otherwise set to ∞
w_u^i	constant set to one if the mapping of virtual node i to substrate node u is feasible; otherwise set to ∞
X^i	set of x_p^i variables belonging to a particular virtual node i
x_p^i	variable $\in [0, 1]$ expressing whether the virtual node i is assigned to peering node p
x_u^i	variable $\in [0, 1]$ expressing whether the virtual node i is mapped to substrate node u
$y_{pp'}^{ij}$	variable $\in [0, 1]$ expressing whether the virtual nodes i and j are assigned to the peering nodes p and p', respectively
α	computing demand weight in VN embedding cost
β	bandwidth demand weight in VN embedding cost

3.5 VN PARTITIONING

In this section, we present formulations for the VN request partitioning problem. We first describe the objective of VN partitioning and provide an MIQP/ILP model (Section 3.5.1), which we subsequently transform into an LP model (Sections 3.5.2 and 3.5.3) in order to reduce time complexity. The notations used in this Section are listed in Table 2.

3.5.1 *MIQP/ILP Model*

For VN request partitioning, we seek the assignment of the requested virtual nodes to the disclosed peering nodes, i. e., $V_V \rightarrow P$, such that the cost for the SP is minimized. To this end, we define a formulation that takes as inputs the VN request, the node and link costs advertised by InPs, and the virtual node to peering node assignment feasibility obtained from the resource matching phase. In this respect, let c_p^i denote the cost associated with the assignment of virtual node i to peering node p. This cost is basically obtained by multiplying the request demand (i. e., compute units) with the unitary virtual node cost from the InP. We further introduce a weight $w_p^i \in \{1, \infty\}$ to specify the feasibility of assigning virtual node i to p. Hence, $w_p^i \leftarrow \infty$ if the mapping is not feasible (i. e., none of the advertised node types satisfies the requested virtual node specification). We use a binary variable x_p^i to indicate the assignment of the virtual node i to peering node p. Similarly, $c_{pp'}^{ij}$ denotes the link cost for the assignment of the pair of virtual nodes i, j to the pair of peering nodes u, v. The variable $c_{pp'}^{ij}$ comprises the multiplication of the bandwidth demand and the unitary link costs along the path. Such link costs are specified separately per traffic direction. This is especially relevant for peering links. In the presence of multiple paths, the path with the lowest cost is selected. We particularly inhibit path splitting, since the VN provider does not have any information about the residual capacity of substrate network paths. A VN is partitioned correctly if all virtual nodes are assigned to a peering node, logically expressed by $\bigvee_{p \in P} x_i^p = 1 \ \forall i \in V$. Altogether, the VN partitioning problem can be initially expressed as an MIQP formulation:

$$\text{Minimize} \quad \sum_{i \in V} \sum_{p \in P} w_p^i c_p^i x_p^i + \sum_{\substack{i,j \in V \\ (i \neq j)}} \sum_{p,p' \in P} w_p^i w_{p'}^j c_{pp'}^{ij} x_p^i x_{p'}^j \tag{3}$$

Subject to

$$\sum_{p \in P} x_p^i = 1 \qquad \forall i \in V \tag{4}$$

$$x_p^i, x_{p'}^j \in \{0,1\} \qquad \forall i,j \in V, \forall p,p' \in P \tag{5}$$

The objective function (3) represents the sum of the costs for the assignment of virtual to peering nodes plus the accumulated link costs between feasibly assigned nodes. Constraint (4) ensures that each virtual node i is assigned to exactly one of the peering nodes and finally, condition (5) expresses the binary domain constraints for the variables x_p^i and $x_{p'}^j$.

Next, we eliminate in the objective function the quadratic term $x_p^i x_{p'}^j$ by substitution to a new binary variable $y_{pp'}^{ij} = x_p^i \wedge x_{p'}^j$. This variable indicates whether a pair of virtual nodes i,j is assigned to a pair of peering nodes p,p'. The binding between x_u^i and $y_{pp'}^{ij}$ is realized by an additional constraint. For any used combination of virtual node i to peering node p a constant number of counterparts, depending on the VN size must exist:
$\sum_{p' \in P} \sum_{j \in V_V} y_{pp'}^{ij} + y_{p'p}^{ji} = 2(|V_V| - 1) \cdot x_p^i$. Multiplication by x_p^i yields zero if there is no virtual node mapping. In this case, no link mapping will take place as well. Finally, we obtain the following ILP formulation:

$$\text{Minimize} \quad \sum_{i \in V_V} \sum_{p \in P} w_p^i c_p^i x_p^i + \sum_{\substack{i,j \in V_V \\ (i \neq j)}} \sum_{p,p' \in P} w_p^i w_{p'}^j c_{pp'}^{ij} y_{pp'}^{ij} \tag{6}$$

subject to:

$$\sum_{p \in P} x_p^i = 1 \quad \forall i \in V_V \tag{7}$$

$$\sum_{p' \in P} \sum_{j \in V_V} y_{pp'}^{ij} + y_{p'p}^{ji} = 2(|V_V| - 1) \cdot x_p^i \qquad i \neq j, \forall i \in V_V, \forall p \in P \tag{8}$$

$$x_p^i, y_{pp'}^{ij} \in \{0,1\} \quad \forall i,j \in V_V, \forall p,p' \in P \tag{9}$$

The objective function (6) is the linear equivalent to Equation (3). Constraint (7), which ensures that each virtual node i is assigned to exactly one of the peering nodes, is identical with (4). Constraint (8) preserves the integrity of the

VN request, i. e., for each assigned virtual node $i \in V_V$ there is connectivity with all $(|V_V| - 1)$ virtual nodes in the request. This does not necessarily imply a full mesh; instead, if required by the VN specification, bandwidth demands between pairs of virtual nodes may be set to zero. Finally, condition (9) expresses the binary domain constraints for the variables x_p^i and $y_{pp'}^{ij}$.

3.5.2 *LP Relaxation*

We reduce the time complexity of the VN request partitioning ILP by relaxing the binary domain constraints and subsequent rounding of the variables x_p^i and $y_{pp'}^{ij}$ to binaries $\{0, 1\}$. To this end, we first relax the domain constraint (9), as follows:

$$x_p^i, y_{pp'}^{ij} \geqslant 0 \quad \forall i, j \in V_V, \forall p, p' \in P \tag{10}$$

By relaxing the variable $y_{pp'}^{ij}$,, the constraint (8) no longer ensures the assignment of all virtual links, especially when peering link costs are different for each direction. We, therefore, add the constraint (14) to ensure the assignment of each virtual link to exactly one pair of peering nodes. In this respect, we summarize the following complete LP formulation at which Equations 11 – 13 are identical to their ILP equivalents.

$$\text{Minimize} \sum_{i \in V_V} \sum_{p \in P} w_p^i c_p^i x_p^i + \sum_{\substack{i,j \in V_V \\ (i \neq j)}} \sum_{p,p' \in P} w_p^i w_{p'}^j c_{pp'}^{ij} y_{pp'}^{ij} \tag{11}$$

subject to:

$$\sum_{p \in P} x_p^i = 1 \quad \forall i \in V_V \tag{12}$$

$$\sum_{p' \in P} \sum_{j \in V_V} y_{pp'}^{ij} + y_{p'p}^{ji} = 2(|V_V| - 1) \cdot x_p^i \qquad i \neq j, \forall i \in V_V, \forall p \in P \tag{13}$$

$$\sum_{p,p' \in P} y_{pp'}^{ij} = 1 \quad \forall i, j \in V_V \tag{14}$$

$$x_p^i, y_{pp'}^{ij} \geqslant 0 \quad \forall i, j \in V_V, \forall p, p' \in P \tag{15}$$

The LP solver returns values for x_p^i and $y_{pp'}^{ij}$ within the interval [0, 1]. These values can be seen as probabilities for the different assignments. Solving the VN request partitioning problem requires fixing, respectively rounding, the val-

ues of x_p^i and $y_{pp'}^{ij}$. In the following section, we present an algorithm for the rounding of these values.

3.5.3 *LP Rounding*

A naive rounding approach consists in rounding x_p^i for all $i \in V_V$ to a binary in one step. This approach, however, misses a substantial optimization potential as dependencies of *actual* virtual node assignments are ignored. Instead, in the following we describe a more efficient rounding technique for iteratively fixing the node assignments x_p^i and repeating the assignment optimization (Algorithm 1).

Algorithm 1 VN partitioning with LP

Require: $P, V_V, w_p^i, c_p^i, c_{pp'}^{ij}$
1: **repeat**
2: $\{x_p^i, y_{pp'}^{ij}\} \leftarrow$ **Solve_LP(..)**
3: $V_V' \leftarrow \{i \in V_V \mid max_{p \in P}(x_p^i) \neq 1\}$
4: **for all** $i \in V_V'$ **do**
5: $X^i \leftarrow \{x_1^i, \cdots, x_{|P|}^i\}$
6: **Sort_Descending(**X^i**)**
7: **end for**
8: $i_{fx} \leftarrow argmax_{i \in V_V'} \frac{X^i(1)}{X^i(2)}$
9: $p_{fx} \leftarrow argmax_{p \in P}(x_p^{i_{fx}})$
10: **Add_LP_Constraint**("$x_{p_{fx}}^{i_{fx}} = 1$")
11: **until** $V_V' = \emptyset$
12: **return** $\{x_p^i, y_{pp'}^{ij}\}$

First, for each virtual node i with more than one feasible assignment, we insert the solutions $\{x_1^i, \ldots, x_{|P|}^i\}$ in the vector X^i in descending order. Next, we determine the significance of each solution using the metric $X^i(1)/X^i(2)$ for $i \in V_V'$, i. e., the ratio of the largest to the second largest value. This measure leads to a solution $X^i(1)$ that dominates the remaining solutions $X^i(2), \ldots, X^i(|P|)$. In every iteration, we fix the most significant node assignment x_p^i until all nodes have been assigned. This algorithm requires at most $|V_V|$ iterations for its completion. The LP formulation in Section 3.5.2 fixes the values of all $y_{pp'}^{ij}$ variables after the termination of Algorithm 1.

We now compare the efficiency of the proposed rounding technique against the naive approach. Figure 13 illustrates the VN partitioning cost (i. e., the cost expressed in the objective function (11)) of naive rounding relative to the proposed solution for a VN size of 10 nodes. After each iteration, the VN partition-

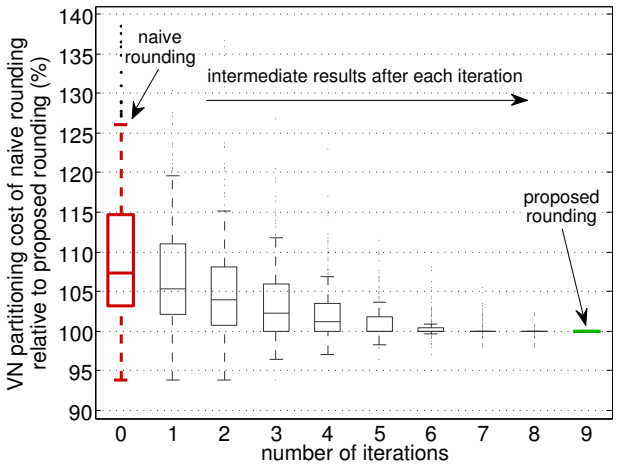


Figure 13: Naive rounding yields a higher VN partitioning cost compared to the proposed rounding with further iterations.

ing cost for the naive rounding variant is derived from the current LP solution and compared to the final solution of the proposed rounding algorithm. The leftmost group represents only naive rounding, while from the left to the right, Figure 13 depicts the relative partitioning cost after each iteration. The rightmost group represents the final solution, as nine fixed nodes implicitly yield the result for the last node. Naive rounding achieves mean VN partitioning costs higher than the proposed rounding. Rounding with Algorithm 1 iteratively converges to a VN partitioning solution that yields low cost.

We further perform a comparison between the ILP and LP in terms of VN partitioning cost. The LP incurs an extra cost of 0.2% to 7.3% for the interquartile range between the 25^{th} and 75^{th} percentiles while the median extra cost is 3.0%. Therefore, the LP incurs a minimal extra cost, while achieving run time which is four magnitudes lower compared to the ILP for VN size of 10. In absolutes values, the LP solver run time usually does not exceed 40ms.

3.6 VN SEGMENT MAPPING

In this section, we turn to the VN segment mapping problem that arises upon VN request partitioning. We provide a problem formulation that captures the mapping of the assigned VN segments to InP substrate networks subject to requirements on virtual nodes and link bandwidths. First, we provide an MILP formulation (Section 3.6.1) that is followed by a relaxed LP variant (Section 3.6.2). Finally, we make the case for the relaxed variant considering its efficiency and showing negligible difference of the empirical results of MILP and LP. We use

the same notation from the previous Section 3.5 for substrate networks and VN requests. Table 2 contains a complete list of the notations used in this section.

3.6.1 *MILP Model*

The VN segment mapping problem comprises of (i) the mapping of virtual nodes to phyisical nodes and (ii) the mapping of virtual links, i. e., traffic flows, to a set of physical links forming each a path between pairs of virtual nodes. The former can be described similar to the virtual node mapping as in the VN partitioning formulation (Section 3.5.1) while the latter is commonly formulated as multi-commodity flow problem (MCF, [22, 101]). In this context, each traffic flow requirement corresponds to a commodity $\{i, j, d^{ij}\}$, where $i, j \in V_V^k$ represent the source and destination nodes, respectively, while d^{ij} is the flow demand from source to destination. The flow variable f_{uv}^{ij} denotes the total amount of flow units on the substrate link (u, v) for the bandwidth demand between the virtual nodes i and j, with $u, v \in V_S$.

To indicate whether the virtual node i is mapped to the substrate node u we use the binary variable x_u^i, i. e., a value of one denotes the assignment. Since not all substrate nodes may fulfill the requirements of a virtual node, we use the weight $w_u^i \in \{1, \infty\}$ to denote the feasibility of mapping the virtual node i to the substrate node u, i. e., we set $w_u^i = \infty$ to denote that the mapping is not feasible. Furthermore, we define α and β to adjust the balance between CPU and bandwidth for each VN segment.

The objective of our VN segment mapping formulation (16) is the assignment of all virtual nodes within V_V^k and the computation of flows f_{uv}^{ij} with $i, j \in V_V^k$ and $u, v \in V_S$, such that the VN embedding cost (i. e., the normalized sum of CPU and bandwidth resources allocated to the VN segment) is minimized. The MILP problem formulation is as follows:

$$\text{Minimize } \alpha \sum_{u \in V_S} \sum_{i \in V_V^k} w_u^i d^i x_u^i + \beta \sum_{\substack{(u,v) \in E_S \\ (u \neq v)}} \sum_{\substack{i,j \in V_V^k \\ (i \neq j)}} w_u^i w_v^j f_{uv}^{ij} \tag{16}$$

subject to:

$$\sum_{u \in V_S} x_u^i = 1 \quad \forall i \in V_V^k \tag{17}$$

$$\sum_{v \in V_S} f_{uv}^{ij} - \sum_{v \in V_S} f_{vu}^{ij} = d^{ij}(x_u^i - x_u^j) \quad i \neq j, \forall i, j \in V_V^k, u \neq v, \forall u \in V_S \tag{18}$$

$$\sum_{i \in V_V^k} d^i x_u^i \leqslant r_u \quad \forall u \in V_S \tag{19}$$

$$\sum_{i,j \in V_V^k} f_{uv}^{ij} \leqslant r_{uv} \quad \forall u, v \in V_S \tag{20}$$

$$x_u^i \in \{0, 1\} \quad \forall u \in V_S, \forall i \in V_V^k \tag{21}$$

$$f_{uv}^{ij} \geqslant 0 \quad \forall u, v \in V_S, \forall i, j \in V_V^k \tag{22}$$

Next we discuss the constraints (17)–(22). First, constraint (17) ensures that each node $i \in V_V^k$ is mapped to exactly one of the substrate nodes. Constraint (18) implies flow conservation, i. e., the summation of flows entering or leaving a substrate node, which is not a source or a sink, must be zero. Given bandwidth demand specifications for all pairs of virtual nodes in both traffic directions, the summation of flows must be zero in any substrate node that does not host a virtual node. Constraint (19) implies a capacity cap for each substrate node u that we denote as the residual capacity limit. Similarly, constraint (20) implies a capacity limit on substrate links. Condition (21) expresses the binary domain constraints for the variable x_u^i, i. e., the mapping of virtual node i to substrate node u. Finally, constraint (22) ensures causality of the flows f_{uv}^{ij}.

3.6.2 *LP Relaxation and Rounding*

We relax the MILP formulation (17)–(22) to an LP formulation by extending the solution space of x_u^i in (21) as:

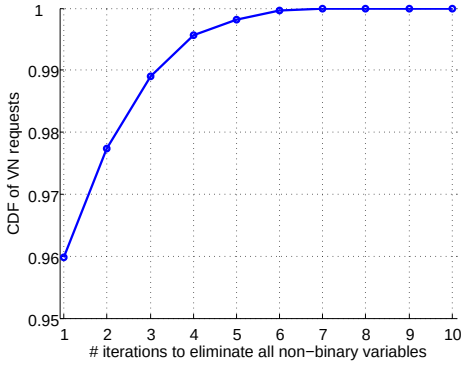
$$0 \leqslant x_u^i \leqslant 1 \quad \forall u \in V_S, \forall i \in V_V^k \tag{23}$$

Those LP solutions may contain non-binary values for x_u^i, i. e., virtual nodes mapped to multiple substrate nodes, what we consider to be not supported by the InPs. Preliminary tests with non-expiring VN requests indicated that four to five percent of the requests yield such non-binary solutions. In the following, we seek an appropriate rounding technique of the node mappings x_u^i. To this end, we propose the rounding technique in Algorithm 2, where we round to the largest fraction value. An application of the rounding Algorithm 1 for VN partitioning to this problem is possible but generally requires a longer run time. The original VN mapping formulation already contains non-integer variable, and thus, relaxation is only required for the x_u^i. In other words, relaxation has less impact on accuracy compared to the VN partitioning case and the proposed algorithm is already sufficiently accurate.

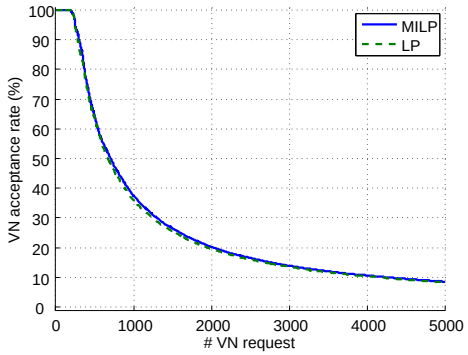
Algorithm 2 VN mapping with LP

Require: $V_S, V_V^k, \alpha, \beta, w_u^i, d^i, d^{ij}, r_u, r_{uv}$
1: **repeat**
2: $\{x_u^i, f_{uv}^{ij}\} \leftarrow$ **Solve_LP(..)**
3: $X \leftarrow \{x_u^i \mid x_u^i \notin \{0, 1\}\}$
4: $\{i_{fx}, u_{fx}\} \leftarrow \mathrm{argmax}_{\{i \in V_V^k, u \in V_S\}} X$
5: **Add_LP_Constraint**("$x_{u_{fx}}^{i_{fx}} = 1$")
6: **until** $X = \emptyset$
7: **return** $\{x_u^i, y_{uv'}^{ij},\}$

Next, we corroborate these claims by the evaluation of the relaxed LP variant. In order to avoid any impact of the preceding VN partitioning phase, we run tests only with VN request mapping onto a single substrate network of 125 nodes. Around 96% of the initial LP solutions have already all x_u^i set to $\{0, 1\}$ which means that no rounding is required at all. The CDF of Figure 14a shows further that the fraction of LP solutions with non-binary x_u^i decreases drastically with each iteration. Figure 14b shows the VN acceptance rate for non-expiring requests of the original MILP and the relaxed LP variant. We observe no significant difference in the results. In addition, the VN embedding cost is equal for MILP and LP in at least 90% of the considered 5000 requests. The significant gain from the relaxed LP variant is apparent when comparing the solver runtimes.



(a) rounding iterations for VN mapping with LP



(b) acceptance rate for VN mapping with MILP/LP

Figure 14: Only 4% of the VNs experience non-binary solutions that require another LP iteration. VN mapping with MILP and LP achieves a comparable VN acceptance rate.

Initially, i. e., for an unutilized substrate network, we measured a maximum LP solver runtime of 5 ms compared to an increase of at least two orders of magnitude in solver runtime for MILP in the same scenario.

We argue that the relaxed LPs for VN request partitioning and VN segment mapping (Sections 3.5 and 3.6) are more suitable for VNE, especially with very large VNs or substrate networks, where the explosion of ILP solver runtime essentially prohibits the use of ILPs. As such, we employ the LPs for our VNE evaluations in Section 3.7.

3.7 EVALUATION

We use our VNE control plane implementation as presented in Section 6.2 for the simulation of multi-provider VNE.[5] In this section, we first explain evaluation parameters and introduce metrics (Sections 3.7.1–3.7.2) before we discuss our simulation results, focusing on the suboptimality in VNE due to limited information disclosure (Section 3.7.3).

3.7.1 *Parameters*

In the following, we discuss the parameters used for our VNE simulations.

Substrate Network. The substrate networks considered in the following are synthesized using the IGen network topology generator [102]. We ran, in addition, tests with topologies based on real InP networks (Figure 15 shows such an

5 Our tests are conducted on a server with two Intel Xeon quad-core CPUs at 2.53 GHz and 12 GB of main memory.

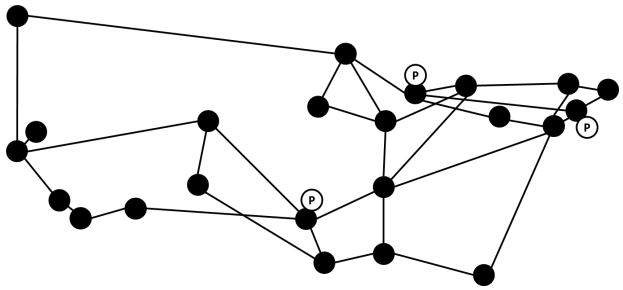


Figure 15: Example of a real InP substrate topology.

example) delivering results that are similar to the results from IGen topologies. The following setting is used throughout the evaluation section if not stated otherwise. The number of participating InPs is between 5 and 10. We exemplary fix the number of substrate nodes for each InP to 25 and generate links using the Waxman method [103]. The mean number of intra-provider and peering links per InP is set to 70 and 4, respectively. Each node is associated with a location identifier while the capacity value for each node and link is randomly selected from a uniform distribution. The residual capacities for substrate nodes and links are updated after a new VN request has been embedded or an existing VN has been released. Resource advertisements from an InP to the VN provider are dynamically updated when resources are assigned or released.

VN Request. A VN request consists of (i) virtual node specifications and (ii) the corresponding bandwidth demands between each pair of virtual nodes given by a traffic matrix. The number of virtual nodes for each VN request is randomly sampled from a given uniform distribution. The node capacity and bandwidth demands also follow a uniform distribution. Further, we include a location attribute to virtual nodes using latitude and longitude values. This lends VN requests an additional dimension, i. e., a desirable geographic footprint. Location constraints essentially obviate the assignment of VNs strictly onto a single InP, without significantly restricting the search space, since different InPs may have overlapping geographic presence and consequently any virtual node may still be mapped onto any of the InPs.

In our evaluation, we process a fixed number of VN requests with limited lifetime which is randomly given by a uniform distribution. We model the arrival of VN requests through a Poisson process, which is an established assumption in the literature (e. g., [63, 70, 78, 79, 84]). The evaluation parameters for the substrate network and VN requests are summarized in Table 3.

We point out that the VN provider constitutes an emerging business model that has not yet materialized on the Internet. Given the lack of real-data sources for VN request workloads, our evaluation parameters are adjusted after broad inspection of resource pricing by cloud providers (e. g., Amazon EC2) and the

Table 3: VNE evaluation parameters.

Substrate network:	
InPs	5 to 10
Nodes per InP	25
Intra-provider links per InP	70 (mean)
Peering links per InP	4 (mean)
Link distribution	Waxman method
Node degree	5.6 (mean)
Node capacity	uniform distrib. [200, 300]
Link capacity	uniform distrib. [4000, 6000]
Virtual network requests:	
Virtual nodes	uniform distrib. [10, 20]
Node capacity	uniform distrib. [1, 8]
Bandwidth demand	uniform distrib. [1, 10]
Mean arrival rate	1 request per 100 time units
VN lifetime	uniform distrib. [500, 5000]
Max. geo distance	unif. distr. [250, 500]

input parameters used in other VNE evaluation environments [63, 70, 78, 79, 84].

3.7.2 *Metrics*

Our main goal is to investigate the suboptimality of LID on multi-provider VNE efficiency and its origins, and further provide useful insights into VNE using the detailed logging of our system, as described in Section 6.1.5. Due to the lack of previous work on multi-provider VNE with a well-defined level of resource and topology information disclosure, we compare VNE under LID against a "best-case" scenario (FID) where the complete network topology and resource availability information is attainable by the VN providers. The complete knowledge of the substrate networks allows their composition into a single substrate where VN requests can be directly embedded using any VN mapping algorithm. As such, to embed VNs under FID, we use our VN mapping formulation (Section 3.6) with the required modifications to support node feasibility mapping. In order to provide a fair comparison between VNE under LID and FID, we do not permit path splitting for VN mapping, (i. e., since there is no path splitting in VN partitioning, VNE under LID cannot benefit fully from it).

To compare VNE efficiency under LID and FID, we first define the *embedding cost* of one VN request as:

$$\sum_{k \in K} \left(\alpha \sum_{i \in V_V^k} d^i + \beta \sum_{\substack{(u,v) \in E_S^* \\ (u \neq v)}} \sum_{\substack{i,j \in V_V^k \\ (i \neq j)}} f_{uv}^{ij} \right) + \sum_{\substack{i,j \in V_V \\ (i \neq j)}} \sum_{p,p' \in P^*} c_{pp'}^{ij}, \quad (24)$$

where the superscript $(\cdot)^*$ at the sets P^* and E_S^* denotes the set of assigned peering points and substrate links, respectively. The set K comprises all VN segments of a given VN request. The embedding cost for multiple VN requests is the sum of the individual VN request costs, each given by (24). The first term in (24) represents the normalized sum of CPU and bandwidth resources for all segments of a VN request. The second term in (24) accumulates the bandwidth resources along the assigned peering links between the VN segments.

Following this definition, we further define the following metrics used in our comparative study:

- **Extra Cost.** Extra cost represents the additional embedding cost incurred under LID relatively to the associated cost under FID. We explore the origins of this extra cost through regression models that empirically show the correlation with specific model variables.
- **Acceptance Rate.** The VN acceptance rate is the fraction of incoming VN requests that is successfully embedded. A difference between the acceptance rates for LID and FID can be regarded as capacity gain that is based on the amount of available information. This capacity gain can lead to revenue gain, under certain conditions discussed at the end of this section.
- **Hop Count.** We consider the empirical cumulative distribution function (CDF) of the hop count of embedded virtual links with LID and FID. The hop count of virtual links can be related to the experienced quality of service along that link. In general, longer paths are associated with higher delays, as well as, a higher packet loss probability.

3.7.3 *Results*

Initially, we measure the extra cost with LID across 300 runs. To this end, we consider a large-scale VNE scenario with 250 VN requests across a diverse number of InPs and VN request sizes.

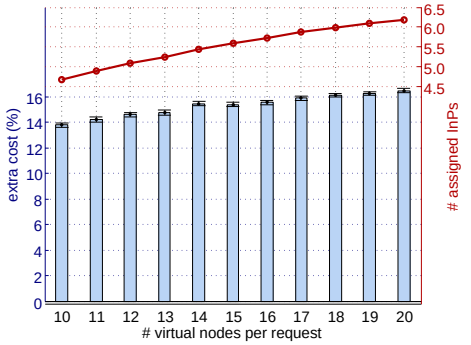
Figures 16a and 16b depict as bar graphs the extra cost for VNE with LID versus the VN size and the number of participating InPs. Accordingly, line graphs

show for LID the number of assigned InPs. Extra cost increases proportional to the VN size while no clear trend can be observed with increasing number of participating InPs. A higher number of them might increase the competition among InPs but also the overall geographic coverage. In the latter case, more VN requests with broader geographic demand are attracted and thus more InPs need to be assigned as the geographic coverage of each single InP remains unchanged. Our tests indicate that this effect is stronger than the effect of potential savings by overlapping regions of competing InP. Figure 16c depicts the extra cost versus the number of assigned InPs and the VN size. These results show that VN size alone does not affect cost significantly if the number of assigned InPs is constant. The increasing extra cost caused by increasing VN size as suggested by Figure 16a is a result of increasing number of assigned InPs. This can again be explained by the broader geographic demand growing with each additional virtual node. Overall, LID incurs 5%-20% extra cost compared to FID. Considering that most resource and substrate topology information is concealed from the VN provider, such cost increase is deemed reasonable. This result also corroborates the efficiency of our multi-provider VNE solution. While there is an increase in the extra cost with more assigned InPs, we do not observe any particular trend in the cost with diverse VN sizes.

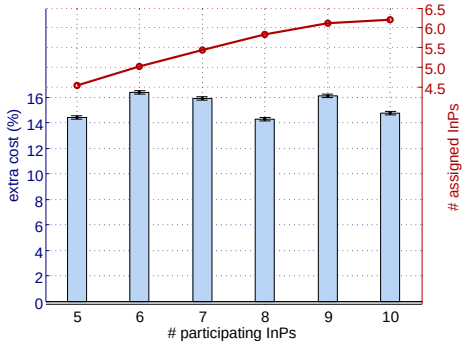
Next, we delve into the origins of the extra cost with LID. We first examine whether LID leads to a larger number of assigned InPs. This may be a possible justification for the additional cost since more InPs incur higher peering link costs. However, Figure 16d provides no statistical evidence of a significant difference of the medians between the two information disclosure levels.

Since the VN embedding cost accumulates all nodes and link costs, we further examine the correlation between the extra cost with (i) the extra node cost, and (ii) the extra link cost. Figure 17a illustrates a scatter plot of extra cost vs. extra link cost, showing a strong correlation between both quantities. This is validated by the coefficients provided by the regression model, i. e., slope $\beta = 0.64$ with $R^2 = 0.912$ and p-value < 0.001. With respect to nodes, our results (not shown here) do not reveal any perceptible increase in the node cost with LID. This occurs because the costs of virtual nodes with certain specifications are advertised to the VN provider and are, therefore, taken into account during VN partitioning. In fact, we observe a few cases where the embedding cost minimization with FID results in the selection of more expensive virtual nodes (compared to the nodes assigned with LID) when the associated link cost is lower.

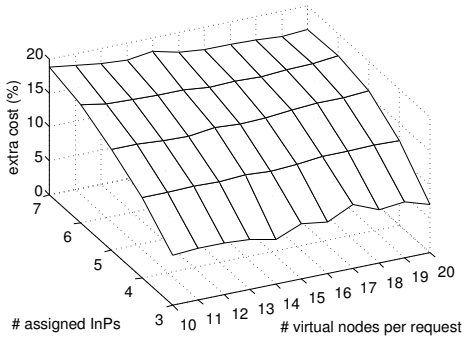
To further investigate the origin of the extra link cost, we present a scatter plot of the extra hop count versus the extra link cost in Figure 17b. We perform a linear regression analysis of this data to obtain the following coefficients: slope



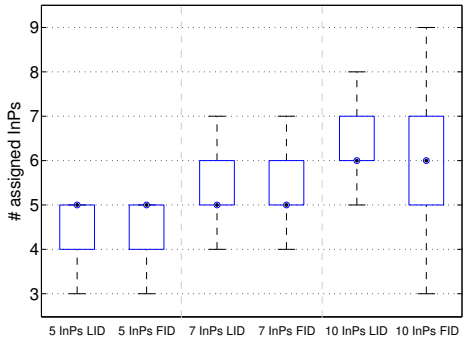
(a) extra cost and number of assigned InPs for VN embedding under LID vs. VN size



(b) extra cost for VN embedding under LID vs. number of participating InPs



(c) extra cost for VN embedding under LID vs. number of assigned InPs and vs. VN size

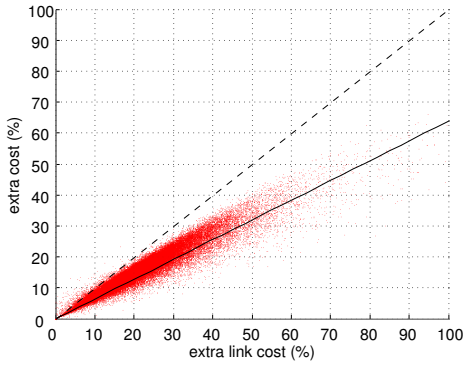


(d) assigned InPs vs. participating InPs

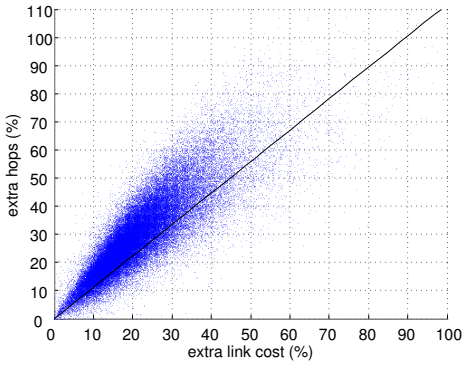
Figure 16: Extra cost and number of assigned InPs for VN embedding under LID increase steadily with the VN size. An increasing number of participating InPs results in a higher number of assigned InPs while no correlation between participating InPs and extra cost under LID can be observed. Extra cost for VN embedding under LID primarily stems from the assignment of additional InPs while the VN size does not affect extra cost at an equal number of assigned InPs. VN embeddings under LID and FID assign the VNs to a comparable number of InPs.

$\beta = 1.12$, $R^2 = 0.73$, and p-value < 0.001. Thereby, we find that the extra link cost is correlated with longer paths (i. e., additional hops). This finding is also confirmed by Figure 17c, which depicts the CDF of the hop count of virtual links both with LID and FID. It is clearly shown that embedding with LID increases the number of hops onto which virtual links are mapped. According to Figure 17c, the maximum increase in the number of hops is three for 95% of the virtual links. Furthermore, Figure 17d shows that virtual links with FID tend to span fewer InPs. For instance, 77% of the virtual links are embedded onto less than three InPs with FID, as opposed to 67% for LID.

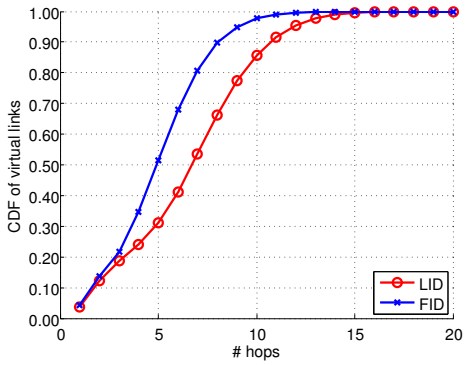
Finally, we compare the acceptance rate of VN requests with LID and FID across 50 runs, each one containing 2000 VN requests. Figure 18a depicts the



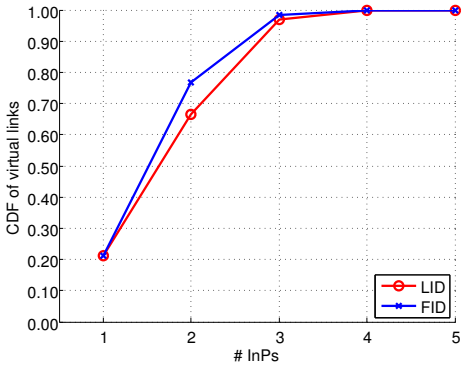
(a) overall extra cost vs. extra link cost



(b) extra hops vs. extra link cost



(c) CDF of hop count of virtual links

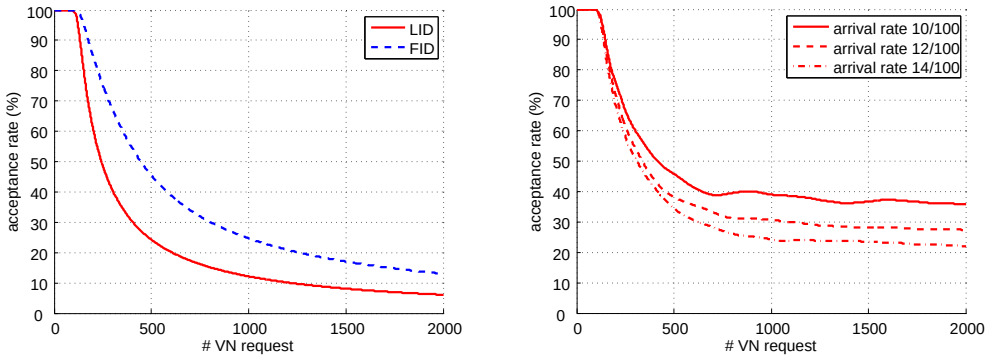


(d) CDF of number of assigned InPs per virtual link

Figure 17: A strong correlation exists between overall extra cost and extra link cost for VN embedding under LID. Extra link cost is in turn correlated with longer paths, i. e., additional hops. In addition, embedding with LID increases the number of hops used per virtual link while its embedding under FID tend to span fewer InPs.

acceptance rate with non-expiring VN requests. Embedding under LID leads to VN request rejection prior to the full information counterpart. The higher acceptance rate for FID implies better resource utilization and essentially higher revenue for InPs. Figure 18b provides a comparison of the acceptance rates for expiring VN requests for different VN request arrival rates under LID, showing that the acceptance rates for all arrival rates reach a steady state. This indicates that a significant fraction of the VN requests can be embedded for long periods. As depicted in Figure 18b, the steady-state acceptance rate decreases with increasing VN request arrival rates.

Our evaluation results indicate the feasibility of multi-provider VNE with LID. Although most of resource and topology information is concealed from the VN provider, the embedding cost in most cases is moderately higher, while the VN request acceptance rate converges to a steady-state which ensures predictable



(a) non-expiring VN requests.

(b) expiring VN requests for different arrival rates under LID.

Figure 18: In the case of non-expiring VN requests, embedding under FID can embed more VNs than LID. Expiring VN requests allow reassignment of resources then a nearly constant acceptance rate can be achieved.

embedding. The mapping of virtual links onto longer substrate paths mainly accounts for the extra embedding cost.

The higher acceptance rate under FID can increase the revenue for an InP, assuming that the InP's resource pricing is competitive (i. e., such that the InP receives segments of submitted VN requests). This could incentivize InPs to disclose more resource information to VN providers. Although our resource information model captures most of the non-confidential information for InPs, topology abstractions disclosed to VN providers could potentially indicate certain InP preferences, e. g., a preferred path for reaching a peering node. For example, the edges of a topology graph can be annotated with weights assigned by each InP, according to his policy (e. g., network load balancing). This is in line with the multi exit discriminator (MED) attribute of the border gateway protocol (BGP). Furthermore, an InP may wish to incorporate computing resource utilizations into the weights of the adjacent links. This could be achieved via a link weight offset which is dynamically adjusted according to the utilization level. Such an increased level of information disclosure can enable a VN provider to tailor VN partitioning to the InPs (rather than the SP), improving resource efficiency and VN request acceptance rate.

3.8 RELATED WORK

There is a large body of work on embedding VN requests onto a single substrate network [67, 70, 78, 79, 84].[6] These VNE techniques aim at optimizing the mapping of VN topologies and require full knowledge of the available substrate

6 Section 2.5.2 provides an overview of existing VNE algorithms.

resources and network topologies. In a multi-provider setting, they can be used for VN segment mapping, after their coupling with a VN request partitioning mechanism. One limitation of these VNE methods is that they are tailored to topology-based VN requests which may introduce unnecessary restrictions in VNE, possibly excluding efficient solutions. Instead, our VN segment mapping formulation (Section 3.6) supports traffic matrix-based requests which yield higher efficiency in VNE.

VNE across multiple substrate networks has been mainly studied in [60, 63, 65, 87, 104]. Houidi et al. [63] decompose multi-provider VNE into VN request partitioning (carried out by a VN provider) and VN segment mapping (performed by each assigned InP). The main goal of this work is the comparison of exact and heuristic methods in terms of VNE efficiency. Without any inspection of resource information disclosure, VN partitioning is carried out based on a highly abstract view of the substrate network (i. e., AS-level topology), which does not include publicly available information, such as the location of peering nodes. Since there is no binding between virtual nodes and peering nodes (instead, the virtual nodes are only assigned to InPs), the link costs estimated during VN request partitioning may be significantly different from the link costs after the final VNE. Hence, cost-efficient VN embeddings are subject to optimizations during VN segment mapping rather than VN request partitioning. In this respect, the abstract view on the underlay combined with the restrictions of topology-based VN requests can lead to inefficient VN embeddings and increased expenditure for SPs. Our evaluation results indicate that a low degree of information disclosure can increase the embedding cost and reduce the VN acceptance rate and revenue for InPs. Leivadeas et al. [65] focus on networked cloud computing environments with multiple InPs, at which requests are partitioned based on iterated local search (ILS) by a cloud broker. The partitioning phase is followed by the final mapping phase, at which InPs use an MILP model based on the VN mapping formulation in [70]. ILS yields substantially lower runtime compared to an ILP, but may still require a large number of iterations and considerable communication overhead between the broker and the InPs, before an approximate solution has been found. The paper does not discuss the visibility of the broker on the substrate network resources, which we consider critical for the efficiency of VNE across multiple substrate providers. In contrast to these approaches, our work is focused on the impact of LID on VNE efficiency. VN request partitioning is carried out based on a well-defined substrate network view that has been synthesized from network elements that are disclosed by InPs and cloud providers. According to our knowledge, our work comprises the first study on VNE that sheds light on the origins of VNE suboptimality, due to LID.

Recent studies have also employed auction-based techniques for multi-provider VNE. *V-Mart* [60] uses a two-stage Vickrey auction model that allows InPs to bid for virtual resources that have been placed on the market by the SP. Since the auctioned items are heterogeneous, the generalized multi-unit Vickrey auction can result in certain inefficiencies. First, because of the complementarities or substitutabilities that typically exist among different units, each InP is required to submit a bid for each possible collection of resources he may provide. Consequently, the generation and process of all bids can be a highly demanding task, which can entail the disclosure of information that an InP prefers to keep as confidential. Second, the auction can result in high expenditure for the SP, which may further increase if some of the bidders form a coalition and coordinate their bids. Finally, other practical considerations, such as the various constraints that the InPs may introduce, can adversely affect the outcome of the auction [105]. Furthermore, Esposito et al. [104] propose a consensus-based auction mechanism for VNE in a distributed manner. More precisely, the substrate nodes exchange their bids for virtual resources requested by the SP, seeking consensus for the winner. The proposed technique may offer better scalability, especially for large substrate networks; however, InP policies will hinder the collaboration of substrate nodes belonging to different InPs. As such, in a multi-provider setting, the proposed VNE mechanism may constitute an alternative approach only the for VN segment mapping phase (i. e., via iterations between each InP and the VN provider). Another limitation of the proposed VNE is the decoupling of link from node mapping which may result in VNE inefficiency.

Our multi-provider VNE methods are directly applicable to the network virtualization architectures (as discussed in Section 2.4) that rely on a layer of indirection between SPs and InPs [12, 20]. The distributed architecture presented in [62], where VN requests are relayed across InPs until the completion of VNE, also benefits from our VNE framework. In particular, our VN partitioning method (Section 3.5) can be used to determine the InP that should be contacted first, improving the convergence of this distributed VNE approach.

4

POLICY-COMPLIANT VIRTUAL NETWORK EMBEDDING

In the previous chapter, we rely on a centralized coordinator (i. e., VN provider) that partitions requested VNs into multiple segments in a single step based on the resource information stored in his local repository. In this case, no means is given to the InPs to appraise the economic potential of the requested VNs and to attract the more beneficial ones. We fill this gap and propose a policy-compliant VNE algorithm particularly for decentralized VN embedding (VNE) in general and for centralized VNE in combination with auctions [106]. To this end, we focus on VNE profitability from the perspective of the InPs. In particular, we introduce a metric to assess the profitability of VNE (Section 4.1) and provide an algorithm that rejects too costly embeddings by applying a profitability threshold based on the InP's policy (Sections 4.2 and 4.3). We further show that such a profitability threshold provides the means to the InPs for striking a balance between short-term and long-term revenue gains (Section 4.4).

4.1 PROFITABILITY OF VN EMBEDDING

VNE consists in mapping VN topologies onto a shared substrate network. Existing VNE algorithms optimize VN assignment based on objectives, such as maximizing VN acceptance rate, minimizing VN embedding cost, maximizing revenue for the InP or achieving load balancing across the substrate network [67, 70, 78, 79, 84][1]. A common feature among these VNE algorithms is that they embed complete VNs, exactly as requested, as long as there are sufficient substrate network resources. Complete VN request embedding is more appropriate for small VNs that can be mapped onto a single substrate provider with a relatively low embedding cost. Larger VN requests that possibly exceed the geographic footprint of a substrate provider have to be partitioned across multiple InPs.

Basically, two multi-provider VNE architectures (Figure 19) exist as introduced in Section 2.4. Both support the approach to allow the InPs to embed the subset of a VN request that generates more profit. More precisely, in auction-based VNE environments the InP will seek to place a higher bid for the most profitable subset of a VN request [60], while in a distributed VNE framework the InP will seek to identify and embed the profitable VN subset and subsequently relay the rest of the VN request to a neighboring InP, as exemplified

1 See also Section 2.5.2 for an overview of related work on VNE.

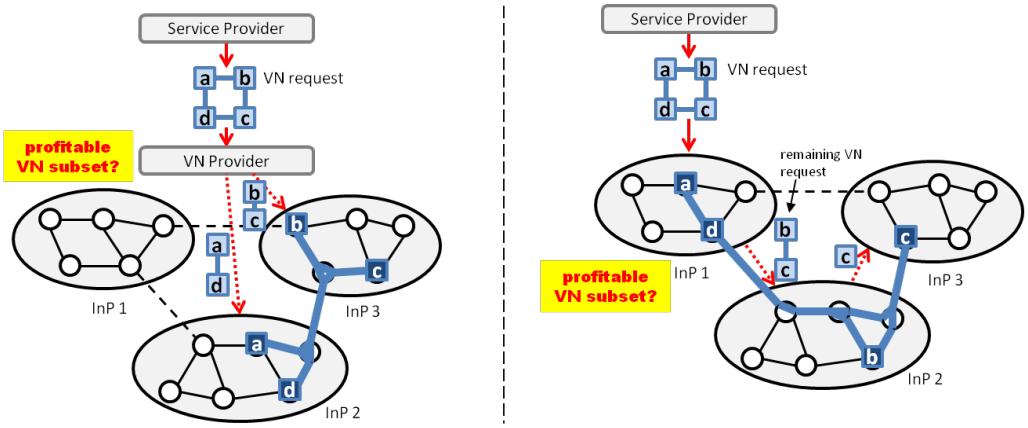


Figure 19: InPs can benefit from policy-compliant VN embedding in both, centralized and distributed architectures, by the identification of the most profitable VN subset.

by PolyViNE [62]. Both VNE frameworks raise the requirement for embedding only the profitable subset of a VN request, which is not fulfilled by any of the existing VNE algorithms, according to our knowledge. We note that the profit gained by embedding a particular VN request subset may vary among different InPs, and consequently, a VN subset relayed to a neighboring InP could be efficiently embedded into this InP's substrate network. Figure 20 depicts an example where InP 1 has embedded a VN subset including virtual nodes a, d, and f and relayed the remaining VN request with b, c, e, and g to his neighboring InP that in turn needs to profitably embed this VN and relay any remaining part to a neighbor of him. Next, we discuss a metric for VNE profitability (Section 4.1.1) and introduce a threshold to enforce the InP's policy on profitability (Section 4.1.2).

4.1.1 *CRR Metric*

InPs might define profitability in different ways depending on their business models. One InP could consider any large VN as profitable and allows oversubscription while another InP aims at attracting premium customers that are willing to pay more for a guaranteed access to resources. We reduce these definitions to a simple form at which profitability is associated with the amount of used resources per requested CPU and bandwidth demand. Figure 21 shows an example of the embedding of a VN consisting of three virtual nodes and two virtual links. Here the virtual nodes a and c are three hops distant to each other and thus the embedding of virtual link (a, c) is more costly than it would be if virtual node c was mapped to a neighboring node of the host of a in the

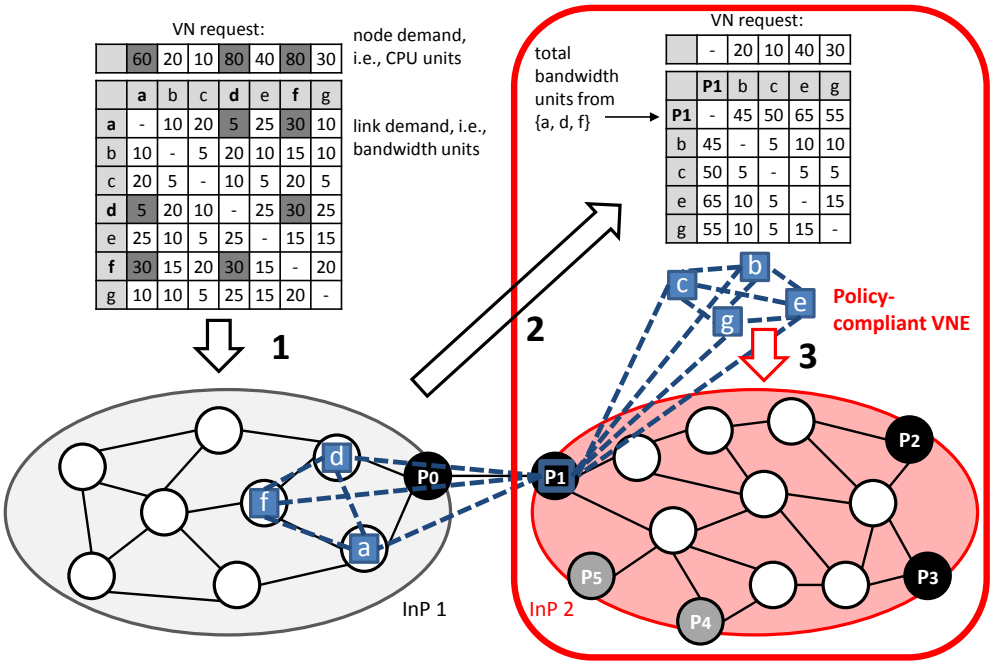


Figure 20: InP 1 receives a VN request from which he identifies the VN subset consisting of the nodes $\{a, d, f\}$ as the profitable subset. He then relays a new VN request consisting of the nodes $\{b, c, e, g\}$ to InP 2. He, in turn, embeds the remaining part of the VN into his substrate network.

substrate topology. Cost alone is not a sufficient metric to express profitability since unexploited resources might *cost* as well. Instead, we consider the potential revenue – the income gain by embedding a VN. From the InP's perspective VN embeddings with lower cost but higher revenue are more profitable.

We refer to the network models and notations introduced in Section 4.2 and define the **revenue** of a VN request initially as:

$$\mathbb{R} = \sum_{i \in V_V} d^i + \alpha \sum_{\substack{i,j \in V_V \\ (i \neq j)}} d^{ij} + \left\{ \frac{\alpha}{2} \sum_{i \in V_V, j \in P'} (d^{ij} + d^{ji}) \right\} \tag{25}$$

Revenue essentially accumulates all the node and link capacity units of the VN request. The term in curly brackets represents the revenue generated from peering links and is split among the involved InPs. Similar to [79], the parameter α is used as a weight in the sum of the computing and the bandwidth revenues. For the purpose of revenue-based node reordering, as required by our algorithm, we further define the revenue gain of the virtual node $i \in V_V$, as follows:

$$\mathbb{R}(i) = d^i + \frac{\alpha}{2} \sum_{\substack{j \in V_V \cup P' \\ (i \neq j)}} (d^{ij} + d^{ji}) \tag{26}$$

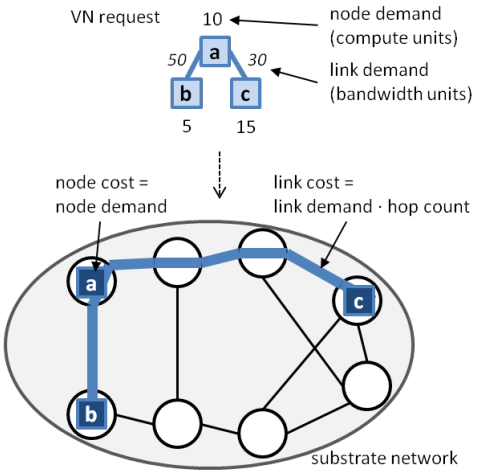


Figure 21: Node cost equals the node demand while link cost requires the link demand be multiplied by the number of used hops.

The **VN embedding cost** is defined as follows:

$$\mathbb{C} = \sum_{i \in V_V} d^i + \alpha \sum_{\substack{i,j \in V_V \cup P' \\ (i \neq j)}} d^{ij} \cdot \mathrm{dist}(u_i, u_j) \tag{27}$$

$$u_i, u_j \in M_V(V_V) \cup P'$$

The VN embedding cost accumulates all node and link costs.

An InP will seek to increase his revenue while maintaining a low embedding cost. As such, we use the

$$\textit{Cost to Revenue Ratio (CRR)} = \mathbb{C}/\mathbb{R} \tag{28}$$

to express VNE efficiency and eventually, how much profit an InP gains by embedding a VN request. Based on the preceding definitions, CRR $= 1$ yields the highest profit while an increasing CRR means decreasing profit.[2]

4.1.2 *CRR trend evolution*

We step back to the illustration of Figure 21 where the bandwidth allocation between the virtual nodes a and c is the worst component in terms of CRR while all other resource assignments are already optimal. It is obvious that the embedding of the full VN request is less resource efficient than the embedding of the VN subset only consisting of the nodes a and b plus the virtual link among.

2 It is premised that virtual nodes are not co-located, and thus, at least, one hop is required for each virtual link.

Next, we investigate CRR trend evolution for an exemplary embedding according to Figure 20 where a four-node VN with a full traffic matrix is requested for the embedding at InP 2. The respective mapping is shown in the Figures 22a – 22d while the CRR is computed in the table in Figure 22e.

The embedding of the single virtual node e (Figure 22a) requires a virtual link to InP 1 in order to reach all the already embedded virtual nodes where the respective bandwidth demand is aggregated from the original virtual links $a - e$, $d - e$, $e - f$. Such virtual links are commonly regarded as less profitable since peering is commonly associated with high cost. Moreover, if two InPs are involved in the mapping of a virtual link, then revenue need to be shared between them.

The second virtual node g lets the embedding of the VN subset (now comprising e and g, Figure 22b) becomes more efficient in terms of CRR since the virtual link $e - g$ which generates full revenue requires merely a single hop for embedding.

Increasing more and more the VN subset size (Figures 22c and 22d) increases the cumulative revenue continuously but at the same time VN embedding cost increases faster with the more distant node mappings requiring longer paths for the virtual links. In our example, the embedding of the three node VN subset was less resource efficient than the embedding of any smaller subset. The same is true for the next larger VN subset - the four node VN originally requested at InP 2.

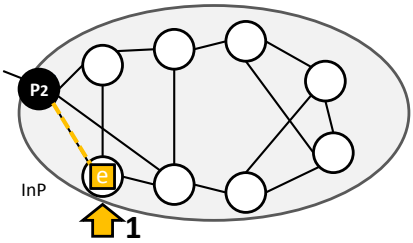
VN request:

	-	20	10	40	30

node demand

	P2	b	c	e	g
P2	-	45	50	65	55
b	45	-	5	10	10
c	50	5	-	5	5
e	65	10	5	-	15
g	55	10	5	15	-

link demand



(a) VN subset: e

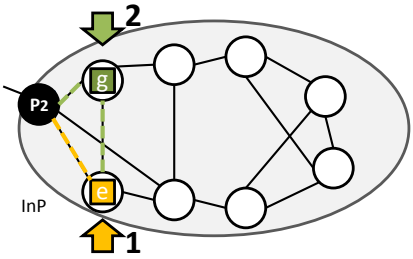
VN request:

	-	20	10	40	30

node demand

	P2	b	c	e	g
P2	-	45	50	65	55
b	45	-	5	10	10
c	50	5	-	5	5
e	65	10	5	-	15
g	55	10	5	15	-

link demand



(b) VN subset: e,g

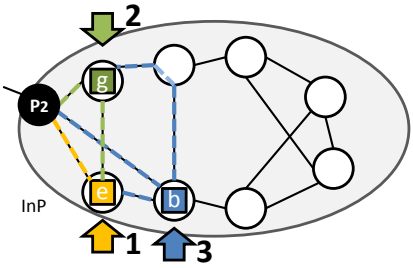
VN request:

	-	20	10	40	30

node demand

	P2	b	c	e	g
P2	-	45	50	65	55
b	45	-	5	10	10
c	50	5	-	5	5
e	65	10	5	-	15
g	55	10	5	15	-

link demand



(c) VN subset: e,g,b

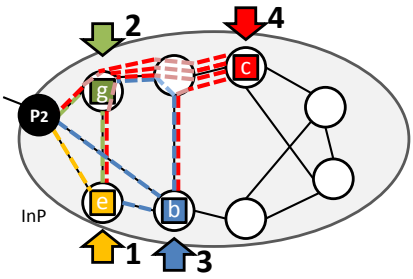
VN request:

	-	20	10	40	30

node demand

	P2	b	c	e	g
P2	-	45	50	65	55
b	45	-	5	10	10
c	50	5	-	5	5
e	65	10	5	-	15
g	55	10	5	15	-

link demand

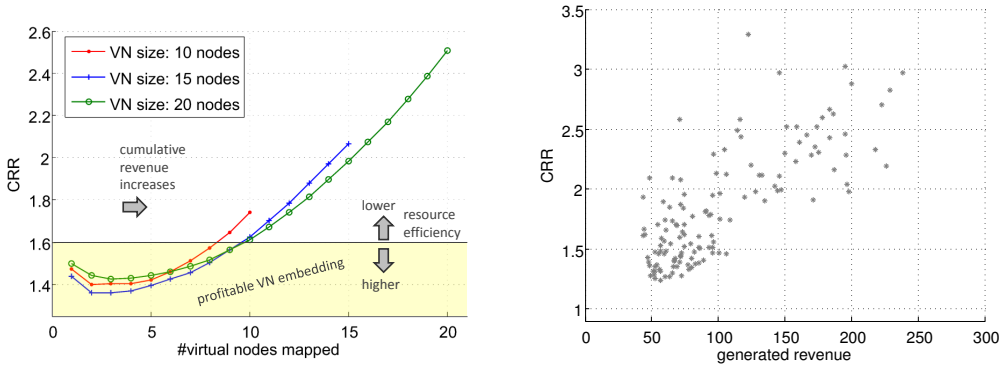


(d) VN subset: e,g,b,c

Figure	VN subset	ℝ	ℂ	CRR
22a	e	105	170	1.62
22b	e,g	220	340	1.55
22c	e,g,b	325	600	1.85
22d	e,g,b,c	415	980	2.36

(e) VNE statistics for the example in Figs. 22a – 22d

Figure 22: High CRR values indicate the mapping of long paths in the substrate topology to the virtual links. This in turn can be observed particularly for larger VNs.



(a) CRR evolution trend

(b) correlation between CRR and generated revenue

Figure 23: A general trend in CRR evolution can be observed. Larger VNs can be embedded less efficiently, and any CRR threshold impacts the generable revenue. CRR is further correlated with the generable revenue of a VN.

We ran additional tests and observed a general trend in CRR evolution. Figure 23a illustrates the evolution of CRR during the iterative embedding of VN requests originally with 10, 15, and 20 nodes. Irrespective of the VN size, CRR increases as a larger subset of the VN is being mapped. We observe a more severe increase in the CRR for larger VNs (i. e., 20 nodes). As such, embedding of complete VN requests of large size incurs a penalty in terms of resource efficiency. This penalty stems from the increasing hop count of virtual links, which counterbalances the extra revenue generated by embedding additional virtual nodes and links. Figure 23b also corroborates this observation, uncovering a correlation between the CRR and the VN revenue.

Based on these observations, we rely on CRR to identify the most profitable subset of a VN request. As such, we seek the embedding of the largest subset of a VN request that does not violate a predefined CRR threshold. In principle, this threshold should be adjusted by the InP according to its policy. The CRR threshold represents a trade-off between revenue and resource efficiency. More precisely, a low CRR threshold is expected to yield higher resource efficiency but will generate revenue at a lower rate, since only small subsets of VN requests will be embedded. Conversely, a high CRR threshold will generate revenue faster but may impair resource efficiency. To this end, in the following section, we present a new algorithm that enables the embedding of subsets of VN requests according to an adjustable CRR threshold. The impact of CRR threshold adjustment on resource efficiency and generated revenue is further discussed in Section 4.4.2.

4.2 NETWORK MODEL

In the following, we introduce the substrate and virtual network models used in the VNE algorithm description (Section 4.3).

Substrate Network Model. The substrate network is represented as a weighted directed graph $G_S = ((V_S \cup P), E_S)$, where V_S is the set of the substrate nodes and E_S is the set of substrate links between these nodes. The set P comprises all the nodes of the substrate network where peering has been established with other network providers. Each substrate node $u \in V_S$ is associated with the residual capacity $r_u \in R_N$. Each substrate link $(u, v) \in E_S$ between two substrate nodes u and v is associated with the residual capacity denoted by $r_{uv} \in R_E$.

VN Request Model. A VN request consists of a combined set of the virtual nodes V_V and the used peering nodes $P' \subset P$ and a set of bandwidth demands $d^{ij} \in D_E$ between any pair of virtual nodes $i, j \in V_V$ and between any virtual node $i \in V_V$ and peering node $j \in P'$. We use a traffic matrix to express all bandwidth demands. Compared to topology-based VN requests that are typically used by the majority of VNE algorithms, traffic-matrix based VN requests allow more flexibility in VNE and a higher level of abstraction which can simplify the specification of network service requirements [10, 87]. We include the peering nodes P' in order to accumulate the bandwidth demands between virtual nodes assigned to different InPs, as shown in Figure 20. Furthermore, each virtual node $i \in V_V$ is associated with the requested capacity denoted by $d^i \in D_V$.

We express the mapping M_V of the virtual nodes to the substrate nodes as $M_V : V_V \to V_S$.

Table 4: Notations used in the Sections 4.1–4.3.

Symbol	Description
C	set of candidate substrate nodes for root node assignment
$cost$	set of virtual link embedding costs for all candidate substrate nodes
CRR	set of CRR values computed after the assignment of each virtual node
CRR_{max}	CRR threshold
CRR_{sol}	set of CRR values for the best solution for each VN subset
d_{from}, d_{to}	inbound / outbound bandwidth demand at a substrate node
d^{i}	computing demand for virtual node i
d^{ij}	bandwidth demand from virtual node i to virtual node j
$dist$	hop-count of shortest paths between each pair of substrate nodes
$dist_{P,min}$	minimum distance (number of hops) to all peering nodes
$dist_{P,max}$	maximum allowable distance (number of hops) to all peering nodes
D_L	bandwidth demands for each pair of virtual nodes
D_N	set of CPU demands for each virtual node
E_S	set of substrate links
M_V	temporary VN mappings
M_{V_sol}	final VN mapping
N_{cand}	set of candidate substrate nodes
V_S	set of substrate nodes
V_V	set of virtual nodes
P	set of peering nodes
P'	set of peering nodes specified in a VN request
r_{from}, r_{to}	available inbound / outbound bandwidth at a substrate node
R_L	residual link capacity (bandwidth units)
$R_{L,rollback}$	link capacity stored for rolling back incomplete mappings
R_N	residual node capacity / CPU units
$R_{N,rollback}$	node capacity stored for rolling back incomplete mappings
S	set of candidate substrate nodes
u_{cand}	candidate for virtual node mapping
u_{root}	candidate for root node mapping
α	balancing factor
θ	distance tolerance to all peering nodes in relation to $dist_{P,min}$
ζ	maximum number of hops from the root node

4.3 POLICY-COMPLIANT VN EMBEDDING ALGORITHM

In this section, we describe the proposed policy-compliant VNE algorithm. The main objective of this algorithm is to identify and embed the subset of a VN request that generates the highest revenue without violating a predefined CRR threshold. To this end, the algorithm seeks feasible solutions by iteratively embedding virtual nodes and their adjacent links. In order to maximize the generated revenue, the algorithm uses the virtual node revenue, as defined in Eq. (26), to sort the nodes of the VN request in terms of revenue in decreasing order. As such, the embedding starts with the virtual node that generates the highest revenue (called *root node*). Since virtual links spanning multiple hops increase the CRR, the assignment of each virtual node aims at minimizing the hop count of the adjacent virtual links. Therefore, for the assignment of the root node, among the substrate nodes with sufficient capacity, the algorithm selects the one with the minimum total distance from the peering nodes (e. g., $\{P_1, P_2, P_3\}$ for InP 2 in Figure 20). Similarly, the remaining virtual nodes are assigned iteratively. Each iteration comprises an evaluation of candidates for virtual node mapping by computing the shortest paths for each virtual link connecting the candidate node with every assigned virtual and peering node. The candidate with the minimum additional embedding cost is selected, and its respective node and link mappings are taken into account when the remaining virtual nodes are mapped onto the substrate network. We reduce the solution space by defining a substrate network region that contains all nodes with a maximum distance from the substrate node on which the root node was mapped.

The algorithm keeps track of the evolution of CRR, after each virtual node has been assigned, and checks the feasibility of the current solution. In the case that the complete VN request can be embedded without violating the CRR threshold, the algorithm returns this solution and terminates its execution. Otherwise, the algorithm computes and stores the largest subset of the VN request that can be embedded without exceeding the threshold for all possible assignments of the root node. To speed up the algorithm execution, we do not consider solutions whose root node mapping is very distant from the peering nodes, since this incurs a significant penalty in terms of CRR. Finally, among all feasible solutions, the algorithm returns the VN subset that generates the highest revenue.

In the following, we present an exemplary VN embedding (Section 4.3.1) and provide further details on the algorithm (Section 4.3.2).

4.3.1 *An exemplary VN Embedding*

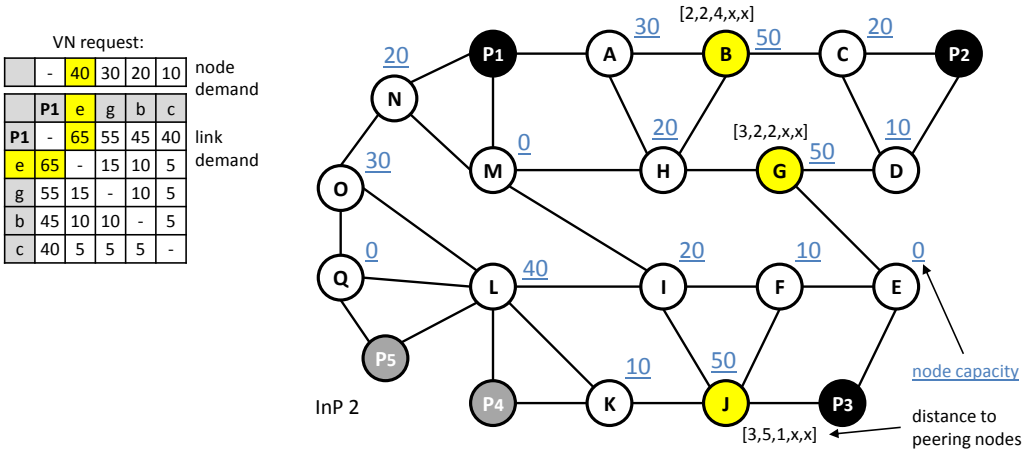
Hereby, we illustrate an exemplary VNE with our algorithm. Figure 20 shows two InPs and a VN request being submitted to InP 2. Assume that three virtual nodes (i. e., a, d, f) previously belonging to this VN request have been already assigned to InP 1. The VN request consists of virtual node demands (i.e., expressed in terms of compute units) and virtual link demands (i. e., expressed in terms of bandwidth units) formulated as a symmetric traffic matrix[3]. Besides the bandwidth demands between each pair of virtual nodes, the traffic matrix also includes the bandwidth demand between each virtual node and the peering node P_1. This essentially expresses the sum of bandwidth demands between the virtual nodes assigned to InP 1 (i. e., a, d, f) and the nodes of this request (i.e., b, c, e, g). Note that InP 2 does not have any information about the substrate topology and the virtual node mappings on InP 1. In this example, we assume that any subset of the VN request that cannot be embedded (because its mapping is either infeasible or unprofitable) will be relayed to a neighboring InP reachable through peering nodes P_2 or P_3. Figures 24 – 25d illustrate the steps of our algorithm for the embedding of this request onto InP 2 that has adjusted his CRR threshold to 3.65.

Virtual node e is the root node of this VN request as it generates the highest revenue. The algorithm first identifies potential hosts $\{B, G, J\}$ for this virtual node which are not too distant from the peering nodes and which fulfill the capacity requirements (Figure 24). The basic idea of this approach is to test different regions in a substrate network. In the following, we go through the example with the variant $e \rightarrow G$.

The mapping of virtual node e to substrate node G is depicted in Figure 25a. The virtual link $e - P_1$ is mapped to the shortest path between the substrate node G and the peering node P_1. Although the current CRR value 4.10 violates the threshold, the algorithm proceeds with the mapping of the virtual node with the 2^{nd} largest revenue, i.e., virtual node g, since CRR is still expected to improve further according to the general CRR trend evolution (Section 4.1.2).

To this end, the algorithm defines a region containing all substrate nodes with a maximum distance from the root node (set to two in our example). This designates the following candidate set $S := \{A, B, C, D, E, F, H, M\}$, from which all the previously assigned substrate nodes (to the same VN request) are removed. In addition, the nodes with insufficient node capacity are eliminated from this list, thus, the 2^{nd} virtual node g can only be mapped to A or B. The algorithm

3 A symmetric traffic matrix is shown for simplicity. Our algorithm can process and embed VN requests expressed with asymmetric traffic matrices.



	-	40	30	20	10

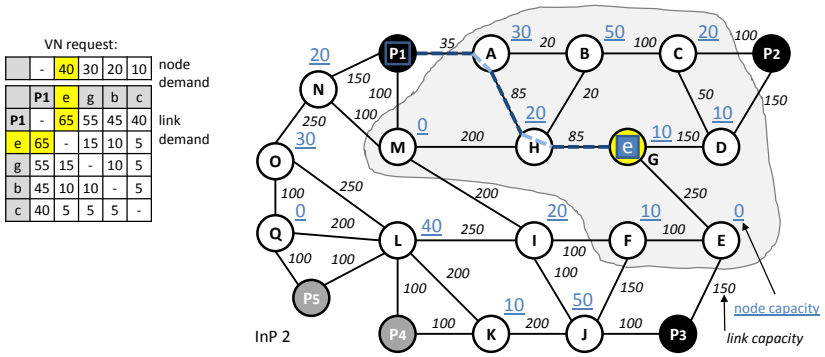
	P1	e	g	b	c
P1	-	65	55	45	40
e	65	-	15	10	5
g	55	15	-	10	5
b	45	10	10	-	5
c	40	5	5	5	-

Figure 24: At the beginning, potential root node candidates B, G, J are identified based on their distance to the peering nodes.

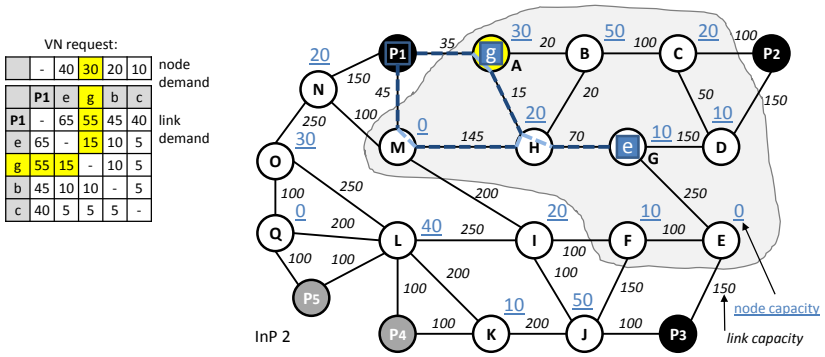
eventually selects the substrate node A, since it incurs lower embedding cost than node B (Figure 25b).

Afterwards, the algorithm examines the assignment of the virtual node b, as depicted in Figure 25c. The candidate node set is $S = \{B, C, H\}$ from which H was finally selected for the mapping of the 3^{rd} virtual node b.

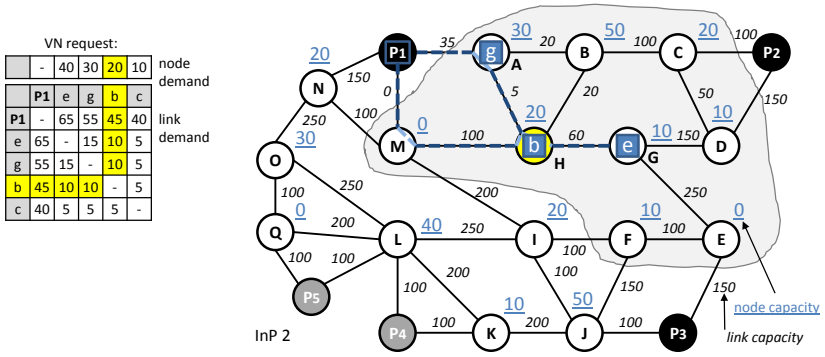
The 4^{th} and last virtual node c can finally be mapped to a node out of $S = \{B, C, D, F\}$. F yields the mapping with the lowest CRR of 3.69. The CRR trend is now positive thus even higher values can be expected for the larger VN subsets according to general CRR trend evolution as discussed in Section 4.1.2. For this reason, the algorithm stops if CRR threshold is violated. This is the case for the 4^{th} virtual node as its best achievable CRR of 3.69 is greater than the CRR threshold of 3.65 (Figure 25d).



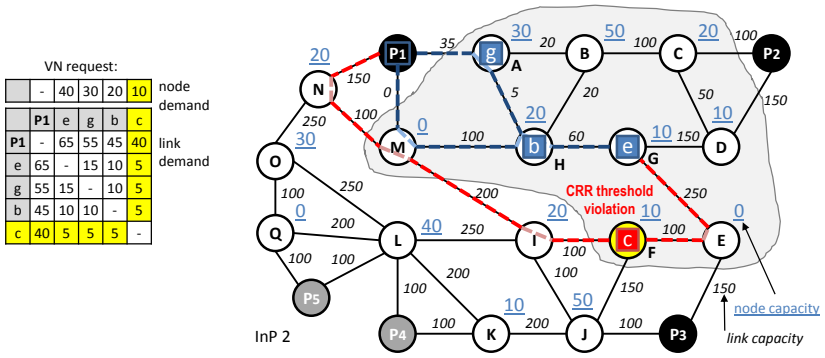
(a) The first virtual node is mapped to the root node, in this case, root node G.



(b) The second virtual node is mapped to the node A.



(c) The third virtual node is mapped to the node H.



(d) The CRR threshold is violated by stepping from a VN subset of three nodes to a full VN with full nodes, since the mapping of the fourth virtual node, introduces too long paths.

Figure 25: Algorithm steps for the root node candidate G.

Table 5: VNE statistics for the example in Figures 25a – 25d with root node → G.

Figure	VN subset mapping	$\mathbb{R}$	$\mathbb{C}$	CRR
25a	e→G	105	430	4.10
25b	e→G, g→A	220	850	3.86
25c	e→G, g→A, b→H	325	1090	3.35
25d	e→G, g→A, b→H, c→F	415	1510	3.69

Table 6: VNE statistics for the example in Section 4.3.1 with diverse root node assignments.

Root node mapped to		G	B	J
Distance to peering nodes		7	8	9
VN subset	$\mathbb{R}$	CRR		
e	105	4.10	9.05	4.10
e,g	220	3.86	6.09	3.86
e,g,b	325	3.35	4.86	3.63
e,g,b,c	415	3.69	4.72	3.73

The algorithm then repeats the steps shown in the Figures 25a – 25d with the other candidate nodes for the root node assignment: B and J. The CRR values for each VN subset are shown in Table 6. The maximum number of virtual nodes that can be embedded without violating the CRR threshold is 3. The variant with the root node assigned to substrate node G achieves the best CRR. Thus, the algorithm returns the mapping result $\{e \rightarrow G, g \rightarrow A, b \rightarrow H\}$ with $\{e, g, b\}$ as the most profitable VN subset.

4.3.2 *Pseudocode Description*

The algorithms 3–5 illustrate the pseudocode of the policy-compliant VNE algorithm. Algorithm 3 performs the VNE computation while Algorithms 4 and 5 include auxiliary functions to identify candidate nodes with sufficient residual capacity and low link embedding cost.

Hereby, we explain Algorithm 3 in detail. In the lines 1–8, an initialization phase takes place. All virtual nodes are sorted in terms of revenue and subsequently, the set of candidate substrate nodes for the root node is being identified using the function "Candidate Substrate Node Preselection" in Algorithm 4. Any candidate node exceeding a maximum distance from all the peering nodes (computed based on the minimum distance and a relative threshold θ, which is

adjusted to 0.1 by default) is removed from the candidate set and is no longer considered for the placement of the root node.

In the lines 10–15, the algorithm checks whether the root node can be assigned to a substrate node within the candidate set based on the availability of substrate network paths (between this node and the peering nodes) satisfying the bandwidth demands. If this is feasible, the algorithm proceeds with the assignment of the following nodes of the VN requests as long as the CRR threshold is not violated (i.e., lines 16–37); otherwise, the assignment of the root node to the next candidate node is being checked (i.e., lines 9–12). The first step for the assignment of each additional virtual node is to identify the substrate node with the minimum embedding cost of all adjacent links, including the links attached to the peering nodes, (i.e., line 17) using the function "Candidate Substrate Node Selection" in Algorithm 5. The assignment is restricted to the substrate nodes whose distance (i.e., the number of hops) from the root node does not exceed a predefined threshold denoted by ζ. This essentially eliminates all inefficient solutions, i.e., substrate nodes that would significantly increase the CRR. We discuss the adjustment of the threshold ζ in Section 4.4.2. Furthermore, we take into account the link embedding cost by computing the corresponding capacity-constrained shortest path. Eventually, the most cost-efficient candidate node is being selected.

Upon the assignment of each virtual node j, the algorithm updates the CRR value and checks for any CRR threshold violations. Instead of making decisions based only on the current CRR value (i.e., CRR(j)), we further take into account the CRR value of the previous virtual node (i.e., CRR(j-1)) in order to capture any trends in the CRR evolution. Figure 26 illustrates the decision-making process with an example of a CRR curve, similar to Figure 23a. In particular, although $j = 1$ exceeds the CRR threshold, there is a decreasing trend in CRR, and therefore, the algorithm permits the assignment of this virtual node. A case of CRR threshold violation is represented by $j = 6$, where the current CRR value exceeds the threshold with an increasing trend. The conditions for the CRR threshold violation are shown in line 19.

If the last node of the VN request is reached and the corresponding mapping of this node is successful, then the current mapping will replace any previous solutions with higher CRR value (i.e., lines 22–28). The "else" branch (i.e., lines 29–36) is reached after the VN embedding becomes either unprofitable or infeasible with virtual node j. In this case, the VN subset of iteration $j - 1$ can be considered as a solution. The current mapping is considered as a better solution only if it is associated with a larger VN subset or if it yields a lower CRR compared to the previous solution. The final VNE solution is returned in M_{V_sol}.

Algorithm 3 Policy-compliant VNE

Input: $V_S, P, E_S, R_N, R_L, V_V, P', D_N, D_L, dist$

$j_{max} \leftarrow -1$, $M_{V\_sol} \leftarrow \emptyset$
$CRR = \{c_0, ..., c_{|V^V|-1} | c_i = \infty\}$, $CRR_{sol} \leftarrow CRR$
$R_{N,rollback} \leftarrow R_N$, $R_{L,rollback} \leftarrow R_L$
Sort the nodes $i \in V_V$ by descending revenue $\mathbb{R}(i)$
$C \leftarrow$ **Candidate Substrate Node Preselection (0)**
$dist_{P,min} \leftarrow min_{u \in C}(\sum_{p \in P} dist_{p,u})$
$dist_{P,max} \leftarrow dist_{P,min} \cdot (1 + \theta)$
$C \leftarrow C \setminus \{u \in C | \sum_{p \in P} dist_{p,u} > dist_{P,max}\}$
while $C \neq \emptyset$ **and** $j_{max} < |V_V| - 1$ **do**
  $M_V \leftarrow \emptyset$
  $u_{root} \leftarrow arg\ min_{u \in C}(\sum_{p \in P} dist_{p,u})$
  if Shortest paths with sufficient capacity exist between u_{root} and all $u \in P'$ **then**
    $M_{V,0} \leftarrow u_{root}$
    Update residual capacity for node u_{root} and for all links used between u_{root} and all $u \in P'$
    Compute $CRR(0)$
    for $j := 1..|V_V| - 1$ **do**
      $u_{cand} \leftarrow$ **Candidate Substrate Node Selection (j)**
      Compute $CRR(j)$
      if $u_{cand} \neq \emptyset$ **and** $(CRR(j) \leqslant CRR_{max}$ **or** $CRR(j-1) > CRR_{max})$ **then**
        $M_{V,j} \leftarrow u_{cand}$
        Update residual capacity for all links between u_{cand} and all nodes $u \in \{M_V, P'\}$
        if $j = |V_V| - 1$ **then**
          if $CRR(j) < CRR_{sol}(j)$ **then**
            $j_{max} \leftarrow j$
            $M_{V\_sol} \leftarrow M_V$
            $CRR_{sol} \leftarrow CRR$
          end if
        end if
      else
        if $CRR(j-1) \leqslant CRR_{max}$ **and** $(j - 1 > j_{max}$ **or** $CRR(j-1) < CRR_{sol}(j-1))$ **then**
          $j_{max} \leftarrow j - 1$
          $M_{V\_sol} \leftarrow M_V$
          $CRR_{sol} \leftarrow CRR$
        end if
        break: exit the for loop
      end if
    end for
  end if
  $C \leftarrow C \setminus u_{root}$
  $R_N \leftarrow R_{N,rollback}$, $R_L \leftarrow R_{L,rollback}$
end while
return M_{V_sol}

Algorithm 4 Candidate substrate node preselection

Input: i global: $V_S, R_N, R_L, V_V, D_N, D_L, M_V$

Find for the mapping of virtual node i all nodes out of V_S with sufficient node capacity. Exclude all nodes definitively having not sufficient link capacity and nodes that have already been used for mapping.

$N_{cand} \leftarrow \emptyset$
$d_{from} \leftarrow 0, \quad d_{to} \leftarrow 0$
for all $j \in V_V$ **do**
  if $M_{V,j} = \emptyset$ **then**
    $d_{from} \leftarrow d_{from} + d^{ij}$
    $d_{to} \leftarrow d_{to} + d^{ji}$
  end if
end for
for all $u' \in V_S$ **do**
  if $u' \notin M_V$ **then**
    if $d^i \leqslant r'_u$ **then**
      $r_{from} = \sum_{\{u,v\} \in E_S, u = u'} r_{uv}$
      $r_{to} = \sum_{\{u,v\} \in E_S, v = u'} r_{uv}$
      if $d_{from} \leqslant r_{from}$ **and** $d_{to} \leqslant r_{to}$ **then**
        $N_{cand} \leftarrow N_{cand} \cup u$
      end if
    end if
  end if
end for
return N_{cand}

Algorithm 5 Candidate substrate node selection

Input: j global: $V_S, R_L, V_V, P', D_L, M_V, u_{root}$

Find for the mapping of virtual node j the substrate node out of V_S, which requires minimum additional link embedding cost.

$u_{cand} \leftarrow \emptyset$
$S \leftarrow$ **Candidate Substrate Node Preselection (j)**
for all $u \in S$ **do**
  if $dist_{u_{root},u} \leqslant \zeta$ **then**
    $cost_u \leftarrow 0$
    for all $i \in \{V_V, P'\}$ **do**
      if $i \in P'$ **then**
        $v \leftarrow i$
      else
        $v \leftarrow M_{V,i}$
      end if
      if $v \neq \emptyset$ **then**
        if Shortest path with sufficient capacity exists between u and v **then**
          $cost_u \leftarrow cost_u + dist_{u,v} \cdot d^{ij}$
          $cost_u \leftarrow cost_u + dist_{v,u} \cdot d^{ji}$
        else
          $cost_u \leftarrow \infty$
          break: exit the for loop
        end if
      end if
    end for
  end if
end for
if $\min_{u \in S}(cost_u) \neq \infty$ **then**
  $u_{cand} \leftarrow \arg\min_{u \in S}(cost_u)$
end if
return u_{cand}

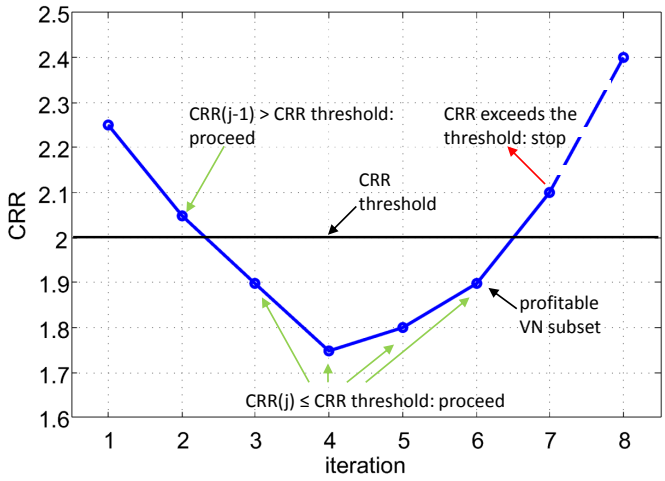


Figure 26: The algorithm increases stepwise the number of considered virtual nodes and compares the CRR with the CRR threshold as long as CRR increases. The algorithm terminates with the last feasible mapping result if CRR threshold is violated and CRR increased since the last iteration. A decreasing CRR results in a further iteration even if CRR threshold is violated.

4.4 EVALUATION

We use our C/C++ based VN embedding framework (as presented in Chapter 6) for VNE simulation.[4] We evaluate the efficiency of the proposed VNE algorithm against a variant of our algorithm that embeds only full VN requests (Section 4.4.2). We further shed light into VNE policy configuration by investigating the impact of diverse CRR threshold adjustments on revenue and VN request acceptance rate (Section 4.4.3).

4.4.1 *Parameters and Metrics*

Next, we explain the parameters of our substrate network and VN request models used for VNE simulation.

Substrate Network. *IGen* [102] was used to generate synthetic substrate network topologies for our simulations. We particularly ran our tests on a substrate network with 200 nodes and 400 links which are distributed based on the two-trees method [107]. We also designated eight substrate nodes for peering with other substrate providers. Initially, all substrate network resources are unutilized. The residual capacity of substrate nodes and links is updated after the embedding of a VN request.

4 Our tests are conducted on a server with two Intel Xeon quad-core CPUs at 2.53 GHz and 12 GB of main memory.

VN Request. A VN request consists of the CPU requirements for each virtual node and the bandwidth demands between all pairs of virtual nodes, represented as a traffic matrix. The traffic matrix further contains the bandwidth requirements between each virtual node and each peering node (i.e., three peering nodes are randomly assigned among the eight available nodes designated for peering). As such, we take into account the embedding cost of virtual links spanning multiple substrate providers. The number of virtual nodes per VN request is randomly sampled from a uniform distribution, between 10 and 30. In each simulation run, we generate and process a sequence of 1000 non-expiring VN requests, which gradually utilize most of the substrate network resources and allow to assess VNE efficiency under various network utilization levels. Each of the following evaluation results is based on 50 simulation runs.

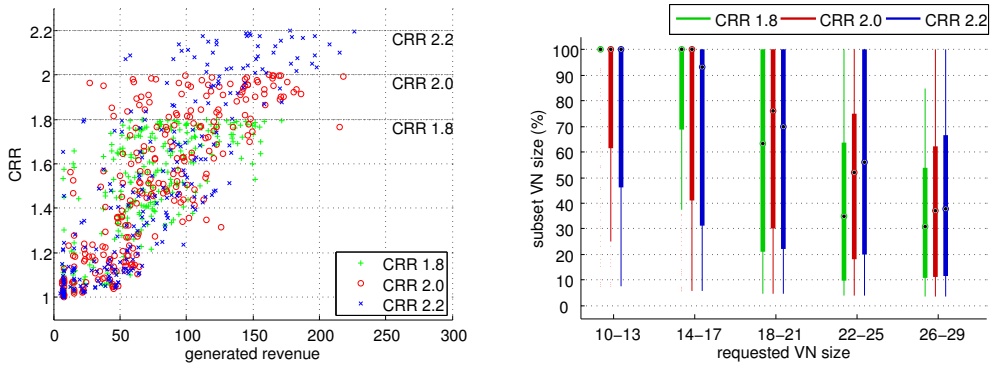
We use two main metrics for the evaluation of VNE efficiency:

- **Acceptance rate** is the number of successfully embedded requests over the total number of requests. Requests from which only a subset is embedded are considered as *accepted* as well.
- **Revenue** accumulates the CPU and bandwidth units leased to SPs.

4.4.2 *Results*

Initially, we discuss the efficiency of our VNE algorithm. Figure 27a illustrates the CRR versus the generated revenue with three CRR threshold adjustments (i. e., 1.8, 2.0, and 2.2). This scatter plot validates the operation of our algorithm, as in each case the CRR does not exceed the predefined threshold. We observe a correlation between the CRR and the generated revenue, i. e., embedding larger VNs incurs a penalty in terms of resource efficiency. More precisely, setting the CRR to 1.8 allows the embedding of a VN subset with revenue up to 180. Adjusting the CRR to 2.0 or 2.2 permits the embedding of VNs with larger revenue (i. e., up to 220). Furthermore, Figure 27b depicts the proportion of the embedded VN size over the VN request size with these CRR threshold adjustments. The results are classified into five different groups of VN request sizes. In many cases, small VN requests (i. e., 10-13 nodes) are fully embedded, especially for a CRR threshold of 1.8. For larger VN requests, only a subset is usually being embedded, and the relative subset size decreases as the VN request size becomes larger. As shown in the figure, lower CRR threshold adjustments result in embedding smaller VN subsets.

Next, we evaluate the efficiency of embedding subsets of VN requests. Figure 28a depicts the cumulative revenue generated by embedding full requests



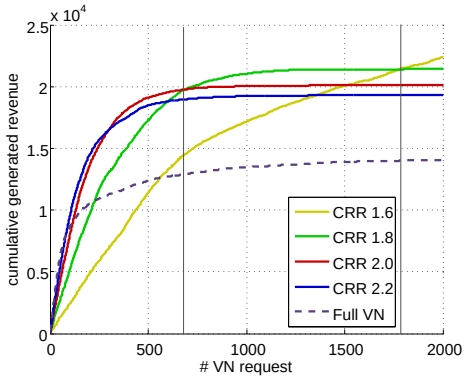
(a) correlation between CRR and generated revenue

(b) relative VN subset size vs. VN request sizes

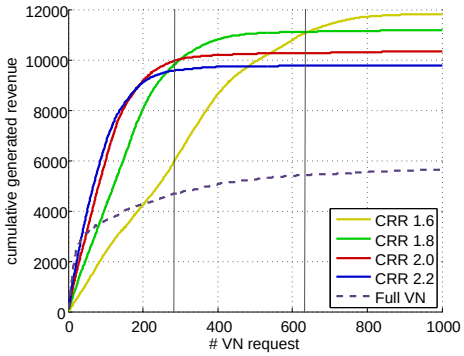
Figure 27: CRR thresholds hinder the embedding of the VNs with high potential revenue. Small VN requests are often embedded at full size while large VNs tend to be accepted at smaller VN subsets sizes. This trend is strengthened by decreasing CRR threshold.

and the most profitable subset of a VN request according to the CRR threshold adjustment. Our VNE algorithm generates much higher revenue compared to full VN embedding, since it embeds the VN request subsets that exhibits higher efficiency. In contrast, full VN embedding results in low revenue due to inefficient resource utilization, as shown in Figure 28c. According to this plot, for a given level of resource utilization, subset VN embedding generates more revenue, although it may require processing a larger number of requests compared to full VN embedding. Full VN embedding initially generates revenue at the maximum rate as it accepts all incoming requests as long as sufficient capacity is available but after the first 150 VN requests VN subset embedding with high CRR threshold values (i. e., 2.0 and 2.2) generates more revenue. Comparing the CRR threshold values, low threshold adjustments restrict revenue generation when the substrate network is underutilized, but achieve slightly more revenue for medium and high utilization levels and eventually generate more revenue for the provider in the long run. Essentially, a high CRR threshold value results in a greedier behavior, generating revenue faster which may be suited to providers that do not anticipate a large number of VN requests. On the other hand, a low CRR threshold value is deemed more beneficial for smaller substrate networks, in which the computing resources can be saturated with a smaller number of VN requests. Among the various CRR threshold adjustments, for 1000 requests, 1.8 achieves the highest revenue, since it exhibits less tolerance to resource inefficiencies. Our simulation logs indicate that the resource utilization penalty for full VN embedding stems from the increased virtual link hop count.

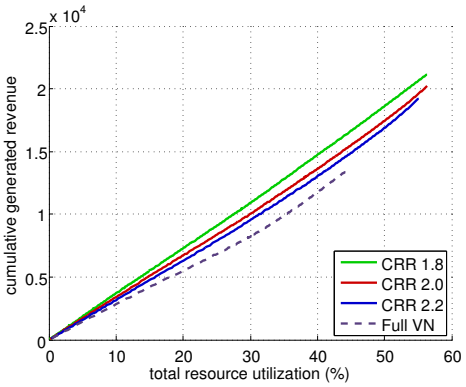
Furthermore, Figure 28d shows the VN request acceptance rate with full and subset VN embedding. The acceptance rate of full VN requests drops quickly, due to the inefficient resource utilization. Depending on the CRR threshold adjustment, our algorithm rejects the VN requests that are not profitable, even if the substrate network is not saturated. However, in the long run, embedding VN subsets leads to a higher acceptance rate.



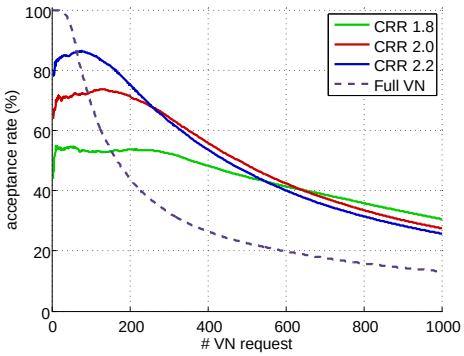
(a) cumulative generated revenue with 200 substrate nodes.



(b) cumulative generated revenue with 100 substrate nodes.



(c) generated revenue vs. resource utilization



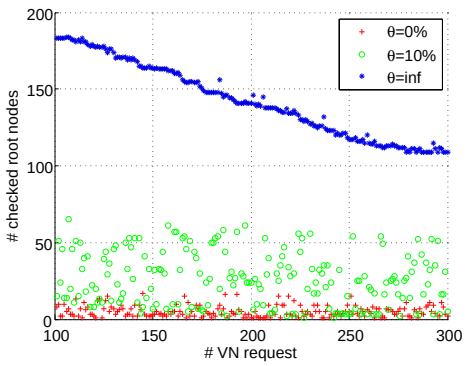
(d) VN request acceptance rate

Figure 28: Depending on the substrate network size and the number of arriving VN requests, a different CRR threshold yields the maximum revenue. In an exemplary substrate network of 200 nodes, a CRR threshold of 1.8 yields the most profitable VN embeddings if around 700 to 1800 VN requests are directed to the InP. These numbers vary for different substrate networks. For example, if 1000 VN requests arrive in a smaller substrate network of 100 nodes then the CRR threshold should be decreased from 1.8 to 1.6 to generate more revenue. It is further shown that lower CRR thresholds yield higher resource efficiency which in turn translates into more revenue gain from an equal amount of physical resources. In addition, full VN embedding greedily embeds all incoming requests while subset VN embedding rejects unprofitable VN requests at the beginning. VN embedding with CRR threshold benefits in the long run from those saved resources as it finally accepts more VN requests.

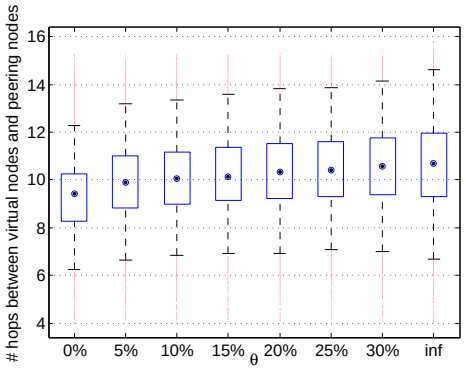
4.4.3 *Parameter Adjustments*

Finally, we discuss the impact of substrate network size and the adjustment of the algorithm parameters θ and ζ. The already described Figure 28a goes beyond the originally considered 1000 requests in order to show the effect of even smaller CRR thresholds – here 1.6 achieves even higher revenue with 2000 requests. Depending on the number of arriving requests, the CRR threshold needs to be adjusted accordingly. Out of the shown CRR thresholds, 1.8 yields the highest revenue in the range between around 700 and 1800 VN requests, as highlighted by the two vertical lines. This range is shifted to around 300 to 600 VN requests when a smaller 100-node substrate network is used (Figure 28b).

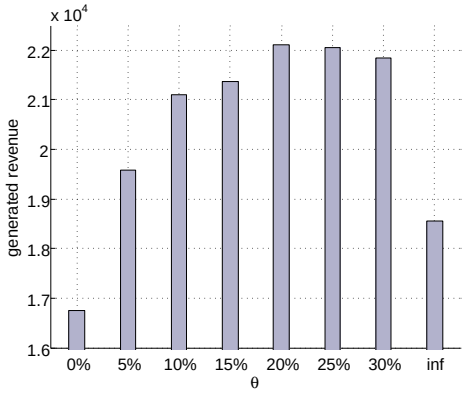
The threshold θ restricts the distance from root node candidates to the potential peering nodes. Figure 29a shows for the first 300 requests the number of potential substrate nodes considered for any root node. Among the two extremes to consider either only the candidates with the shortest distance to the peering nodes ($\theta = 0$) or to accept all candidates ($\theta = \infty$), a θ of 10% yields a reasonable candidate set with a size between one and around one quart of the available substrate nodes. A higher θ can directly be translated into longer paths for the domain-crossing virtual links. For 50% of these virtual links, 9 to 12 hops are typically required for the embedding in the substrate topology while a strict θ threshold results in 8 to 10 hops for each virtual links (Figure 29b). Reducing hop count for multi-domain virtual links should not be the only aim of θ adjustment as it increases the risk of rejections due to resource shortages incurred by the restricted search space. Instead, a moderate θ of 10-30% results in higher cumulative revenue, in the long run, as illustrated in Figure 29c. As shown in the plot, adjusting θ to a value greater than 10% does not lead to notable revenue gains, while it increases the solver runtime. As such, we use 10% as the default value for θ.



(a) checked root nodes

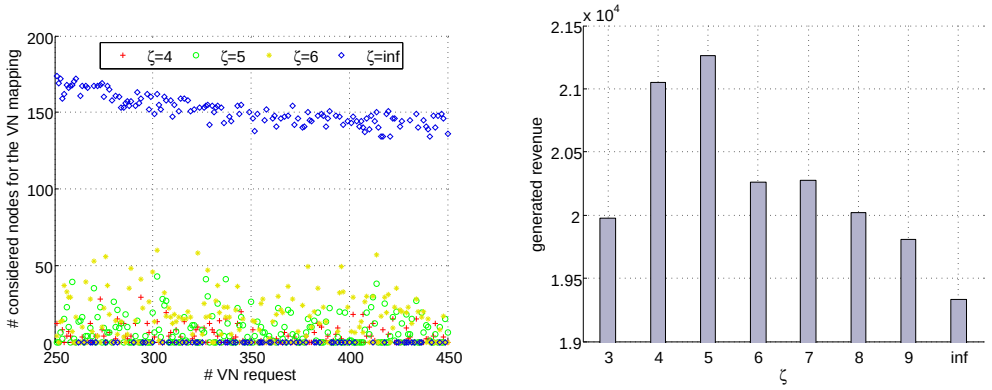


(b) hop count per multi-provider virtual link



(c) generated revenue

Figure 29: θ adjustments increase or decrease the number checked root nodes and the number of hops per multi-provider virtual link. These adjustments finally impact the generable revenue where $\theta = 20\%$ achieves the maximum revenue.



(a) locally considered candidate nodes

(b) generated revenue

Figure 30: ζ adjustments increase or decrease the number of locally considered candidate nodes which in turn impacts the generable revenue. These adjustments finally aim at maximizing revenue – here achieved with $\zeta = 5$.

The other threshold ζ restricts the distance between the hosts of the root node and the other virtual nodes each after a root node has been assigned to a substrate node. In contrast to θ where only those virtual links that cross the domain border are taken into account, ζ impacts only the embedding of the other group of virtual links that are completely embedded in the current domain. ζ drastically reduces the number of considered candidates (Figure 30a) while cumulative revenue, in the long run, is generally higher than without such a threshold ($\zeta = \infty$, Figure 30b). Among the different ζ adjustments, $\zeta = 5$ generates the highest revenue in the used substrate topology.

Although we used static CRR threshold values in our simulations, providers are envisaged to apply policies that require the dynamic adjustment of the CRR threshold depending on the network utilization and resource demand. Similarly, the algorithm parameters θ and ζ need to be readjusted for the different substrate topologies. The insights gained from our simulation results can be useful for the specification of such VNE policies.

4.5 RELATED WORK

A large body of work on embedding VN requests onto a substrate network exist (e. g., [67, 70, 78, 79, 84]). [5] These VNE techniques aim at optimizing the mapping of VN topologies and always embed full VN requests when this is feasible. As opposed to these techniques that ignore the policies of substrate providers, we take a different approach by tailoring VNE to the provider's policy.

5 Section 2.5.2 provides an overview of existing VNE algorithms.

In particular, our VNE algorithm restricts the solution space according to the policy.

Our approach can be used in the multi-provider VNE architectures introduced in Section 2.4. In this regard, substrate providers can benefit from our policy-compliant VNE algorithm. In PolyViNE [62], a substrate provider can use our VNE algorithm to identify and embed the most profitable subset of a VN request. Similarly, in auction-based VNE environments such as V-Mart [60], our algorithm can assist a provider in adjusting his bid for the embedding of VN requests.

5

MULTI-PROVIDER SERVICE CHAIN EMBEDDING

In Chapter 3, we present our centralized multi-provider VN embedding framework and investigate the impact of limited information disclosure of the InPs on VN embedding efficiency [108]. As a next step, we take into account network service chains comprised of network functions (NFs) which we embed into multiple data centers (DCs) again due to location dependencies of the *NF providers (NFPs)*. In this context, we introduce a service model that facilitates the specification of service chain requirements by the clients, e. g., enterprises (Section 5.1). Since NFP policies will hinder interoperability with third parties and especially competitors, we consider a centralized coordinator (i. e., *network service composition layer, NSCL*), between the requester and the NFPs, that relies on an abstract network view (Section 5.2). In this respect, we provide a framework for multi-provider network service embedding (NSE) in which a centralized coordinator partitions the requested service chain into NF subsets each to be finally embedded into the substrate by the assigned NFPs (Section 5.3). According to that, we present MILP and LP based approaches for solving the request partitioning problem (Section 5.4) and the resource mapping problem (Section 5.5). Finally, we evaluate NSE efficiency with different objectives and make a comparison with a greedy variant (Section 5.6).

5.1 SERVICE MODEL

We aim at defining a service model that simplifies the specification of service requirements by clients and the estimation of NF computing and bandwidth demands. The difficulty in computing resource requirements stems from the effects of NFs on traffic. More precisely, appliances, such as redundancy eliminators and caches, conserve bandwidth, while other NFs (e. g., packet multiplication, encryption) amplify traffic. The level of bandwidth conservation or traffic amplification depends on various factors, such as the size and hit ratio for caches, the amount of duplicate content for redundancy elimination appliances, and the volume of traffic filtered by firewalls and intrusion detection systems (IDS). In this respect, Table 7 summarizes the effect of widely-used NFs on traffic and further shows the range of bandwidth saving or traffic amplification for each NF, collected from various studies [109–111]. Based on these observations, we introduce φ_p^i which denotes the ratio of outbound traffic at port p of NF i

Table 7: Some network functions amplify or reduce traffic. The table shows network functions with confirmed range of outbound/inbound traffic ratio.

Network function	**Bandwidth preservation**	**Outbound/inbound traffic ratio (φ)**
Flow monitoring	Yes	-
Load balancer	Yes	-
NAT	Yes	-
RE	No	40–70% [109], 59–74% [110]
VPN (IPsec)	No	105–228% (for 64–1500-byte packets) [111]

over the aggregate inbound traffic at all ports. We particularly consider the traffic ratio per output port, since traffic may be split between multiple output ports depending on the outcome of packet inspection.

Our network service model consists of an NF graph in which each NF i is associated with a traffic ratio φ_p^i per port p (Figure 31). Essentially, φ_p^i is used for the estimation of the bandwidth demand over each link, given the aggregate inbound traffic rate at each NF. The adjustment of φ_p^i for a given NF can be derived based on traffic statistics from middleboxes with the same functionality, deployed on the client's premises. In case such information is not available, φ_p^i can be adjusted based on statistics available from middlebox studies [109–111] or other network operators. Since achieving a very accurate estimation of φ_p^i may be difficult, φ_p^i can be set to the lowest bandwidth saving or the highest level of traffic amplification (assuming a known range of bandwidth saving or traffic amplification, as shown in Table 7). This approach ensures that bandwidth allocation will be sufficient while any spare bandwidth can be distributed proportionally to the clients. After φ_p^i has been adjusted for each NF in the service chain, the client simply needs to specify the rate of the traffic generated at each endpoint.

The computational requirements for each NF can be derived using the inbound traffic rate and the resource profile of each NF (i. e., CPU cycles per packet). Resource profiles are available for a wide range of NFs [112, 113] while existing profiling techniques [114] can be applied to any flow processing workloads whose computational requirements are not known. This obviates the need to specify any computing demands for the NFs in the service chain.

Next, we formulate a **service chain request model** which considers the service descriptions above. We use a directed graph $G_F = (V_F, E_F)$ to express a network service request. The set of vertices V_F includes all NFs and endpoints that comprise the service request. Each NF i is associated with an outbound-/inbound traffic ratio per port p, denoted by φ_p^i. Each endpoint is associated

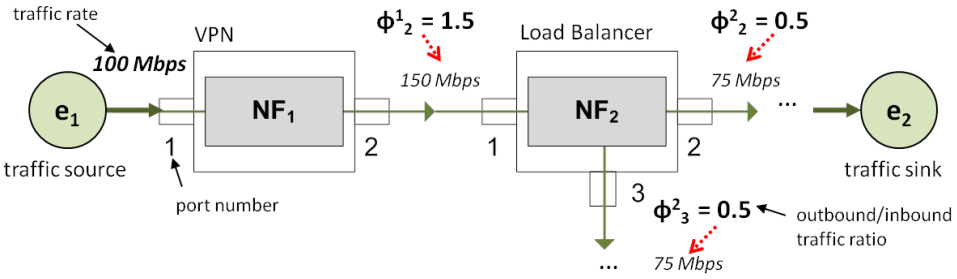


Figure 31: The service chain can be represented as a graph with φ^i_p denoting the outbound to inbound traffic ratio at the port p of the NF i. The traffic rate on each link in this NF graph can then be determined in dependence on the source traffic rate.

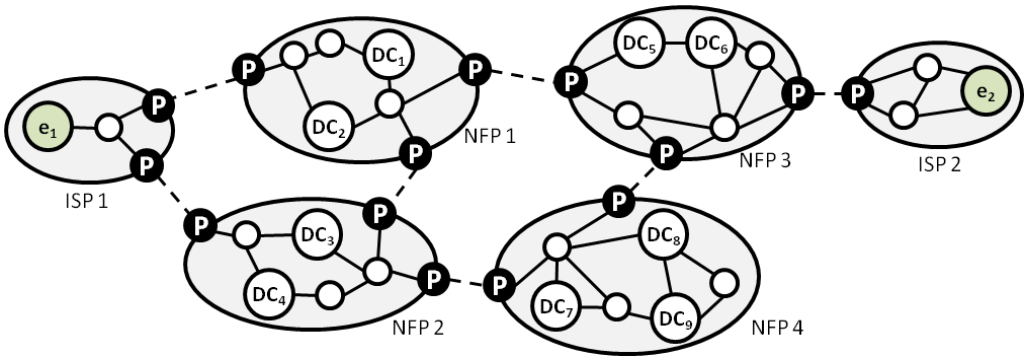
with a traffic generation rate, which, combined with φ^i_p, gives the bandwidth demand d^{ij} for each edge $(i,j) \in E_F$. The computing demand d^i of each NF is estimated based on the inbound traffic rate and the NF resource profile (i. e., CPU cycles per packet). NF location dependencies are expressed by the distance tolerance, denoted by λ^i. This request model is used for request partitioning and mapping in the Sections 5.4 and 5.5.

5.2 TOPOLOGY ABSTRACTIONS AND SUBSTRATE NETWORK MODEL

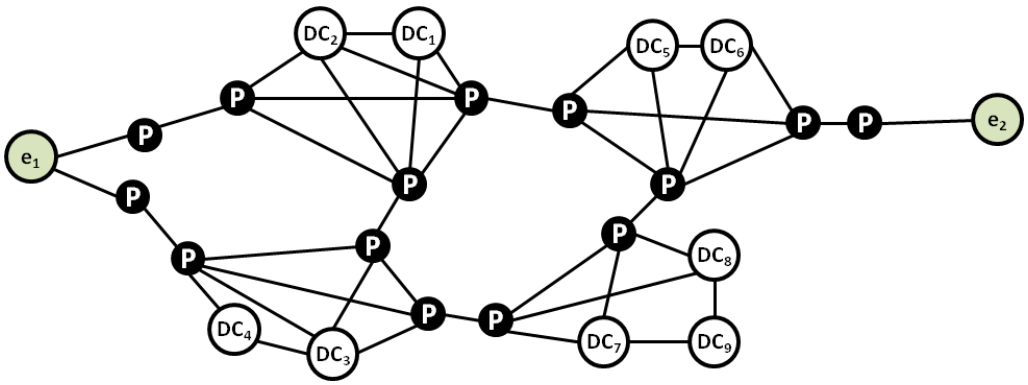
Topology abstractions are crucial for the generation of efficient embeddings considering the information disclosure policies of NFPs. We seek to identify topology abstractions that conceal any information deemed as confidential by NFPs. To this end, we rely on information disclosed by ISPs (i. e., InPs) and cloud providers. ISPs often publish simplified PoP-level topologies [90], while cloud providers advertise resource types across different availability zones [89].

We depart from a PoP-level topology view that includes the Internet access points, NFV PoPs (i. e., DCs), and peerings with neighboring networks. Figure 32a depicts such a topology spanning four NFPs and two ISPs. Since the endpoints (i. e., e_1, e_2) are fixed, we need a network view that simplifies the estimation of the link costs between the endpoints and the DCs. Based on Figure 32a, we derive an abstract network view that obscures the Internet access points and represents (i) the connectivity between DCs and peering nodes within each NFP, and (ii) the peerings among NFPs (Figure 32b). This topology abstraction combined with NF computing and link costs provides the necessary means for the estimation of the overall NSE cost.

The edges of this network graph can be annotated with weights assigned by each NFP, according to the NFP's policies (e. g., load balancing), similarly



(a) PoP-level topology



(b) Topology abstraction

Figure 32: An abstract network view obscures the Internet access points and simplifies the estimation of the link costs between the endpoints of a service chain. These link costs are considered for the service chain partitioning.

to the multi exit discriminator (MED) attribute of the border gateway protocol (BGP). An NFP may wish to incorporate DC utilizations into the weights of the adjacent links, avoiding the explicit advertising of DC utilization information. We particularly consider a link weight offset which is dynamically adjusted according to the DC utilization level. Link weights are used by our request partitioning formulation variant which is tailored to NFPs (Section 5.4).

In the following, we formally describe a **substrate network model** which facilitates the abstractions as discussed beforehand. We rely on an undirected graph $G_S = (V_S, E_S)$ for the description of topology abstractions and substrate network topologies. We use α_u and β_{uv} to express the monetary cost of NFs and links, respectively. Each graph edge $(u, v) \in E_S$ is associated with a weight, denoted by w_{uv}, which is assigned by the NFP. Furthermore, substrate nodes and links are associated with their residual capacity, represented by r_u and r_{uv}, respectively. To enforce NF location constraints, we further introduce l_u^i which represents the distance between the preferred location (e. g., close to an endpoint) and the DC u assigned to NF i. This substrate network model is used for request partitioning and mapping in the Sections 5.4 and 5.5.

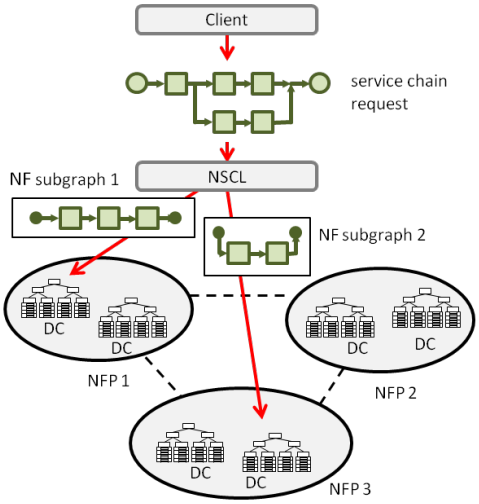


Figure 33: Our NSE control plane is distributed across the NSCL, the NF providers, and the DCs deployed by each NFP. Service chain requests are partitioned by the NSCL across multiple NFPs (i. e., DCs) that in turn assign resources to the NF subgraphs of the service chain.

5.3 SERVICE CHAIN EMBEDDING FRAMEWORK

In this section, we provide an overview of our NSE framework and discuss the sequence of steps for the embedding of network service requests. Our NSE framework processes and embeds requests specified based on the service model presented in Section 5.1. The topology abstraction in Section 5.2 represents the view of the NSCL on the substrate network topologies. To embed network service requests, we use our NSE control plane, which is distributed across the NSCL, the NFPs, and the DCs deployed by each NFP, as shown in Figure 33. Along these lines, our NSE framework decomposes NSE into the following steps:

Graph Rendering. Graph rendering consists in the transformation of detailed topology graphs into topology abstractions that facilitate request partitioning while obscuring any confidential information for NFPs. Each NFP generates the topology abstraction for his own network and subsequently annotates the edges of the graph with the link costs (i. e., expressed as cost per bandwidth unit) and optionally with weights representing link and DC preferences. The NSCL collects the graphs from all participating NFPs and stitches them together constructing an abstract network view that spans all NFPs (i. e., Figure 32b). New topology abstractions are generated upon significant substrate topology changes or the participation of new NFPs. Link weights are updated on the existing network graphs in response to changes in resource utilization levels or NFP policies.

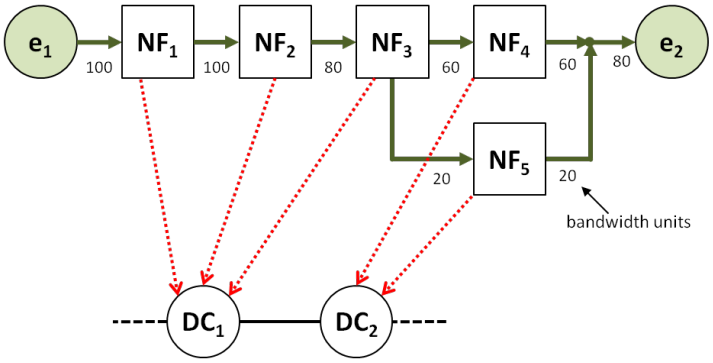
Service Chain Request Partitioning. Network service requests are partitioned among NFPs when there is no single NFP that satisfies the location dependen-

cies of all NFs in the request. More precisely, the NSCL identifies a list of DC candidates for each requested NF by matching NF location constraints against each NFP's footprint. Subsequently, the NSCL uses two variants of an ILP/LP model for request partitioning, tailored to (i) the client (i. e., expenditure minimization) or (ii) the NFPs (i. e., network load balancing), exploiting link and DC preferences disclosed by NFPs. The request partitioning formulations are discussed in detail in Sections 5.4.1 and 5.4.2.

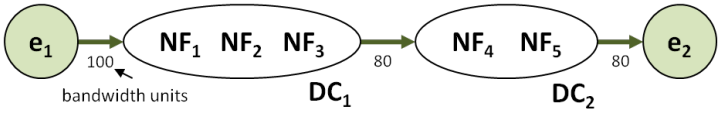
The request segments are derived from the ILP/LP solver output, i. e., the NF-to-DC assignment (Figure 34a). First, the NSCL computes the total inbound and outbound bandwidth demand for each request segment (Figure 34b). Next, the NSCL generates an NF subgraph, in which all inter-segment traffic traverses a virtual gateway (VGW), as shown in Figure 34c. This subgraph allows the binding of the VGW with the DC network gateway, augmenting the mapping of each request segment onto the assigned DC.

NF Subgraph Mapping. Each NF subgraph is mapped onto the assigned DC network by the corresponding NFP. This process does not require any topology abstractions, since each NFP has a complete view of the DC network topologies and the utilization of servers and links. We particularly consider 2-level hierarchical DC network topologies (similar to Figure 2, Section 2.1.2) that provide sufficient capacity for data transfers between the few hundreds of servers deployed within each micro-DC. Nevertheless, our NF subgraph mapping methods are also applicable to 3-layer fat-tree topologies, used for larger DCs.

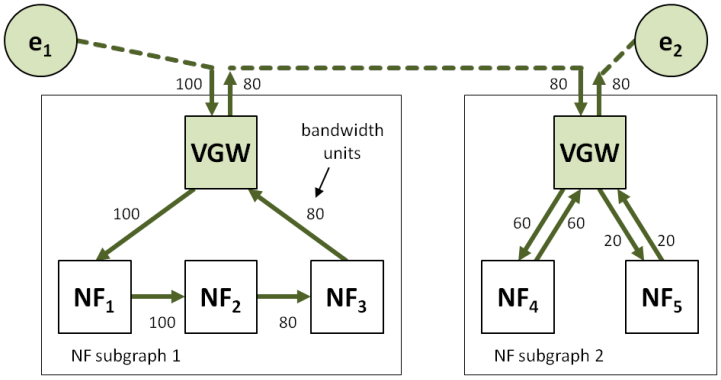
We also employ an MILP/LP solver for the assignment of NF subgraphs to DC networks – here with the combined objective of NF consolidation level maximization and inter-rack traffic minimization. Further details on these NF subgraph mapping methods are given in Sections 5.5.1 and 5.5.2.



(a) Assignment of NFs to DCs



(b) Bandwidth demand aggregation



(c) Generation of subgraphs with virtual gateways (VGW)

Figure 34: Request partitioning comprises sub-steps that finally provide reformulated NF subgraphs that are subsequently sent as new requests to each corresponding NFP (i. e., DC).

Table 8: Notations used in Sections 5.4 and 5.5.

Symbol	Description
α_u	monetary server cost at DC u in \$/GHz
β_{uv}	monetary cost of link (u, v) in \$/Mbps
d^i	computing capacity demand of NF i in GHz
d^{ij}	bandwidth demand of edge (i, j) in Mbps
E_F	set of NF graph edges
E_S	set of substrate links
f_{uv}^{ij}	flow demand of edge (i, j) assigned to the intra-DC link (u, v) in Mbps
φ_p^i	outbound/inbound traffic ratio per port p for NF i
i_{fx}	NF to be fixed to a certain node
l_u^i	distance between the preferred location and the DC u assigned to NF i in km
λ^i	distance tolerance of NF i in km
r_u	residual capacity of server u in GHz
r_{uv}	residual capacity of link (u, v) in Mbps
u_{fx}	node to be fixed to a certain NF
V_F	set of NFs
V_S	set of substrate nodes or DCs
w_{uv}	weight of link (u, v)
X	set of x_u^i
x_u^i	assignment of NF i to DC or server u
y_{uv}^{ij}	mapping of NF graph edge (i, j) onto PoP-level graph edge (u, v)
z_u	assignment of any NF to server u

5.4 SERVICE CHAIN PARTITIONING

In this section, we present an ILP formulation for request partitioning (Section 5.4.1) which is complemented by a relaxed LP variant for the faster computation of near-optimal solutions (Section 5.4.2). We use the request and network models introduced in the Sections 5.1 and 5.2. The notations used in this Section are listed in Table 8.

5.4.1 *ILP Model*

Request partitioning will be subject to objectives, such as service cost minimization or network load balancing. In this respect, we provide two ILP formulation variants tailored to the client and the NFPs. The ILP variants differ only in the objective function. The objective function **Min-C** minimizes the overall mon-

etary cost for the client, by accumulating all the monetary NF and link costs. On the other hand, the objective function **Min-W** minimizes the link weights, disclosed by NFPs. Since link weights express the utilization of links and DCs, link weight minimization essentially leads to network load balancing.

In the ILP formulations, we use the binary variable x_u^i to express the assignment of NF i to the DC u. Similarly, the binary variable y_{uv}^{ij} indicates whether the NF graph edge $(i,j) \in E_F$ has been mapped onto the PoP-level graph edge $(u,v) \in E_S$. The request partitioning ILP is defined as follows:

Min-C:

$$\text{Minimize} \sum_{u \in V_S} \alpha_u \sum_{i \in V_F} d^i x_u^i + \sum_{\substack{(u,v) \in E_S \\ (u \neq v)}} \beta_{uv} \sum_{(i,j) \in E_F} d^{ij} y_{uv}^{ij} \tag{29}$$

OR

Min-W:

$$\text{Minimize} \sum_{\substack{(u,v) \in E_S \\ (u \neq v)}} w_{uv} \sum_{(i,j) \in E_F} d^{ij} y_{uv}^{ij} \tag{30}$$

subject to:

$$\sum_{u \in V_S} x_u^i = 1 \quad \forall i \in V_F \tag{31}$$

$$\sum_{\substack{v \in V_S \\ (u \neq v)}} (y_{uv}^{ij} - y_{vu}^{ij}) = x_u^i - x_u^j$$

$$i \neq j, \forall (i,j) \in E_F, \forall u \in V_S \tag{32}$$

$$l_u^i x_u^i \leqslant \lambda^i \quad \forall i \in V_F, \forall u \in V_S \tag{33}$$

$$x_u^i \in \{0,1\} \quad \forall i \in V_F, \forall u \in V_S \tag{34}$$

$$y_{uv}^{ij} \in \{0,1\} \quad \forall (i,j) \in E_F, \forall (u,v) \in E_S \tag{35}$$

Hereby, we briefly discuss the ILP constraints. Constraint (31) ensures that each NF i is mapped exactly to one DC. Condition (32) preserves the binding between the NF and the link assignments. More precisely, this condition ensures that for a given pair of assigned nodes i, j (i. e., NFs or endpoints), there is a path in the network graph where the edge (i,j) has been mapped. Condition (33) enforces NF location constraints. Finally, the conditions (34) and (35) express the binary domain constraints for the variables x_u^i and y_{uv}^{ij}. In addition, we fix

the assignment of each endpoint k in the request to its respective location u by setting $x_u^k \leftarrow 1$.

We rely on the branch-and-cut method for solving the ILPs. The request partitioning ILP solver yields a mean runtime of 210 ms (with the evaluation parameters shown in Table 9). Time complexity and solver runtime can be reduced by employing relaxation and rounding techniques at the cost of suboptimality. We apply such relaxation and rounding techniques in the following Section 5.4.2.

5.4.2 *LP Relaxation and Rounding*

We transform the request partitioning ILP model to an LP model by relaxing the integer domain constraints. We replace the Equations (34) and (35) by:

$$x_u^i \geqslant 0 \quad \forall i \in V_F, \forall u \in V_S \tag{36}$$

$$y_{uv}^{ij} \geqslant 0 \quad \forall (i,j) \in E_F, \forall (u,v) \in E_S \tag{37}$$

Finally, the request partitioning LP formulation consists of the objective functions (Equations (29) and (30)) and the constraints from the original ILP (Equations (31) – (33)) plus the above-mentioned non-integer domain constraints (Equations (36) and (37)).

Relaxation of integer variables might require further changes to the LP. In particular, Constraint (33) will not hold if x_u^i is too small since the actual geo distance to a candidate could appear too close due to multiplication with any $x_u^i < 1$. We decided not to extend the LP model and rather to exclude those infeasible solutions afterwards during the rounding phase (Algorithm 6, Lines 6 - 10). Constraint (32) holds even with non-binary x_u^i and y_{uv}^{ij}. The relaxed domain constraint (36) together with the Constraint (31) forces $0 \leqslant x_u^i \leqslant 1$ where the non-binary solutions are not yet associated with a final decision for or against any node candidate. Instead, these values express uncertainty due to the relaxed LP model.

In the following we present a rounding algorithm for the LP based generation of near-optimal solutions for the request partitioning problem. Algorithm 6 iteratively fixes (or excludes) a node mapping which is probably part of a near-optimal solution (i. e., $\max x_u^i$) if the respective geo constraint is not violated (or violated) as long as the LP solution after each iteration is feasible and contains non-binaries.

We now compare the efficiency of the original ILP variant and the LP variant that uses a rounding algorithm. To this end, we partition 25K service chains

Algorithm 6 Service chain partitioning with LP

repeat
  $\{x_u^i, y_{uv}^{ij}\} \leftarrow$ **Solve_LP(..)**
  $X \leftarrow \{x_u^i \mid x_u^i \notin \{0,1\}\}$
  if $X \neq \emptyset$ **then**
    $\{i_{fx}, u_{fx}\} \leftarrow argmax_{\{i \in V_F, u \in V_S\}} X$
    if $l_{u_{fx}}^{i_{fx}} \leqslant \lambda^i$ **then**
      Add_LP_Constraint("$x_{u_{fx}}^{i_{fx}} = 1$")
    else
      Add_LP_Constraint("$x_{u_{fx}}^{i_{fx}} = 0$")
    end if
  end if
until $(X = \emptyset) \vee$ NoFeasibleSolutionLP
return $\{x_u^i, y_{uv}^{ij}\}$

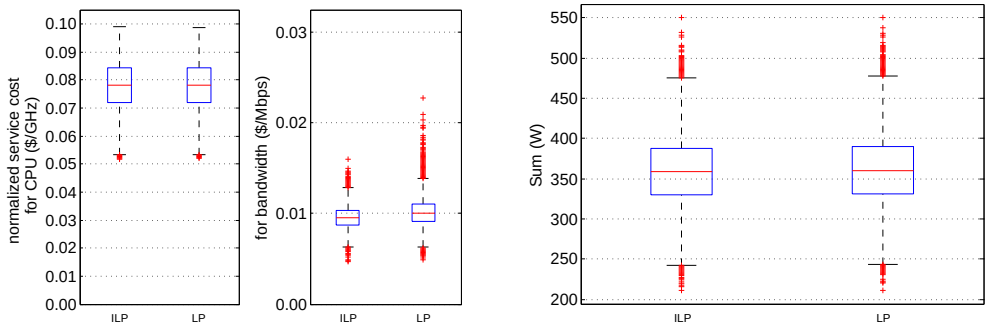
across 50 DCs, using both the ILP and LP variant. Further evaluation parameters used for this test are the same with our NSE evaluation in Section 5.6.1 (see Table 9). We run tests with both, the *Min-C* and the *Min-W* objective function. Figure 35a shows the normalized resource unit costs (i. e., for CPU, bandwidth) after the partitioning with *Min-C*.

There is no significant increase in CPU cost for the LP variant while bandwidth cost shows suboptimality in particular for the links which are generally more expensive. In total, LP yields 4.4% higher bandwidth cost compared to the ILP variant. The suboptimality for LP relaxation for the other objective (*Min-W*) is shown in Figure 35b. The weight update function, as used for *Min-W* is deactivated for a fair comparison. We determine a mean sum of weights per service chain of 358.5 and 360.5 for the ILP and LP variant. Eventually, the LP yields only marginal suboptimality compared to the ILP while the LP exhibits a substantially lower runtime (165 ms) compared to the ILP (944 ms)[1], which we deem that outweighs its suboptimality.

5.5 NF SUBGRAPH MAPPING

In this section, we formulate an MILP model for the mapping of NF subgraphs onto DC networks (Section 5.5.1). We further transform this model into an LP model by relaxing the integer variables and present the respective rounding algorithm (Section 5.5.2). The notations used in this Section are listed in Table 8.

1 Tests are carried out on a server with 2.53 GHz Intel Xeon CPU where each test is assigned to a single CPU core.



(a) Min-C: normalized resource cost

(b) Min-W: weight sum

Figure 35: In the case of cost-minimized service chain partitioning, LP results in slightly higher bandwidth costs while compute costs do not significantly differ from the ILP results. Similarly, LP yields only marginal suboptimality in terms of weight sum compared to the ILP variant under weight-minimized partitioning.

5.5.1 *MILP Model*

The MILP for NF subgraph mapping aims at maximizing NF co-location, i. e., minimizing the number of used servers while minimizing the traffic within the DC. In this respect, the binary variable z_u indicates whether any NF has been assigned to server u, i. e., $z_u = 1$ if server u is used and $z_u = 0$ otherwise. z_u depends on another variable x_u^i which denotes the assignment of NF i to server u. x_u^i is not relevant for the objective function as its sum is always constant. Instead, it will be implicitly bound to z_u in Equation (41). Based on the multi-commodity flow problem formulation, we use the term "commodity", defined as $Com^{ij} = \{i, j, d^{ij}\}$, to express bandwidth demands d^{ij} between a pair of NFs i, j. In this context, the flow variable f_{uv}^{ij} denotes the amount of flow (i. e., bandwidth units) over the DC link (u, v) for the NF graph edge $(i, j) \in E_F$. The objective function (38) consists of two terms, i. e., the number of assigned servers and the accumulated flow divided by the total bandwidth demand. Essentially, the second term yields one if all NF graph edges $(i, j) \in E_F$ are mapped onto single-hop links. The normalization of the second term provides a balance against the first term in the objective function. The NF subgraph mapping MILP model is formulated as follows:

Minimize

$$\sum_{u \in V_S} z_u + \frac{1}{\sum\limits_{(i,j) \in E_F} d^{ij}} \cdot \sum_{\substack{(u,v) \in E_S \\ (u \neq v)}} \sum_{(i,j) \in E_F} f_{uv}^{ij} \tag{38}$$

subject to:

$$\sum_{u \in V_S} x_u^i = 1 \quad \forall i \in V_F \tag{39}$$

$$\sum_{v \in V_S} f_{uv}^{ij} - \sum_{v \in V_S} f_{vu}^{ij} = d^{ij}(x_u^i - x_u^j)$$
$$i \neq j, \forall i, j \in V_F, u \neq v, \forall u \in V_S \tag{40}$$

$$\sum_{i \in V_F} d^i x_u^i \leqslant r_u \cdot z_u \quad \forall u \in V_S \tag{41}$$

$$\sum_{i,j \in V_F} f_{uv}^{ij} \leqslant r_{uv} \quad \forall (u,v) \in E_S \tag{42}$$

$$x_u^i \in \{0,1\} \quad \forall i \in V_F, \forall u \in V_S \tag{43}$$

$$z_u \in \{0,1\} \quad \forall u \in V_S \tag{44}$$

$$f_{uv}^{ij} \geqslant 0 \quad \forall (i,j) \in E_F, \forall (u,v) \in E_S \tag{45}$$

We further discuss the constraints (39)–(45) in the MILP. Condition (39) ensures that each NF $i \in V_F$ is mapped exactly to one server. Constraint (40) enforces flow conservation, i. e., the sum of all inbound and outbound traffic in switches and servers that do not host NFs should be zero.

The constraints (41) and (42) ensure that the allocated computing and bandwidth resources do not exceed the residual capacities of servers and links, respectively. Equation (41) is further used for the binding between the two binary variables z_u and x_u^i. The allocated capacity equals the demand $d^i x_u^i = d^i$ if NF i is not mapped to server u or zero otherwise. Similarly, the residual capacity is multiplied by z_u a variable which tends to zero as it is part of the objective function. If the left-hand side of Equation (41) is greater than zero, then the binary z_u on the right-hand side of the same equation is forced to 1. Conversely,

allocated capacity cannot exceed the residual capacity of any combination of NF i and server u.

Finally, the conditions (43) and (44) express the binary domain constraints for the variables x_u^i and z_u, while constraint (45) ensures that the flows f_{uv}^{ij} are always positive. We further assume that the first element in V_F represents the virtual gateway which we bind to the physical gateway GW by setting $x_{GW}^{V_F(1)} \leftarrow 1$.

5.5.2 *LP Relaxation and Rounding*

We transform the NF graph mapping MILP model to an LP model by relaxing the integer domain constraints. We replace Equations (43) and (44) by:

$$x_u^i \geqslant 0 \quad \forall i \in V_F, \forall u \in V_S \tag{46}$$

$$z_u \geqslant 0 \quad \forall u \in V_S \tag{47}$$

In the following, we discuss the impact of non-binary x_u^i and z_u on the model. Constraints (39) and (40) contain x_u^i where a non-boolean value represents the infeasible splitting of an NF to server mapping. However, Equation (39) implicitly enforces $x_u^i \leqslant 1$, and the binding between x_u^i and f_{uv}^{ij} (Equation (40)) impacts the values in x_u^i in the way that higher values rather indicate preferred (w.r.t. objective function) mapping combinations. Thus, suboptimal mapping results can be obtained from rounded x_u^i. The node capacity constraint (41) could be violated if $x_u^i < 1$ or if $z_u > 1$. On the contrary, with $z_u < 1$ nodes with sufficient capacity could be ignored which yields merely suboptimal but feasible combinations. Hence, we prefer rounding of x_u^i over z_u and modify the domain constraint of z_u (Equation (47)) to

$$0 \leqslant z_u \leqslant 1 \qquad \forall u \in V_S \tag{48}$$

while the additional constraint

$$z_u \geqslant x_u^i \quad \forall i \in V_F, \forall u \in V_S \tag{49}$$

yields again binary values for z_u after the rounding of all corresponding x_u^i. The latter is needed in order to accumulate the correct number of servers in the objective function.

Altogether, the final LP model is expressed as follows:

Minimize

$$\sum_{u \in V_S} z_u + \frac{1}{\sum_{(i,j) \in E_F} d^{ij}} \cdot \sum_{\substack{(u,v) \in E_S \\ (u \neq v)}} \sum_{(i,j) \in E_F} f_{uv}^{ij} \tag{50}$$

subject to:

$$\sum_{u \in V_S} x_u^i = 1 \quad \forall i \in V_F \tag{51}$$

$$\sum_{v \in V_S} f_{uv}^{ij} - \sum_{v \in V_S} f_{vu}^{ij} = d^{ij}(x_u^i - x_u^j)$$
$$i \neq j, \forall i, j \in V_F, u \neq v, \forall u \in V_S \tag{52}$$

$$\sum_{i \in V_F} d^i x_u^i \leqslant r_u \cdot z_u \quad \forall u \in V_S \tag{53}$$

$$\sum_{i,j \in V_F} f_{uv}^{ij} \leqslant r_{uv} \quad \forall (u,v) \in E_S \tag{54}$$

$$z_u \geqslant x_u^i \quad \forall i \in V_F, \forall u \in V_S \tag{55}$$

$$x_u^i \geqslant 0 \quad \forall i \in V_F, \forall u \in V_S \tag{56}$$

$$0 \leqslant z_u \leqslant 1 \quad \forall u \in V_S \tag{57}$$

$$f_{uv}^{ij} \geqslant 0 \quad \forall (i,j) \in E_F, \forall (u,v) \in E_S \tag{58}$$

We designed a rounding algorithm (Algorithm 7) which is based on previous work. It basically fixes each the most probable assignment of an NF to a server (x_u^i) and repeats solver runs as long as there are feasible LP solutions and non-binary results for x_u^i. If an NF to node mapping becomes infeasible due to the already assigned demand, then the corresponding mapping will be precluded by setting this $x_u^i \leftarrow 0$.

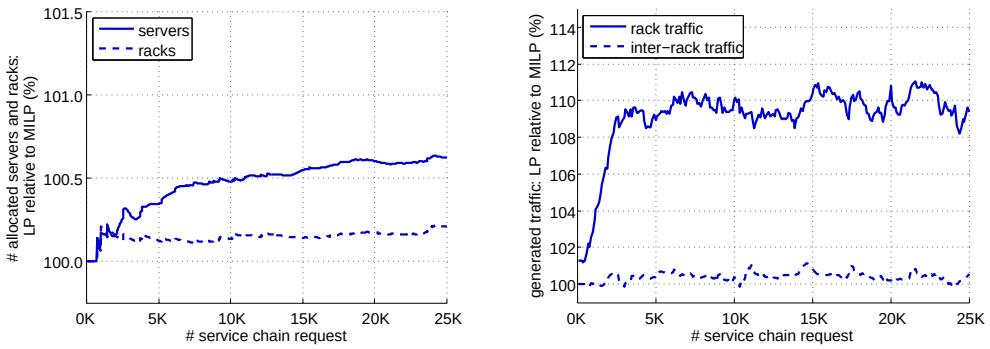
We hereby investigate the suboptimality of the LP compared to the MILP variant, in terms of DC mapping efficiency. To this end, we assign 25K NF graphs onto a DC, using both mapping variants. Each NF graph contains 3 to 20 NFs. Other evaluation parameters used for this test are the same with our NSE evaluation in Section 5.6.1 (see Table 9). For comparison between the two variants

Algorithm 7 NF graph mapping with LP

repeat
  $\{z_u, x_u^i, f_{uv}^{ij}, y_r^s\} \leftarrow$ **Solve_LP(..)**
  $Y \leftarrow \{y_r^s \mid y_r^s \notin \{0,1\}\}$
  $X \leftarrow \{x_u^i \mid x_u^i \notin \{0,1\}\}$
  if $Y \neq \emptyset$ **then**
    $\{s_{fx}, r_{fx}\} \leftarrow argmax_{\{s \in S, r \in R\}} Y$
    Add_LP_Constraint("$y_{r_{fx}}^{s_{fx}} = 1$")
  else if $X \neq \emptyset$ **then**
    $\{i_{fx}, u_{fx}\} \leftarrow argmax_{\{i \in V_F, u \in V_S\}} X$
    if $\sum_{i \in \{V_F \mid x_{u_{fx}}^i = 1\}} x_{u_{fx}}^i d^i + d^{i_{fx}} \leqslant r_u$ **then**
      Add_LP_Constraint("$x_{u_{fx}}^{i_{fx}} = 1$")
    else
      Add_LP_Constraint("$x_{u_{fx}}^{i_{fx}} = 0$")
    end if
  end if
until $(X = \emptyset \wedge Y = \emptyset) \vee$ NoFeasibleSolutionLP
return $\{z_u, x_u^i, f_{uv}^{ij}, y_r^s\}$

we stress the system at a level at which the original ILP variant, with always the optimal mapping result starts dropping incoming requests due to resource shortages. Therefore, we set the request arrival rate to 10 per hour which in turn yields a mean acceptance rate of 99.87% and 99.86% for the MILP and LP variant. We further aim at comparing the resource efficiency of both NF graph mapping variants. In this respect, Figure 36a illustrates the number of used servers and racks for the LP variant relative to the MILP variant. The LP results in a marginally higher number of servers and a negligible higher number of racks. This appears plausible since the NF graph mapping objective function aims at minimizing link cost implicitely by co-locating NFs preferably in a server, if possible, or in a rack otherwise. We further investigate whether the LP variant generates additional traffic within the DC, compared to the MILP. According to Figure 36b, the LP results in marginally higher volume of inter-rack traffic and a more perceptible increase (i. e., 8 to 11%) in the traffic within the racks. However, rack traffic is less expensive than inter-rack traffic and as mentioned above, this does not impact the request acceptance rate and, therefore, the generable revenue. Eventually, the LP yields only marginal suboptimality compared to the MILP while the LP exhibits a substantially lower runtime (62 ms) compared to the MILP (650 ms)[2], which we deem that outweighs its suboptimality.

2 Tests are carried out on a server with 2.53 GHz Intel Xeon CPU where each test is assigned to a single CPU core.



(a) server and rack allocation

(b) DC traffic generation

Figure 36: NF graph mapping with MILP/LP: LP results in (i) marginally higher number of servers and a negligible higher number of racks and (ii) in marginally higher volume of inter-rack traffic and a more perceptible increase in the traffic within the racks.

5.6 EVALUATION

In this section, we assess the efficiency of NSE across multiple NFPs with our NSE control plane implementation as presented in Section 6.2. We mainly focus on service chain partitioning and particularly on the impact of different partitioning objectives. To this end, we rely on the two request partitioning LP variants, introduced in Section 5.4. Upon partitioning, the mapping of NF subgraphs to DCs is done by applying the LP based mapping method as introduced in Section 5.5. We explain evaluation parameters and metrics (Section 5.6.1) before we discuss our simulation results (Section 5.6.2).

5.6.1 *Parameters and Metrics*

Below, we provide further details on the substrate network and service chain specifications, as used in our evaluations.

Substrate Network. We generated a PoP-level substrate topology with 12 NFPs covering a region with the size of the US state California. The substrate spans 50 homogeneous DCs, each one containing 200 servers in 10 racks. For each DC, we have generated a 2-level hierarchical network topology, similar to Figure 2 as presented in Section 2.1.2.

Service chains. Network service requests are generated based on service chain templates. These templates are composed of NFs that correspond to real middlebox applications (e. g., firewall, load balancing, RE). Each NF is associated with the outbound/inbound traffic rate (φ), adjusted according to the statistics summarized in Table 7. The NF computational requirements and bandwidth

Table 9: NSE evaluation parameters.

Substrate network (PoP-level topology):	
NFPs / DCs	12 / 50
Intra-domain link cost	unif. distrib. [0.002, 0.006] $/Mbps
Peering link cost	unif. distrib. [0.006, 0.018] $/Mbps
Server cost	unif. distrib. [0.05, 0.10] $/GHz
Substrate network (data center topology):	
Core switches	5
Racks per DC	10
Servers per rack	20
Server capacity	$16 \cdot 2$ GHz
ToR-to-server link capacity	4 Gbps
Inter-rack link capacity	16 Gbps
Service chains:	
Number of NFs	uniform distrib. [10, 20]
Traffic generation rate	uniform distrib. [10, 100] Mbps

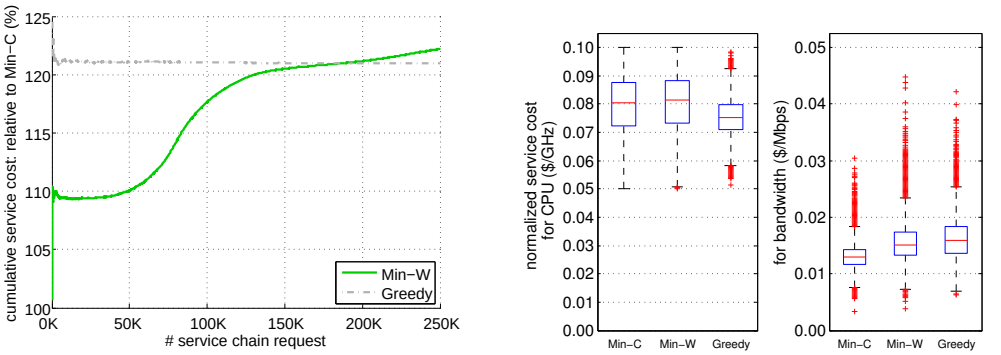
demands are derived from our network service model (Section 5.1), given the φ adjustments and the traffic rate at the endpoints. The traffic rate is randomly sampled from a uniform distribution. The endpoints are randomly selected out of 50 possible locations with a minimum distance of 250km to each other. Table 9 provides a list of the evaluation parameters.

We use the following metrics for the evaluation of NSE efficiency:

- **Service cost** represents the client's expenditure for the network service.
- **DC load balancing level** is defined as the maximum over the average server CPU load across the DCs.
- **Acceptance rate** is the number of successfully embedded requests over the total number of requests.
- **Revenue** accumulates the CPU and bandwidth units leased to clients.

5.6.2 *Results*

We perform a comparative study of the two request partitioning variants (Section 5.4), i. e., embedding cost minimization (*Min-C*) and link weight minimization (*Min-W*). In addition, we use a greedy algorithm as baseline. This algorithm binds each NF with one of the endpoints, depending on the NF location



(a) Service cost evaluation of NSE.

(b) normalized service cost with cost-minimized, weight-minimized, and greedy request partitioning

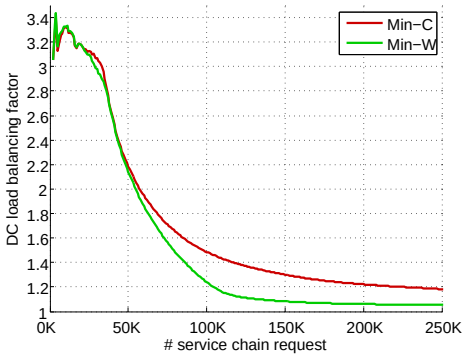
Figure 37: Both Min-W and the greedy algorithm yield a higher service cost, compared to the cost-minimized request partitioning. The lower service cost of Min-C stems from the significantly lower bandwidth cost.

constraint or the order in the service chain (for NFs without location dependencies), and assigns each NF to the DC which is most proximate to the corresponding endpoint.

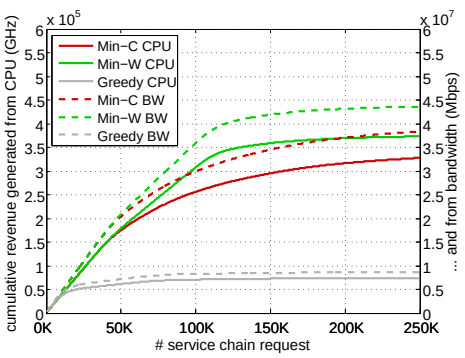
Figure 37a illustrates the evolution of the cumulative service cost with 250K non-expiring requests. Both *Min-W* and the greedy algorithm yield a higher service cost, compared to *Min-C*, which is formulated for service cost minimization. In particular, *Min-W* exhibits an increase in the service cost (relatively to *Min-C*) with the number of requests, eventually converging to 20% additional service cost, which is steadily incurred by the greedy algorithm.

The boxplots in Figure 37b illustrate the decomposition of service cost into the CPU and bandwidth cost, normalized per resource unit. The lower service cost of *Min-C* stems from the significantly lower bandwidth cost (Figure 37b), considering that in absolute terms the fraction of bandwidth cost is one magnitude higher than the fraction of CPU cost. Essentially, *Min-C* achieves cost savings with the selection of DCs which are reachable over less costly paths. The greedy algorithm yields an average CPU cost of 0.075\$/GHz, which corresponds to the average CPU cost across all NFPs, since DC selection is bound to randomly assigned endpoints.

So far, *Min-C* appears very appealing for clients, since it minimizes their expenditure. However, *Min-C* may entail suboptimality for NFPs which we investigate in the following. In this respect, Figure 38a depicts the evolution of load balancing level across the DCs. Since the greedy selection of DCs close to the endpoints does not lead to load balancing, we focus on the load balancing levels of the two LP variants. According to Figure 38a, *Min-W* converges to near-optimal load balancing after 100K requests, exploiting the DC utilization



(a) load balancing level with cost- and weight-minimized request partitioning



(b) cumulative revenue with cost-minimized, weight-minimized, and greedy request partitioning

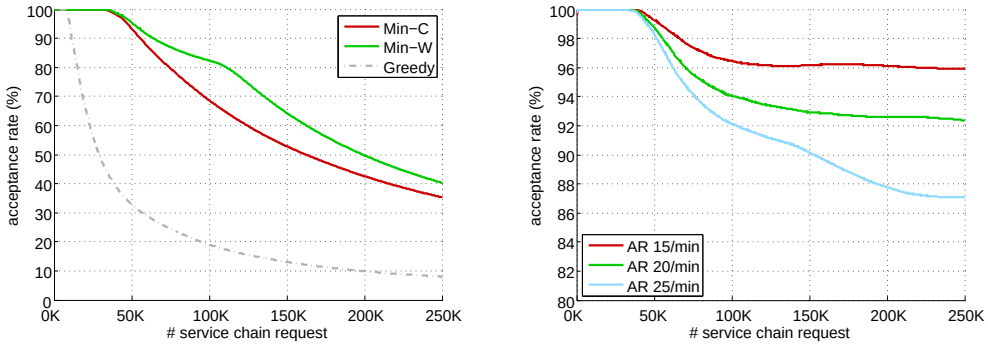
Figure 38: Min-W converges to near-optimal load balancing after 100K requests while Min-C yields a perceptible suboptimality. The LP variants generate substantially higher revenue from CPU and bandwidth, compared to the greedy algorithm.

levels disclosed via the link weights. In comparison, *Min-C* yields a perceptible suboptimality. For instance, after 250K requests the highest server load is 5.3% and 18.2% above the average DC utilization, for *Min-W* and *Min-C*, respectively.

Figure 39a shows the request acceptance rates for the three request partitioning methods. Optimizing DC selection based on the disclosed weights (*i. e.*, *Min-W*) inhibits the assignment of NF subgraphs to highly utilized DCs, which usually leads to request rejections. As such, *Min-W* yields a higher request acceptance rate. Specifically, after 100K requests (which corresponds to a server utilization level of 80% across DCs), *Min-W* can embed 23% more requests than *Min-C*. On the other hand, the greedy algorithm suffers from a large number of rejections, due to the restrictions in DC selection.

Figures 39a and 38b show a strong correlation between the acceptance rate and generated revenue. The LP variants generate substantially higher revenue from CPU and bandwidth, compared to the greedy algorithm. For *Min-W*, the highest acceptance rate is translated to a higher revenue, i. e., up to 14% more than *Min-C*. This essentially designates *Min-W* as the preferred request partitioning method for NFPs.

Finally, we measure the acceptance rate of *Min-W* with 250K expiring requests and diverse arrival rates. Figure 39b shows that acceptance rates converge to a steady state, irrespective of the arrival rate. This further indicates the efficiency of *Min-W* LP and our DC mapping algorithm for NSE across multiple NFPs.



(a) cost-minimized, weight-minimized, and greedy non-expiring request partitioning

(b) diverse arrival rates of weight-minimized expiring request partitioning

Figure 39: Min-W yields the highest request acceptance rate compared to the other variants. The greedy algorithm suffers from a large number of rejections, due to the restrictions in DC selection. In the case of expiring requests, the acceptance rate converges towards a steady state irrespective of the arrival rate.

5.7 RELATED WORK

Existing work on NSE has mainly focused on NF graph mapping onto DC networks. STRATOS [46] and CloudNaaS [115] propose heuristic mapping algorithms that seek to minimize inter-rack traffic within DC networks. A similar approach is also taken by Oktopus [116], SecondNet [117] and CloudMirror [118] for the assignment of virtual clusters to DCs. Huang et al. propose a distributed algorithm for network service placement assuming the ability to deploy NFs in the data path [119]. MIDAS [120] employs a heuristic algorithm for order-preserving NF assignment to middleboxes deployed along the data path. Several studies have tackled the problem of embedding VN topologies onto a shared substrate network.[3] However, these embedding techniques are designed for arbitrary virtual and substrate network topologies, and are not optimized for NF graph mapping onto DC network topologies. Compared to all these approaches, our NSE framework provides a holistic solution for the NSE problem across multiple NFPs, including request partitioning, generation of NF subgraphs with DC gateway bindings, and NF subgraph mapping.

Furthermore, with respect to the service chain partitioning, similar works related to VN embedding exist that apply a similar embedding problem decomposition, i. e., VN request partitioning among providers, followed by request segment mapping onto each provider's network (e. g., [63, 65, 87], see also Section 2.5.1). These works do however not cope with the requirements of service chain embedding.

3 See section 2.5.2 for an overview of existing VN embedding algorithms.

6 IMPLEMENTATION

All results in this thesis have been obtained from our own simulation environment. A set of C/C++ based libraries, compiled from nearly 18,000 lines of source code, represents the core of our implementation work (Section 6.1). We used these libraries in our universal control plane implementation for our proposed VNE/NSE frameworks (Section 6.2). In addition to that, we visualize the substrate network views of the different actors of VNE/NSE and provide a graphical real-time evaluation module (Section 6.3).

6.1 FUNCTIONALITY

We have implemented a set of libraries which provide the basic functions of the VNE and NSE frameworks presented in the Chapters 3 and 5. An API provides various functions which can be classified into

- generators for requests and topologies (Sections 6.1.1 and 6.1.2),
- request processing (Sections 6.1.3 and 6.1.4),
- logging and statistics (Section 6.1.5).

The implementation which is based on C++ classes facilitates internal data structures that are used for the modeling of logic elements such as network topologies including network devices and servers. This data can be accessed by a file I/O class using XML data schemes.

6.1.1 *Request Generator*

Two different types of requests – VNs and service chains can be generated. Requests are generally associated with an abstract time stamp defining their starting time following a Poisson distribution. The duration of each request can either be infinite, i. e., non-expiring, or follow a uniform distribution between a minimum and a maximum number of abstract time units. For each virtual node of a VN the CPU demand is randomly set and a traffic matrix consisting of virtual links (bandwidth demands) among the virtual nodes is computed accordingly. Preferred geographic coordinates can be generated optionally if the coverable region is given by an existing substrate topology file. VN requests

contain a traffic matrix with fixed bandwidth demands between all the virtual nodes while service chain requests merely contain the source traffic rate. We randomly set the source traffic rate for the request and generate a set of subchains each consisting of different NFs. CPU and bandwidth demands are then computed dependent on the inbound traffic rate at each NF, the NF type and therewith the traffic amplification rate (Section 5.1).

6.1.2 *Substrate Network Generator*

Our implementation supports the import from *.rig* files which can be generated by substrate topology generators such as IGen [102]. Non-supported but required attributes, such as CPU capacity or monetary link cost can be added to a customized XML format upon .rig file import. DC topologies are further covered by our own topology generator supporting, for example, 3-layer fat-tree topologies of individual size (number of core switches, racks, servers, and so forth).

6.1.3 *Request Partitioning*

Multi-provider VNE/NSE requires the partitioning of the original VN/service chain into segments to be embedded by different InPs/NFPs. This is a combinatorial optimization problem which we formally describe in Sections 3.5 and 5.4. We rely by default on *IBM ILOG CPLEX* [121] to solve the ILP/LP models but offer a wrapper class for both the CPLEX and the *GNU Linear Programming Kit (GLPK)* [122] optimizers. The latter should be used if the commercial product CPLEX [121] is not available. The rounding algorithms for obtaining near-optimal results from LPs as well as a greedy algorithm for service chain partitioning are implemented as well. For a few intermediate steps, we rely on the Boost Graph Library [123] which we use for graph computations, for example, to find the shortest path in a substrate topology.

6.1.4 *Resource Mapping*

The resource mapping problems discussed in this work are solved using either MILP/LP-based models or by a heuristic algorithm:

- VN (segment) mapping (MILP/LP, Section 3.6),
- subset VN mapping (heuristic algorithm, Section 4.3),
- NF subgraph mapping in a DC (MILP/LP, Section 5.5).

For solving the resource mapping combinatorial optimization problem we rely again on CPLEX and apply rounding algorithms as we do for the request partitioning problem (previous Section 6.1.3).

6.1.5 *Logging and Statistics*

Apart from the logging of partitioning and mapping results, we keep track of changes in the substrate networks (e. g., with respect to resource utilization, weight and cost updates, or request expirations). This information is partially aggregated for further processing, i. e., the average utilization of all servers in a DC or the cumulative generated revenue. This step simplifies the generation of statistics and enables access to intermediate statistic results of a running simulation. Most data is available in the *.csv* format.

6.2 UNIVERSAL CONTROL PLANE FOR MULTI-PROVIDER EMBEDDING

We have implemented a universal control plane for multi-provider embedding that supports VNs and service chains. This control plane is distributed across multiple management nodes, allowing the physical separation of all actors. In our implementation, each resource provider (e. g., InP or NF provider) exposes a control interface to the centralized coordinator (VN provider in the case of VNs), allowing (i) resource advertisements from each resource provider to the coordinator, and (ii) VN segment or NF subgraph submission from the coordinator to each assigned resource provider. Similarly, the coordinator exposes a control interface to the client or SP, such that they can submit requests and receive the embedding result. All these functions are executed through remote procedure calls. We use an XML-based schema to represent resource requirements, network topologies, and resource availability. This schema is used by all actors to specify and exchange virtual and physical resource information for resource advertisements and the submission of requests.

The basic workflow for multi-provider VNE/NSE, as implemented in our system, is depicted in the sequence diagram of Figure 40. All resource providers are required to advertise all offered resource types to the coordinator, that maintains this information in a local resource repository. Initially, the requester generates and submits a request to the coordinator, that in turn partitions the request into segments as described in the Sections 3.5 and 5.4. Subsequently, the coordinator generates requests for the segments, which are relayed to the assigned resource provider. Each resource provider evaluates the feasibility of the segment mapping onto his network and returns the result to the coordinator (using the methods discussed in the Sections 3.6 and 5.5). After successful

mapping of all segments, the coordinator declares the request as accepted and returns the result to the requester. Otherwise, the request is rejected, and the requester is informed of this outcome. For those resource providers that are only used for transit, we compute, between the respective peering nodes, the shortest path that satisfies the bandwidth demand. To this end, we adapt the problem formulation of Section 3.6 and eliminate the node capacity constraint.

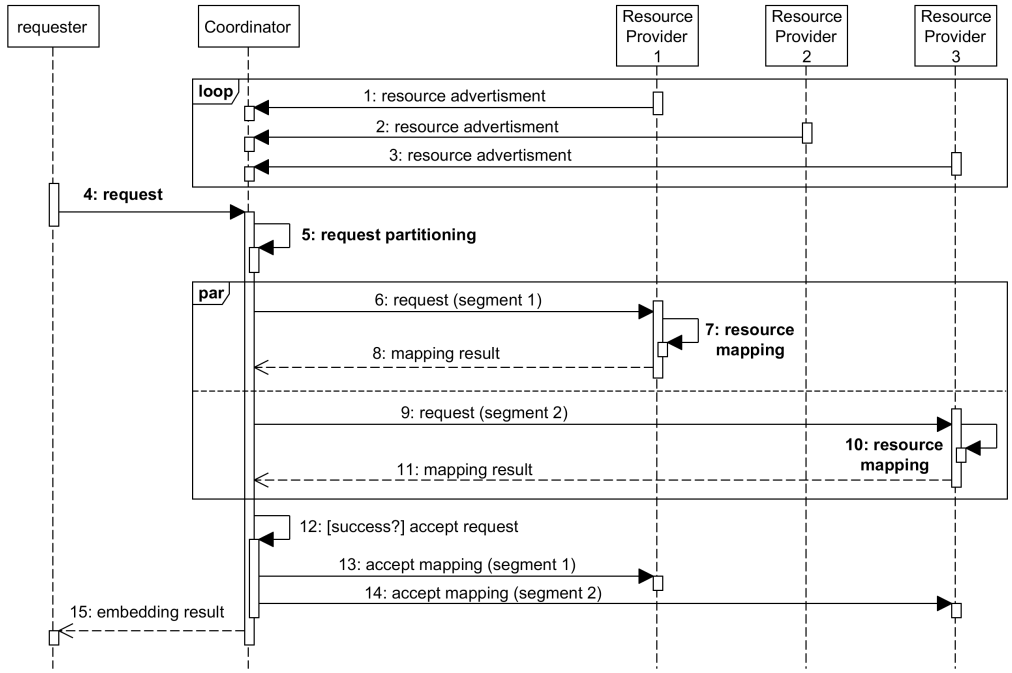


Figure 40: Multi-provider VNE/NSE requires the continuous advertisement of resources by the resource providers. The coordinator partitions any incoming request across the resource providers that subsequently try to map the segments of the original request to their physical resources.

We use a non-distributed version of our universal control plane implementation for evaluation of multi-provider VNE (Section 3.7) and NSE (Section 5.6).

6.3 VISUALIZATION AND REAL-TIME EVALUATION

We implemented a fully-automated framework for VNE/NSE across multiple substrate networks which is equipped with Qt-based GUIs and a real-time evaluation module [124]. This framework comprises a realistic evaluation environment for VN and service chain partitioning and resource mapping algorithms. It facilitates investigation of various VNE/NSE aspects, such as the impact of information disclosure on embedding efficiency and the suitability of different request models. Specifically, we simulate the different actors (requester, coordinator, and resource providers) in separate instances of our control plane implementation (Section 6.2). Each instance provides a GUI for optional manual control. According to the embedding workflow as depicted in Figure 40), the requester instance generates and forwards requests automatically in a given time interval. The requests can be parameterized in real-time, for example, in terms of VN size or service chain length, traffic rate, the level of geographic restrictions. Upon arrival of the request at the coordinator, the partitioning and subsequent reformulation of VN segments or NF subgraphs take place. Then the coordinator sends each of these reformulated requests to the assigned resource provider. All involved resource providers automatically map the requested VN or NF subgraph to their physical resources and report the result to the coordinator. In addition, partitioning and mapping results are received by the evaluation module which generates statistics such as acceptance rate, total hop count, and generated revenue. The evaluation module also simulates direct embedding of requests without any restriction on physical resource visibility. Thus, a visual comparison between the realistic embedding under limited information disclosure and the best-case scenario with full information disclosure is provided. In this context, Figure 41 shows a screenshot of an evaluation setup for VNE with SP, VN provider, and 3 InPs.

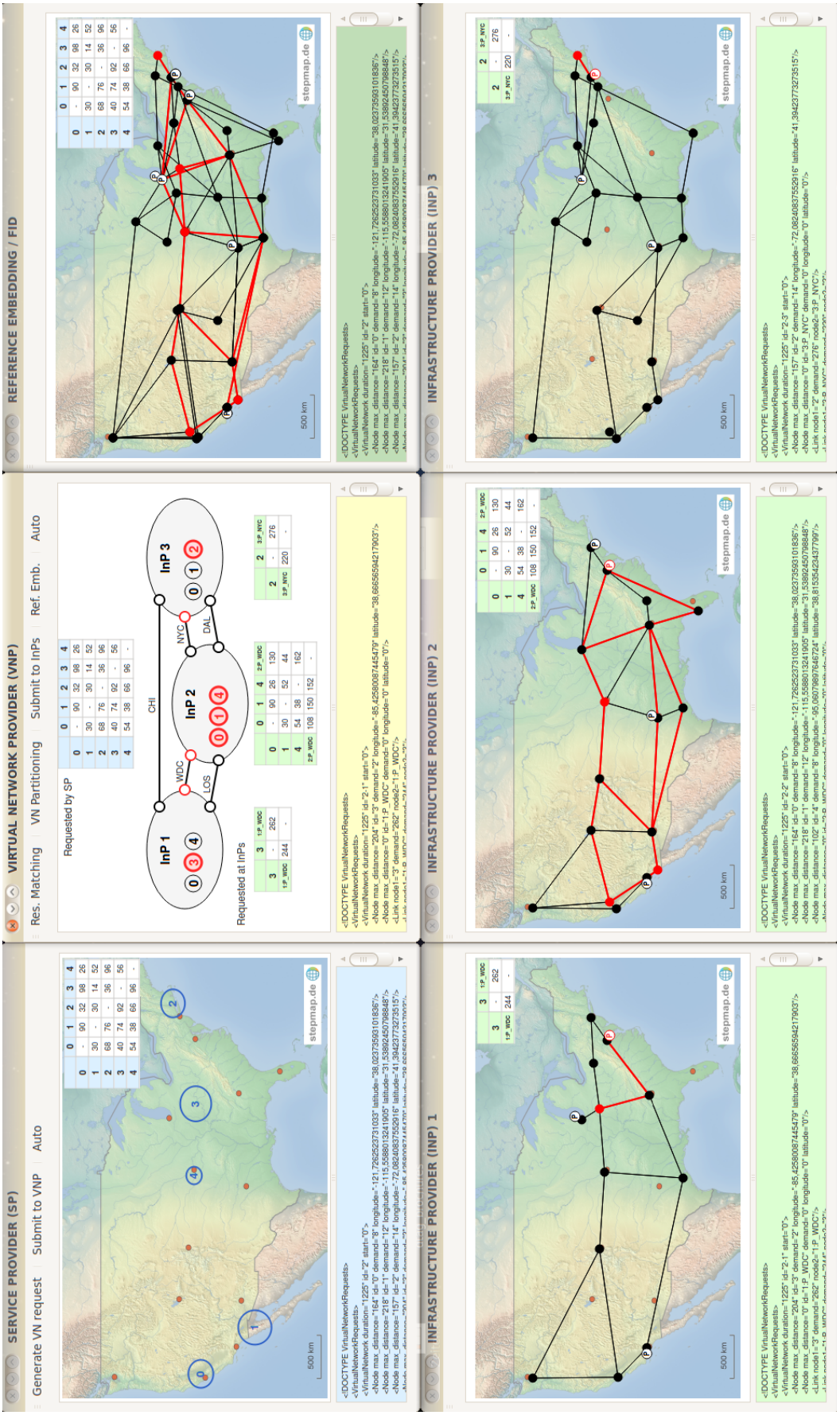


Figure 41: The automated framework for the real-time evaluation of multi-provider VNE/NSE is equipped with Qt-based visualization. The screenshot shows windows for request generation (SP), request partitioning (VNP), reference embedding based on full information disclosure (FID), resource mapping at the 3 participating InPs.

Our implementation also supports the distributed setup of VNE/NSE evaluation with multiple InPs [125]. Such a setup for VNE is depicted in Figure 42 where we use separate PCs/laptops (i. e., five in total) for the SP, the VN provider, and for each one of the InP management nodes. Communication between the different actors is XML-based and displayed immediately. Further, for each request the optimal solution in terms of cost is computed immediately and visualized in the evaluation module that further generates and displays statistics.

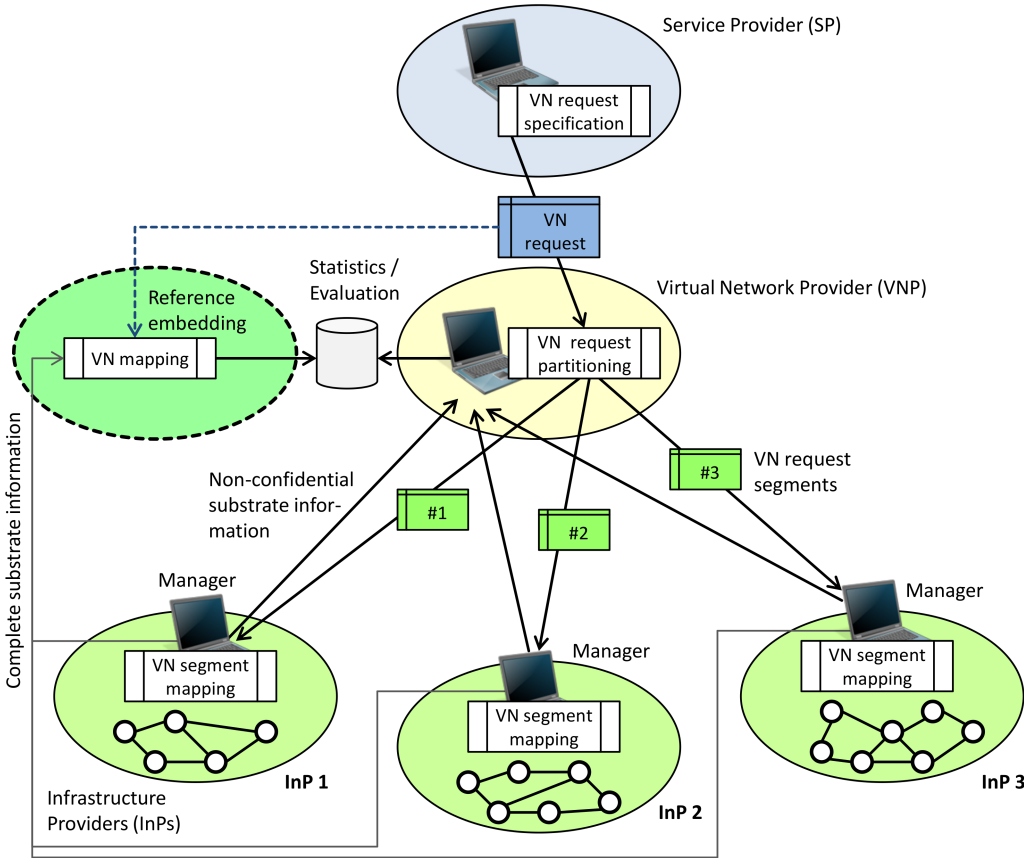


Figure 42: A distributed setup of VNE/NSE framework can be realized by the exclusive assignment of a single PC to all of the actors (i. e., SP, VN provider, InPs).

7

CONCLUSIONS

We tackled the challenges of multi-provider VN and service chain embedding (VNE/NSE) with respect to (i) limited information disclosure of the infrastructure providers, (ii) consideration of middlebox policies and the variety of network functions (NFs) in service chains, and (iii) profitability of VNEs from the perspective of the InPs.

We studied the feasibility of multi-provider VNE with limited view on the substrate networks. In order to conduct this study, we developed a VNE framework that uses an abstraction model only including non-confidential substrate information. We accommodated into this framework (i) the VN request partitioning fulfilled by the VN provider with limited view on the substrate networks and (ii) the subsequent VN segment mapping fulfilled by all the involved InPs with detailed substrate network information. Our study also questioned the suitability of VN topologies for VNE by simulations. The more we enforced the mapping of a VN to the substrate topology to strive for similarity to the requested VN graph, the lower was the acceptance rate, especially for larger VNs. For this reason, we motivated the use of traffic matrix instead of topology-based VN formulations.

In addition, we devised a framework for the embedding of service chains based on our experience with the multi-provider VNE framework. We particularly paid attention to (i) the traffic rate amplification (or reduction) caused by the different NFs, geographic interdependencies between NFs, and (ii) to the confidentiality of information with regard to DCs and inter-DC networks. In this respect, we devised a service model that simplifies the specification of the network service requests and a topology abstraction to be used by a centralized coordinator (NSCL) for service chain partitioning. We subsequently used for the mapping of NF subgraphs to DC servers, i. e., within the NF provider's domain, fully visible substrate information.

We further provided linear programming formulations for the combinatorial optimization problems that arose from the request partitioning and resource mapping phases each of the VNE and the NSE. We also elaborated on the profitability of VN requests from the perspective of an InP and introduced a policy dimension to VNE. For this reason, we developed a heuristic algorithm that enforces a certain level of profitability in terms of resource efficiency. According to that, we defined the cost to revenue ratio (CRR) threshold and evaluated

possible adjustments in order the strike a balance between short- and long-term revenue gains for the InP.

7.1 KEY FINDINGS

Our simulation results showed that multiprovider VNE under limited information disclosure is feasible at the price of moderate extra cost of 5-20% in comparison to a "best-case" scenario where all substrate network information is available to VN providers. We uncovered that the VNE suboptimality under limited information disclosure stems from the increased hop count of embedded virtual links. This seemed reasonable as we observed an increasing extra cost for those VNs with a higher number of assigned InPs. In contrast to that, the number of participating InPs and the VN size did not significantly impact the extra cost. We also found a remarkable increase of VNE efficiency in terms of acceptance rate and revenue if VNs were formulated as traffic matrices, instead of topologies.

Further simulation results indicated that service chain deployment across multiple NF providers (and DCs) is feasible. Our insights into request partitioning uncovered a trade-off between service cost minimization and resource efficiency. In particular, service cost minimization can potentially lead to 10-20% cheaper NFaaS offerings, attracting more clients, but at the same time generates suboptimal embeddings that restrict the revenue of NF providers. Conversely, partitioning optimizations driven by NF provider policies yield higher resource efficiency, increasing the NF providers' revenue by up to 14%, but also entail more expensive and, thus, less competitive NFaaS offerings.

Moreover, we evaluated policy-compliant embedding of VN subsets at which profitability is enforced by a CRR threshold. Simulation results corroborated our assumption that policy-compliant VNE uses resources more efficiently and generates much higher revenue. More precisely, 1.5x higher revenue compared to full VN request embedding can be generated with policy-compliant VNE. We also investigated the effect of different CRR threshold adjustments. Lower CRR thresholds increased the resource efficiency at the beginning, thus, in the long run, more revenue was generated. Conversely, higher CRR thresholds generated more revenue in the short run at the price of lower resource efficiency. Overall, we showed that CRR threshold is a viable means towards policy-compliant VNE with the goal to dynamically achieve balancing between resource efficiency and revenue.

7.2 FUTURE WORK

In the following, we sketch future research directions.

- **Service chain dynamics.** Network service chains will be subject to dynamic variations after their initial resource allocation. These variations can be basically due to changes in services demands, traffic loads, and physical resources. Examples for that are changes to policies (e. g., for firewalls, load balancing), additional endpoints or changes to the geographic region to be covered by the network service. Altogether, this raises the need for dynamic scaling of NFs. This means that NF instances need to be added or removed, and NFs need to be reprovisioned in terms of server resources such as CPU, memory, and storage. In addition, NF scaling results in changing traffic rates throughout the service chain that require reprovisioning of bandwidth accordingly. Altogether, service chain dynamics represents one of the major challenges of future research in network service embedding.

- **InP privacy.** We take advantage of the fact that InPs advertise, at least to a limited extent, information about their network topology and resources. For the future, we expect that they could unveil more information to attract more requests. Regardless of the information disclosure level, there exists the serious risk that third parties circumvent the InPs' privacy policy. Specifically, they could draw conclusions from the outcome of previous requests. In addition, they could direct to an InP specific test requests with the purpose to explore the InP's secret network topology and resource information, or even pricing policies. A study on the potential exploitation of request results will be highly desired by the InPs.

- **Network function verification.** The correct function of a service chain needs to be ensured permanently. Minor changes can have serious implications on the function of a service chain. For example, loops in the traffic path could cause NFs be traversed not in the correct order, which in turn could accidentally disarm firewall rules. Furthermore, outsourcing NFs to external cloud providers raises the issues of performance assurance. For example, latency might be increased due to longer paths. In addition, the client should be able to verify the correct accounting for the used physical resources such as the consumed CPU cycles. We believe that functional verification of service chains is another major field of future research.

Part II

APPENDIX

BIBLIOGRAPHY

[1] Y. Rekhter, T. Li, and S. Hares. *A Border Gateway Protocol 4 (BGP-4)*. RFC 4271 (Draft Standard). Updated by RFCs 6286, 6608, 6793, 7606, 7607. Internet Engineering Task Force, Jan. 2006. URL: http://www.ietf.org/rfc/rfc4271.txt (cit. on pp. 3, 10).

[2] J. Moy. *OSPF Version 2*. RFC 2328 (INTERNET STANDARD). Updated by RFCs 5709, 6549, 6845, 6860, 7474. Internet Engineering Task Force, Apr. 1998. URL: http://www.ietf.org/rfc/rfc2328.txt (cit. on p. 3).

[3] Scott Shenker. "Fundamental design issues for the future Internet." In: *IEEE Journal on Selected Areas in Communications* 13.7 (1995), pp. 1176–1188 (cit. on p. 3).

[4] David D Clark, John Wroclawski, Karen R Sollins, and Robert Braden. "Tussle in cyberspace: defining tomorrow's internet." In: *IEEE/ACM Transactions on Networking (ToN)* 13.3 (2005), pp. 462–475 (cit. on p. 3).

[5] Anja Feldmann. "Internet clean-slate design: what and why?" In: *ACM SIGCOMM Computer Communication Review* 37.3 (2007), pp. 59–64 (cit. on p. 3).

[6] Mark Handley. "Why the Internet only just works." In: *BT Technology Journal* 24.3 (2006), pp. 119–129 (cit. on p. 3).

[7] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. "Xen and the art of virtualization." In: *ACM SIGOPS Operating Systems Review* 37.5 (2003), pp. 164–177 (cit. on pp. 3, 13).

[8] *SWSoft, OpenVZ, Server Virtualization Open Source Project, 2005*. URL: https://openvz.org/ (cit. on pp. 3, 13).

[9] Norbert Egi, Adam Greenhalgh, Mark Handley, Mickael Hoerdt, Felipe Huici, and Laurent Mathy. "Towards high performance virtual routers on commodity hardware." In: *Proceedings of the 2008 ACM CoNEXT Conference*. ACM. 2008, p. 20 (cit. on p. 3).

[10] Cong Wang, Shashank Shanbhag, and Tilman Wolf. "Virtual network mapping with traffic matrices." In: *2012 IEEE International Conference on Communications (ICC)*. IEEE. 2012, pp. 2717–2722 (cit. on pp. 3, 26, 58).

[11] Sapan Bhatia, Murtaza Motiwala, Wolfgang Muhlbauer, Yogesh Mundada, Vytautas Valancius, Andy Bavier, Nick Feamster, Larry Peterson, and Jennifer Rexford. "Trellis: A platform for building flexible, fast virtual networks on commodity hardware." In: *Proceedings of the 2008 ACM CoNEXT Conference*. ACM. 2008, p. 72 (cit. on p. 3).

[12] 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*. ACM. 2009, pp. 63–72 (cit. on pp. 4, 5, 17, 23, 49).

[13] *GENI: Global Environment for Network Innovations*. URL: `http://www.geni.net/` (cit. on p. 4).

[14] Justine Sherry, Shaddi Hasan, Colin Scott, Arvind Krishnamurthy, Sylvia Ratnasamy, and Vyas Sekar. "Making middleboxes someone else's problem: network processing as a cloud service." In: *ACM SIGCOMM Computer Communication Review* 42.4 (2012), pp. 13–24 (cit. on p. 4).

[15] B. Carpenter and S. Brim. *Middleboxes: Taxonomy and Issues*. RFC 3234 (Informational). Internet Engineering Task Force, Feb. 2002. URL: `http://www.ietf.org/rfc/rfc3234.txt` (cit. on p. 4).

[16] Justine Sherry, Sylvia Ratnasamy, and Justine Sherry At. "A survey of enterprise middlebox deployments." In: (2012) (cit. on pp. 4, 14).

[17] *ETSI, European Standards Organization, Network Functions Virtualization*. URL: `http://www.etsi.org/technologies-clusters/technologies/nfv` (cit. on pp. 4, 14).

[18] *T-NOVA Project*. URL: `http://www.t-nova.eu/` (cit. on pp. 4, 14).

[19] Benjamin Frank, Ingmar Poese, Yin Lin, Georgios Smaragdakis, Anja Feldmann, Bruce Maggs, Jannis Rake, Steve Uhlig, and Rick Weber. "Pushing cdn-isp collaboration to the limit." In: *ACM SIGCOMM Computer Communication Review* 43.3 (2013), pp. 34–44 (cit. on pp. 4, 14).

[20] Yaping Zhu, Rui Zhang-Shen, Sampath Rangarajan, and Jennifer Rexford. "Cabernet: connectivity architecture for better network services." In: *Proceedings of the 2008 ACM CoNEXT Conference*. ACM. 2008, p. 64 (cit. on pp. 5, 17, 49).

[21] David G Andersen. "Theoretical approaches to node assignment." In: *Carnegie Mellon University, Computer Science Department, Paper 86* (2002), p. 86. URL: `http://repository.cmu.edu/compsci/86` (cit. on p. 6).

[22] Shimon Even, Alon Itai, and Adi Shamir. "On the complexity of time table and multi-commodity flow problems." In: *1975 16th Annual Symposium on Foundations of Computer Science*. IEEE. 1975, pp. 184–193 (cit. on pp. 6, 37).

[23] Nick Feamster, Jay Borkenhagen, and Jennifer Rexford. "Guidelines for interdomain traffic engineering." In: *ACM SIGCOMM Computer Communication Review* 33.5 (2003), pp. 19–30 (cit. on p. 9).

[24] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. "A view of cloud computing." In: *Communications of the ACM* 53.4 (2010), pp. 50–58 (cit. on p. 10).

[25] K. Lougheed and Y. Rekhter. *Border Gateway Protocol (BGP)*. RFC 1105 (Experimental). Obsoleted by RFC 1163. Internet Engineering Task Force, June 1989. URL: http://www.ietf.org/rfc/rfc1105.txt (cit. on p. 10).

[26] Renata Teixeira, Aman Shaikh, Tim Griffin, and Jennifer Rexford. "Dynamics of hot-potato routing in IP networks." In: *ACM SIGMETRICS Performance Evaluation Review* 32.1 (2004), pp. 307–319 (cit. on p. 10).

[27] Timothy G Griffin, F Bruce Shepherd, and Gordon Wilfong. "The stable paths problem and interdomain routing." In: *IEEE/ACM Transactions on Networking (ToN)* 10.2 (2002), pp. 232–243 (cit. on p. 10).

[28] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. "A scalable, commodity data center network architecture." In: *ACM SIGCOMM Computer Communication Review* 38.4 (2008), pp. 63–74 (cit. on p. 11).

[29] Thomas Anderson, Larry Peterson, Scott Shenker, and Jonathan Turner. "Overcoming the Internet impasse through virtualization." In: *Computer* 4 (2005), pp. 34–41 (cit. on p. 12).

[30] NM Mosharaf Kabir Chowdhury and Raouf Boutaba. "A survey of network virtualization." In: *Computer Networks* 54.5 (2010), pp. 862–876 (cit. on p. 12).

[31] Avi Kivity, Yaniv Kamay, Dor Laor, Uri Lublin, and Anthony Liguori. "kvm: the Linux virtual machine monitor." In: *Proceedings of the Linux Symposium*. Vol. 1. 2007, pp. 225–230 (cit. on p. 12).

[32] *VMware*. URL: http://www.vmware.com/ (cit. on p. 12).

[33] Hasan Fayyad-Kazan, Luc Perneel, and Martin Timmerman. "Full and Para-Virtualization with Xen: A Performance Comparison." In: *Journal of Emerging Trends in Computing and Information Sciences* 4.9 (2013) (cit. on p. 13).

[34] *Docker*. URL: https://www.docker.com/ (cit. on p. 13).

[35] C. Perkins. *IP Encapsulation within IP*. RFC 2003 (Proposed Standard). Updated by RFCs 3168, 6864. Internet Engineering Task Force, Oct. 1996. URL: http://www.ietf.org/rfc/rfc2003.txt (cit. on p. 13).

[36] R. Housley and S. Hollenbeck. *EtherIP: Tunneling Ethernet Frames in IP Datagrams*. RFC 3378 (Informational). Internet Engineering Task Force, Sept. 2002. URL: http://www.ietf.org/rfc/rfc3378.txt (cit. on p. 13).

[37] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina. *Generic Routing Encapsulation (GRE)*. RFC 2784 (Proposed Standard). Updated by RFC 2890. Internet Engineering Task Force, Mar. 2000. URL: http://www.ietf.org/rfc/rfc2784.txt (cit. on p. 13).

[38] E. Rosen, A. Viswanathan, and R. Callon. *Multiprotocol Label Switching Architecture*. RFC 3031 (Proposed Standard). Updated by RFCs 6178, 6790. Internet Engineering Task Force, Jan. 2001. URL: http://www.ietf.org/rfc/rfc3031.txt (cit. on p. 13).

[39] A. Farrel, J.-P. Vasseur, and A. Ayyangar. *A Framework for Inter-Domain Multiprotocol Label Switching Traffic Engineering*. RFC 4726 (Informational). Internet Engineering Task Force, Nov. 2006. URL: http://www.ietf.org/rfc/rfc4726.txt (cit. on p. 13).

[40] Christoph Werle, Panagiotis Papadimitriou, Ines Houidi, Wajdi Louati, Djamal Zeghlache, Roland Bless, and Laurent Mathy. "Building virtual networks across multiple domains." In: *ACM SIGCOMM Computer Communication Review*. Vol. 41. 4. ACM. 2011, pp. 412–413 (cit. on p. 13).

[41] Adam Greenhalgh, Felipe Huici, Mickael Hoerdt, Panagiotis Papadimitriou, Mark Handley, and Laurent Mathy. "Flow processing and the rise of commodity network hardware." In: *ACM SIGCOMM Computer Communication Review* 39.2 (2009), pp. 20–26 (cit. on p. 14).

[42] Vyas Sekar, Sylvia Ratnasamy, Michael K Reiter, Norbert Egi, and Guangyu Shi. "The middlebox manifesto: enabling innovation in middlebox deployment." In: *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*. ACM. 2011, p. 21 (cit. on p. 14).

[43] Vyas Sekar, Norbert Egi, Sylvia Ratnasamy, Michael K Reiter, and Guangyu Shi. "Design and implementation of a consolidated middlebox architecture." In: *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*. USENIX Association. 2012, pp. 24–24 (cit. on p. 14).

[44] Justine Sherry, Shaddi Hasan, Colin Scott, Arvind Krishnamurthy, Sylvia Ratnasamy, and Vyas Sekar. "Making middleboxes someone else's problem: network processing as a cloud service." In: *ACM SIGCOMM Computer Communication Review* 42.4 (2012), pp. 13–24 (cit. on p. 14).

[45] Glen Gibb, Hongyi Zeng, and Nick McKeown. "Outsourcing network functionality." In: *Proceedings of the first workshop on Hot topics in software defined networks*. ACM. 2012, pp. 73–78 (cit. on p. 14).

[46] Aaron Gember, Robert Grandl, Ashok Anand, Theophilus Benson, and Aditya Akella. "Stratos: Virtual middleboxes as first-class entities." In: *UW-Madison TR1771* (2012) (cit. on pp. 14, 16, 99).

[47] Nick McKeown. "Software-defined networking." In: *INFOCOM keynote talk* 17.2 (2009), pp. 30–32 (cit. on p. 14).

[48] *The OpenFlow Switch Consortium*. URL: `http://www.openflowswitch.org` (cit. on p. 14).

[49] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. "OpenFlow: enabling innovation in campus networks." In: *ACM SIGCOMM Computer Communication Review* 38.2 (2008), pp. 69–74 (cit. on p. 14).

[50] Natasha Gude, Teemu Koponen, Justin Pettit, Ben Pfaff, Martín Casado, Nick McKeown, and Scott Shenker. "NOX: towards an operating system for networks." In: *ACM SIGCOMM Computer Communication Review* 38.3 (2008), pp. 105–110 (cit. on p. 14).

[51] David Erickson. "The beacon openflow controller." In: *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM. 2013, pp. 13–18 (cit. on p. 14).

[52] *OpenDayLight*. URL: `https://www.opendaylight.org/` (cit. on p. 14).

[53] *Open vSwitch*. URL: http://openvswitch.org/ (cit. on p. 14).

[54] Andreas Blenk, Arsany Basta, Martin Reisslein, and Wolfgang Kellerer. "Survey on Network Virtualization Hypervisors for Software Defined Networking." In: *IEEE Communications Surveys & Tutorials* (2015) (cit. on p. 15).

[55] Rob Sherwood, Michael Chan, Adam Covington, Glen Gibb, Mario Flajslik, Nikhil Handigol, Te-Yuan Huang, Peyman Kazemian, Masayoshi Kobayashi, Jad Naous, et al. "Carving research slices out of your production networks with OpenFlow." In: *ACM SIGCOMM Computer Communication Review* 40.1 (2010), pp. 129–130 (cit. on p. 15).

[56] Zdravko Bozakov and Panagiotis Papadimitriou. "AutoSlice: Automated and Scalable Slicing for Software-defined Networks." In: *Proceedings of the 2012 ACM Conference on CoNEXT Student Workshop*. ACM, 2012 (cit. on p. 15).

[57] Andreas Blenk, Arsany Basta, and Wolfgang Kellerer. "HyperFlex: An SDN Virtualization Architecture with Flexible Hypervisor Function Allocation." In: *IFIP/IEEE International Symposium on Integrated Network Management*. 2015 (cit. on p. 15).

[58] Jeffrey C Mogul and Lucian Popa. "What we talk about when we talk about cloud network performance." In: *ACM SIGCOMM Computer Communication Review* 42.5 (2012), pp. 44–48 (cit. on p. 15).

[59] Zafar Ayyub Qazi, Cheng-Chun Tu, Luis Chiang, Rui Miao, Vyas Sekar, and Minlan Yu. "SIMPLE-fying middlebox policy enforcement using SDN." In: *ACM SIGCOMM Computer Communication Review*. Vol. 43. 4. ACM. 2013, pp. 27–38 (cit. on p. 16).

[60] Fida-E Zaheer, Jin Xiao, and Raouf Boutaba. "Multi-provider service negotiation and contracting in network virtualization." In: *Network Operations and Management Symposium (NOMS), 2010 IEEE*. IEEE. 2010, pp. 471–478 (cit. on pp. 18, 48, 49, 51, 77).

[61] Paul Klemperer. "Auctions: theory and practice." In: *SSRN 491563* (2004) (cit. on p. 18).

[62] 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*. ACM. 2010, pp. 49–56 (cit. on pp. 18, 49, 52, 77).

[63] Ines Houidi, Wajdi Louati, Walid Ben Ameur, and Djamal Zeghlache. "Virtual network provisioning across multiple substrate networks." In: *Computer Networks* 55.4 (2011), pp. 1011–1023 (cit. on pp. 18–20, 26, 27, 30, 41, 42, 48, 99).

[64] Panagiotis Papadimitriou, Ines Houidi, Wajdi Louati, Djamal Zeghlache, Christoph Werle, Roland Bless, and Laurent Mathy. "Towards large-scale network virtualization." In: *Wired/Wireless Internet Communication*. Springer, 2012, pp. 13–25 (cit. on p. 18).

[65] Aris Leivadeas, Chrysa Papagianni, and Symeon Papavassiliou. "Efficient resource mapping framework over networked clouds via iterated local search-based request partitioning." In: *IEEE Transactions on Parallel and Distributed Systems* 24.6 (2013), pp. 1077–1086 (cit. on pp. 19, 48, 99).

[66] Andreas Fischer, Juan Felipe Botero, Michael Till Beck, Hermann De Meer, and Xavier Hesselbach. "Virtual network embedding: A survey." In: *IEEE Communications Surveys & Tutorials* 15.4 (2013), pp. 1888–1906 (cit. on p. 19).

[67] Ines Houidi, Wajdi Louati, and Djamal Zeghlache. "A distributed virtual network mapping algorithm." In: *IEEE International Conference on Communications (ICC'08)*. IEEE. 2008, pp. 5634–5640 (cit. on pp. 19, 26, 30, 47, 51, 76).

[68] 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)*. IEEE. 2011 (cit. on pp. 19, 20).

[69] Chrysa Papagianni, Aris Leivadeas, Symeon Papavassiliou, Vasilis Maglaris, Cristina Cervello-Pastor, and Alvaro Monje. "On the optimal allocation of virtual resources in cloud computing networks." In: *IEEE Transactions on Computers* 62.6 (2013), pp. 1060–1071 (cit. on p. 20).

[70] NM Chowdhury, Muntasir Raihan Rahman, and Raouf Boutaba. "Virtual network embedding with coordinated node and link mapping." In: *IEEE INFOCOM 2009*. IEEE. 2009, pp. 783–791 (cit. on pp. 20, 26, 30, 41, 42, 47, 48, 51, 76).

[71] Flavio Esposito and Ibrahim Matta. "A decomposition-based architecture for distributed virtual network embedding." In: *Proceedings of the 2014 ACM SIGCOMM workshop on Distributed cloud computing*. ACM. 2014, pp. 53–58 (cit. on p. 20).

[72] Abdallah Jarray and Ahmed Karmouch. "Decomposition Approaches for Virtual Network Embedding With One-Shot Node and Link Mapping." In: *IEEE Transactions on Networking* 23.3 (2015), pp. 1012–1025 (cit. on p. 20).

[73] Jing Lu and Jonathan Turner. "Efficient mapping of virtual networks onto a shared substrate." In: *Technical Report, WUCSE-2006-35, Washington University* (2006) (cit. on p. 20).

[74] Xiang Cheng, Sen Su, Zhongbao Zhang, Hanchi Wang, Fangchun Yang, Yan Luo, and Jie Wang. "Virtual network embedding through topology-aware node ranking." In: *ACM SIGCOMM Computer Communication Review* 41.2 (2011), pp. 38–47 (cit. on p. 20).

[75] Ilhem Fajjari, Nadjib Aitsaadi, Guy Pujolle, and Hubert Zimmermann. "VNE-AC: Virtual network embedding algorithm based on ant colony metaheuristic." In: *2011 IEEE International Conference on Communications (ICC)*. IEEE. 2011 (cit. on p. 20).

[76] Rashid Mijumbi, Juan-Luis Gorricho, Joan Serrat, Maxim Claeys, Filip De Turck, and Steven Latré. "Design and evaluation of learning algorithms for dynamic resource management in virtual networks." In: *2014 IEEE Network Operations and Management Symposium (NOMS)*. IEEE. 2014 (cit. on p. 20).

[77] Xuzhou Chen, Yan Luo, and Jie Wang. "Virtual network embedding with border matching." In: *2012 Fourth International Conference on Communication Systems and Networks (COMSNETS)*. IEEE. 2012 (cit. on p. 20).

[78] 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*. ACM. 2009, pp. 81–88 (cit. on pp. 20, 26, 30, 41, 42, 47, 51, 76).

[79] Minlan Yu, Yung Yi, Jennifer Rexford, and Mung Chiang. "Rethinking virtual network embedding: substrate support for path splitting and migration." In: *ACM SIGCOMM Computer Communication Review* 38.2 (2008), pp. 17–29 (cit. on pp. 20, 26, 30, 41, 42, 47, 51, 53, 76).

[80] Carlo Fuerst, Stefan Schmid, and Anja Feldmann. "Virtual network embedding with collocation: Benefits and limitations of pre-clustering." In: *Cloud Networking (CloudNet), 2013 IEEE 2nd International Conference on*. IEEE. 2013, pp. 91–98 (cit. on p. 20).

[81] Zhiping Cai, Fang Liu, Nong Xiao, Qiang Liu, and Zhiying Wang. "Virtual network embedding for evolving networks." In: *2010 IEEE Global Telecommunications Conference (GLOBECOM 2010)*. IEEE. 2010 (cit. on p. 20).

[82] Abdallah Jarray, Yihong Song, and Ahmed Karmouch. "Resilient virtual network embedding." In: *2013 IEEE International Conference on Communications (ICC)*. IEEE. 2013, pp. 3461–3465 (cit. on p. 20).

[83] Matthias Rost, Stefan Schmid, and Anja Feldmann. "It's About Time: On Optimal Virtual Network Embeddings under Temporal Flexibilities." In: *2014 IEEE 28th International Parallel and Distributed Processing Symposium*. IEEE. 2014, pp. 17–26 (cit. on p. 20).

[84] Yong Zhu and Mostafa H Ammar. "Algorithms for Assigning Substrate Network Resources to Virtual Network Components." In: *IEEE INFOCOM*. 2006 (cit. on pp. 21, 26, 30, 41, 42, 47, 51, 76).

[85] Juan Felipe Botero, Xavier Hesselbach, Andreas Fischer, and Hermann De Meer. "Optimal mapping of virtual networks with hidden hops." In: *Telecommunication Systems* 51.4 (2012), pp. 273–282 (cit. on p. 21).

[86] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998 (cit. on p. 22).

[87] David Dietrich, Amr Rizk, and Panagiotis Papadimitriou. "Multi-domain virtual network embedding with limited information disclosure." In: *IFIP Networking Conference, 2013*. IEEE. 2013 (cit. on pp. 23, 48, 58, 99).

[88] David Dietrich, Amr Rizk, and Panagiotis Papadimitriou. "Multi-Provider Virtual Network Embedding with Limited Information Disclosure." In: *IEEE Transactions on Network and Service Management* 12.2 (2015), pp. 188–201 (cit. on p. 23).

[89] *Amazon EC2 Instance Types*. URL: http://aws.amazon.com/ec2/instance-types/ (cit. on pp. 24, 81).

[90] Neil Spring, Ratul Mahajan, David Wetherall, and Thomas Anderson. "Measuring ISP topologies with Rocketfuel." In: *IEEE/ACM Transactions on Networking* 12.1 (2004), pp. 2–16 (cit. on pp. 24, 81).

[91] Benoit Donnet and Timur Friedman. "Internet topology discovery: a survey." In: *IEEE Communications Surveys & Tutorials* 9.4 (2007), pp. 56–69 (cit. on p. 24).

[92] *University of Oregon Route Views Project*. URL: http://www.routeviews.org/ (cit. on p. 24).

[93] *The Internet2 Community*. URL: https://www.internet2.edu/products-services/advanced-networking/ (cit. on p. 24).

[94] *Dr. Peering*. URL: http://www.drpeering.net/ (cit. on p. 24).

[95] *DE-CIX - German Internet Exchange*. URL: http://www.de-cix.net/ (cit. on p. 24).

[96] *AMS-IX - Amsterdam Internet Exchange*. URL: https://ams-ix.net/ (cit. on p. 24).

[97] Bernhard Ager, Nikolaos Chatzis, Anja Feldmann, Nadi Sarrar, Steve Uhlig, and Walter Willinger. "Anatomy of a large European IXP." In: *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*. ACM. 2012, pp. 163–174 (cit. on p. 24).

[98] *Peering DB*. URL: http://www.peeringdb.com/ (cit. on p. 24).

[99] Houssem Medhioub, Ines Houidi, Wajdi Louati, and Djamal Zeghlache. "Design, implementation and evaluation of virtual resource description and clustering framework." In: *Advanced Information Networking and Applications (AINA), 2011 IEEE International Conference on*. IEEE. 2011, pp. 83–89 (cit. on p. 28).

[100] 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*. ACM. 2010, pp. 41–48 (cit. on p. 28).

[101] JA Tomlin. "Minimum-cost multicommodity network flows." In: *Operations Research* 14.1 (1966), pp. 45–51 (cit. on p. 37).

[102] *IGen Network Topology Generator*. URL: http://informatique.umons.ac.be/networks/igen (cit. on pp. 40, 69, 102).

[103] Bernard M Waxman. "Routing of multipoint connections." In: *IEEE Journal on Selected Areas in Communications* 6.9 (1988), pp. 1617–1622 (cit. on p. 41).

[104] Floriana Esposito, Donato Di Paola, and Ibrahim Matta. "A general distributed approach to slice embedding with guarantees." In: *IFIP Networking Conference, 2013*. IEEE. 2013 (cit. on pp. 48, 49).

[105] Lawrence M Ausubel, Paul Milgrom, et al. "The lovely but lonely Vickrey auction." In: *Combinatorial auctions* 17 (2006), pp. 22–26 (cit. on p. 49).

[106] David Dietrich and Panagiotis Papadimitriou. "Policy-compliant virtual network embedding." In: *Networking Conference, 2014 IFIP*. IEEE. 2014 (cit. on p. 51).

[107] Bruno Quoitin, Virginie Van den Schrieck, Pierre François, and Olivier Bonaventure. "IGen: Generation of router-level internet topologies through network design heuristics." In: *2009 21st International Teletraffic Congress. ITC 21 2009*. IEEE. 2009 (cit. on p. 69).

[108] David Dietrich, Ahmed Abujoda, and Panagiotis Papadimitriou. "Network service embedding across multiple providers with nestor." In: *IFIP Networking Conference (IFIP Networking), 2015*. IEEE. 2015 (cit. on p. 79).

[109] Neil T Spring and David Wetherall. "A protocol-independent technique for eliminating redundant network traffic." In: *ACM SIGCOMM Computer Communication Review* 30.4 (2000), pp. 87–95 (cit. on pp. 79, 80).

[110] Bhavish Agarwal, Aditya Akella, Ashok Anand, Athula Balachandran, Pushkar Chitnis, Chitra Muthukrishnan, Ramachandran Ramjee, and George Varghese. "EndRE: An End-System Redundancy Elimination Service for Enterprises." In: *NSDI*. 2010, pp. 419–432 (cit. on pp. 79, 80).

[111] Christos Xenakis, Nikolaos Laoutaris, Lazaros Merakos, and Ioannis Stavrakakis. "A generic characterization of the overheads imposed by IPsec and associated cryptographic algorithms." In: *Computer Networks* 50.17 (2006), pp. 3225–3241 (cit. on pp. 79, 80).

[112] Mihai Dobrescu, Norbert Egi, Katerina Argyraki, Byung-Gon Chun, Kevin Fall, Gianluca Iannaccone, Allan Knies, Maziar Manesh, and Sylvia Ratnasamy. "RouteBricks: exploiting parallelism to scale software routers." In: *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*. ACM. 2009, pp. 15–28 (cit. on p. 80).

[113] Mihai Dobrescu, Katerina Argyraki, and Sylvia Ratnasamy. "Toward Predictable Performance in Software Packet-Processing Platforms." In: *In Proc. NSDI*. 2012 (cit. on p. 80).

[114] Qiang Wu and Tilman Wolf. "Runtime task allocation in multicore packet processing systems." In: *IEEE Transactions on Parallel and Distributed Systems* 23.10 (2012), pp. 1934–1943 (cit. on p. 80).

[115] Theophilus Benson, Aditya Akella, Anees Shaikh, and Sambit Sahu. "CloudNaaS: a cloud networking platform for enterprise applications." In: *Proceedings of the 2nd ACM Symposium on Cloud Computing*. ACM. 2011, p. 8 (cit. on p. 99).

[116] Hitesh Ballani, Paolo Costa, Thomas Karagiannis, and Ant Rowstron. "Towards predictable datacenter networks." In: *ACM SIGCOMM Computer Communication Review*. Vol. 41. 4. ACM. 2011, pp. 242–253 (cit. on p. 99).

[117] Chuanxiong Guo, Guohan Lu, Helen J Wang, Shuang Yang, Chao Kong, Peng Sun, Wenfei Wu, and Yongguang Zhang. "Secondnet: a data center network virtualization architecture with bandwidth guarantees." In: *Proceedings of the 6th International Conference*. ACM. 2010, p. 15 (cit. on p. 99).

[118] Jeongkeun Lee, Myungjin Lee, Lucian Popa, Yoshio Turner, Sujata Banerjee, Puneet Sharma, and Bryan Stephenson. "CloudMirror: Application-aware bandwidth reservations in the cloud." In: *USENIX HotCloud* 20 (2013) (cit. on p. 99).

[119] Xin Huang, Sivakumar Ganapathy, and Tilman Wolf. "A scalable distributed routing protocol for networks with data-path services." In: *IEEE International Conference on Network Protocols, 2008. ICNP 2008*. IEEE. 2008, pp. 318–327 (cit. on p. 99).

[120] Ahmed Abujoda and Panagiotis Papadimitriou. "MIDAS: Middlebox Discovery and Selection for On-Path Flow Processing." In: *IEEE COMSNETS, Bangalore, India* (2015) (cit. on p. 99).

[121] *IBM ILOG CPLEX Optimizer*. URL: `http://www.ibm.com/software/commerce/optimization/cplex-optimizer` (cit. on p. 102).

[122] *GNU Linear Programming Kit*. URL: `http://www.gnu.org/software/glpk/` (cit. on p. 102).

[123] *Boost C++ Libraries - The Boost Graph Library*. URL: `http://www.boost.org/doc/libs/` (cit. on p. 102).

[124] *Qt project*. URL: `http://www.qt-project.org` (cit. on p. 105).

[125] David Dietrich, Amr Rizk, and Panagiotis Papadimitriou. "AutoEmbed: automated multi-provider virtual network embedding." In: *ACM SIGCOMM Computer Communication Review*. Vol. 43. 4. ACM. 2013, pp. 465–466 (cit. on p. 107).

PUBLICATIONS

D. Dietrich, A. Rizk, and P. Papadimitriou. "Multi-Domain Virtual Network Embedding with Limited Information Disclosure." In: *Proc. IFIP Networking*, New York, USA, May 2013.

D. Dietrich, A. Rizk, and P. Papadimitriou. "AutoEmbed: Automated Multi-Provider Virtual Network Embedding." Demo, *ACM SIGCOMM*, Hong Kong, China, August 2013.

D. Dietrich and P. Papadimitriou. "Policy-Compliant Virtual Network Embedding." In: *Proc. IFIP Networking*, Trondheim, Norway, June 2014.

D. Dietrich, A. Abujoda, and P. Papadimitriou. "Embedding Network Services across Multiple Providers with Nestor." In: *Proc. IFIP Networking*, Toulouse, France, May 2015.

D. Dietrich, A. Rizk, and P. Papadimitriou. "Multi-Provider Virtual Network Embedding with Limited Information Disclosure." In: *IEEE Transactions on Network and Service Management*, Vol.12, No.2, pp.188-201, June 2015.

CURRICULUM VITAE

Name	David Dietrich
Day of birth	22 August 1977

Education

January 2010 to present	**Ph.D.** student Leibniz Universität Hannover. Thesis Title: Multi-Provider Network Service Embedding
April 2003 to March 2006	**M.Sc., Dipl.-Ing.** in Electrical Engineering Universität Kassel Thesis Title: Automated Measurement Data Processing for Building Process Models
October 1998 to March 2003	**Dipl.-Ing.(FH)** in Computer Science Hochschule Bremen (University of Applied Sciences) Thesis Title: Implementation of a CRM Database System (at the IT-Management Department of Daimler AG, Bremen)

Work Experience

January 2010 to present	**Leibniz Universität Hannover**, Institute of Communications Technology *Research Assistant*
June 2006 to December 2009	**Carts GmbH**, Kassel *Software Engineer*
April 2003 to September 2006	**Universität Kassel**, Institute of Drive Technology *Technical Staff*
August 2002 to February 2003	**Daimler AG**, IT-Management Department, Bremen *Internship*
September 2000 to February 2001	**EADS / Astrium GmbH**, Space Infrastructure Division, Bremen *Internship*

Teaching Experience

	Leibniz Universität Hannover, Faculty of Electrical Engineering and Computer Science
Academic year of 2011 to 2016	Exercises on "Network Management"
Academic year of 2010 to 2014	Exercises on "Future Internet Communication Technologies"
Academic year of 2011 to 2016	Co-advising several student projects, B.Sc. and M.Sc. theses

Research Projects

2014 to 2016	T-NOVA: Network Functions as-a-Service Over Virtualised Infrastructures *EU Framework Programme 7 (FP7)*
2014	CONFINE: Community Networks Testbed for the Future Internet *EU Framework Programme 7 (FP7)*
2013	ProCloud: Resource Procurement for Network Slicing in Multi-Provider Clouds *L3S / Ministry of Science and Culture, Lower Saxony, Germany*
2011 to 2012	LAVINET: Enabling Large-Scale Network Virtualization *L3S / Ministry of Science and Culture, Lower Saxony, Germany*
2010 to 2011	G-Lab VIRTURAMA: Virtual Routers: Architecture, Management and Applications *Federal Ministry of Education and Research, Germany*
2003 to 2004	Power Optimised Aircraft (POA) *EU Framework Programme 5 (FP5)*

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*".

Final Version as of March 21, 2016 (`classicthesis`).