Design and Evaluation of Cloud Network Optimization Algorithms

Dallal Belabed

UNIVERSITÉ PIERRE ET MARIE CURIE
Laboratoire d'Informatique de Paris 6

Design and Evaluation of Cloud Network Optimization Algorithms

Doctoral Dissertation of:
Dallal Belabed

Advisor:
Dr Stefano Secci

2015

Thèse

Présentée pour obtenir le grade de docteur
de l'Université Pierre et Marie Curie
Spécialité: Informatique

Dallal BELABED

Optimisation des réseaux virtuels : conception et évaluation

Soutenue le 24 avril 2015 devant le jury composé de:

Rapporteurs	Dr Guido MAIER	Ecole Polytechnique de Milan, Italie
	Prof. Michele NOGUEIRA	Université Fédérale du Paraná, Brésil.
Examinateurs	Dr Bernardetta ADDIS	Université de Lorraine, France.
	Dr Mathieu BOUET	Thales Communications & Security S.A.S, France.
	Dr Prosper CHEMOUIL	Orange Labs, France.
	Prof. Deep MEDHI	University of Missouri-Kansas City, Etats-Unis.
	Prof. Guy PUJOLLE	Université Pierre et Marie Curie, France.
Président	Dr Prosper CHEMOUIL	Orange Labs, France.
Directeur de thèse	Dr Stefano SECCI	Université Pierre et Marie Curie, France.

UNIVERSITÉ PIERRE ET MARIE CURIE
Laboratoire d'Informatique de Paris 6

Design and Evaluation of Cloud Network Optimization Algorithms

Author: Dallal BELABED

Defended on April 24, 2015, in front of the committee composed of:

Referees: Dr Guido MAIER (Politecnico di Milano, Italy).
Prof. Michele NOGUEIRA (Federal University of Paraná, Brazil).

Examiners: Dr Bernardetta ADDIS (Université de Lorraine, France).
Dr Mathieu BOUET (Thales Communications & Security S.A.S, France).
Dr Prosper CHEMOUIL (Orange Labs, France).
Prof. Deep MEDHI (University of Missouri-Kansas City, USA).
Prof. Guy PUJOLLE (Université Pierre et Marie Curie, France).

President: Dr Prosper CHEMOUIL (Orange Labs, France).

Advisor: Dr Stefano SECCI (Université Pierre et Marie Curie, France).

Remerciements

En tout premier lieu ma sincère reconnaissance va à Monsieur Stefano Secci de m'avoir fait l'honneur d'accepter d'être l'encadrant de la présente thèse au courant d'un certain mois de Juin de l'an 2011 et d'avoir permis, à celle-ci, de longer les rails qui l'ont menée sous la lumière du jour et la font exister aujourd'hui. Aussi, j'adresse un grand respect et un remerciement chaleureux à l'égard de Guy Pujolle qui m'a accueilli au sein de son équipe. Je les remercie également de m'avoir accompagnée durant cette aventure en guidant mes pas, en m'accordant leur confiance et en orientant notre collaboration durant ces années. Sans leur implication, ce travail n'aurait pas abouti.

Ma gratitude s'adresse à Deep Medhi pour son accueil bienveillant à Kansas City, pour son soutien constant, pour le travail accompli ensemble, pour sa disponibilité, pour ses encouragements, pour son immense savoir et pour ses précieux conseils. Ainsi je ne saurais omettre Monsieur Guido Maier et Madame Michele Nogueira d' avoir agréé de reporter cette thèse.

Une particulière émotion pleine d'humilité et de gratitude va à l'adresse de Monsieur Prosper Chemouil qui a bien voulu honorer la conclusion de mon travail à travers sa participation à ma soutenance et via la présidence du jury, sachant qu'il vient à cela en défiance d'un repos personnel, en interrompant ses propres vacances pour répondre à cette mission qu'il s'incombe avec altruisme et je ne saurais assez le remercier pour cela.

Je remercie aussi Madame Bernardetta Addis et Mathieu Bouet pour leur travail collaboratif ainsi que d'avoir accepté de constituer le jury devant lequel je soumets ma thèse.

Je remercie également mes chers instituteurs et professeurs, avec le regret de ne pas pouvoir citer tous leurs noms, eux qui ont fait de moi ce que je suis devenue: Mme Boumrar, Mme Adel, Mme Djabra, Mme Hamza, Mr Hamrour, Professeur Ighilaza, Professeur Mohamed Bentarzi, Professeur Bénédicte le grand, à la mémoire d'un grand homme Professeur M'hamed MEKLATI, je vous serai à jamais

reconnaissante.

Mes remerciements vont à tous les membres des équipes Phare à l'UPMC et du Computer Science & Electrical Engineering Department à l'université de Kansas-City avec qui j'ai eu le plaisir d'échanger pendant ma thèse, sans oublier aussi les membres des autres équipes du LIP6, particulièrement, les membres de l'équipe Complex Networks, merci au 15mn max.

Chaleureuses pensées à mes amis des deux cotés de la rive de la méditerranée, mes amis d'enfance et ceux qui le sont devenus plus tard, pour avoir été là pendant toutes ces années.

Les mots ne suffisent pas pour remercier mes parents, mon frère et mes soeurs sans oublier ma grand mère mes oncles et ma tante et toute ma famille pour leur appui et pour avoir toujours cru en moi - une pensée particulière à ma soeur chérie Hanane.

Enfin, le meilleur pour la fin : merci à l'amour de ma vie pour son soutien inconditionnel pour son amour fou, il faut l'être pour épouser une doctorante.

Ce travail est aussi le fruit de leur patience. Merci

Abstract

The rapid development of software virtualization solutions have led to huge innovation in Data Center Networks (DCN), with additional capabilities to perform advanced network functions via software elements. Novel protocols designed in the recent years allow for a large path diversity at the edge server level by means of multipath forwarding protocols and virtual bridging functions. Moreover, increasing features are given to DCN elements that become programmable so as to allow, for instance, turning off and on virtual servers for energy consolidation, migrating virtual machines on demand or automatically based on variation of system and network states to improve performance, etc. In this scope, this dissertation addresses a number of research questions.

The first research question to which we answer is at which extent and in which situations performing multipath forwarding and virtual bridging in DCNs can be beneficial when performing data center optimizations meeting traffic engineering and energy efficiency goals. We formally formulate the problem of virtual machine placement aware of server and network states, as well as aware of the capability to perform multipath forwarding and virtual bridging. We propose a heuristic approach for its resolution. We give many insights, showing in particular that virtual bridging brings high performance gains when traffic engineering is the primary goal, and should be deactivated when energy efficiency becomes important. We also determine that multipath forwarding brings relevant gains only when energy efficiency is the primary goal and virtual bridging is not enabled.

In a second contribution our focus moves toward the analysis of the relationship between novel flattened and modular DCN architectures and congestion control protocols. In fact, one of the major concerns in congestion control being the fairness in the offered throughput, the impact of the additional path diversity, brought by the novel DCN architectures and protocols, on the throughput of individual endpoints (servers) and aggregation points (edge switches) was an aspect not clearly addressed in the literature. Our contribution consists in providing a novel com-

prehensive mathematical programming formulation of the throughput optimization problem based on the proportional fairness principle of the Transport Control Protocol (TCP), and in an extensive series of experiments conducted following the model. We find how much and why the traffic allocation fairness is impacted by the type of DCN architecture employed and by the adopted TCP variant.

Finally, in the third contribution we investigate a novel rising virtualization resource orchestration problem in Network Functions Virtualization (NFV) architectures for carrier networks. We define the rising problem of optimally placing Virtual Network Functions and chaining virtualized network functions in the delivery of carrier network services. We propose a mathematical programming formulation taking into consideration both NFV and traffic engineering costs, and describe a matheuristic approach. We draw quantitatively and qualitatively many insights on the design of NFV infrastructures across carrier network layers (access, aggregation, core) also depending on the type of virtualized network function.

Résumé en Langue Française

L'émergence rapide des solutions de virtualisation ont grandement participé au développement et à l'amélioration des réseaux de centre de données, en tirant profit de nouveaux concepts, comme l'utilisation des fonctions du réseau implémentée en logiciel. Ces dernières années, de nouveaux protocoles permettant une grande diversité de chemins au niveau de la bordure du réseau ont été conçus, et de nouvelles fonctionnalités telles que la virtualisation des commutateurs ont été introduites. Ainsi, de plus en plus de programmabilité est développée pour les réseaux de centre de données et, plus génériquement, pour les réseaux d'opérateurs. Ceci, à titre d'exemple, permet d'éteindre des serveurs virtuels pour économiser de l'énergie, en migrant les machines virtuelles à la demande ou automatiquement en fonction de l'état du réseau afin d'améliorer les performances. C'est dans cette optique que notre thèse s'inscrit afin de proposer des méthodes de résolution de plusieurs problématiques d'optimisation des réseaux.

L'objet initial de notre quête se concentre sur l'impact de certaines de ces fonctionnalités dans l'optimisation des centres de données, et plus précisement dans le placement de machines virtuelles. Nous étudions dans quels cadres et dans quelles situations la diversité des chemins et la virtualisation des bridges dans les réseaux de centre de données sont-elles bénéfiques vis-à-vis des objectifs d'optimisation des réseaux, comme par exemple en termes d'ingénierie de trafic et d'efficacité énergétique. Pour répondre à ce type d'intérrogation, nous formulons le problème de placement des machines virtuelles, en prenant en compte des contraintes liées à la capacité des serveurs, mais aussi, des contraintes liées à l'état du réseau, ainsi que celles introduites par l'utilisation du multi-chemin et de l'utilisation des commutateurs virtuels, que nous résolvons en utilisant un algorithme héuristique. Nous avons trouvé, à travers de multiples simulations, que l'utilisation de commutateurs virtuels a un impact positif sur la performance, mais cela seulement quand des objectifs d'ingénierie du trafic sont adoptés dans l'optimisation, et doit être désactivée quand le but est l'économie d'énergie. Concernant la diversité des chemins, elle

apporte un gain mais seulement dans le cas où l'économie d'énergie est l'unique but de l'optimisation et la commutation virtuelle est désactivée.

Ensuite, dans la deuxième partie de la thèse, nous adressons notre attention vers une meilleure compréhension de l'impact de nouvelles architectures sur le contrôle de congestion, et vice versa. En effet, l'une des préoccupations majeures dans le contrôle de congestion est l'équité dans le débit offert. L'impact du routage multi-chemins introduit par les nouvelles architectures de centre de données, et par les nouveaux protocoles, sur le débit des points terminaux (serveurs) et des points d'agrégations (commutateurs), n'a pas été étudié dans l'état de l'art. C'est pourquoi dans cette partie nous essayons de répondre à ces intérrogations. En outre, notre contribution consiste à fournir une formulation mathématique du problème, en apportant un modèle linéaire qui intègre à la fois le principe de l'équité proportionnelle dans le protocole TCP (Transport Control Protocol) et aussi la possible de transmettre le traffic via plusieurs chemins en parallèle. Nous avons effectué plusieurs séries de tests dans différents scénarii et avec différentes architectures. Nous avons qualifié comment l'équité proportionnelle de TCP est impactée par le type d'architecture DCN employé et par la variante de TCP adoptée.

Enfin, dans la troisième partie de la thèse nous ménons une étude préliminaire sur un nouveau paradigme dans les réseaux d'opérateur introduit par la possibilité de virtualiser les fonctions de réseaux nommées "Virtual Network Functions (VNF)", au sein des réseaux d'opérateurs. Nous avons formulé le problème en un modèle linéaire générique, incluant les nouvelles contraintes liées aux particularités introduites par les VNFs, comme la variation de la latence de commutation en fonction du volume du trafic et du débit en fonction du type de VNF, en intégrant deux objectifs d'optimisation, un suivant les besoins en termes d'ingénierie de trafic et de minimisation du cout de virtualisation. Nous avons appliqué une approche dite de 'math-heuristique " de manière à trouver le meilleur itinéraire à travers le réseau e le placement de VNFs pour servir au mieux les demandes des clients.

Contents

List of Figures

List of Tables

Acronyms

3GPP: 3rd Generation Partnership Project
AMSTP: Alternative Multiple Spanning Tree Protocol
BGP: Border Gate Protocol
BID: Bridge Identifier
BPDU: Bridge Protocol Data Unit
CMS: Cambridge Monitor System
CP: Control Program
CPE: Customer Premises Equipment
CPU: Central Processing Unit
DC: Data Center
DCN: Data Center Networking
DEI: Drop eligible indicator
ECMP: Equal Cost Multi-Path
EE: Energy Efficiency
ETSI: European Telecommunications Standards Institute
EVC: Ethernet Virtual Connections
FEC: Forwarding Equivalence Class FIB: Forwarding Information Base
GMPLS: Generalized Multi-Protocol Label Switching
GOE: Global Open Ethernet
GRE: Generic Routing Encapsulation
HER: Head-End Replication
IaaS: Infrastructure as a Service
IETF: Internet Engineering Task Force
IGP: Interior Gateway Protocol
IMS: IP Multimedia Subsystem
IP: Internet Protocol
ISIS: Intermediate System to Intermediate
ISL: Inter-Switch Link

IST: Internal Spanning Tree
IVT: Intel Virtualization Technology
L2LSP: Layer 2 Label Switched Paths
LAN: Local Area Network
LISP: Locator/ Identifier Separation Protocol
LP: Linear Programming
LSP: Label Switched Path
MAN: Metropolitan Area Network
MDC: Modular Data Center
MIP: Mixed Integer Programming
MISTP: Multiple Instances Spanning Tree Protocol
MMF: Max-Min Fairness
MST: Multiple Spanning Tree
MSTP: Multiple Spanning Tree Protocol
MTCP: Multipath TCP
NaaS: Network as a service
NaaS: Network as a Service
NFV: Network Functions Virtualization
NGN: Next Generation Network
NIC: Network interface card
NSP: Network Service Providers
NVGRE: Network Virtualization Using Generic Routing Encapsulation
ONF: Open Networking Foundation
OS: Operating System
OSPF: Open Shortest Path First
OTT: over-the-top content
PaaS: Platform as a Service
PB: Provider Bridges
PBB-TE: Backbone Bridges with Traffic Engineering
PBB: Provider Backbone Bridges
PCP: Priority Code Point
PE: Provider Edge
PF: Proportional Fairness
RB: Routing Bridge
RBridges: Routing Bridge
RSTP: Rapid Spanning Tree Protocol
RSVP: Resource Reservation Protocol
SaaS: Software as a Service
SCTP: Stream Control Transmission Protocol

SDE: Software Defined Everything
SDN: Software Defined Networking
SIP: Session Initiation Protocol
SN: Substrate Network
SPB: Shortest Path Bridging
STP: Spanning Tree Protocol
STT: State less Transport Tunneling
TCI: Tag Control Information
TCP: Transmission Control Protocol
TE: Traffic Engineering
TPID: Tag Protocol IDentifier
TRILL: Transparent Interconnection of Lots of Links
TTL: Time to Live
UDP: User Datagram Protocol
UMTS: Universal Mobile Telecommunications System
VDC: Virtual Data Center
VID: VLAN identifier
VLAN: Virtual Local Area Network
VM: Virtual Machines
VMM: Machine Monitor
VN: Virtual Network
VNE: Virtual network embedding
VNFaaS: Virtual Network Function as a Service
VSID: Virtual Subnet ID
VXLAN: Virtual Extensible LAN
WAN: Wide Area Network

Chapter 1

Introduction

In recent years, cloud computing has attracted major attention in computer networking and telecommunications due to the important gains and novel features it can bring to both end users and network providers. The gains, the deployment of cloud computing technologies can induce, can be of different nature. For example, it can lower the capital expenditures in operating a network thanks to the reduction of physical space needed to deploy network and service elements. Moreover, exploiting the fact that servers are commonly not fully used all the time, it allows a better share of the available physical resources among multiple virtual servers. Finally, it can lower the operational expenditures by allowing remote, virtual and automated management of large information systems networks running on the top of virtual machines and virtual network controllers.

Cloud computing solutions cover a wide range of functionalities and advantages, as reported in Figure 1.1. Costumers can be given remote access to their computing and storage resources, and in some setting they can develop and run their own network management platforms. Indeed, cloud computing supports the relocation of entire computer infrastructures and applications from the customer premises and devices to the cloud provider data center infrastructure. This evolution is readily in accordance with the end-to-end principle in Internet working, which suggests that service intelligence is left to edge network elements.

In so-called Infrastructure as a Service (IaaS) solutions, customers can get direct access to a set of virtual machines, with a certain guarantee of isolation on the traffic among them from other IaaS virtual networks. In IaaS offers, service-level-agreements on network and service performance are typically exposed and contractualized. A large set of choices is typically given on the Operating System (OS), levels of system resources (e.g., live memory, processors, storage, etc), levels of link resources (ingress and egress bandwidth, latency, etc), and the platforms and applications supported. A set of functionalities limited to software platform,

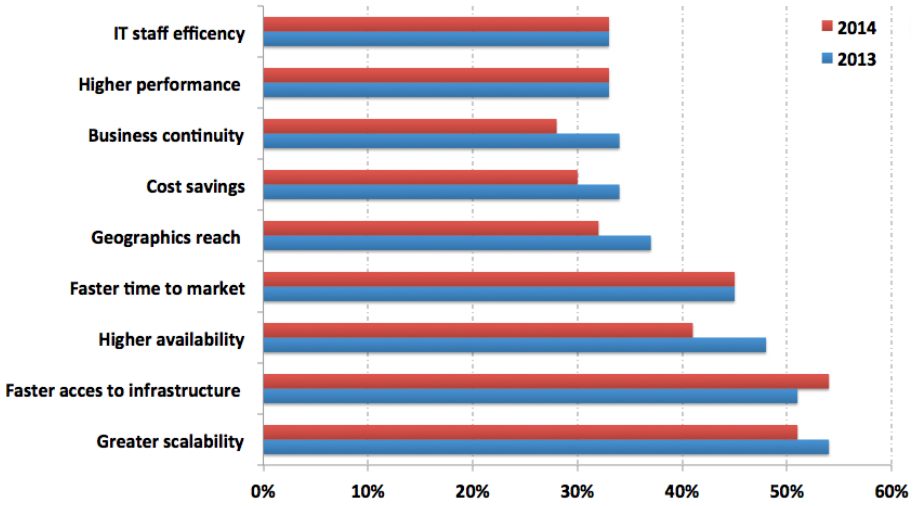


Figure 1.1: Survey results: which are the major cloud benefits. Source: RightScale 2014 state of the cloud report [2].

applications and OS variations are given to so-called Platform as a Service (PaaS) and Software as a Service (SaaS) solutions. In this dissertation we focus on IaaS solutions because they have a direct impact on data-center networking operations.

The emergence of network virtualization solutions offers several advantages to organizations in terms of both operational and capital expenditures related to IaaS operations in cloud networking [1]. The transition from physical independent networks to virtual delocalized networks operated in the cloud can be facilitated if, besides security concerns, network performance and efficiency are at an acceptable level and show desirable fairness properties.

In the following of this chapter, we will first give a general overview of what is meant with system and network virtualization. Thereafter, we introduce major issues in cloud networking and related advances in network programmability, i.e., Software Defined Networking (SDN) protocols and Network Function Virtualization (NFV).

1.1 Server Virtualization

The actual Internet architecture is facing a standoff. In the nineties, the explosion of Internet Protocol (IP)-based networks was essentially driven by the fact that the connection-less packet-switching nature of IP networks introduced a much more scalable and flexible environment to operate networks and to develop added-value services such as video-over-IP, voice-over-IP and web services. Similarly, with the emergence of cloud computing, actually the legacy IP-centric Internet architecture

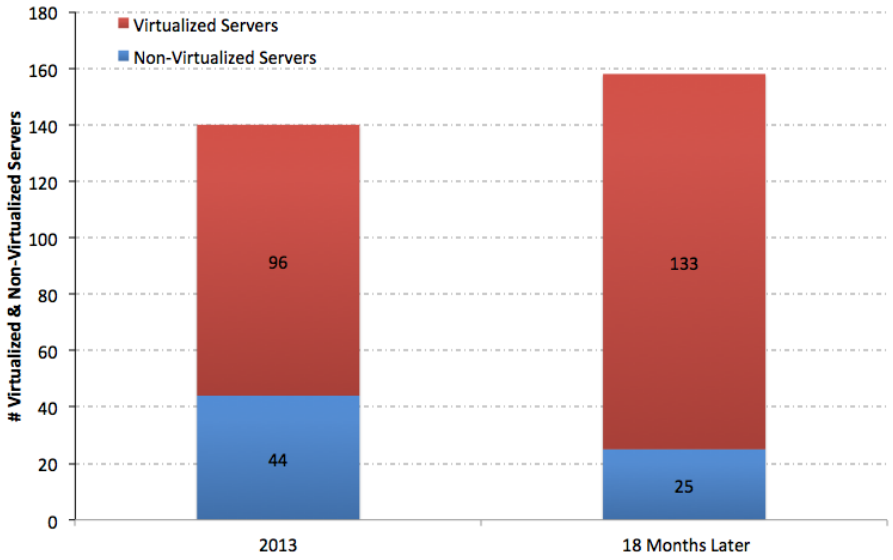


Figure 1.2: Survey results: growth of software virtualization in the Internet. Source: Wikbon survey (August 2013) [3].

appears as not sufficiently flexible to readily support the cloud computing virtualization principle.

Before presenting what virtualization technically means, let us show how it has evolved recently in terms of deployment. Figure 1.2 depicts the result of a recent survey [3] showing that the ratio of virtualized servers (i.e., servers running on the top of a physical virtualization server) worldwide has grown by 15% in only 18 months time from 2013 to 2015, while the global number of servers (virtualized or not) grew by 14%. Figure 1.3 from another survey shows that this trend is underpinned by the fact that organizations and companies move to virtualization solutions in order to reduce capital and operational expenditures: 40% of data center (DC) cloud companies believe that server virtualization has the greatest impact in the reduction of DC costs.

Finally, Figure 1.4 shows the result of another survey dating back to 2012: 64.9% of organizations, that did an early deployment of network virtualization solutions, still had challenges to address in application throughput and end-user experience; 11.7% were facing serious network complexity challenges and only 23.4% were fully satisfied. In the mean time, during the period this thesis work spanned from 2011 to 2015, huge innovation happened in this sector.

1.1.1 Virtualization: historical facts

One can set 1964 as the year of beginning of the virtualization adventure. In that year IBM began investing R&D effort in developing the 'Control Program' CP-40 [6],

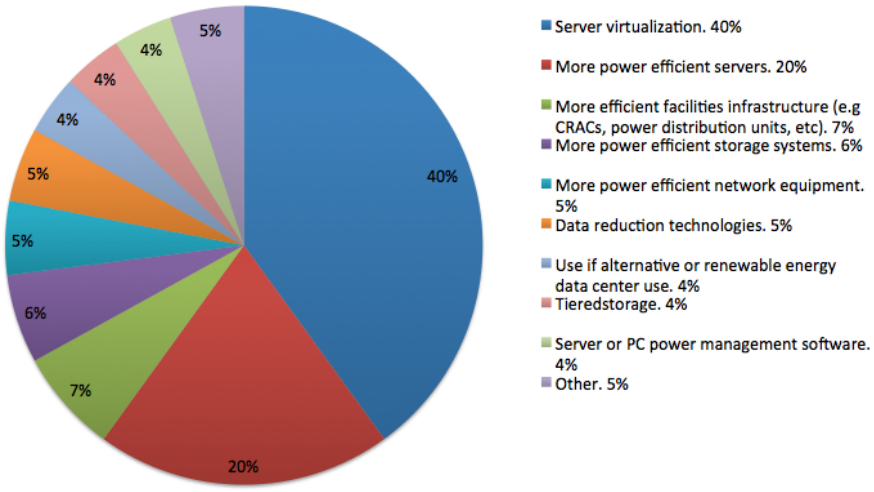


Figure 1.3: Survey results: which technology will have the greatest impact on reducing data center costs Source: SYS-CON Media [4].

that was the parent of IBM's virtual machine solution family, and the precursor of the CP-67 for the OS IBM System/360 Model 67, a 32-bit Central Processing Unit (CPU) with virtual memory hardware announced by IBM on August 1965. Essentially, the CP-40 can run multiple OS clients, in particularly the Cambridge Monitor System (CMS), where CMS is a simple interactive single-user operating system. These primitive virtualization solutions were therefore strongly relying on hardware support. Afterwards, the adventure of virtualization continues for almost two decades. The aim of virtualization at that time was to reduce the cost of the mainframe by efficiently using very expensive computing resources.

In the 1980s, the cost of computers began to decrease following the emergence of personal computers. In practice, the former virtualization solutions disappeared until late 1990s. In fact, at that period workstations and servers started to be more powerful, as captured by Moore's law.

In the beginning of the new century, novel virtualization then appeared, having as main difference with respect with 80's solutions that their virtualization control program has a much lower dependence on the hardware (in some cases, no dependence at all). Even though the technology and use cases have evolved, the core meaning of virtualization remains the same: enabling a computing environment to run multiple independent operating systems at the same time. Nowadays, the Control Program that allows physical machines to run Virtual Machines (VM) is called 'hypervisor' or 'Virtual Machine Monitor (VMM)'. VMM consists of an abstraction software layer that divides the host (physical machine) into VMs with different OSs, which share the hardware resources offered by the host.

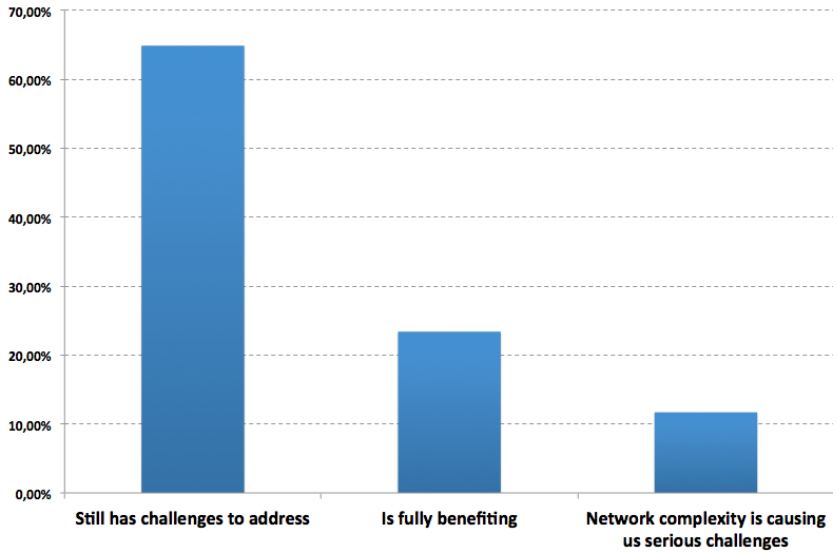


Figure 1.4: Survey results: how your data-center virtualization solution is complete. Source: Extreme Networks [5].

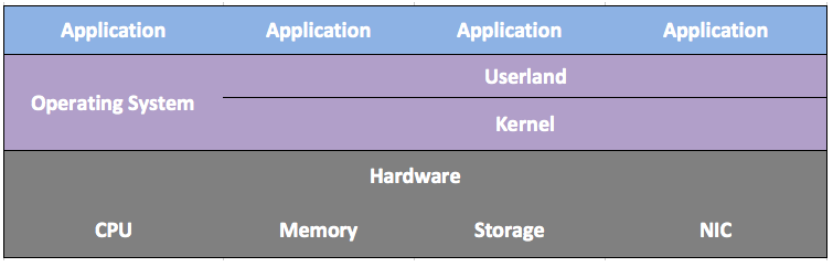


Figure 1.5: A non virtualized equipment.

1.1.2 Hypervisor or Virtual Machine Monitor Architecture

Figure 1.5 depicts the way applications access to physical interfaces via the OS. The hardware layer includes the physical resources and interfaces such as CPU, memory, storage and Network Interface Card (NIC). The operating system leaves to the kernel space the direct access to interfaces, and to the userland (or user space) the interface with the application code running.

With recent software-driven virtualization solutions, we can distinguish two hypervisor architecture modes: the hosted type and the native or bare metal mode.

Hosted Mode

The hosted mode consists of adding a software application on top of the host's OS layer to act as hypervisor. In each virtual machine, a guest OS can be installed and run any application. Figure 1.6 shows an example of the hosted mode. The

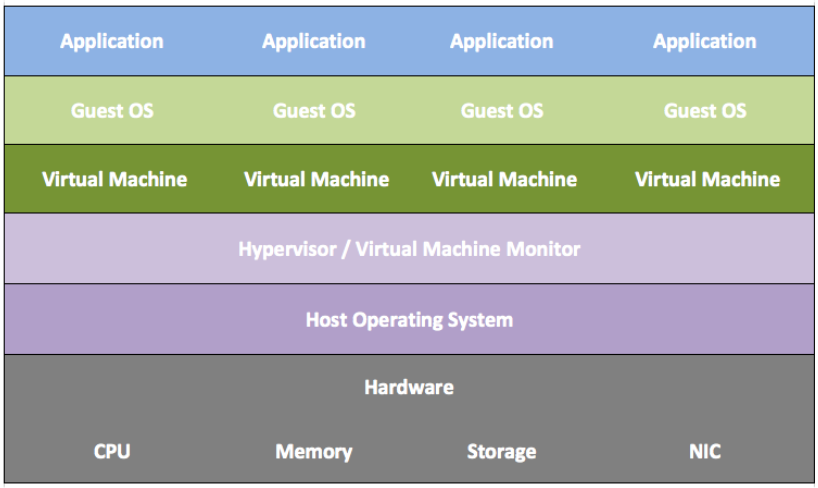


Figure 1.6: The hosted virtualization mode.

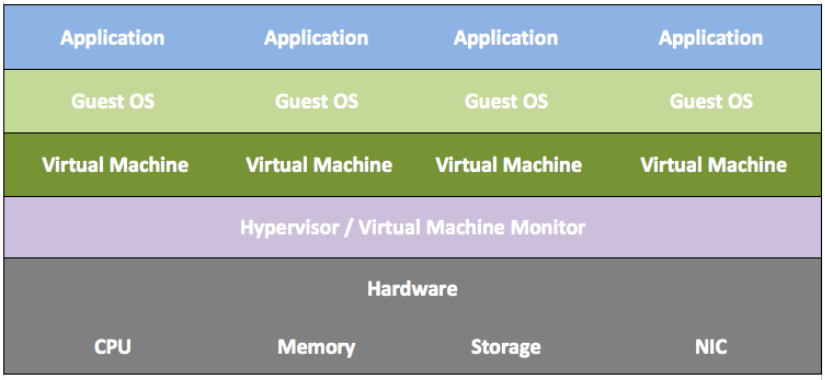


Figure 1.7: The native virtualization mode.

peculiarity is therefore that the hypervisor software is run as an application and is not run at the OS kernel level.

The hosted mode therefore targets end users and not industrial-grade virtualization servers. A well-known example of a hosted hypervisor is Oracle VM VirtualBox. Others include VMWare Workstation, Microsoft Virtual PC, and Parallels.

Native Mode

In the native virtualization mode, the hypervisor runs at the kernel level of the physical machine OS. Examples of a native mode hypervisors are: Xen, Microsoft Hyper-V server, VMWare server (ESX/NSX).

The native mode offers higher efficiency, and presented many technical difficulties at first implementations. In fact, the most widely used processors at that time were the x86, which were not compatible with native virtualization techniques.

In 1974, Popek and Glodberg [7] have introduced the conditions to be fullled by the OS to be completely virtualized. They propose principles they estimated should be fulfilled by virtualization systems. Those principles were around fairness in resource sharing among VMs (even access to resources to VMs) and application execution efficiency (possibility to give to VMs direct access to some physical resources without passing though the hypervisor). For that purpose, they classified the processor instructions to three categories:

- Privileged instructions: a processor 'op-code' (assembler instructions), which can only be executed in supervisor mode and requires a trap when used under another mode. These instructions are independent of the virtualization process.
- Sensitive instructions: instructions directly impacting the 'virtualizability' of particular machines. They are defined in two types:
  - Control sensitive: these instructions englobe all the instructions that attempt to modify the resource configuration, as mapping the virtual memory on the physical memory or configuring the global registry without passing by the subroutine trap.
  - Behavior sensitive: these instructions depend on the current machine's mode.

In 2005/2006, both Intel and AMD, with the Intel Virtualization Technology (IVT) and AMD-V solutions have introduced new instruction capabilities at the CPU level as extensions to the x86 architecture, which include enabling the complete virtualization features.

1.2 The Dawn of Cloud Networking

Executing servers as virtual machines has a direct impact on network operations. The first tangible impact is the fact that virtual machines also need to be addressed from an IP and Ethernet network perspective, transparently with respect to the physical server, by means of its virtual network interfaces. Moreover, when IaaS networks need to be operated, traffic from and between VMs of a same IaaS needs to be isolated from other virtual networks and uniquely identified at the data plane level. Furthermore, the virtualization servers expose novel traffic patterns in data-center network to support novel operations such as VM migration, storage synchronization, VM redundancy, etc.

Traditionally, data center network architectures were relying on a rather simple tree-like layer-2 Ethernet fabric, using the Spanning Tree Protocol (STP) [9], which

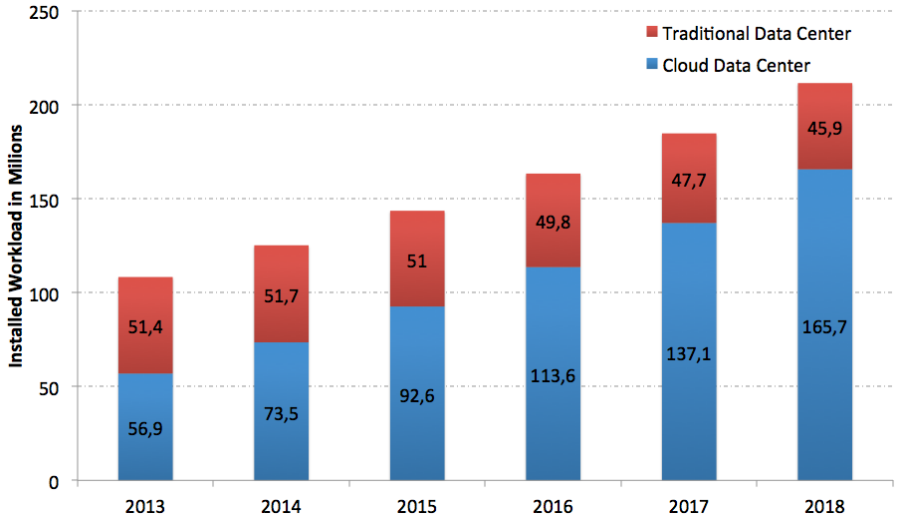


Figure 1.8: Traditional data center vs cloud data center in terms of installed computing workload. Source: Cisco global cloud index, 2013-2018 [8].

for hierarchical tree-like topologies can be considered as sufficiently meeting the connectivity requirement. With virtualization, the emergence of IaaS virtual networks with many VMs located at many distinct edge servers creates a huge amount of horizontal traffic among edge servers, especially if VM migrations happen across the servers. As later explained in the next chapter, novel data-center network topologies have been defined and deployed to meet these new requirements.

In order to qualify the impact of virtualization on data-center network architecture and operations, Figure 1.8 shows the estimated growth of computing workloads of data centers from 2013 to 2018. While for traditional DCs the growth is estimated as rather stable (slightly decreasing), for cloud DCs (based on virtualization servers) the growth is expected to nearly triple. This increase of interest in cloud networking by integrating cloud computing solutions into data-center networks, is certainly supported by economic motivations. The Figure 1.9 reports the result of a market research [10], showing that virtualization alone can bring up to 36% in capital expenditures, and that it can further unplug additional optimizations in energy reduction.

In this context, technically, many cloud networking protocols have been developed these last years in order to face novel cloud network requirements. The requirements as well as the various major cloud network overlay protocol solutions are presented later in the next chapter in section 2.1.2. Briefly, most of them rely on forms of encapsulation of packets and data frames in order to guarantee isolation, virtual network identification, separate addressing and localization functions, support multipath communications, etc.

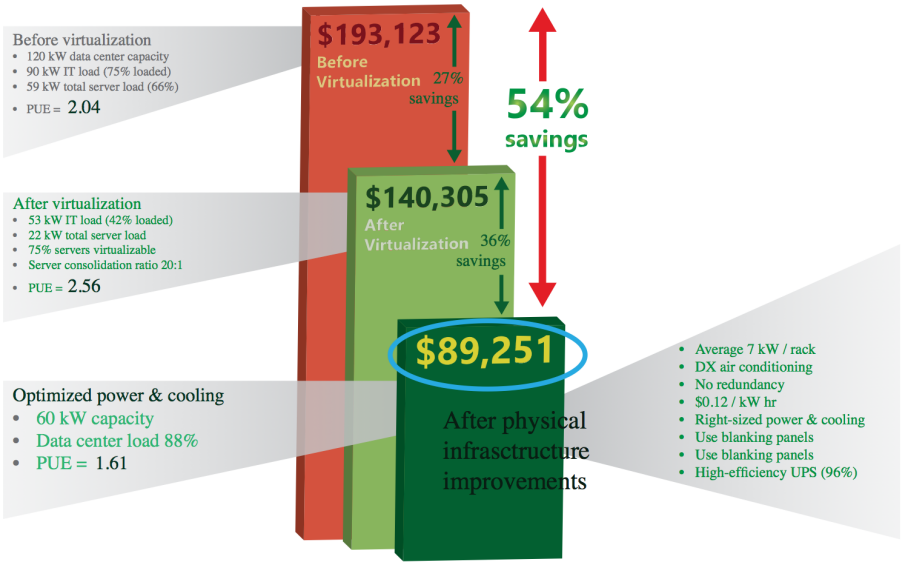


Figure 1.9: Cost benefits of virtualization. Source: tradeoff tool - TT9 Rev 0 'virtualization energy cost calculator', APC by Schneider Electric [10].

Cloud network overlay protocols can be seen as the medium term answer to the management of cloud networking issues. Recently, an increasing interest is devoted to software-defined-network (SDN) solutions, alternative and also complementary at some extent, which often do not imply changes to the data-plane as for cloud network overlay protocols.

Another recent evolution in cloud networking is also the emergence of advanced techniques to virtualize not only user-accessed servers, but also network functions such as routing, firewalling, cyphering, and other functions that are typically implemented via physical nodes. This recent trend is referred to as Network Functions Virtualization (NFV), which can also be supported by SDN solutions, as described in the following.

1.2.1 Software Defined Networking

Historically, routing protocols running in operational networks work in a 'push' mode, i.e., when a packet arrives at a routing equipment, the equipment simply applies the routing or forwarding rules already given in a pre-set routing table. The routing table is typically built by means of a distributed signaling mechanism disseminating routing information following a distance-vector procedure (such as with the Border Gateway Protocol [11]) or a link-state procedure (such as with the Open Shortest Path First protocol [12]), or a mix of these procedures, to distribute useful information needed to determine a path to reach any known destination. In

order to avoid too high signaling and too large routing table, the routing table semantic (i.e., based on which information the destination and the next hop or shortest path are determined) is limited to a single layer (for instance, to only IP or to only Ethernet addressing information).

The push mode underlying legacy routing and switching protocols has been criticized in the last decade because not flexible enough, as opposed to alternative protocols working under a 'pull' mode that, instead, suggests to avoid the dynamic synchronization of routing information by allowing a node to query a logically centralized control-plane only when needed, i.e., only when the next hop for a given packet or frame is not known based on local information. In this way, the control-plane is able to take routing decision based on the specific routing request, opening the path to much powerful traffic engineering and policing options.

The idea of separating the network control plane from the data plane firstly appeared in operational networks such as Universal Mobile Telecommunications System (UMTS) networks and IP Multimedia Subsystem (IMS) networks in the beginning of the 21^{st} century. In both cases, a clear separation between switching and routing decision functions, between data-plane and control-plane, were implemented to better meet quality of service requirements of voice and multimedia services for the landline or mobile phones.

In the last years, a protocol named OpenFlow [13] was defined as a pull routing and switching architecture such that the switching rule which is given back to the requester node can contains a very rich semantic. A switching rule can be defined as a function of multiple layer 2, 3 and 4 protocol header fields, and even further protocol layers in custom variations. In 2011, an organization named Open Networking Foundation (ONF) [14] was created in order to promote OpenFlow, referring to it as the main candidate to implement the so-called Software Defined Networking (SDN) principle (i.e., pull cross-layer routing and switching with a logically centralized control-plane), and also tackling the design of advanced controllers independent of OpenFlow.

More precisely, following ONF norms, a SDN architecture shall be:

- Directly programmable: network control is directly programmable because decoupled from forwarding functions.
- Agile: abstracting control from forwarding let administrators dynamically adjust network-wide traffic distribution to meet changing needs.
- Centrally managed: network intelligence is (logically) centralized in software-based SDN controllers that maintain a global view of their network domain, which appears to applications and policy engines as a single, logical switch.
- Programmatically configured: SDN lets network managers configure, manage, secure, and optimize network resources very quickly via dynamic, automated SDN programs, which they can write themselves because the programs do not depend on proprietary software.
- Open standards and vendor-neutral solutions: when implemented through open standards, SDN simplifies network design and operation because instructions are provided by SDN controllers instead of multiple vendor-specific devices and protocols.

Beside OpenFlow, other protocols for the switch-to-controller interface exist. Proprietary protocols and controllers exist from vendors, that typically support both OpenFlow and open source controllers, and the proprietary protocols and controllers. These same vendor products typically are also exposing cloud network overlay protocol interfaces, such as those described in chapter 2.1.2. Finally, open source data-center management platforms such as OpenStack tend to integrate many features that can be found in a SDN controller.

The general feeling in the research community about OpenFlow is that it is very attractive in terms of additional functionalities and in particular the multi-layer switching features, but it fails in being sufficiently scalable (especially for inter-domain scopes) and reliable. On the other hand, it seems to be a broad consensus on the fact that the SDN principles, and in particular the separation from the data-plane and the control-plane and their programmability by means of open software node elements, should lead to novel advanced and scalable SDN protocols. Active research is ongoing to capture these requirements (scalability, reliability and programmability). Such evolution is in particular driven by novel interesting capabilities offered by cloud computing, such as the possibility to virtualize not only end-servers access by end-users but also network functions accessed for network and cloud provider needs.

1.2.2 Network Functions Virtualization

To gain or to avoid losing market shares, telecommunication providers need to introduce regularly new services, but the deployment of new features is often complex and slow. Actually, the dependence towards equipment lifecycle impacts the architecture evolution, and results in its lack of flexibility. The dream of agility may become true by the use of well-defined and practiced concepts in the IT domain, such as server virtualization and cloud computing. Network nodes thus become applications served by common hardware infrastructure distributed across data centers. Such disruptive trend promises CAPEX (low cost hardware) and OPEX (shared infrastructure, software upgrade, etc.) savings. In addition, this new flexibility allows a Telco to improve the added value of its network by offering new services such as the opening of its virtualized infrastructure in a mobile virtual network operator fashion. Scalability is another benefit of this technological tandem: depending on the load, the VM provisioning (creation or suppression) allows to optimize the resource usage, and to face the demand while guaranteeing QoS levels, e.g., as stated in a service level agreement.

The network programmability, virtualization and 'cloudification' phenomena have been accelerated since the publication in 2012 of a common white paper [15] written by a group of network operators under the ETSI umbrella, calling for 'network functions virtualization' (NFV). NFV aims at leveraging standard IT Virtualization technology to consolidate network appliances (switching, DPI, firewall, cache, load balancer, cipher etc.) onto industry standard high-volume servers, switches and storage, which could be located in datacenters, network nodes or end-user premises (see Figure 1.10). By implementing such network functions in software, NFV thus enables the deployment of virtualized network applications on shared infrastructure. Furthermore, it brings new ways to implement resilience, service assurance, test and diagnosis and security surveillance, new methods for interlinking virtualized services and functions, but it also transforms the way network operators architect and operate their networks. NFV, highly complementary but not dependent on SDN, is applicable to a wide variety of networking functions in both fixed and mobile networks.

In [15], the ETSI defends nine fundamental functional elements and design priorities for NFV systems:

- Network Functions Virtualization Infrastructure as a Service: a specific IaaS service infrastructure.
- Virtual Network Function as a Service (VNFaaS).
- Virtual Network Platform as a Service (VNPaaS).

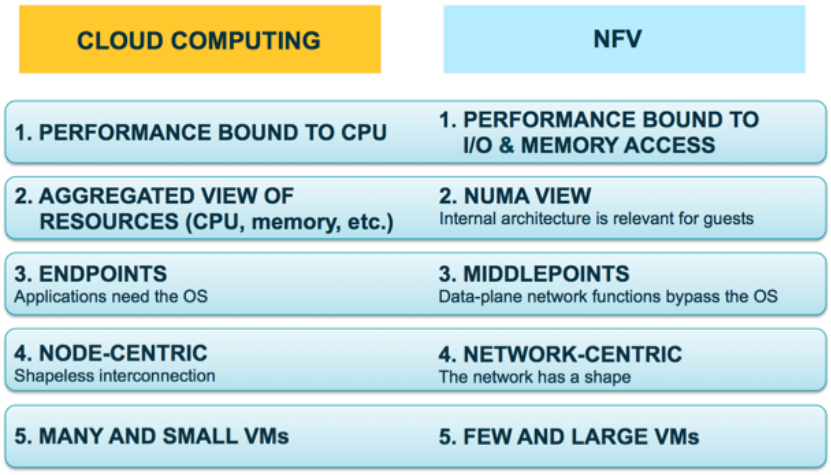


Figure 1.10: Specificities of NFV computing requirements with respect to typical cloud services.

- Virtual Network Function Forwarding Graphs.
- Virtualization of mobile core network elements such as IMS.
- Virtualization of cellular base stations.
- Virtualization of fixed access and home environment network nodes.
- Virtualization of Content Delivery Network (CDN) elements (vCDN).

There is a gap between what NFV asks in terms of network flexibility and what is available with legacy protocol. For this reason, SDN solutions are expected to be strongly interrelated with NFV systems, so that NFV can become viable only if adequate support is offered by a SDN architecture. NFV could therefore be the killing application to guide the deployment of SDN in telecommunication provider networks. In a hybrid SDN/NFV environment, represented in Figure 1.11, the controller/orchestrator performs the advanced tasks (from complex routing decisions to orchestration of NFV resources). In order to overcome security, vulnerability, and availability shortcomings due to the centralization, self-diagnosis and self-healing properties are very important for NFV/SDN environments, since the automated and intelligent fault management that these tools provide can reduce the impact of malfunctions at a lower cost and much more efficiently. Self-healing systems are commonly agreed to be the candidate solutions for enabling intelligence and automation on the current fault management processes. Self-healing systems are composed of three blocks: detection, recovery and diagnosis [16]. The detection block is in charge of discovering fault events while the recovery block aims at repairing the network after a failure. The recovery is possible thanks to the diagnosis

block that relates the observed symptoms of the network to the real cause of the malfunction. Its accuracy is hence critical as it determines the recovery strategy to be carried.

Given the vulnerability of the control plane in SDN, the automatic detection of any anomaly on the control plane allows for a rapid recovery and thus prevents (or mitigates) any impact on subsequent forwarding problem on the data plane. However, problems in the control plane affect the underlying switches, to the point that, if no alternative forwarding mechanism is provided (i.e. hybrid switches count on Ethernet forwarding except the OpenFlow-enabled ones if not expressly configured), they do not know how to forward the incoming packets and become completely inoperative.

At present, there is a limited literature on the design of NFV systems. Some practical work concerns the virtualization of routing functions in residential networks [17] and CPE (Customer Premises Equipment) [18] for diverse access technologies (like ADSL, DOCSIS, and L2ONT). The lack of guidelines or examples of fault management mechanisms and solutions in NFV is a noticeable fact. Most of the recovery and diagnosis solutions for SDN are OpenFlow centric, ignoring legacy equipment. Hence, several fault-tolerance mechanisms [19], [20], [21] propose solutions for OpenFlow-based equipment and do not seem to be extensible to other technologies. In addition, there is an increasing trend in the use of programming languages over the northbound interface, such as Lithium [22], Frenetic [16] and FatTire [23], all of them based on OpenFlow. These languages enable a transparent management and modification of the OpenFlow flows through high-level policies. For example, Lithium is an event-based language that applies different high-level policies to the incoming packets when certain events occur (i.e., a load balancing application for redirecting incoming packets with a non-authenticated IP source). Replication techniques, very similar to re-incarnation [24], are popular for coping with problems on the controller, performing a hot-swapping (i.e., a transition to a back-up controller in the presence of a fault on the primary one).

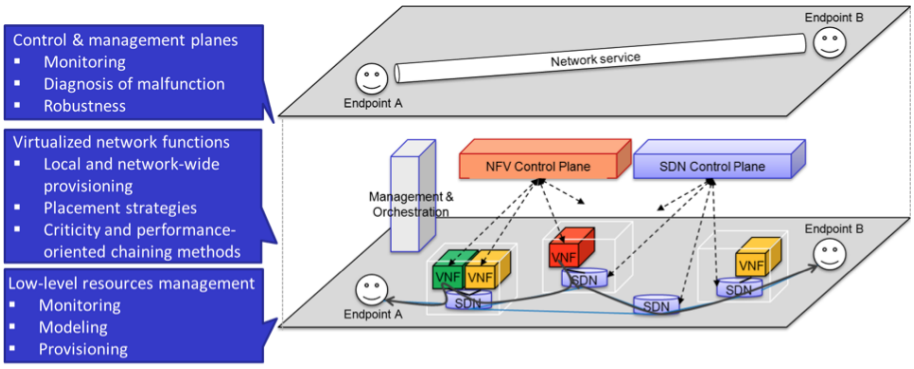


Figure 1.11: Coordination between NFV and SDN systems.

Structure of the dissertation

In this first chapter, we gave a general introduction on the broad context of this thesis, describing the increasing softwarization and cloudification of data center and provider networks. We described the general trends in networking, namely cloud networking and software defined networking protocols and network function virtualization systems.

In Chapter 2, we provide a more specific and technical state of the art on the technologies either directly covered in our contributions or indirectly related to them, in terms of functional support or desired network property.

In Chapter 3, we present the first contribution of this dissertation. We formalize the problem of placement of virtual machines in a data center network, fully integrating virtualization solutions as well as advanced cloud networking and SDN protocols. We describe a repeated matching heuristic and we propose to solve the problem for large instances. Then, we report the results of a detailed simulation analysis, focusing on how virtual machine placement is influenced by optimization goals and the presence of multipath forwarding and virtual switching features.

As there is major attention today devoted to the adoption of multipath forwarding protocols and of data-center topologies able to offer a large amount of paths to such protocols, an open question is whether this additional path diversity comes at the expense of traffic fairness desirable properties for data center networking. We investigated this research question in Chapter 4, which describes our second contribution. We formulate the traffic fairness problem based on TCP proportionally fairness and integrating different multipath modes. We give then relevant and counterintuitive insights on possible data-center fabric design choices.

Chapter 5 describes our third and last contribution. We adopted a medium/long-term vision where the data-center network facility exits from its legacy environment and expands into the carrier network to support NFV systems. We define the virtual network function chain routing problem in a carrier network as the problem of finding the best route in a carrier network where customer demands have to pass through a number of NFV nodes inside the telecommunication access, aggregation and core network, taking into consideration the unique constraints set by NFV. Our simulation results show at which extent this scope is viable and interesting for network carriers.

Chapter 6 concludes the dissertation and contains perspectives for further work.

Appendix I describes in detail some additional blocks for the heuristic described in Chapter 3.

Chapter 2

Related Work

We synthetically provide in this section a brief overview about data center fabrics and optimization algorithms. We start by briefly describing recently proposed data center network topologies as well as legacy and recent switching protocols. Then, we synthetically describe the state of the art on virtual network embedding and consolidation algorithms, as well as traffic engineering and fairness protocols and models.

2.1 Data Center Fabrics

The expressions 'data center fabric' or 'cloud fabric' derive from a jargon typically used in storage area networks. Since the dawn of cloud networking, it is increasingly used to refer to a comprehensive architecture including specific type of network elements, specific topologies, specific standard or proprietary protocols, etc. In the following, we describe topologies that have been proposed and partially deployed in industrial environments in the last decade. We then provide a brief state of the art on data center switching architecture and cloud network overlay protocols.

2.1.1 Topologies

The common legacy DC topology is the *3-tier* topology [25]. It has three layers: access, aggregation and core layers. At the access layer, servers or server units (also called 'blades') are attached to the network, via access switches; at the aggregation layer, access switches connect to aggregation switches; at the core layer, each aggregation switch is connected to multiple core switches. Such an architecture typically relies on legacy Spanning Tree Protocol (STP) switching [9]. STP is the control-plane protocol actually installing switching rules; while simple and fast, it leads to underutilize the network resources because of port blocking to avoid loops. Even if

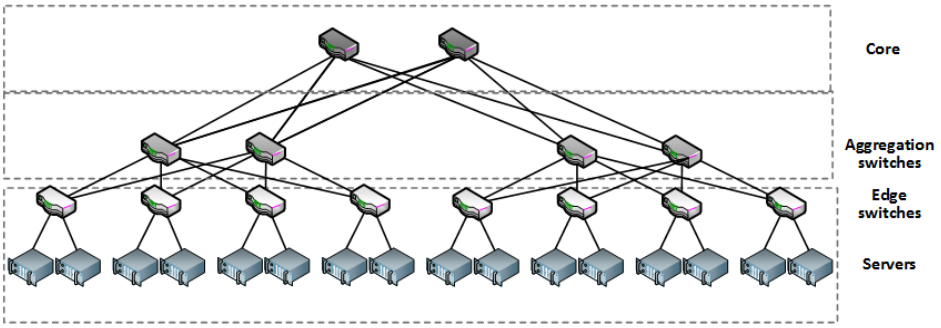


Figure 2.1: 3-layer topology.

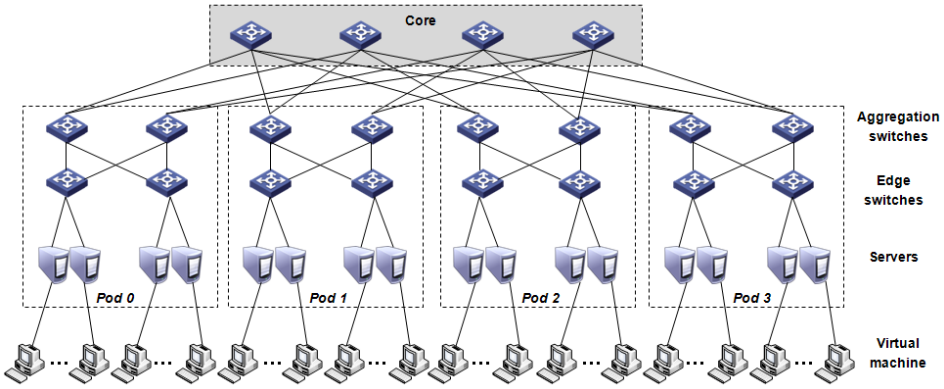


Figure 2.2: Fat-tree topology with 4 pods.

traffic engineering extensions such as Multiple STP, root bridge priority and port cost optimization methods exist, major problems still persist, namely in terms of convergence time upon failures, routing, and physical topology changes.

Such problems are even more important with less hierarchical and more horizontal alternative topologies proposed in recent years to offer high bandwidth and path availability to edge traffic among virtualization servers. Authors in [26] proposed a special instance of a Clos topology called "fat-tree" to interconnect commodity Ethernet switches as $k-$ary fat-tree. As depicted in Figure 2.2, all switches are identical and are organized on two layers: the core layer and the pod layer. Generally, at the pod layer there are k pods, each one containing two layers of $\frac{k}{2}$ switches: edge switches and aggregation switches. Each k-port switch in the lower layer (edge layer) is directly connected to $\frac{k}{2}$ hosts. Each of the remaining $\frac{k}{2}$ ports is connected to $\frac{k}{2}$ of the k ports in the aggregation layer. Concerning the core layer, there are $(\frac{k}{2})^2$ k-port core switches. Each core switch has one port connected to each of the k pods. The i^{th} port of any core switch is connected to the i^{th} pod so that consecutive ports in the aggregation layer of each pod switch are connected to the core switches on $(\frac{k}{2})$ strides. Figure 2.2 shows a fat-tree example for $k = 4$. The authors have

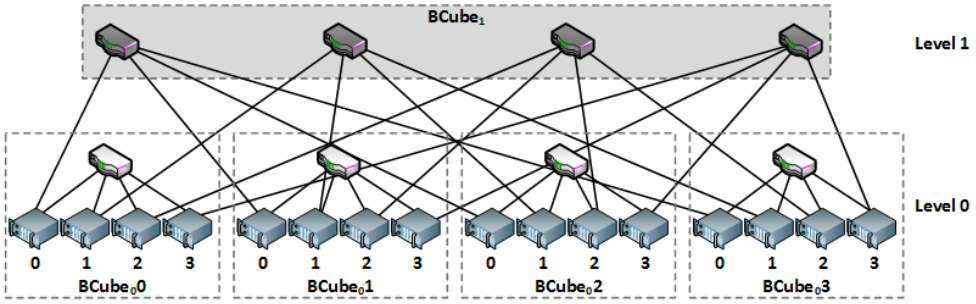


Figure 2.3: BCube_1 with n=4.

introduced a two-level concept route lookups for a multipath routing inspired by OSPF and ECMP. In this protocol, the flows to the hosts connected to the same lower-layer switch are forwarded, and the flows to the others destination are routed.

Another proposed topology that captures major attention is BCube [27], a recursive modular architecture. As depicted in Figure 2.3, BCube has server devices with multiple ports (typically no more than four). Multiple layers of cheap COTS switches are used to connect those servers. A BCube_0 is composed of n servers connected to an n-port switch. A BCube_1 is constructed from n BCube_0s and n n-port switches. More generally, a BCube_k ($k \geq 1$) is constructed from n BCube_{k-1}s and n^k n-port switches. For example, in a BCube_k with n n-port switches, there are $k+1$ levels of switches. Each server has $k+1$ ports numbered from level-0 to level-k. Hence, BCube_k has $N = n^{k+1}$ servers. Each level has n^k n-port switches. The construction of a BCube_k is as follows. One numbers the n BCube_{k-1}s from 0 to $n-1$ and the servers in each BCube_{k-1} from 0 to $n^k - 1$. Then one connects the level-k port of the i^{th} server ($i \in [0, n^k - 1]$) in the j^{th} BCube_{k-1} ($j \in [0, n-1]$) to the j^{th} port of the i^{th} level-k switch. It is worth noting that BCube requires virtual bridging in the containers to operate. Fig. 2.3 shows an example of a BCube_1, with $n = 4$. Regarding to the routing protocols the authors have introduced a single path routing protocol where the longest shortest path is equal to $k+1$. They also introduced a multipath routing protocol where only the parallel paths are valid. The authors mean by parallel paths two paths where all the nodes are disjoint, which mean no common server or switch. Thus, as an example we can have only two paths with BCube_1. .

Similarly to BCube, DCell [28] has servers equipped with many interfaces and COTS switches. A DCell server is connected to several other servers and a switch. A high-level DCell is constructed from low-level DCells. The connection between DCells can make use of virtual bridging. A DCell_k ($k \geq 0$) is used to denote a level-k DCell. DCell_0 is the building block to construct larger DCells. It has n servers and a switch ($n = 4$ for DCell_0 in Figure 2.4). All servers in DCell_0 are connected to

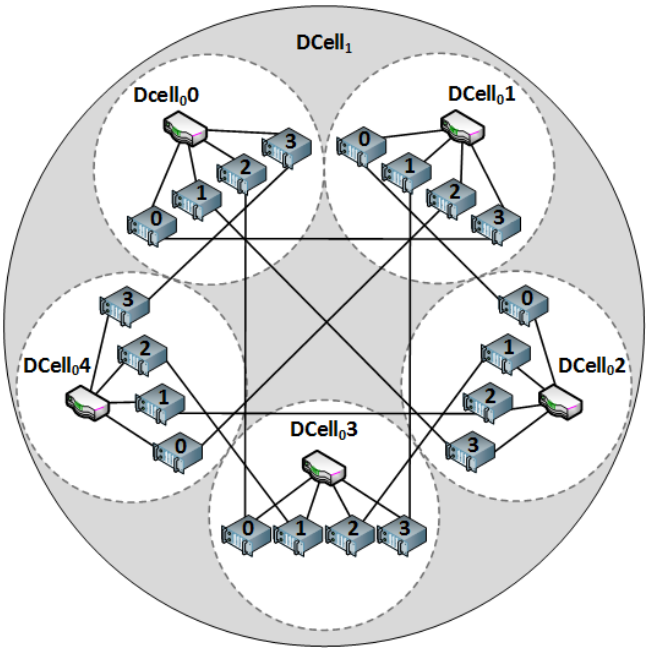


Figure 2.4: DCell_1 with n=4.

the switch. In DCell_1, each DCell_0 is connected to all the other DCell_0s with one link; the Figure 2.4 shows a DCell_1 example. DCell_1 has $n+1=5$ DCell_0s. DCell connects the 5 DCell_0s as follows. It assigns each server a 2-tuple $[a_1, a_0]$, where a_1 and a_0 are the level-1 and level-0 IDs, respectively. Thus a_1 and a_0 take values from $[0, 5)$ and $[0, 4)$, respectively. Then two servers with 2-tuples $[i, j-1]$ and $[j, i]$ are connected with a link for every i and every $j > i$. Each server has two links in DCell_1. One connects to its switch, and the other to a server in another DCell_0. In DCell_1, each DCell_0 is fully connected with every other virtual node to form a complete graph. Moreover, since each DCell_0 has n inter-DCell_0 links, a DCell_1 can only have $n+1$ DCell_0s. A DCell_k is constructed in the same way to the above DCell_1 construction in a recursive procedure [28] that is more complex than the BCube one.

Regarding to the routing protocols the authors have introduced the DCellRouting protocol for a single path routing protocol based on the recursive structure of the DCell. However, they have introduced a theorem that DCell server can utilize its multiple links to achieve high throughput using services such as Google FIle System.

2.1.2 Ethernet Switching

Ethernet switching protocols were originally designed for enterprise Local Area Networks (LANs). The standard solution was based on the 3F concept: upon reception

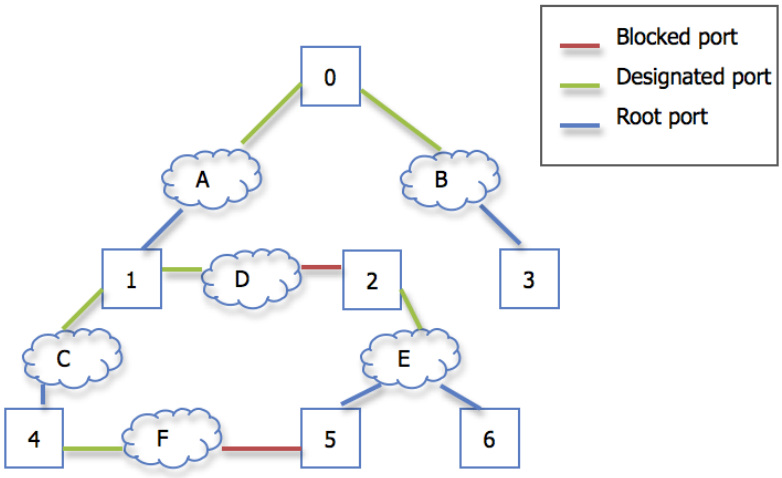


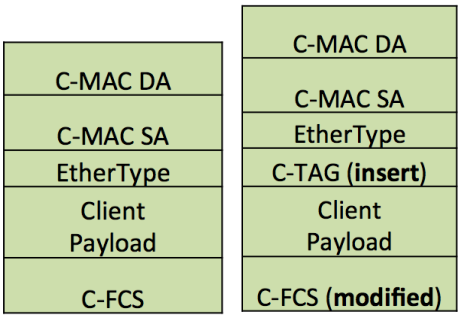
Figure 2.5: Spanning Tree Protocol.

of an Ethernet frame, an Ethernet bridge either Filter it, or Forward it, Flood it. If the Media Access Control (MAC) destination of the incoming frame is beyond the port where the frame comes from, the switch does not forward the frame, dropping it (i.e., 'filtering' it). If the destination is beyond another port, after looking at its MAC table it will then forward it to the appropriate segment. If the destination address is not in the MAC table at all, it will then be flooded on all of its ports except the port it was received on.

A major issue appearing with 3F forwarding is the creation of forwarding loops in case of multiple paths between Ethernet bridges. In this is context the Spanning Tree Protocol (STP) [9], IEEE 802.1D, was specified and implemented. STP avoids loops by deactivating ports so that the enabled ports form a tree spanning all the nodes of the network, as represented in the example of Figure 2.5. At a given node, the ports that are active on the spanning tree are those that are along the shortest path toward the root node, the root node being manually set or automatically discovered by arbitrary rules (highest priority or lowest MAC address).

Enhancements to IEEE 802.1D

While STP has the advantage of being simple to understand and implement, it has major shortcomings in terms of performance. The performance requirements typically required in common LAN environments are not stringent, and link capacity is largely sufficient. However, in data center network environments it is more challenging due to the very high density of servers, the low latency required, and the high bit rates of connections and links. In such situations, STP is offers low reliability, scalability and traffic engineering features.



(a) Basic frame. (b) 802.1Q frame.

Figure 2.6: Basic and IEEE 802.1Q frame formats.

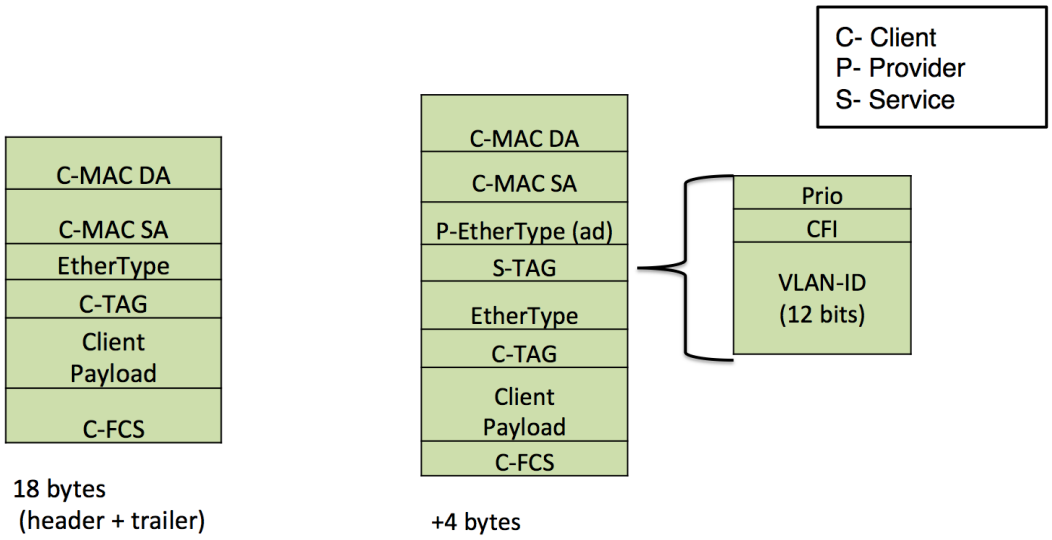
In terms of reliability, by default STP convergence upon topology changes such as due to failures may take many dozen of seconds and even minutes for medium-size networks, depending on signaling cycles and the network topology diameter. In 1998, Rapid Spanning Tree Protocol (RSTP), IEEE 802.1w [29], was presented to decrease the convergence time upon failure, by introducing the notion of backup and alternative ports associated to the root port and designated ports.

The first attempt to add traffic management features in Ethernet switching was the introduction of Virtual Local Area Network (VLAN) switching in IEEE 802.1Q [30]. The Ethernet data-plane is altered with a VLAN tag (also called Client tag, C-TAG, see Figure 2.6), based on which independent switching MAC tables are used at each Ethernet bridge. The VLAN binding can be based on ports, MAC address, and even layer 3 and 4 information for advanced switches. With IEEE 802.1Q also the possibility to add switching priorities has been introduced.

Even with VLAN switching, traffic engineering (i.e., how to explicitly route traffic where resources and capacity are available) is limited only to STP root election control. Moreover, in terms of scalability, Ethernet switching has important limitations when conceiving its extension to metropolitan area network domains. These shortcomings have been addressed by further modifying the data-plane format and some control-plane features as described hereafter.

Ethernet Carrier Grade

With the standardization and commercialization of Gbps-order Ethernet links and switches in early 2000s', the industrial sector did show interest in the potential extension of Ethernet switching beyond the LAN boundaries, with therefore much more machines to address, much more traffic to switch and more stringent constraints on latency and reliability. A similar interest was also expressed a few years later for data center networks. The first shortcomings that were addressed at the



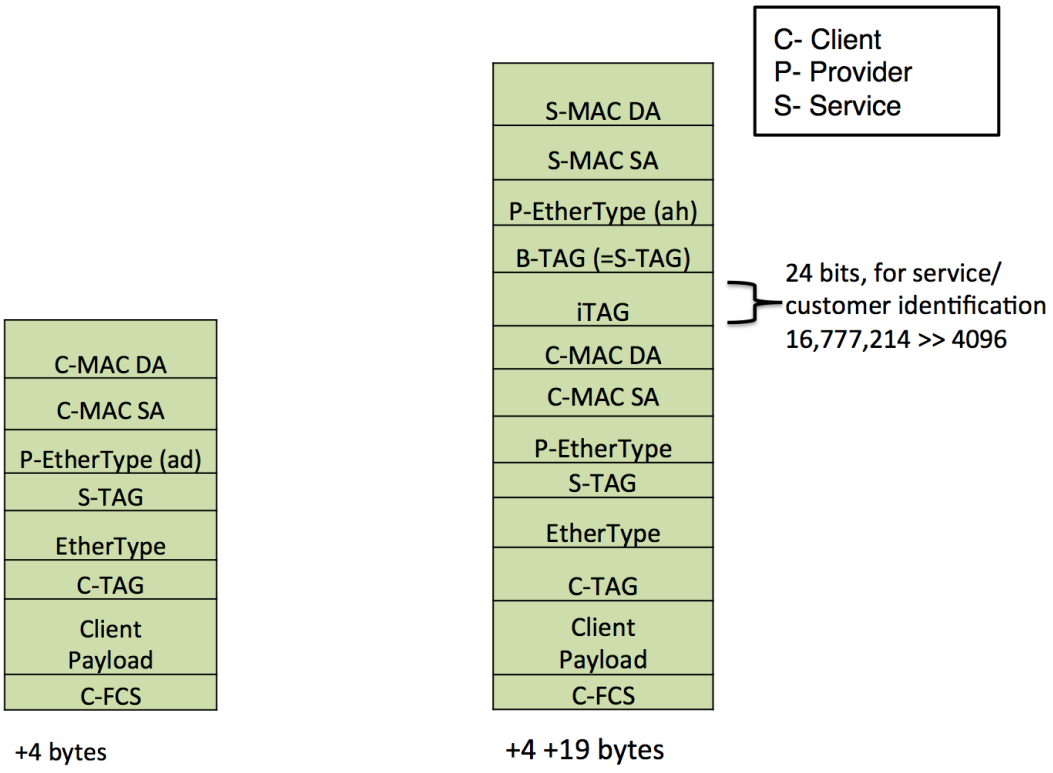
(a) 802.1Q frame. (b) 802.1ad frame

Figure 2.7: IEEE 802.1ad frame format.

IEEE were in terms of scalability: the VLAN addressing space is limited and cannot be shared between enterprise networks and a metropolitan area network (MAN) or DC network provider providing interconnection of many Ethernet highlands sites via a high-speed Ethernet network; the MAC table size would explose at MAN and DC scopes because of the much higher number of connected terminals. These two issues are addressed by the Provider Bridge (PB) node, IEEE 802.1ad 'QinQ' [31], and the Provider Backbone Bridges (PBB) node, IEEE 802.1ah [32] 'M-in-M'.

IEEE 802.1ad introduces an additional VLAN tag, also called Service-provider tag (S-TAG), for network provider usage only, as presented in Figure 2.7, to allow the network provider to uniquely identify customers. In DC networks, this operation is typically performed at the access switch level, where traffic from servers is aggregated. The Ethernet switching logic is then not altered under IEEE 802.1ad: traffic is distributed following the 3F modes and the STP. At each PB node, a separate MAC table for each S-TAG is maintained.

IEEE 802.1ah introduces MAC in MAC layer encapsulation of endpoint frames into PBB frames in order to isolate the addressing of endpoints from the addressing of core bridges (PBB), hence reducing the PBB MAC table size. 802.1ah and 802.1ad can coexist in a same frame, and in such a case the S-TAG is copied upwards in the 802.1ah header part as shown in Figure 2.8. In addition, an Informational tag (I-TAG) of 24 bits is added, to either extend the VLAN addressing space or to allow proprietary usages. In the PBB domain, traffic is then distributed following the 3F modes and the STP. In DC networks, IEEE 802.1ah is commonly performed at the Top of Rack (ToR) switch level, even if some hypervisors offer the possibility to do it



(a) 802.1ad frame

(b) 802.1ah frame

Figure 2.8: IEEE 802.1ah frame format.

directly in the virtualization server for traffic coming from virtual machines. In such cases, the I-TAG can be used as virtual network (IaaS) identifier, which somehow anticipates similar functionalities offered by cloud network overlay protocols (as presented later in this chapter).

In order to reinforce the Ethernet switching architecture with traffic engineering features, there have been various attempts. Short term attempts essentially consist in marginal adaptations of the legacy architectures. One way is to aggregate multiple physical links as a single virtual one for the STP using the Link Aggregation Group or the multi-chassis EtherChannel protocols [33]. Another protocol acting at the The Multiple Spanning Tree Protocol (MSTP) [34] is one solution largely deployed (many other variants of MSTP, also proprietary, exist [35, 36]): it allows distributing Ethernet traffic on multiple spanning trees concurrently, with a VLAN:ST N:1 assignment.

The assignment of VLANs to STs can be the result of a TE optimization policy as presented in [34]. Such extensions allow reaching a higher traffic distribution and a higher bandwidth efficiency, avoiding to disable entire links as done in STP, which can be very costly for both DC and MAN network environments. However,

there is a very high level of complexity in the management of MSTP solutions, in terms of signaling, state management and VLAN:ST multiplexing optimizations. Considered as short-term patches, for DC network environments MSTP is de-factor today overcome by novel protocols conceived for implementing routing of Ethernet frames similarly to how routing is performed in IP networks, as described hereafter.

Toward Ethernet Routing

The real bottleneck in performing TE efficiently in an Ethernet switching context being the spanning tree bridging of Ethernet traffic, eventually the STP control-plane has been removed from more recent cloud fabric solutions implementable for DC Networks (DCNs). The protocols that have been largely commercialized are: the IEEE Provider Backbone Bridges with Traffic Engineering (PBB-TE) [37], where centralized control servers push MAC tables to backbone switches (in a similar philosophy OpenFlow [13] does so too); the IETF Layer 2 Label Switched Paths (L2LSP) [38] effort suggesting to use the VLAN fields as MultiProtocol Label Switching (MPLS) label fields; the IEEE 802.1aq Shortest Path Bridging (SPB) [39] and IETF Transparent Interconnection of a Lot of Links Protocol (TRILL) [40] protocols where the control-plane is distributed adapting a layer-3 link state routing protocol, the Intermediate System to Intermediate System (ISIS) protocol, to work with the Ethernet data-plane.

While differing in terms of scalability and deployability, the latter three solutions (L2LSP, SPB, TRILL) have proven to be viable ones and have been adopted by many vendors. Notably, these protocols enabled multipath routing and forwarding of Ethernet frames, and hence opened the way to active load-balancing over multiple paths across virtual and physical switches. A problem that is addressed in the data-plane of SPB and TRILL is the fact that, without STP, loops can appear during ISIS convergence. To cope with loops, a Time to Live (TTL) field has been included in both protocols. While SPB needs the whole DCN backbone to be upgraded with new switches, TRILL can be implemented only at key points (e.g., at the hypervisor level only using software implementations) and is therefore considered more scalable. As nodes in this context are no longer simple bridges since they perform a routing function in TRILL as well as in this dissertation, the term Router-Bridges (RBs) is adopted.

Cloud Network Overlay Protocols

All the Ethernet carrier grade and routing protocols presented above strictly work with an Ethernet data-plane. Those performing Ethernet frame encapsulation in another Ethernet frame, i.e., PBB, PBB-TE, SPB, TRILL, are in particular inter-

Table 2.1: Features of cloud network overlay protocols.

Cloud network overlay feature	SPB	TRILL	LISP	VXLAN	NVGRE	STT
Encapsulation	Ethernet over Ethernet	Ethernet over Ethernet	IP over IP	Ethernet over IP	Ethernet over IP	Ethernet over TCP/IP
Inter-data center link	Ethernet	Ethernet	IP	IP	IP	IP
Intra-data center link	Ethernet	Ethernet	IP	IP	IP	IP
User device integration	None	None	Yes	None	None	None
Virtual network segmentation	Yes	Limited	Yes	Yes	Yes	Yes
Firewall friendliness	Very high	Very high	High	High	Low	Very low
Incremental deployability	Low	High	Very high	High	Low	Low
Multipath and load balancing	Native	Native	Native	Partial	Partial	Partial
Multicast	Native	Native	Ongoing	Native	Partial	Partial

esting for DCN environment because the association they create between the address and the routing locator of the destination is very beneficial for the reduction of the routing table size and for the support of VM migrations. In networking, protocols performing encapsulation at the same layer or across layers are typically referred to as 'overlay protocols'. Those having as direct application the DCN environment (e.g., PBB, PBB-TE, SPB) are commonly referred to as 'cloud network overlay' or 'virtual network overlay' protocols [41]. Other cloud network overlay protocols exist. Some of them are compared in Table 2.1.

The only standard protocol performing IP-in-IP encapsulation with a complete control-plane architecture decoupled by the data-plane is the Locator/ Identifier Separation Protocol (LISP) [42]. LISP has intermediate UDP and LISP shim headers - UDP allows passing transparently through middle-boxes. Despite it has been originally designed to cope with Internet routing, it is actually used as a cloud network overlay protocol in commercial solutions, such as Cisco and NU@GE Cloud solutions, because its functionalities allow efficiently managing virtual machines and IaaS isolation.

Other protocols encapsulate Ethernet frame into higher layer packets such as IP, UDP and TCP. The Virtual Extensible LAN (VXLAN) [43] protocol encapsulates Ethernet packets into IP packets. It shares with LISP most of its data-plane, apart the inner packet that is IP in LISP and Ethernet+IP in VXLAN. The VXLAN shim header, as well as the LISP one, allows transporting a virtual network identifier (on 24-bit as for the IEEE 802.1ah I-TAG) useful for IaaS segmentation. The Network Virtualization Using Generic Routing Encapsulation (NVGRE) [44] works similarly than VXLAN, but it does not uses UDP - it uses instead its own shim header for virtual network identifier, and it has therefore issues in passing through middle-boxes. Finally, the Stateless Transport Tunneling (STT) [45] performs Ethernet

encapsulation over a TCP/IP packet, where TCP functions are de-facto non used in practice. The TCP header is added to exploit TCP offloading and optimization present in some commercial NICs.

Table 2.1 briefly summarizes the six cloud network overlay protocols we briefly presented. As the table suggests, they all natively support multipath forwarding and load balancing, at least to some extent. The most promising protocols (e.g., VXLAN, LISP), however, are also incrementally deployable, support virtual network segmentation, natively pass through IP networks and easily cross firewalls.

2.2 Data Center Network Optimization

We review in the following a selection of the state of the art on data center network optimization algorithms and protocols that see real applications in data center networking today. We classify as virtual network embedding, virtual machine placement, and traffic engineering methods.

2.2.1 Virtual Network Embedding

Virtual Network Embedding (VNE) is a term adopted since the inception of advanced software virtualization in early 2000s'. A generic definition of the virtual network embedding problem is to assign to a set of virtual network demands a virtual network built over a same physical network infrastructure, which technically is supposed to offer isolation and possibly deterministic service level specifications at both nodes and links. In the literature the term Virtual Data Center (VDC) is also frequently used, often associated to an embedding algorithm. A representation of a virtual network is given in Figure 2.9. Each virtual network is represented by a graph that has to be embedded over a physical network graph that shall have sufficient idle resources to serve the demands. A virtual network demand can be rejected in case of resource unavailability.

VNE algorithms at the state of the art can be classified as static or dynamic. Static approaches do not consider the remapping possibility of some VNs to improve the embedding cost as a function of changing network states, while the dynamic approach allows it, taking into consideration several metrics such as network resources fragmentation. In fact, over time, some VNs can expire and release their resources, which cause fragmentation; this fragmentation can increase the rejection rate of future VN embedding requests. Hence the dynamic approach allows a better network resource allocation efficiency. In this context, the authors of [46] have shown that most VN request rejections can be due to bottlenecked links.

VNE algorithms can also be classified as centralized or distributed. In centralized approaches, a computation server is logically present and responsible for computing

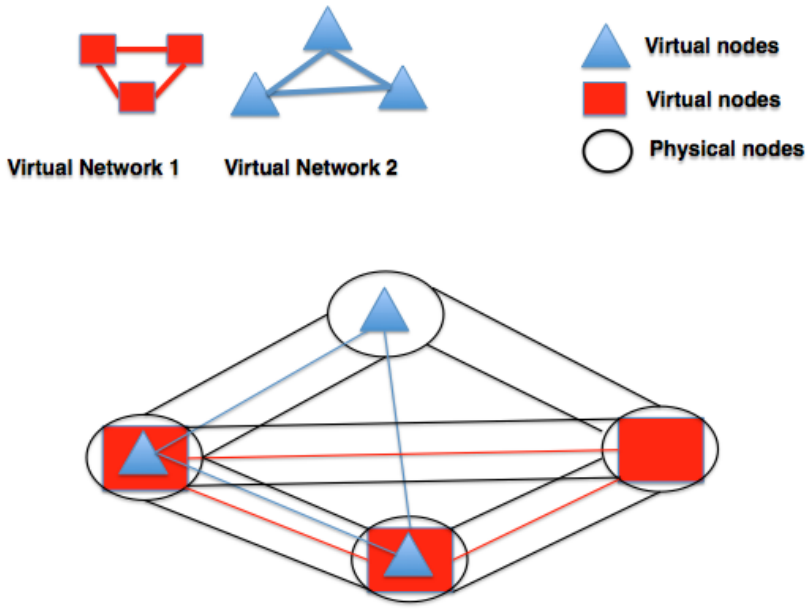


Figure 2.9: A sample of virtual network embedding.

the VNE, while in distributed approaches collaboration takes place between different computation nodes to compute VNE solutions.

The literature on VNE is vast. Many propositions have been proposed to solve many variations of the VNE problem. Most of the proposals have adopted mathematical programming approaches [47,48], often applicable only for small instances of the problem because it is NP-hard. Many others propose heuristic or metaheuristic algorithms [49–55]

The performance goal is most of the time expressed through a single fitness metric. A mapping request cost or a request acceptance/rejection rate are commonly defined as abstract metrics in [48, 56]. Others consider quality of service metrics such as the 'stress level'(function of the amount of virtual nodes or links embedded over the same physical nodes or links) in [57], or the path length [58], the system utilization (e.g., in terms of CPU and RAM) in [59], or the link utilization in [52], or the throughput in [50]. Some papers as [55, 60] include two metrics in the objective function. The VNE problem can be seen as composed of two sub-problems, nodes mapping and link mapping. Hence, it can be solved by solving the two sub-problems separately, as done in [48], also called the uncoordinated approach, or with a coordination with the two sub-problems where each solution affect the other as in [61].

2.2.2 Virtual Machine Placement

With the emergence of server virtualization and Cloud network overlay protocols, adaptive, online or recurrent virtual machine placement and replacement procedures have been designed and implemented in Cloud orchestration platforms.

Server consolidation is one of the first VM placement optimization problems of interest for industrial data centers: - it aims to reduce the number of enabled (turned-on) virtualization servers by regrouping the VMs on the same server whenever possible. Therefore itallows taking direct monetary benefits from server virtualization, thanks to lower energy consumption granted by adaptive hibernation of unused servers. The VM consolidation problem is derived from capacitated facility location problem and/or bin-packing problems and is therefore a NP-hard optimization problem. In order to allow its quick resolution also including online environments some of them proposed extensions of simple greedy algorithms such as First Fit Decreasing [62], Least full first [63], Most Full First [63], Best Fit [64], to avoid attempting optimal yet too time consuming mathematical programming approaches; they show, however, a non-deterministic guarantee on the optimality gap. In [65] the authors proposed Ant Colony Optimization a metaheuristic to minimize two objectives the resource wastage and the power consumption. Also, in [66] they solved the problem by using Ant Colony Optimization and only considered CPU and memory constraints.

A few works detach from specific straightforward modeling and resolution algorithms of the virtual machine placement problem, taking into consideration other parameters than energy consumption only, and in particular networking constraints. In [67], the authors proposed a VM placement solution also considering network resource consumption. They designed a two-tier approximation algorithm that efficiently solves the VM placement problem. They assumed that a VM container could be divided into CPU-memory slots, where each slot could be allocated to any VM. They considered the number of VMs as equal to the number of slots; if the number of slots was higher than the number of VMs, they added dummy VMs (with no traffic), and did not affect the algorithm. Due to a communication cost between slots, defined as the number of forwarded frames among them, the objective was set as the minimization of the average forwarding latency. They also assumed static single-path routing and focused on two traffic models. A dense one where each VM sent traffic to every VMs at an equal and constant rate, and a sparse Infrastructure as a Service (IaaS)-like one with isolated clusters so that only VMs in the same IaaS could communicate. Similarly in [68], the authors minimized the energy consumption of active servers, bridges and links, to maximize the global EE. The authors converted the VM placement problem into a routing problem, so as to address the joint network and server optimization problem (where there is no tread-off between

network and server objectives).

Some other works proposed genetic algorithms to solve the problem. In [69], the authors proposed a genetic algorithm based approach, namely GABA, the authors proposed an online self-reconfiguration approach for reallocating VMs in large-scale data centers, and just considered the CPU resources. In [70], the authors also propose a genetic algorithm for resolving the VM placement problem. They formulate the problem as a multi-objective optimization minimizing the total resource wastage, the power consumption, and thermal dissipation costs, and they proposed genetic algorithm to incrementally search the better solution combining possibly conflicting objectives.

Many works have formulated the problem as a bin-packing problem. Apart those already mentioned, in [71], the authors consolidated VM placement considering a non-deterministic estimation of bandwidth demands. The bandwidth demand of VMs was set to follow normal distributions as the authors assumed that server consolidation usually occurs at weekly or monthly timescale. They formulated the consolidation in a Stochastic Bin Packing problem and introduced an online heuristic approach to resolve it. In [72] the authors also formulated the problem as a bin-packing problem considering only server resources constraints, and they proposed an optimized First-Fit-Decreasing, where instead of placing directly the new arrived workload into the first node that can accommodate, the heuristic tries to reorganize the workloads smaller than the newcomer. The heuristic also reorganizes the workloads when workload departs.

In [73], the authors revisited the virtual embedding problem by distinguishing between server and bridge nodes with respect to the common formulation. They proposed an iterative 3-step heuristic: during the first step an arbitrary VM mapping was done; the second step mapped virtual bridges to bridge nodes, and the third one mapped virtual links accordingly. If one of these steps failed, the heuristic would come back to the previous one until a solution was found. The quality of the solution seemed dependent on the first step, the other steps just minimized the impact of the previous step. Furthermore, there may have been a scaling problem due to the uncontrollable backtracking. More generally, virtual embedding approaches in the literature often discarded specificities of the network control-plane such as the routing protocol and TE capabilities.

In [74], the authors considered network constraints in addition to CPU and memory constraints in the VM placement problem. They defined a network-aware VM placement optimization approach to allocate VM placement while satisfying predicted traffic patterns and reducing the worst case cut load ratio in order to support time-varying traffic. Interested by network cuts, they partitioned the set of hosts into non-empty connected subsets, which are bottlenecks for the traffic demand

between VMs placed in different sides of the cut. In [75], the authors optimized jobs placements where each job required a number of VMs; the objective function minimized the network and the node costs. The authors did not handle the link capacity constraints, and did not consider multipath forwarding capabilities instead of multipath routing with one single egress path. In [68], the authors minimized the power energy consumption of activated servers, bridges and links, to maximize the global energy saving. The authors converted the VM placement problems into a routing problem, and so they addressed the network and server optimization problem as a single one. So, there was no trade-off between the network-side and server-side optimization objective.

Most of the studies on the state of the art focused on a single criterion. Some of these studies ignored link capacity constraints, [62, 63, 63, 64], others excluded dynamic routing as in [67], or just considered the traffic volume to reduce the number of containers as in [71], or just the network resources as in [67] and [74], only [68] considered multipath forwarding capabilities.

Some of these studies excluded dynamic routing functionalities; for example in [67] the authors propose a VM placement solution considering network resource consumption, wherein the objective is set as the minimization of the average forwarding latency. Others studies take only the network resources into account, as in [67] [74], or just consider the traffic volume to reduce the number of containers as in [71] where authors propose a VM placement considering a non-deterministic estimation of bandwidth demands, formulating the problem as a Stochastic Bin Packing problem, and introducing a new heuristic approach to resolve it. In [73], where the authors revisited the virtual embedding problem by distinguishing between server and switching nodes; they do not handle the link capacity constraints, and they also do not consider multipath forwarding (load balancing), but a multipath routing with single egress path. Commonly, because of the relatively recent employment of virtual bridging for transiting traffic at the server level, virtual bridging capabilities for external traffic forwarding were ignored.

2.2.3 Traffic Engineering

As defined by [76] 'Internet traffic engineering is defined as that aspect of Internet network engineering dealing with the issue of performance evaluation and performance optimization of operational IP networks. Traffic Engineering encompasses the application of technology and scientific principles to the measurement, characterization, modeling, and control of Internet traffic...'. This definition by the IETF is quite broad, and indeed encompasses a large number of different components. We review in the following two TE features that are recalled in the next chapters of this dissertation.

Explicit routing

Starting by the end of the nineties, increasing interest was addressed to traffic engineering (TE) techniques that are able to better distribute traffic load on existing links with respect to standard link-state routing. The first article proposing to optimize Interior Gateway Protocol (IGP) metrics as a function of network load is [77]. Given the NP-hardness of the problem when considering link capacity constraints in the multi-commodity flow formulation, heuristics were proposed therein and in subsequent works [78–80]. In [78], the problem of finding the optimal set of metrics also accounting for transient link failures is formulated. Another important work is the one in [80] where the metric set is computed also in order not to induce route deviations in BGP due to the coupling between IGPs and the Border Gateway Protocol (BGP).

The typical objective in TE metric optimization for IP networks is the minimization of the maximum link utilization. This goal is opposed to opposite goals used in transport optical networks, typically the maximization of circuit acceptance or minimization of blocking rate. The reason resides in the major difference between the connection-less packet-switching nature of IP routing and the connection-oriented circuit-switching buffer-less nature of optical switching. In IP networks, there is no a so certain guarantee on the bitrate as it can happen in optical networks, so that the routing solution should be as far as possible from the risk of congestion to support real-time services. Indeed, real-time services are not be able to recover data stream loss while users are exploiting the service.

There is therefore the need to better master network impairments such as link failure and traffic variations. About the latter, in [79] robust IGP metrics are computed to take into account the traffic matrix variations, by the definition of incertitude sets integrated in the optimization engine. Another way is to (easily) include in previous TE methods the maximization of load-balancing for a same destination prefix. This can be reached by selecting for the final IGP metric set the metrics that not only satisfy the TE goal, but that also maximizes the usage of Equal Cost Multi-Path (ECMP) routing, i.e., the load-balancing of traffic over multiple paths when these paths have equal IGP path cost.

In order to master link failures, the proactive optimization of IGP metrics so that they are robust against link failure, i.e., so that the TE objective is minimized also taking into account the occurrence of link failure, as done in In [78], is, however, not sufficient. The reason is the IGP routing convergence time upon failure that can be too high in very-high capacity networks such as DC and carrier networks. In IGP link-state routing, the convergence time upon failure can be of many seconds, essentially due to a combination of the time needed to detect the failure, announce it everywhere and to allow for the recomputation of the new shortest paths and

their writing in the forwarding information base.

Also to answer to faster rerouting upon failure, major interest has been devoted to label-switching solutions for IP networks based on the Multi-Protocol Label Switching (MPLS) protocol and its generalizations [81]. In MPLS, data-plane switching is done using label-switching tables that associate input port and label information to output port and label, where the label is an identifier of the destination pre-distributed using specific control-plane protocols. The routing decision behind the input/output port and label associations can rely on legacy IGP routing or can also be the result of explicit path computation logics.

Advanced constrained path computation solutions have been conceived for MPLS networks extended so that multiple metrics can be associated to each single link. Such extensions, referred to as MPLS with Traffic Engineering extensions (MPLS-TE), rely on IGP-TE extensions [82,83], and on the Resource ReSerVation Protocol with TE extensions (RSVP-TE) [84]. This allows imposing explicit paths that completely bypass legacy IGP routing, while being able to maintain in the same switching table instructions related to unconstrained plain IGP routing and instructions computed using multiple constraints. Moreover, with MPLS-TE, fast-rerouting solutions can be implemented in relative straightforward way exploiting the fact that labels have all a local significance: at each hop, the external label of incoming packets changes (swapped with another label, replaced with a new pushed label on the top of it, or popped out from the IP stack). Thanks to local significance, upon detection of a failure a node can immediately react by pushing a new external label and swapping the port for the packets to the destinations beyond the failed link, so that the value used for the label upon failure has been pro-actively pre-signaled. In this way, recovery upon failure can stay below 50 ms [85].

These same TE features can be implemented also at lower layers, and notably at the Ethernet switching level [81]. With Layer-2 Label Switching (L2LSP), a IEEE 802.1ad data-plane can host at the S-TAG level an MPLS label. Other Cloud fabric protocols presented in the previous section, such as PBB-TE, TRILL, SPB and OpenFlow, are not as powerful as layer-2 MPLS in terms of fast rerouting based on current specifications. TRILL, SPB and OpenFlow have however the potential of acquire needed extensions to support fast-rerouting, yet this appears as much simpler with OpenFlow, which shares with MPLS-TE the logically centralized TE path computation way. Indeed, in MPLS-TE, path computation can be externalized to external computation servers called Path Computation Elements (PCEs) [86].

2.2.4 Traffic Fairness

Explicit routing TE methods act at the aggregate network level. TE policies, including ECMP and fast rerouting, do apply so that a same transport-level flow sees its

packets routed over a same path, and that load-balancing is performed for different flows or even for different groups of destination networks (based on the definition of the MPLS forwarding equivalence class, for instance).

Another concern in operational networks is to guarantee that different flows sharing the same network links subject to congestions suffer from the same level of bandwidth reduction, at stable equilibrium situations. This concern is typically referred to as traffic fairness. Since the first steps of the Internet, it was recognized that unrestricted access to the Internet resulted in poor performance in the form of low network utilization and high packet loss rates [87]. These phenomenons were called congestion collapses, and led to the development of congestion control algorithms appropriate to the Internet. The basic dominating idea was to detect the congestion in the network through packet losses, and integrate this information in the way throughput is offered to greedy applications; upon detecting a packet loss, the source reduces its transmission rate, or otherwise it increases the transmission rate. Eventually, many versions of the Transport Control Protocol (TCP) [88] use the lack of acknowledgement of packets as an alert of packet loss.

In a network with multiple competing TCP sessions sharing links, several studies [89–92] have shown that TCP implicitly solves a utility problem in equilibrium. This utility problem is formally described as a maximization of an aggregate utility subject to capacity constraints:

$$\max_{X \geq 0} \sum_{j \in \mathcal{J}} U(X_j) \tag{2.1}$$

subject to

$$\sum_{j \in \mathcal{J}} \delta_{je} X_j \leq c_e, \quad e = 1, 2, ..., E \tag{2.2}$$

The above model maximizes the utility function $U(X_j)$ of each session $j \in \mathcal{J}$ where X_j denotes the rate of session j while δ_{je} is the indicator that takes the value 1 if session j uses link e, 0 otherwise.

Multiple approaches in order to address the fairness between costumers exist. In [93] two approaches are described, the *Max-Min Fairness* (MMF) approach, which maximizes the minimal assignment of capacity to demands. In fact the solution can be resumed in iterative steps; first, MMF assigns the same minimal volume to all demands, until some free capacity is still present, each minimal assignment is increased whenever possible, and so on until the maximization does not improve any longer the situation.

To illustrate MMF, the authors consider a network where V is the set of nodes, E is the set of links and D is the set of demands between nodes. Each node is identified by v_i such as $v_i \in V$ and $i = 0, 1, \ldots, |V|$ and each link is identified by e_j such as $e_j \in E$ and $j = 0, 1, \ldots, |E|$ and finally each demand is identified by d_k

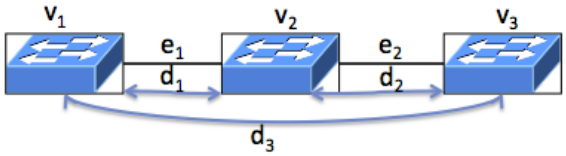


Figure 2.10: A reference network example for MMF and PF allocations. Source: Routing, Flow, and Capacity Design in Communication and Computer Networks book [93].

such as $d_k \in D$ and $k = 0, 1, \ldots, |D|$. Each demand is identified by an origin and a destination between two nodes. A route is specified by a sequence of links. Let x_d be the rate at which source v is allowed to transmit data. Each link e in the network has a capacity c. Given the capacity constraints on the links, the source allocation problem is to assign a rate x_d to the users in a fair manner.

To illustrate the difficulties in defining a fair allocation, we consider the example in Fig. 2.10. Supposing a network with three links, two nodes and three demands. The network consists of two links e_1 and e_2 and three nodes v_1, v_2 and v_3 and three demands d_1, d_2 and d_3. d_1 and d_2 cover only one link, e_1 and e_2, respectively. The third demand d_3 covers both links, e_1 and e_2. Suppose that the capacity of link e_1 is $c_1 = 2$, of link e_2 is $c_2 = 1$. A resource allocation that satisfies the link capacity constraints is $x_2 = x_3 = 0.5$ and $x_1 = 1.5$. Indeed, MMF attempts to divide the capacity of each link among the demands. Then, the demands d_1 and d_3 on the link e_1 would get a rate of 1 each. The demand d_2 and d_3 on link e_2, would get a rate of 0.5 each. However, since the demand d_3 covers both links e_1 and e_2 it can only transmit at rate 0.5. Thus, there is still one more unit of capacity remaining to be allocated on link e_1. This remaining capacity will be allocated to the only other demand using that link, which is demand d_1, thus given $x_1 = 1.5$, $x_2 = 0.5$ and $x_3 = 0.5$. Then, supposing that $c_1 = c_2 = 1.5$, the resource allocation that satisfies the link capacity constraints is $x_1 = x_2 = x_3 = 0.75$. Clearly this is not optimal with respect to the throughput. In fact the resulting throughput is equal to $x_1 + x_2 + x_3 = 2.25$.

In the alternative *Proportional Fairness* (PF) [94, 95] allocation that is applicable to TCP, utility $U(x_j)$ is set to $\omega_j \; log \; x_j$, where ω_j is the weight of the session j. Hence, the resource allocation corresponding to this utility function is commonly referred to as *weighted proportionally fair*, or, if all ω_d are equal to one, as *proportionally fair*. Thus, (2.1) for PF becomes:

$$\max_{X \geq 0} \sum_{j \in J} \omega_j \log X_j \tag{2.3}$$

The *log* function avoids the assignment of zero or too low volumes to demands since $log\ x \to -\infty$ as $x \to 0$. Furthermore, it makes it not profitable to assign much volume to any demand. We note that the PF objective function is non-linear with respect to the MMF one. Taking the example in Figure 2.10, the corresponding explicite resource allocation is given by:

$$max\ \{logx_1 + logx_2 + logx_3\} \tag{2.4}$$

subject to:

$$x_1 + x_3 \leq 2, \tag{2.5}$$

$$x_2 + x_3 \leq 1, \tag{2.6}$$

$$x_1, x_2, x_3 \quad \in R^+. \tag{2.7}$$

The corresponding weighted proportionally fair resource allocation is $c_1 = c_2 = 1.5$, which favor shortest flows. The resource allocation that satisfies the link capacity constraints is $x_1 = x_2 = 1$ and $x_3 = 0.5$. From a fairness perspective, the PF solution is less fair than the MMF solution. However, since it favors shorter flows, the PF allocation is more efficient in terms of global throughput, in this case it is equal to $x_1 + x_2 + x_3 = 2.5$.

An open question is what happens in terms of fairness in Cloud fabrics that today expose a large number of paths to applications and servers, also using extended versions of TCP such as Mutlipath TCP [96]. How path diversity coupled with multipath transport can affect TCP fairness is a question to which we try to answer in the next chapters.

Chapter 3

Traffic Engineering and Energy Efficiency in Virtual Machine Placement

The increasing adoption of server virtualization has recently favored three key technology advances in data center networking: the emergence at the hypervisor software level of virtual bridging functions, between virtual machines and the physical network; the possibility to dynamically migrate virtual machines across virtualization servers in the data center network (DCN); a more efficient exploitation of the large path diversity by means of multipath forwarding protocols. In this chapter, we investigate the impact of these novel features in DCN optimization by providing a comprehensive mathematical formulation and a repeated matching heuristic for its resolution. We show, in particular, how virtual bridging and multipath forwarding impact common DCN optimization goals, traffic engineering (TE) and energy efficiency (EE), and assess their utility in the various cases of four different DCN topologies. We show that virtual bridging brings a high performance gain when TE is the primary goal and should be deactivated when EE becomes important. Moreover, we show that multipath forwarding can bring relevant gains only when EE is the primary goal and virtual bridging is not enabled.

3.1 Introduction

The advent of efficient software virtualization techniques allows running server virtualization at competitive performance-cost trade-offs with respect to legacy solutions. The increasing adoption of server virtualization has recently favored three key technology advances in data center networking: the emergence of virtual bridging

functions at the hypervisor software level (between virtual machines and the physical network); a more efficient exploitation of path diversity by means of multipath forwarding protocols; the possibility to dynamically migrate virtual machines across virtualization servers at different places in the data center networking (DCN). In the context of DCN optimization, virtual bridging is useful for the management of Virtual Machines (collocated in the same virtualization server), by offloading inter-Virtual Machine (VM) traffic from access and aggregation switches at the expense of an additional computing load on the virtualization server. Moreover, with the emergence of flat DCN topologies, such as Fat-Tree [26], DCell [28], and BCube [27], offering more path diversity, multipath forwarding can become useful to fully utilize the available paths and capacity and therefore offer higher throughput and resiliency to the servers. The ability to synchronize VM copies and migrate across virtualization servers (referred in the following also as 'containers' or 'VM containers') further adds elasticity to the cloud fabric by allowing fault-restoration and resource consolidation.

Virtual machine placement algorithms typically address a traffic engineering (TE) [75, 97] or a energy efficiency (EE) [98, 99] objective, that is, such as to minimize the maximum link utilization when balancing the traffic load on DCN links or to maximize server utilization to turn off or hibernate some servers to save energy. Addressing TE and EE goals eventually leads to savings in DCN maintenance and planning costs while increasing the performance. The relationship between the presented three recent trends, virtual bridging, multipath forwarding, and VM placement, is a rather unexplored subject that we investigate in our contribution.

The contribution of this chapter is is two-fold:

- Given that, to the best of our knowledge, no work at the state of the art offers a DCN optimization framework supporting virtual bridging and multipath forwarding, we formally formulate the virtual machine placement optimization problem with these features in a novel, compact, and versatile formulation. For its resolution with dense, flat, and large DCN topologies, we propose a repeated matching heuristic.

- We analyze the impact of virtual bridging and multipath forwarding in DCN optimization with TE and EE objectives, and with a detailed sensibility analysis including four different DCN topologies (3-layer, FatTree, BCube, DCell) that cover all possible cases. We draw observations on the case-by-case suitability of virtual bridging and multipath forwarding features with respect to DCN VM placement optimization.

In the following, the DCN optimization model is formulated in Section 3.2, the proposed heuristic in Section 3.3, and simulation results are in Section 3.4.

Table 3.1: Mathematical notations

N	set of VM containers and RBs; $n \in N$.
C	container set; $C \subset N$.
V	VM set; $V \subset N$.
R	RB set; $R \subset N$. $R_a \subset R$ is the access RB set.
T^V	set of VM pairs; $T^V \subset V \times V$.
T^C	set of container pairs; $T^C \subset C \times C$.
T^R	set of RB pairs; $T^R \subset R \times R$.
Variables and Parameters	
$e_{v,c}$	1 when v is at c, 0 otherwise. $v \in V, c \in C$.
b_c	1 if c is enabled, 0 otherwise. $c \in C$.
$a_{c,r}$	1 when c traffic via r. Multipath: $\in [0,1]$. $c \in C, r \in R$.
$\rho^k_{s,d}$	1 if traffic from r_s to r_d transits by the k^{th} path if unipath. Multipath: $\in [0,1]$. $(r_s, r_d) \in T^R$.
t_{c_i,c_j}	traffic from c_i to c_j, $(c_i, c_j) \in T^C$.
t_{r_i,r_j}	traffic from r_i to r_j; $(r_i, r_j) \in T^R$.
$t_{c,r}$	traffic from $c \in C$ to $r \in R$.
U	maximum network link utilization.
K^P_c	power capacity of container $c \in C$.
K^M_c	memory capacity of container $c \in C$.
d^P_v	computing power demand of VM $v \in V$.
d^M_v	memory demand of VM $v \in V$.
t_{v_i,v_j}	traffic from v_i to v_j, $(v_i, v_j) \in T^V$; $t_{v_i,v_i} = 0$.
$K_{i,j}$	(i,j) link capacity, null if no link; $(i,j) \in N \times N$.
$p^{k,s,d}_{i,j}$	1 when k^{th} path from r_s to r_d uses link (r_i, r_j).
α	trade-off coefficient between TE and EE objective, $\alpha \in [0,1]$.

Section 3.5 concludes our contribution.

3.2 Optimization Problem

In the following, we present the mathematical formulation of the target VM placement problem. The optimization problem is to determine how to place VMs at VM containers in a DCN supporting virtual bridging and/or multipath forwarding while satisfying TE and/or EE goals. We first present the formulation with no multipath forwarding and virtual bridging capabilities, and then we show how it can be easily extended to enable these features. The notations are provided in Table 4.1.

Our problem is a bi-criteria optimization problem: the goal is to minimize both

the maximum link utilization (TE goal), and the number of enabled containers (EE goal), simultaneously. This problem can be formulated as follows:

$$\begin{cases} \min\ U \\ \min \sum_{c \in C} b_c \end{cases} \tag{3.1}$$

Subject to the following constraints:

A VM is assigned to only one container:

$$\sum_{c \in C} e_{v,c} = 1 \quad \forall v \in V. \tag{3.2}$$

A container is enabled only if it hosts at least one VM:

$$b_c \geq e_{v,c} \quad \forall c \in C,\ \forall v \in V. \tag{3.3}$$

Each container is assigned to one RB:

$$\sum_{r \in R} a_{c,r} = 1 \quad \forall\ c \in C. \tag{3.4}$$

Traffic between two access RBs is sent over a single path:

$$\sum_{k} q^k_{r_s,r_d} = 1 \quad \forall (r_s, r_d) \in R_a \times R_a. \tag{3.5}$$

A VM is assigned to a container only if there are available residual computing resources:

$$\sum_{v \in V} d^P_v\ e_{v,c} \leq K^P_c \quad \sum_{v \in V} d^M_v\ e_{v,c} \leq K^M_c; \quad \forall c \in C. \tag{3.6}$$

Container-RB traffic does not violate the access link capacity:

$$t_{c,r} \leq K_{c,r} \quad \forall\ c \in C \quad \forall r \in R. \tag{3.7}$$

Similarly for the aggregation-core link capacity:

$$\sum_{r_s,r_d} \sum_{k} t_{r_s,r_d} \rho^k_{r_s,r_d} p^{k,r_s,r_d}_{r_i,r_j} < U\ K_{r_i,r_j} \quad \forall (r_i, r_j) \in T^R \tag{3.8}$$

$$t_{c,r} = \sum_{(v_i,v_j) \in T^V} (t_{v_i,v_j} + t_{v_j,v_i})\ e_{v_i,c}\ a_{c,r} \quad \forall r \in R,\ \forall c \in C$$

$$t_{r_s,r_d} = \sum_{(c_i,c_j) \in T^C}^{i \neq j} t_{c_i,c_j}\ a_{c_i,r_s}\ a_{c_j,r_d} \quad \forall (r_s, r_d) \in T^R$$

$$t_{c_i,c_j} = \sum_{(v_x,v_y) \in T^V} (t_{v_x,v_y} + t_{v_y,v_x})\ e_{v_x,c_i}\ e_{v_y,c_j} \quad \forall\ c_i, c_j \in C.$$

We could solve this bi-criteria optimization problem by searching the Pareto frontier (the set of optimal solutions to this problem), and then select a Pareto optimal solution that offers a good trade-off between our two criteria. However, in this dissertation we propose to simplify the problem by scalarizing it to a single objective function, which is a linear combination of our two criteria weighted by the α trade-off parameter. Hence, data center providers can control the importance they want to give to the one or to other criterion. We note that this method is one of the most common general scalarization methods for multi-objective optimization [100]:

$$\min \ \alpha \ U + (1 - \alpha) \sum_{c \in C} b_c \tag{3.9}$$

Enabling multipath capabilities

Multipath forwarding between containers and RBs (in the place of link aggregation/bonding or similar approaches) can simply be enabled by declaring $a_{c,r}$ as a non-negative real variable. Hence, we add the following integrity constraint:

$$\sum_{k} a_{c,r} = 1 \quad \forall \ c \in C. \tag{3.10}$$

Similarly, multiple paths between RBs can be enabled by declaring $\rho^k_{s,d}$ as a non-negative real variable and adding the following integrity constraint:

$$\sum_{k} \rho^k_{r_s,r_d} = 1 \quad \forall (r_s, r_d) \in R_a \times R_a, \tag{3.11}$$

where the traffic between RBs in (3.8) becomes

$$t_{r_s,r_d} = \sum_{(c_i,c_j) \in T^C}^{i \neq j} t_{c_i,c_j} a_{c_i,r_s} \ a_{c_j,r_d}; \ \forall (r_s, r_d) \in T^R \ \forall \ c_i, c_j \in C.$$

Enabling virtual bridging

Enabling virtual bridging means that the container takes the function of a bridge (typically at the hypervisor level). This feature can be easily included by transforming the variable $a_{c,r}$ in a parameter and extending the RB set including the container nodes. Given that virtual bridging consumes additional power and memory, (4.3) should be slightly changed so that such an additional component (function of the traffic load) is included.

The use of virtual bridging consumes VM container power and memory. The constraints (3.6) change to (3.12) and the traffic between VMs in the same container no longer transits by physical bridges.

$$\tau(t_{c,c} + t_{c,r} + t_{r,c}) + \sum_{v \in V} d^P_v \ e_{v,c} \leq K^P_c; \quad \forall \ c \in C$$

$$\gamma(t_{c,c} + t_{c,r} + t_{r,c}) + \sum_{v \in V} d_v^M \; e_{v,c} \leq K_c^M; \quad \forall \; c \in C. \tag{3.12}$$

This optimization model is an extension of the baseline multi-commodity flow (MCF) problem for network routing with link capacity constraints [93] by taking into account peculiar data center networking constraints due to VM mobility, VM container switching on and off, virtual bridging, and multipath forwarding.

Given the elasticity related to VM migrations and multipath forwarding that require double mapping, between VMs and VM containers and between VM containers and usable paths, our optimization problem, beside being comprehensive and versatile (considering both unipath and multipath modes with and without virtual bridging) is a non-linear one.

To summarize, the problem is

$$\min \;\; (3.9) \;\; \textit{subject to} : (3.2), (3.3), (3.4), (3.5), (4.3), (3.7) \; \textit{and} \; (3.8)$$

where (4.3) is replaced by (3.12) for the virtual bridging case. Contraints (3.10) and (3.11) are added for the multipath case. The problem has $|V| + 2|C| + |R_a|^2$ variables, $|V| + |C| + |R_a|^2$ with multipath forwarding and virtual bridging, and less than $|V| + |C|(2 + |V||R|) + 2|R|^2$ constraints.

3.3 Heuristic approach

Classically, mapping problems reduce to facility location problems; when capacity constraints need to be verified as a function of the type of mapping, there are similarities with the capacitated facility location problem [101], in particular, to the single source facility location problem (SSFLP) [102, 103].

It is easy to derive that our DCN optimization problem can be reduced to the SSFLP, and hence, is NP-hard. In the SSFLP, we have a set of customers that must be served by a single facility, and there is a cost associated with opening a facility in a particular location and a transportation cost from the facility to the customer. Each customer has a particular demand and each facility has a limited capacity. The problem is to find where to locate the facilities to minimize the cost of the network.

Our optimization problem can be reduced to an instance of a SSFLP as follows. Each VM pair can be seen as a customer of the SSFLP, and the VM pair has a global traffic demand that can be seen as the customer demand from a facility of the SSFLP. To translate our problem to a SSFLP instance, let (i) the traffic demand between two VMs of our VM placement problem correspond to a customer demand from a facility in the SSFLP; (ii) the cost of the link between the potential containers (where the two VMs can be located) and each access RBs correspond to half the

cost between the potential facility location and the customer of the SSFLP; (iii) the first assignment of a VM to a container corresponds to the cost of opening a facility of the SSFLP; (iv) the container capacity constraint corresponds to the capacity of the facility in the SSFLP. In this way, the solution of such an SSFLP instance provides us with the solution of our VM placement problem and hence, NP-hard.

3.3.1 Reformulation of the optimization problem

Recently, modeling an optical network design problem as a facility location problem, the authors in [104] extended a repeated matching heuristic described in [102, 103] to solve the SSFLP and proved it can reach good optimality gaps for many large instances of the problem.

Motivated by those results and basic similarities with the problem in [104], we redesign our DCN optimization problem as a repeated matching problem. With respect to the network context of [104], we have more matching sets with peculiar constraints due to fundamental differences between optical networks and DCNs. The double mapping we have to handle in our problem and the multiple capacity constraints we have to care about (at both the link and VM container sides) makes this problem more combinatorial than [104], so that comparison to the optimum is not differently possible than in previous applications [102–104].

In our DCN scope, communications are between VMs that can be hosted behind the same VM container or behind distant containers interconnected by a DCN path. External communications can be modeled introducing fictitious VMs and VM containers acting as an egress point, if needed. When multipath is enabled, multiple paths can be used toward a same destination, and when virtual bridging is enabled, a VM container can transit external traffic if the topology supports it. When communicating VMs are not collocated, inter-VM communication should involve a pair of containers and at least a DCN path between them.

Let a VM container node *pair* be designated by cp, $cp \in T^C$, so that $cp = (c^i, c^j)$, i.e., a container pair is composed of two containers c^i and c^j. When the source (c^i) and the destination (c^j) of the container pair cp are the same ($c^i = c^j$) it is called *recursive container*. A subset of container node pairs is designated by D^C so, $D^C \subseteq T^C$. Let the k^{th} path from RB r^1 to RB r^2 be designated by $rp = (r^1, r^2, k)$. A set of RB paths is designated by D^R so that $D^R \subset T^R$.

Definition 3.3.1. A **Kit** ϕ is composed of a subset of VMs D^V, a VM container pair $cp \in T^C$ and a subset of RB paths D^R. Each VM $v \in D^V$ is assigned to one of the containers in a pair cp (c^1, c^2). A container pair cp (c^1, c^2) is connected by each RB path rp $(r^1, r^2, k) \in D^R$, so that c^1 and c^2 are respectively mapped to r^1 and r^2. The Kit is recursive when its cp is recursive, and in such a case D^R must

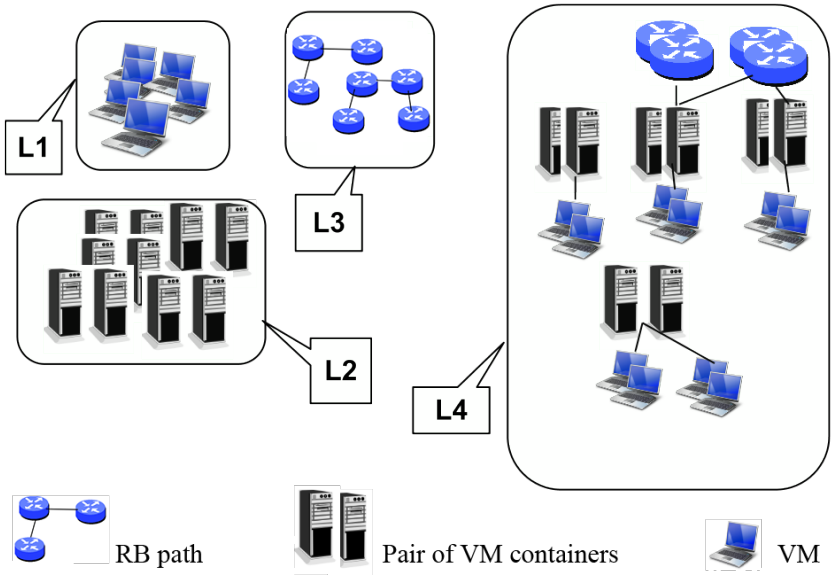


Figure 3.1: Representation of heuristic sets: L_1, L_2, L_3, and L_4.

be empty. When the multipath is not enabled, $|D^R| = 1$. The Kit is denoted by $\phi(cp, D^V, D^R)$.

Definition 3.3.2. A Kit is a **Feasible Kit** if:

- D^V is not empty, i.e., $D^V \neq \oslash$.
- Memory and power demands of each VM are satisfied, i.e.(4.3), restricted to D^V and cp.
- In case of a non-recursive Kit, the link capacity constraints between VM containers are satisfied, i.e., (3.7) is restricted to D^V, D^R and cp.

Definition 3.3.3. $\mathbf{L_1}, \mathbf{L_2}, \mathbf{L_3}$, **and** $\mathbf{L_4}$

- $\mathbf{L_1}$ is the set of VMs not matched with a container pair, i.e., $\mathbf{L_1} = \{v \mid v \in D^V \wedge v \notin \phi\}$.
- $\mathbf{L_2}$ is the set of VM container pairs not matched with an RB path, i.e., $\mathbf{L_2} = \{cp \mid cp \in T^C \wedge cp \notin \phi\}$.
- $\mathbf{L_3}$ is the set of RB paths not matched with a container pair, i.e., $\mathbf{L_3} = \{rp \mid rp \in T^R \wedge rp \notin \phi\}$.
- $\mathbf{L_4}$ is the set of Kits. It is worth mentioning that $\mathbf{L_4}$ becomes Packing when all its kits are feasible.

Definition 3.3.4. A **Packing** $\mathbf{\Pi}$ is a union of Kits in $\mathbf{L_4}$. A Packing is said to be feasible if it contains at least one feasible Kit and $\mathbf{L_1}$ is empty.

3.3.2 Matching Problem

Given the DCN optimization problem elements using the above described sets, it can be reformulated as a matching problem. The classical matching problem can be described as follows. Let A be a set of q elements $h_1, h_2, \ldots, h_q$. A matching over A is such that each $h_i \in A$ can be matched with only one $h_j \in A$. An element can be matched with itself, which means that it remains unmatched. Let $s_{i,j}$ be the cost of matching h_i with h_j. We have $s_{i,j} = s_{j,i}$. Let $z_{i,j}$ be a binary variable equal to 1 if h_i is matched with h_j. The matching problem consists in finding the matching over A that minimizes the total cost of the matched pairs.

$$\min \quad \sum_{i=1}^{q} \sum_{j=1}^{q} s_{i,j}\ z_{i,j} \tag{3.13}$$

$$s.t. \quad \sum_{j=1}^{q} z_{i,j} = 1, \quad i = 1, \ldots, q \tag{3.14}$$

$$\sum_{i=1}^{q} z_{i,j} = 1, \quad j = 1, \ldots, q \tag{3.15}$$

$$z_{i,j} = z_{j,i}, \quad i, j = 1, \ldots, q \tag{3.16}$$

$$z_{i,j} \in \{0, 1\}, \quad i, j = 1, \ldots, q. \tag{3.17}$$

(3.14) and (3.15) ensure that each element is exactly matched with another one. (3.16) ensures that if h_i is matched with h_j, then h_j is matched with h_i. (3.17) sets $z_{i,j}$ as binary.

In our heuristic, one matching problem is solved at each iteration between the elements of L_1, L_2, L_3, and L_4. At each iteration, the number of matchable elements is $n_1 + n_2 + n_3 + n_4$ where n_1, n_2, n_3, and n_4 are the current cardinalities of the four sets, respectively. For each matching iteration, the costs $s_{i,j}$ have to be evaluated. The cost $s_{i,j}$ is the cost of the resulting element after having matched element h_i of L_1, L_2, L_3, or L_4 with element h_j. A basic example of matching is shown in Figure 3.2: L_1 is empty ($n_1 = 0$), L_2 has two containers ($n_2 = 2$), L_3 has two RB paths ($n_3 = 2$), and L_4 has one feasible Kit ($n_4 = 1$). The result of the matching creates an unfeasible Kit, and modifies the existing feasible Kit with an additional RB path ($n_1 = n_2 = n_3 = 0$, $n_4 = 2$). At each iteration, the least-cost matching between the elements has to be determined. The computed matching costs $z_{i,j}$ are stored in a $(n_1 + n_2 + n_3 + n_4) \times (n_1 + n_2 + n_3 + n_4)$ cost matrix Z. Z dimensions change at each iteration, and Z is a symmetric matrix. Given the symmetry, only ten blocks have to be considered. The notation $[L_i - L_j]$ is used hereafter to indicate the matching between the elements of L_i and the elements of L_j as:

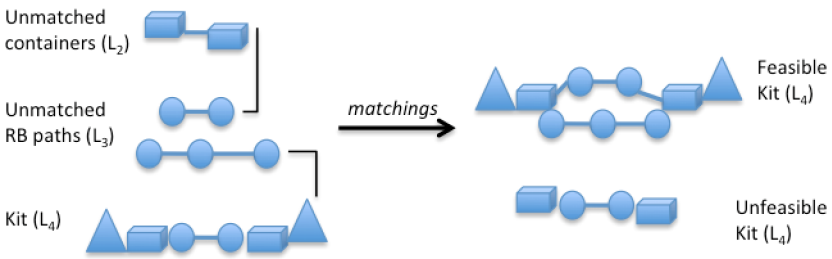


Figure 3.2: A simple example of matching.

$$Z = \begin{pmatrix} [L_1 - L_1] & [-] & [-] & [-] \\ [L_2 - L_1] & [L_2 - L_2] & [-] & [-] \\ [L_3 - L_1] & [L_3 - L_2] & [L_3 - L_3] & [-] \\ [L_4 - L_1] & [L_4 - L_2] & [L_4 - L_3] & [L_4 - L_4] \end{pmatrix}$$
$$= \begin{pmatrix} [1] & [-] & [-] & [-] \\ [2] & [3] & [-] & [-] \\ [4] & [5] & [6] & [-] \\ [7] & [8] & [9] & [10] \end{pmatrix}.$$

Selecting the least cost matching vector enables solution improvements via set transformations in the next iterations. Obviously, $L_1 - L_1$, $L_2 - L_2$ and $L_3 - L_3$ matchings are ineffective. To avoid a matching, e.g., because it is infeasible, its cost is set to infinity (a large number in practice). Matchings corresponding to other blocks without L_4 lead to the formation of Kits. Other matchings involving elements of L_4 shall lead to the improvement of the current Kits that also generate local improvements due to the selection of better VM containers or RB routes. Note that for such blocks, local exchange problems are to be solved for determining an exchange of VMs, VM containers and Kits, between the heuristic sets, while satisfying computing capacity constraints. The details on how to precisely compute each block matching costs are given in the Appendix.

The Kit cost computation has to maintain the same rationale as in the reference optimization problem when setting individual matching costs. The cost needs to be computed to de-motivate under-loading VM containers in terms of CPU and RAM utilization, while avoiding over-loading RB paths in terms of link utilization and respecting computing capacity constraints. The Kit cost function has to appropriately model two opposite forces due to the dual aspects stressing DCNs: computing and network resources. On the one hand, the Kit feasibility in terms of the link capacity constraints as described above, does not need to be enforced during the repeated matching iterations but rather to be motivated via the classical TE costs

inducing the minimization of the maximum link utilization, and hence maximizing the minimum residual link capacity.

On the other hand, the residual computing capacity at the VM container level should be considered as a cost. In fact, it is not suitable for the DCN provider to have idle memory and CPU capacities. The overall Kit cost is not meant to represent a direct monetary cost, but it is so that the repeated matching promotes more efficient Kits. Therefore, to align with the objective function (3.9), and by remembering that the cost of a Packing corresponds to the cost of its Kits, we set the cost of a Kit $\phi(cp, D^V, D^R)$ as:

$$\mu(\phi) = (1-\alpha)\mu^E(\phi) + \alpha\mu^{TE}(\phi). \tag{3.18}$$

Where α is the trade-off scaling factor between the EE and the TE components, that are, respectively:

$$\mu^E(\phi) = \sum_{c^i \in cp} \left(\frac{K^P_{c^i}}{\sum\limits_{v \in D^V_i} d^P_v} + \frac{K^M_{c^i}}{\sum\limits_{v \in D^V_i} d^M_v} + \Gamma T_v \right) \tag{3.19}$$

$$\mu^{TE}(\phi) = \max_{(n_i,n_j) \in rp, rp \in \phi} U_{n_i,n_j}(\Pi). \tag{3.20}$$

Where T_v represents the global traffic v sends and receives, i.e., $T_v = \sum_{v' \in V}^{v \neq v'} t_{v,v'}$. Γ is the additional power and memory, to take into account the impact of traffic to the VM container CPU and memory consumption when virtual bridging is enabled (zero otherwise).

Note that the computing capacity constraints are indirectly enforced within the $[L_4 - L_4]$ matching cost computation. $U_{n_i,n_j}(\Pi)$ is the link utilization of each link used by the current Packing Π solution, so that the maximum link utilization experienced by the Kit RB paths can be minimized. In our heuristic, in order to linearly compute the RB paths link utilization, the aggregation and core links of the RB paths are considered as congestion free, while the access container-RB links are considered as prone to congestion, which adheres to the reality of DCNs today. This is a realistic approximation we believe to be acceptable in a heuristic approach, especially because it allows a significant decrease of the time complexity.

3.3.3 Steps of the repeated matching heuristic

Due to the advantage of repeated matching between the different sets as described above, we can get rid of the non-linearities of the reference optimization problem with a heuristic approach that, based on the state of the art, is geared to achieve low optimality gaps.

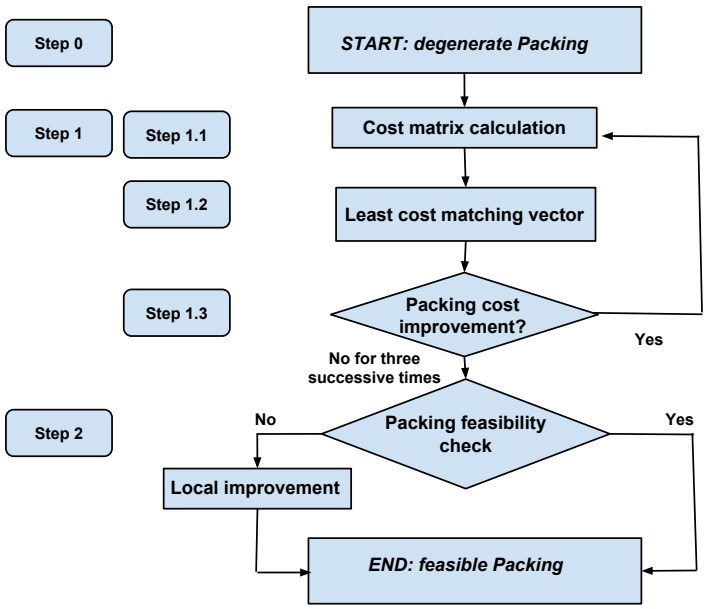


Figure 3.3: Chart of the repeated matching heuristic steps.

A global chart resuming the steps of our repeated matching heuristic is given in Figure 3.3. Its steps are as follows:

- **Step 0**: The algorithm starts with a degenerate Packing with no Kits and all other sets full.
- **Step 1**: A series of Packings is formed.
- **Step 1.1**: The cost matrix Z is calculated for every block.
- **Step 1.2**: The least cost matching vector is selected.
- **Step 1.3**: Go back to 1.1 for a new iteration unless the Packing cost has not changed in the last three iterations.
- **Step 2**: The heuristic stops, and in the case that L_1 is not empty, a local incremental solution is created assigning VMs in L_1 to enabled and available VM containers or, if none, to new containers.

The least cost matching computation (Step 1.2) can be hard to optimally solve because of the symmetry constraint (3.16). In our heuristic, we decided to solve it in a suboptimal way to lower the time complexity. We have implemented the algorithm in [105], based on the method of Engquist [106]. Its starting point is the solution vector of the matching problem without the symmetry constraint (3.16) obtained with the algorithm described in [107] that was chosen for its speed performance. Its output is a symmetric solution matching vector.

For instance, if the resulting Packing in Figure 3.2 does not change for three times, the result composed of the feasible kits (one Kit in the example) is kept as final result. Designing the matching costs in an efficient and rational way, the Packing cost across iterations should be decreasing - the decrease is expected to be monotonic starting by the moment when L_1 gets empty, so that the heuristic converges.

3.3.4 Time complexity

The complexity of the whole heuristic depends on its different sub-algorithms and phases. The calculation of the cost matrix is straightforward except for two blocks of the matrix (see blocks 10 and 8 in the Appendix) where a polynomial swapping problem depends on the number of connections in the network. The resolution of the matching problem operates on the cost matrix through the Forbes's [105] and the Volgenant's [107] algorithms. In the worst case, the first has a $\ominus(n^3)$ complexity while the second one has a $\ominus(n^2)$ complexity, where $n = n_1 + n_2 + n_3 + n_4$.

3.4 Simulation results

We implemented our heuristic using Matlab, and CPLEX for the computation of matching costs of some blocks. The adopted VM containers correspond to an Intel Xeon 5100 server with 2 cores of $2.33GHz$ and $20GB$ RAM and able to host 16 VMs. We use various weights for the TE and EE components in the optimization objective, and we analyze what happens when multipath forwarding and virtual bridging are enabled. We use the different forms of multipath forwarding, encompassing the following cases.

1. Multipath forwarding between RBs (MRB).
2. Multipath forwarding between containers and RBs (MCRB).
3. Both multipath forwarding modes (MRB-MCRB).

We executed our heuristic on the following topologies: 3-layer Figure 2.1, 4-pod Fat-Tree Figure 2.2, BCube Figure 2.3 with $n = 4$ and DCell Figure 2.4 with $n = 4$. We note that BCube and DCell work properly only by employing virtual bridging at the server level. We allowed for a small modification of the topology to allow a reference comparison between them and the other topologies for the cases without virtual bridiging, calling these variations BCube* 3.4 and DCell* 3.6, respectively. We allow the access switches in DCell* to be directly connected to each-other, and we allow each access switch in BCube* to be directly connected to all core switches.

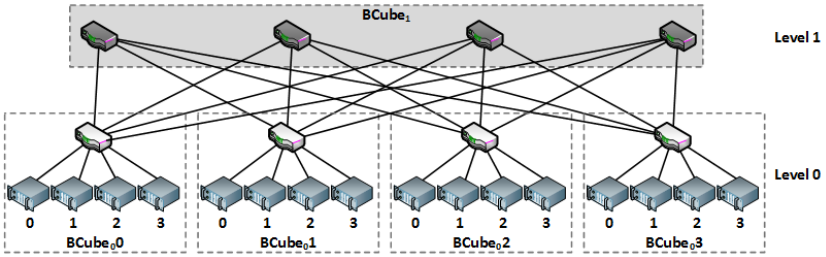


Figure 3.4: BCube* allows Multipath between RBs (MRB).

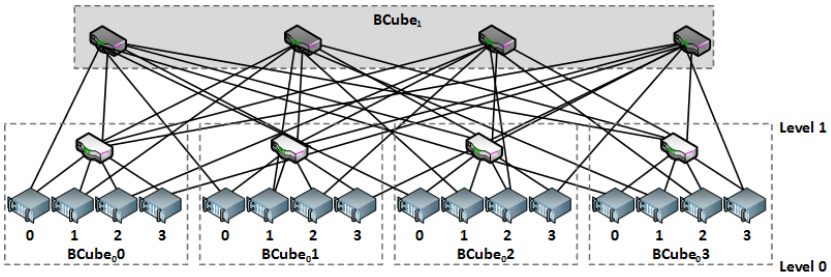


Figure 3.5: BCube** allows Multipath between RBs (MRB) and Multipath between Container and RB (MCRB).

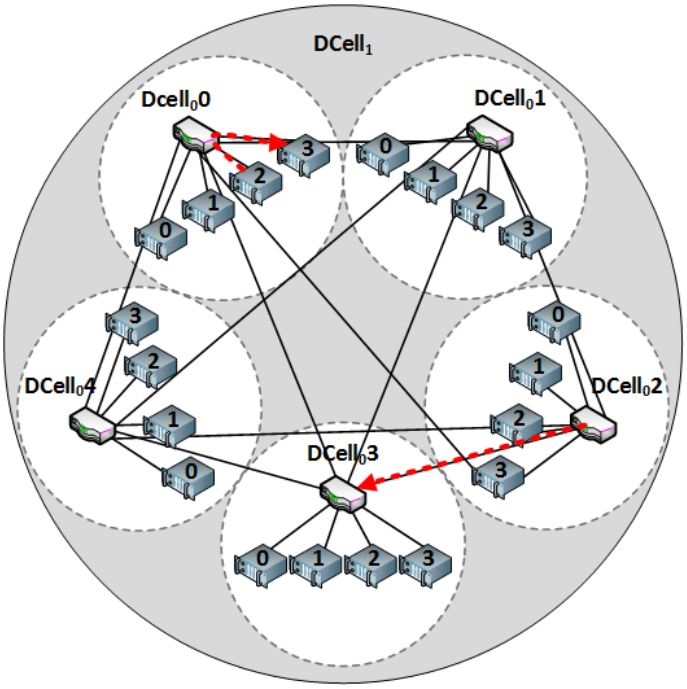


Figure 3.6: DCell* allows Multipath between RBs (MRB).

Table 3.2: Evaluated DCN setting cases.

VB	Multipath mode	Objective	Topologies
yes	MRB	EE	BCube, DCell
		TE	BCube, DCell
		EE+TE	BCube, DCell
no	MRB	EE+TE	3-layer, Fat-Tree, BCube*,DCell*
	MCRB	EE+TE	BCube**
	MRB-MCRB	EE+TE	BCube**

Table 3.3: Evaluated DCN size cases.

Topologies	Container number	Container capacity	VM number
Fat-tree, BCube	16	16	150
DCell	20	16	240

Moreover, the 3-layer, Fat-Tree and DCell topologies have no multipath forwarding capabilities between containers and RBs, because there are no multiple links between containers and RBs (only BCube has that specificity). To also evaluate the case with multipath between RBs and containers, we use another variation to BCube, referred to as BCube** 3.5, where each server is multi-homed with two switches, its pod switch and one core switch. Table 3.2 summarizes the topologies that do or do not support the virtual bridging mode for each multipath forwarding case.

In the simulations, all DCNs are loaded at 70% in terms of computing capacity. Table 3.3 summarizes each topology's container capacity, the number of VMs that a container can host, and the total number of VMs used for our simulations. Note that with all topologies, we allowed for a certain level of overbooking in the resource allocation area for the sake of algorithm fluidity, especially at starting and intermediate iterations. The capacity of the access link was set to 10 *Gbps*.

We ran 30 different instances with different traffic matrices for each case. The reported results have a confidence interval of 95%. Our heuristic reached convergence roughly within a dozen minutes of each execution and successfully reaching steady states.

In the following, we detail the adopted traffic model and related state of the art. Then, we report experiment results we obtained considering the impact of virtual bridging and multipath forwarding on EE-oriented and TE-oriented VM placement. We look at the impact of virtual bridging and multipath forwarding without mixing the TE and EE objectives, and we perform sensibility analysis by varying the EE-TE

α trade-off coefficient.

3.4.1 Traffic model

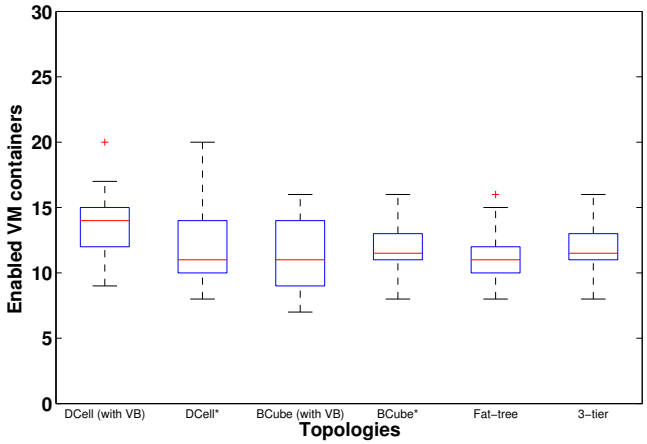
Choosing an appropriate traffic model is important when running DCN simulations. In the state of the art there are a few relevant works. We built our traffic drawing conclusions from the following studies [67,108–110]. Authors in [108,109], collected data from 3-layer-like DCNs and found that the traffic originating from a rack showed an ON/OFF behavior following heavy-tailed distributions. 80% of the DC server-originated traffic stayed within the rack, while between 40% and 90% left the rack. In [110], more than 90% of transfers had a volume from 100 MB to 1 GB. Moreover for 50% of its running time, a VM handled approximately 10 concurrent flows, and at least for 5% of its running time, a VM had more than 80 concurrent flows. In [67], the authors analyzed the incoming and outgoing traffic rates of an IaaS DCN with 17,000 VMs, reporting that 80% of the VMs had an average rate less than 800 KB/min. However, 4% of them had a ten times higher rate. Moreover, the traffic rate standard deviation was 82%, lower than or equal to twice the mean traffic.

Since not all VMs communicate with each other in todays DCNs adopting network virtualization, but instead traffic is segmented by IaaS networks, we built an IaaS-like traffic matrix as in [67], which somehow also takes into account the IaaS traffic rack vicinity trend reported in [108, 109]. We apply the experimental incoming and outgoing traffic distribution of [67], with IaaS clusters of up to 30 VMs communicating with each-other and not communicating with other IaaS VMs. Within each IaaS, the traffic matrix was built according to the traffic distribution of [110].

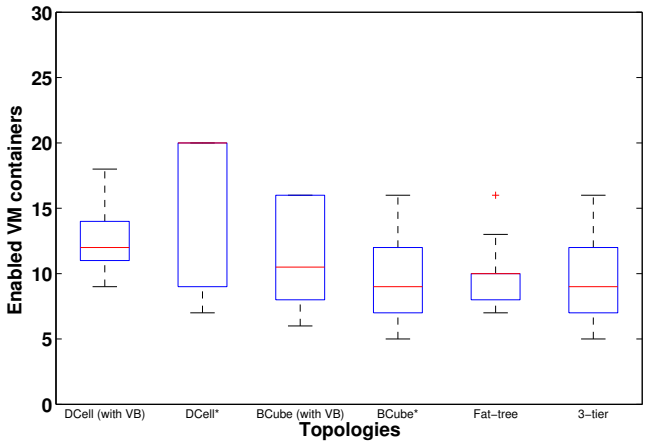
3.4.2 Energy efficiency considerations

Figure 3.7 illustrates the results in terms of enabled VM containers for different topologies when EE is the unique goal (i.e., $\alpha = 0$ in the problem formulations). We report results for both the cases when multipath forwarding is not enabled (i.e., $|D^R| = 1$ for all Kits) and the case where it is enabled. By observing the results we can assess the following:

- *Impact of virtual bridging*: When the EE is the goal, virtual bridging leads to negligible differences in EE performance;
- *Impact of multipath forwarding*: It has a positive impact on EE when virtual bridging is disabled, and seems to be counterproductive when virtual bridging is enabled;



(a) Unipath

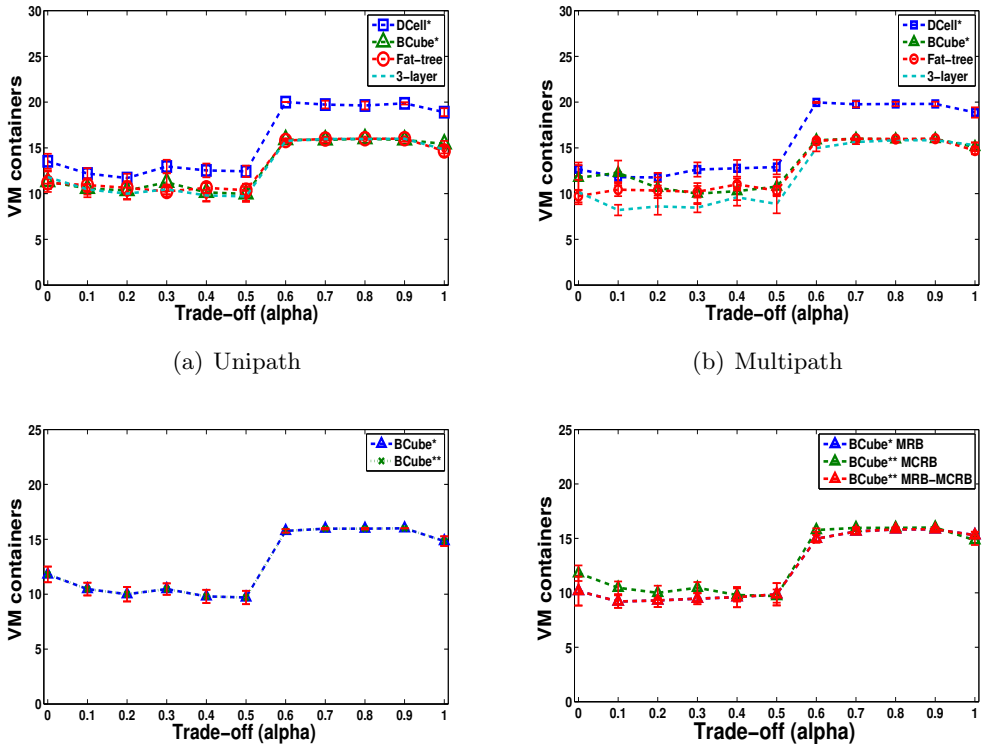


(b) Multipath

Figure 3.7: Number of enabled VM containers Energy Efficiency (EE) results, with EE as single objective (VB=Virtual Bridging).

- *Sensibility to topologies*: The DCell topology shows a better EE performance than the BCube, especially when multipath forwarding is enabled. This can be explained by the higher path diversity at the DCell container. Hierarchical topologies, Fat-Tree, and 3-layer, show an overall worst EE performance for single-path forwarding and a better EE performance for multipath forwarding, with a negligible difference.

This analysis suggests that enabling virtual bridging does not bring any useful EE gain and can even worsen the EE performance when the consolidation EE objective is minimizing the number of enabled VM containers. In the following, we



(a) Unipath (b) Multipath

(c) Unipath (d) Multipath

Figure 3.8: Number of enabled VM containers EE results, without virtual bridging.

discuss the results with an EE perspective when we vary the impact of the EE goal in the VM placement.

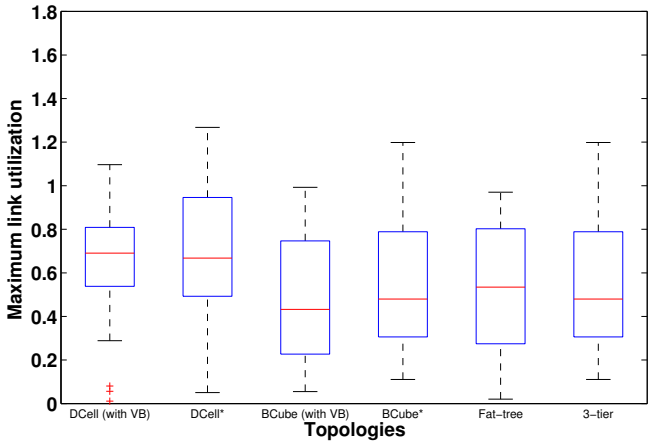
Optimization goal sensibility with respect to multi-path forwarding features

With respect to multipath forwarding features, Figs. 3.8 illustrates the results in terms of enabled VM containers for different values of the trade-off parameter α between the EE and TE goal, ranging from a null value giving full importance to the EE goal, to a maximum value giving importance to the TE goal, with a step of 0.25.

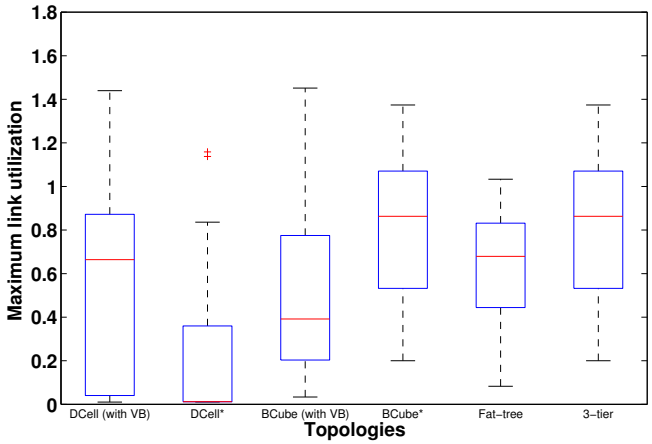
We report the results including the case when multipath is not enabled (i.e., $|D^R| = 1$ for all Kits) and the case where it is enabled. Observing the results we can assess that:

- When EE is discarded ($\alpha = 1$), the number of VM containers reaches its maximum, which could be expected.

- The results for all topologies are similar for MRB - the DCell slope is slightly



(a) Unipath

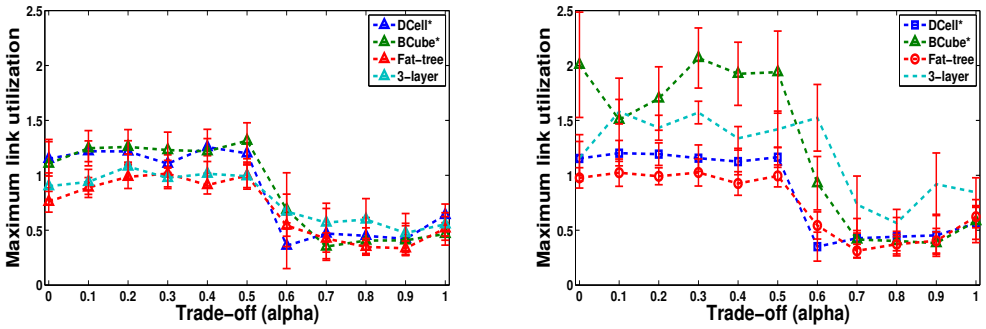


(b) Multipath

Figure 3.9: Maximum link utilization Traffic Engineering (TE) results, with TE as single objective (VB=Virtual Bridging).

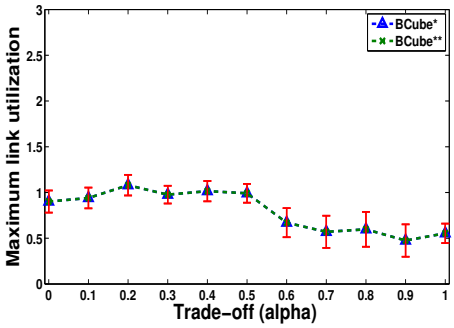
higher then the other ones, which can be explained by the number of containers within DCell topology, equal to 20.

- For MRB, the enabling of multipath forwarding decreases roughly up to 30% the number of enabled VM containers, and only by 20% for MCRB when EE is considered as an important objective.
- The impact of multipath forwarding becomes negligible when EE is not considered important;
- MRB-MCRB gives the same effect as enabling MRB.

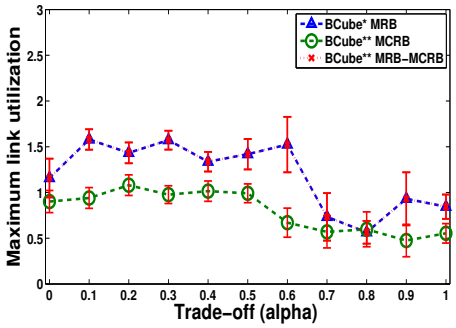


(a) Unipath

(b) Multipath



(c) Unipath



(d) Multipath

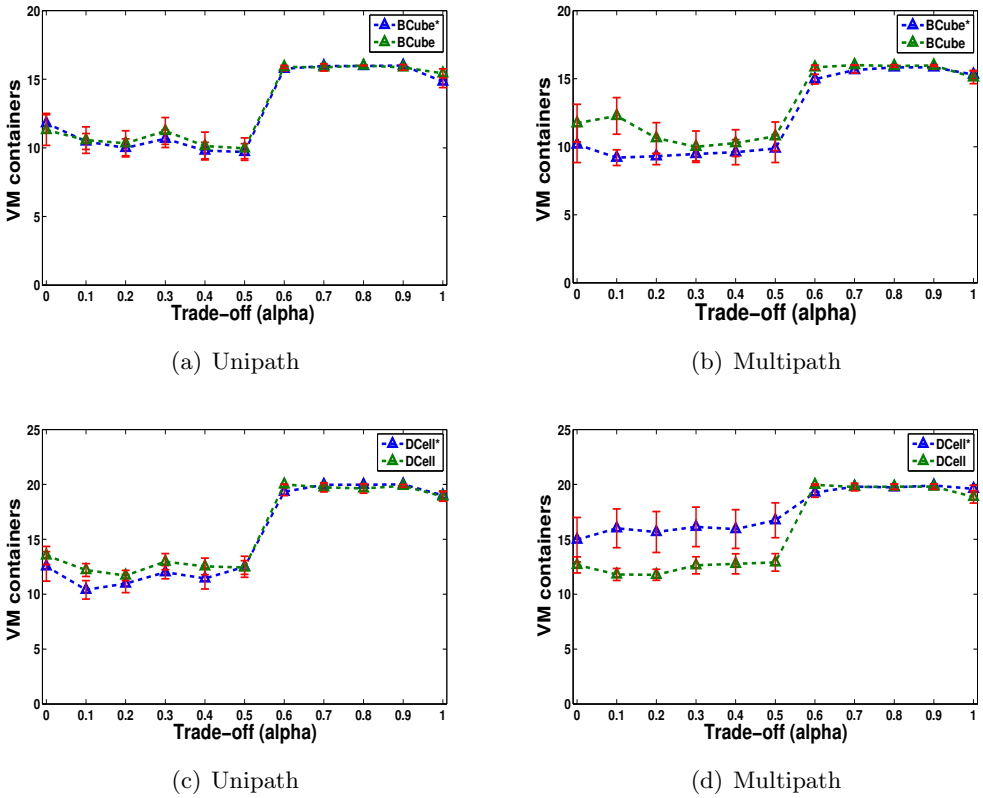
Figure 3.10: Maximum link utilization TE results, without virtual bridging.

These results are counter-intuitive. On the one hand, decreasing the access link bottleneck by enabling multipath L2 forwarding seems to free VMs and allow a better VM consolidation switching off unused containers that lead to energy gains. On the other hand, multipath communications appear not to be useful for that goal when switching off VM containers is either not interesting or not possible.

Optimization goal sensibility with respect to virtual-bridging features

Furthermore, we analyze the results with an EE perspective when both virtual bridging and multipath forwarding features are enabled. Figure 3.11 illustrates the results in terms of enabled VM containers for different values of the trade-off parameter α, ranging from a null value giving full importance to EE goal, to a maximum value giving importance to the TE goal. The figure reports the results for BCube and DCell topologies that support virtual bridging when multipath is enabled or not. Observing the results we can assess that:

- Enabling only virtual bridging (Figure 3.11(a,c)) does not have a relevant impact on the number of enabled VM containers. We note a negligible gain when EE is the goal ($\alpha = 0$), for BCube, and the opposite for DCell topologies



(a) Unipath

(b) Multipath

(c) Unipath

(d) Multipath

Figure 3.11: Number of enabled VM containers EE results, with virtual-bridging.

(both have the same confidence interval).

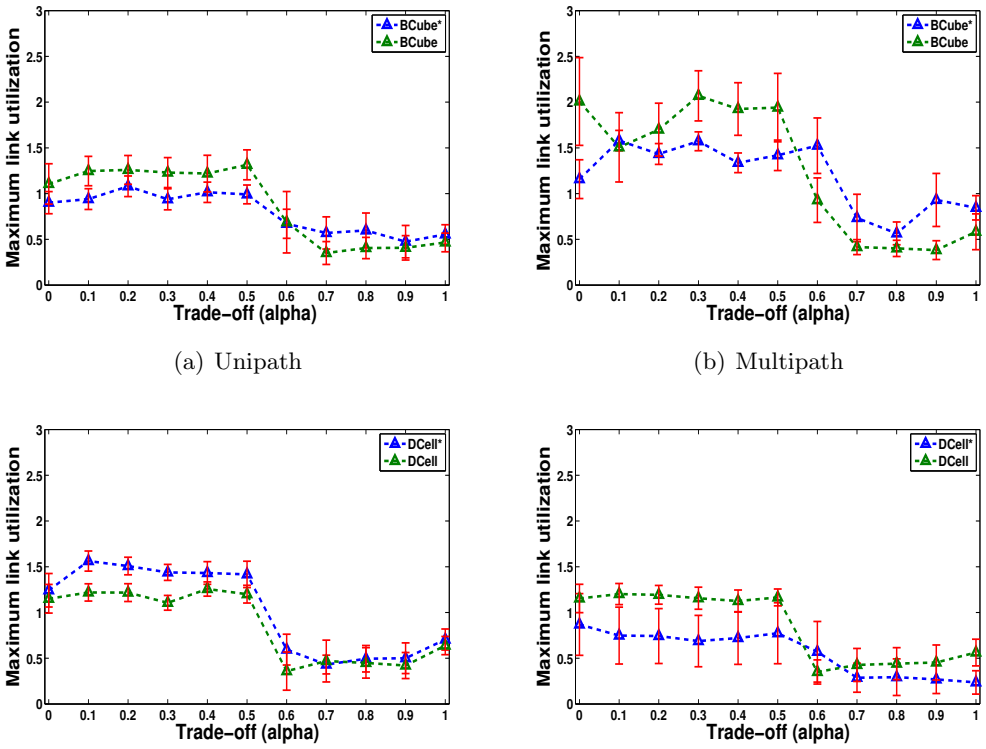
- Enabling multipath forwarding (Figure 3.11(b,d)) has a positive EE impact when virtual bridging is disabled (the two curves have the same confidence interval when virtual bridging is enabled) - this impact becomes negligible when EE is not considered as an important.

- We note that DCell topology shows better EE results than the BCube topology. In fact, the DCell topology used $\sim 55\%$ of the containers while BCube used $\sim 69\%$ of the containers.

These results confirm that, on the one hand, enabling virtual bridging has no impact on the EE goal regardless of the EE-TE trade-off level. On the other hand, multipath forwarding has a positive impact only when the virtual bridging is disabled.

3.4.3 Traffic engineering considerations

As already mentioned, EE goals can be considered the opposite of TE goals. Chasing EE tends to minimize the number of enabled VM containers, yet no care is given



(a) Unipath (b) Multipath

(c) Unipath (d) Multipath

Figure 3.12: Maximum link utilization TE results, with virtual bridging.

to network link utilization.

Figure 3.9 reports the results when TE is the single goal of the virtual machine placement optimization, i.e., $\alpha = 1$, considering singlepath and multipath forwarding for the different topologies. We make the following observations:

- *Impact of virtual bridging*: Virtual bridging lead to sensible TE performance gains.

- *Impact of multipath forwarding*: With singlepath forwarding, the DCell case gets the largest TE gain, from a median of roughly 65% of the maximum link utilization to roughly 45% - this is due to the fact that virtual bridging in the DCell allows for indirectly minimizing the number of links used to interconnect servers (with multipath forwarding, the BCube case gets the largest TE gain, with the maximum link utilization being halved from about 80% to 40%);

- *Sensibility to topologies*: BCube and DCell do have similar TE performances, with a slighter better performance with the BCube probably because the gain in path diversity brought by virtual bridging is higher with the BCube (which keeps a core layer unlike the DCell) - the TE gain with respect to hierarchi-

cal topologies (Fat-tree, 3-layer) is always positive and slightly higher with enabled multipath forwarding.

These TE performance results are not intuitive and appear to be quite relevant. It is interesting to adopt virtual bridging when the primary goal of the virtual machine placement is traffic engineering. Flat topologies show a sensible gain with respect to more hierarchical topologies, which once more motivate the migration to such new topologies for IaaS-based DCNs.

Optimization goal sensibility with respect to multipath forwarding features

Under a TE perspective, the lower the maximum utilization is, the better it is, to ensure highly efficient communications. Under this perspective, we perform a sensibility analysis varying the TE-EE trade-off. Figure 3.10 reports the maximum link utilization for both the unipath and the multipath cases, under different trade-off coefficients.

As expected, the curve decreases with α (oppositely to the previous curve, observing EE performance, in Figure 3.8). We make the following observations:

- MRB can be counterproductive: the unipath case guarantees a better TE performance when TE is not considered as an important goal in DCN optimization (i.e., when $\alpha \to 0$) - the MRB mode induces unacceptable TE performance when TE is not the primary goal.
- The curves of all topologies are similar for the unipath case - DCell has the worst curve when EE is the goal, and all curves converge when the maximum importance is given to TE.
- MRB-MCRB gives the same effect as enabling MRB.

Optimization goal sensibility with respect to virtual-bridging features

Furthermore, we analyse the impact of virtual bridging under a TE perspective. Figure 3.12 illustrates the results in terms of maximum link utilization for different values of the trade-off parameter, considering the case when multipath is enabled and when it is not enabled, for the DCell and BCube topologies. We make the following observations:

- With respect to the BCube case (Figure 3.12a-b), enabling virtual bridging when EE is the goal has a negative impact on the maximum link utilization and a positive one when TE is the goal.

- According to Figure 3.12c, the DCell topology is less impacted by virtual bridging than BCube, which can be explained by the totally flat nature of DCell (with its single access layer).

- Enabling multipath forwarding appears to have a negative impact on TE when TE is not the goal, regardless of the topologies.

We note that virtual bridging has a positive impact when TE is the goal, but it can have a negative TE impact when EE is the goal, and the topology that benefites the most from virtual bridging is BCube.

3.5 Summary

Data center networking is a challenging field of applications of old and new technologies and concepts. In this chapter, we investigate how traffic engineering (TE) and energy efficiency (EE) goals in virtual machine placement can coexist with the emergence of virtual bridging (i.e., the capability to switch traffic at the hypervisor level in virtualization servers) and of multipath forwarding (i.e., the capability to balance the load toward a same destination on multiple egress paths). We provide a versatile formulation of the virtual machine placement problem supporting virtual bridging capabilities and multipath forwarding and propose a repeated matching heuristic for its resolution.

Through extensive simulation of realistic instances with legacy and novel flat DC topologies (i.e., 3-layer, Fat-Tree, BCube, and DCell), we found that:

- *Impact of multipath forwarding*: Multipath forwarding has a positive impact on EE when virtual bridging is disabled, and this positive impact becomes negligible when virtual bridging is enabled or EE is not considered as important.
  Enabling multipath forwarding between virtualization servers (containers) and router-bridges (RBs) does not bring relevant additional performance gains with respect to both EE and TE goals. This suggests that legacy link aggregation/bonding protocols between servers and bridges are sufficient and multipath routing protocols do not need to be brought down to the hypervisor level.

- *Impact of virtual bridging*: When TE is the primary goal, virtual bridging shows notable positive gains, and the TE performance gain can be important and can be improved up to two times, with a maximum link utilization that can be halved for the BCube DCN topology, while remaining important also for the DCell topology. Flat topologies show a notable gain with respect to

more hierarchical topologies (3-layer, Fat-Tree) that once more motivate the migration to such new topologies for IaaS-based DCNs.

- *Sensibility to DCN optimization goals*: When EE is the primary goal of the DCN optimization, both multipath forwarding and virtual bridging can be counter-productive, leading to saturation at some access links. Instead, when TE is the primary goal, multipath forwarding grants only a moderate gain.

We believe these results provide important insights on the possible joint or individual adoption of multipath forwarding protocols and virtual bridging in data center networks, to help data center network designers take the right choices when energy efficiency and traffic engineering performance goals need to be taken into account.

Chapter 4

Traffic Fairness of Data Center Fabrics

A present challenge in data center networks (DCN) is to better understand the impact of novel flattened and modular DCN architectures on congestion control protocols, and vice-versa. One of the major concerns in congestion control is the fairness in the offered throughput: the impact of the additional path diversity and forwarding features, brought by the novel DCN architectures and protocols, on the throughput of individual endpoints (servers) and aggregation points (edge switches) being unclear. This contribution attempts to answer these open questions. Specifically, how best is the allocation of the competing elastic demand flows and how is this allocation impacted with the increase in capacity? We provide a linear programming formulation of the problem based on the proportional fairness principle of TCP. We conducted a series of test scenarios on the fat-tree and BCube data center topologies by considering load balancing and link capacities for different network cases in order to present our analysis on the results. We observed that the traffic allocation fairness is primarily impacted by the weights associated with the TCP implementation in use.

4.1 Introduction

The emergence of network virtualization solutions, such as Infrastructure as a Service (IaaS), offers several advantages to organizations in terms of both operational and capital expenditures [1]. The transition from physical independent networks to virtual de-localized networks operated in the Cloud can be facilitated if, besides security concerns, connectivity performance is at an acceptable level and shows desirable fairness properties.

With the growth in customer volumes, service differentiation and elastic demands, avoiding bottlenecks is a critical point in Data Center Network (DCN) architectures. With the de-facto dominating trend of deploying services using virtualization servers, a non negligible ratio of the traffic is horizontal traffic between virtualization servers, in support of virtual machine migration and storage synchronisation. The amount of intra-DC horizontal traffic can overcome the access vertical traffic volume [109]. This has eventually favored the emergence of novel DCN architectures that expose additional horizontal capacity between server racks and clusters of racks such as fat-tree (Fig. 2.2), and BCube ((Fig. 2.3).

An open question is: how best is the traffic allocation of the competing elastic demand flows for horizontal traffic between edge servers in data center fabrics, and how is this allocation impacted with the increase in capacity? To address this question, we assume that all traffic uses TCP allowing multipath forwarding. More specifically, we are interested in understanding this impact in equilibrium, on other words, we want to study the impact of bandwidth sharing at the stationary state of the congestion control mechanism. It has been shown that several variants of TCP are proportionally fair in equilibrium, which have been verified through simulation [91, 92]. We, therefore, use a proportional fairness model to understand the allocation for competing demands in data center fabrics. Our study is focused on the fat-tree and BCube data center topologies, which are popular ones, and structured so that one distinguish clearly two traffic management segments: intra-pod and inter-pod segments.

The rest of the chapter is organized as follows. Our optimization model is formulated in Section 4.2 and the simulation results are presented in Section 4.3. Section 4.4 concludes the chapter.

Table 4.1: Mathematical notations

Indices	
$d = 1, 2, \ldots D$	demands associated with pairs of edge switches.
$p = 1, 2, \ldots P_d$	candidate paths for demand d.
$e = 1, 2, \ldots E$	links.
Variables	
x_{dp}	amount allocated to path p of demand d.
X_d	amount allocated to d.
y_e	amount of throughput carrying by the link e.
Parameters	
$\delta_{edp} = 1$	if link e belongs to path p of demand d; 0, otherwise.
α	a minimum sub-flow ratio allocated to each path p available to a demand d.
c_e	capacity of link e.
ω_d	weight of demand d (constant).

4.2 Problem Formulation

Following [93], we now generalize the basic proportional fairness model for DCN 2.2.4 allowing multipath forwarding for elastic demands that use TCP. First, while the actual TCP sessions are between edge servers in a DCN, we can consider the model in terms of elastic demands between a pair of edge switches since all such sessions must enter and exit through edge switches (see Figure 2.2). Thus, moving away from sessions (identified by j earlier), we identify a demand between a pair of edge switches by d with the elastic demand denoted by X_d. Secondly, due to multipath forwarding, we identify traffic flow along each path p associated with demand d by using x_{dp} (notations are summarized in Table 4.1). Therefore, for a given demand, the sum of traffic amount allocated to the paths is equal to the total elastic demand X_d given by:

$$\sum_p x_{dp} = X_d \quad d = 1, 2, \ldots, D. \tag{4.1}$$

Next, the sum of all the flows using a particular link e must satisfy the link capacity constraint:

$$\sum_d \sum_p \delta_{edp} x_{dp} - y_e = 0 \quad e = 1, 2, \ldots, E \tag{4.2}$$

$$y_e \leq c_e \quad e = 1, 2, \ldots, E. \tag{4.3}$$

Table 4.2: Linear approximation

Indices	
$k = 1, 2, \ldots, K$	Consecutive pieces of the approximation of $\log x$.
Variables	
f_d	approximation of $\log X_d$.
Parameters	
a_k, b_k	coefficients of the linear pieces of the linear approximation of $\log x$.

The goal is to maximize the utility objective:

$$\max_{X,x\geq 0} \quad F(X) = \sum_d \omega_d \log X_d. \tag{4.4}$$

where ω_d is weight for demand d, which is discussed further in Section 4.2.2.

It is worth noting that while elastic demand X_d is non-negative, the logarithm function ensures that no elastic demand takes the value zero, i.e. every demand must get its fair share while satisfying the network constraints.

In addition to the above model, we are also interested in understanding the behavior when for each demand d we enforce the usage of multiple paths, which can be imposed using the following additional constraint:

$$x_{dp} \geq \alpha X_d \quad d = 1, 2, \ldots, D \quad p = 1, 2, \ldots, P. \tag{4.5}$$

Here, each available path has to carry at least αX_d, i.e. the minimum of the rate allocated to demand d on each candidate path.

4.2.1 Linear approximation of the objective

We note that in the previous formulation, the objective function is non-linear due to the logarithm function. We use a linear approximation [93] of the logarithm function as follows:

$$\log X_d = \min_{k=1,2,\ldots,K} \{a_k X_d + b_k\}. \tag{4.6}$$

Then, the optimization objective becomes:

$$\max_{X,x,f\geq 0} \quad F = \sum_d \omega_d f_d. \tag{4.7}$$

subject to (4.1)-(4.3).(4.5) also applies for the case where multipath is enforced.

$$f_d \leq a_k X_d + b_k \quad d = 1, 2, \ldots, D \quad k = 1, 2, \ldots, K. \tag{4.8}$$

The advantage of this approximation is that it leads to a linear programming problem that can be solved using well-known linear programming solvers such as CPLEX.

4.2.2 On weights w_d

We now elaborate on the role of w_d. It allows taking into consideration two different interpretations of TCP Vegas [92,111] at the state-of-the-art: the one based on bytes per *round trip time*, and the other based on bytes per *second*, hence using to different utility functions.

$$U(X_d) = \log X_d \tag{4.9}$$

$$U(X_d) = \bar{\omega}_d \log X_d. \tag{4.10}$$

Here $\bar{\omega}_d$ corresponds to the propagation delay of session d. The first situation (4.9) does not weight the session: we refer to it as the *fixed-delay case*. The second situation (4.10) weights the propagation delay by means of $\bar{\omega}_d$ for session d: we refer to it as the *weighted-delay case*. Besides the two valid implementations of TCP Vegas, FAST TCP follows the weighted-delay case [91]. For comparison purposes, we use a simplification for the weighted-delay case by setting $\bar{\omega}_d$ to be based on the number of hops between the source and the destination, to serve as a rough approximation of the delay being the number of hops [112].

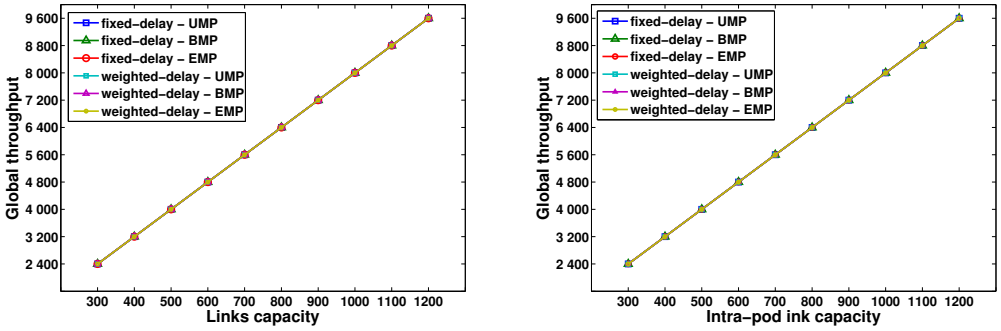
4.3 Performance evaluation

We evaluate the proposed fairness model for DCNs using the fat-tree [26] and the BCube [27] DCN topologies (at one level, $k = 1$, and with different sizes, $n = \{4, 6, 8\}$), and under different settings as explained hereafter.

4.3.1 Study cases

In order to assess traffic fairness with different levels of the horizontal DCN capacities, we consider the two following DCN dimensioning cases:

- Uniform capacity: all link capacities are set equal. In this case, we consider different capacity configurations, increasing the capacity on all the links from 300 to 1200 units in increments of 100 units.



(a) Uniform capacity case

(b) Asymmetric capacity case

Figure 4.1: Global throughput, fat-tree (n=4)

- Asymmetric capacity: the starting configuration has an equal capacity of 300 units per link. We then increase the capacity only on the intra-pod links (link between edge and aggregation switches) from 300 to 1200 units by increments of 100 units. The capacity of links between the aggregation and core switches ("extra-pod links") remains fixed at 10 units.

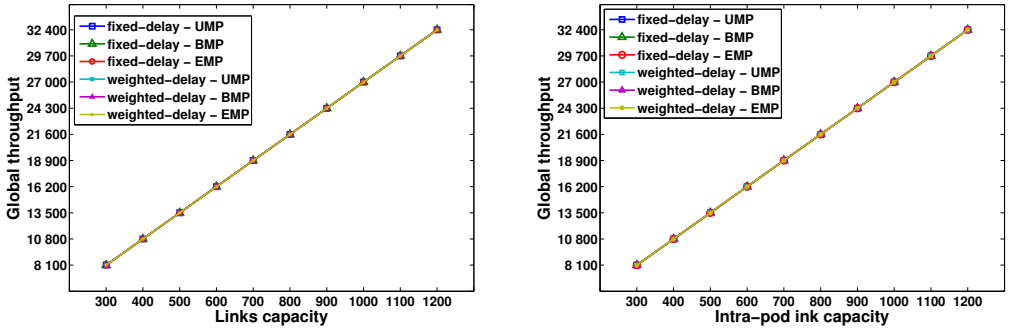
We run the simulations for different values of α, i.e. the minimum sub-flow traffic ratio allocated to each path available to a demand d. We consider the following cases:

- Unbounded MultiPath (UMP) case, with $\alpha = 0$, so that multipath forwarding is not forced for any demand, but can be used.

- Equi-distribution MultiPath (EMP) case, with α being replaced by $\alpha_d = 1/N_d$ in (4.5), where N_d is the number of paths available to demand d, so that traffic distribution is forced to be even over the paths available to each demand.

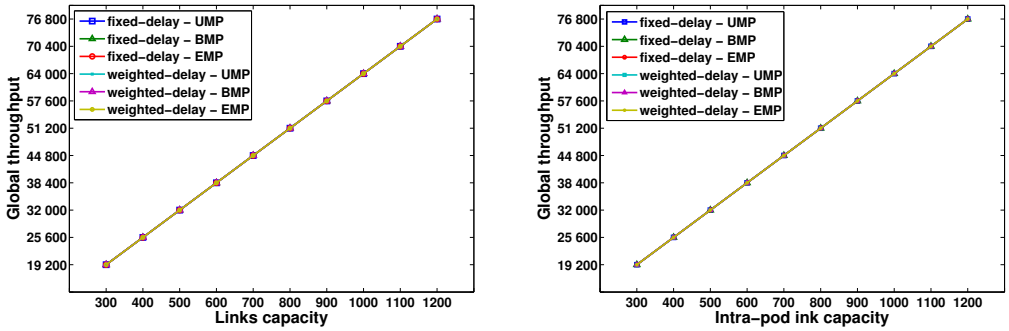
- Bounded MultiPath (BMP) case, with $\alpha = 1/(\max_d(N_d)/2)$, for all the demands, where $\max_d(N_d)$ represents the highest number of paths available for all the demands. Hence, multipath forwarding is lightly forced on all available paths for all demands, and can be freely used.

It is worth noting that for the fat-tree topology, intra-pod traffic can have two paths, while inter-pod traffic can use four paths for $n = 4$, three and nine for $n = 6$ and four and eight for $n = 8$. Regarding the BCube architecture, only the parallel paths are valid; in $BCube_l$, between any access node pairs there are $l+1$ paths, one path in each level. Hence, every access server has two paths.



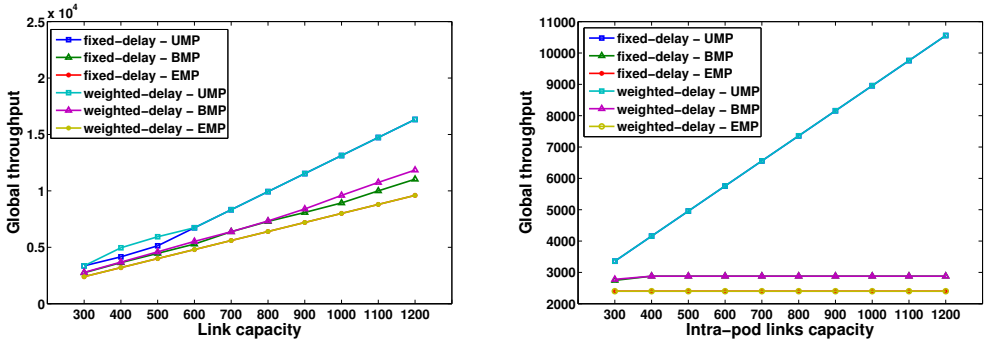
(a) Uniform capacity case (b) Asymmetric capacity case

Figure 4.2: Global throughput, fat-tree (n=6).



(a) Uniform capacity case (b) Asymmetric capacity case

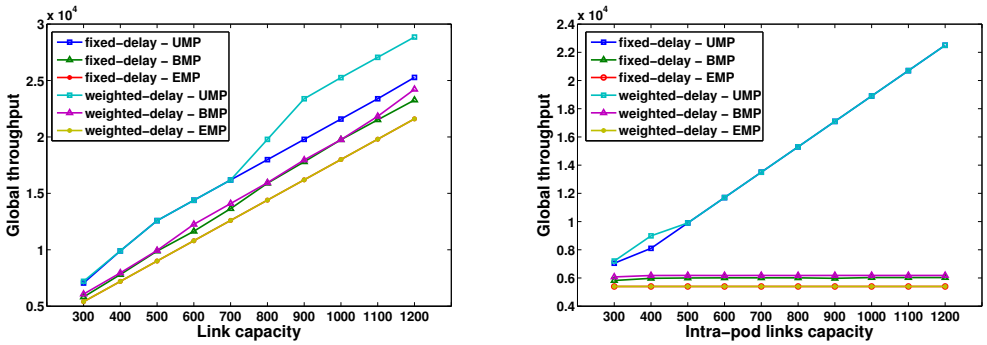
Figure 4.3: Global throughput, fat-tree (n=8).



(a) Uniform capacity case (b) Asymmetric capacity case

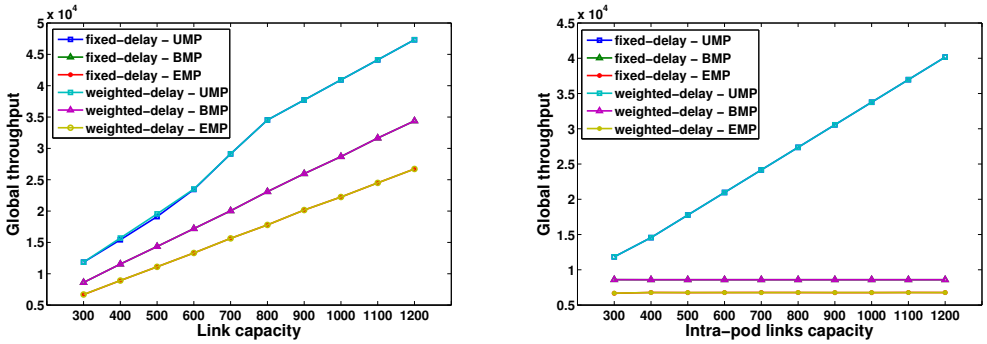
Figure 4.4: Global throughput, BCube (n=4).

Moreover, we evaluate the results for both the utility functions presented in Section 4.2: the fixed-delay situation ($\bar{\omega} = 1$) given by (4.9) and the weighted-delay situation ($\bar{\omega}_d =$ hop count) given by (4.10). These two options allow us to see how



(a) Uniform capacity case (b) Asymmetric capacity case

Figure 4.5: Global throughput, BCube (n=6).



(a) Uniform capacity case (b) Asymmetric capacity case

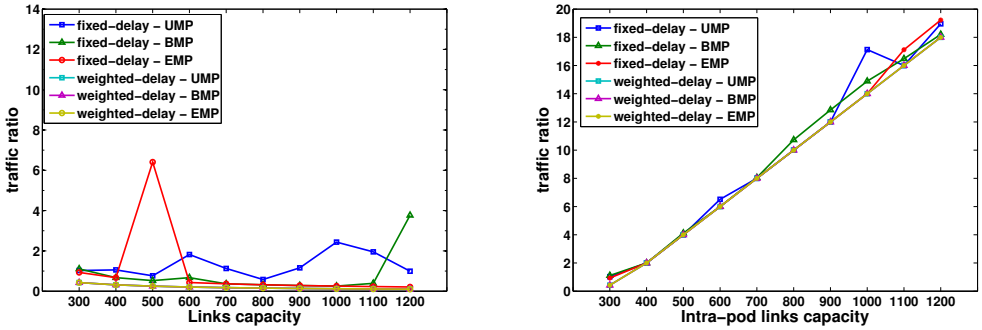
Figure 4.6: Global throughput, BCube (n=8).

fairness is guaranteed for intra-pod and inter-pod traffic. More importantly, a data center provider can decide to deploy its preferred TCP implementation (as far as it owns the computing infrastructure) by taking advantage of the lessons learned from this study, and accordingly allocate jobs to servers to target traffic fairness. In other words, this study could also help the Cloud provider to decide on fine-grained scheduling of jobs that meets traffic fairness requirements.

In order to show the impact on throughput allocation between intra-pod and inter-pod edge switches, we measure the intra-to-inter-pod traffic allocation ratio. Finally, we measure the path diversity of the solution for the UMP case for all the edge-to-edge demands, where each edge has a demand to all the edges.

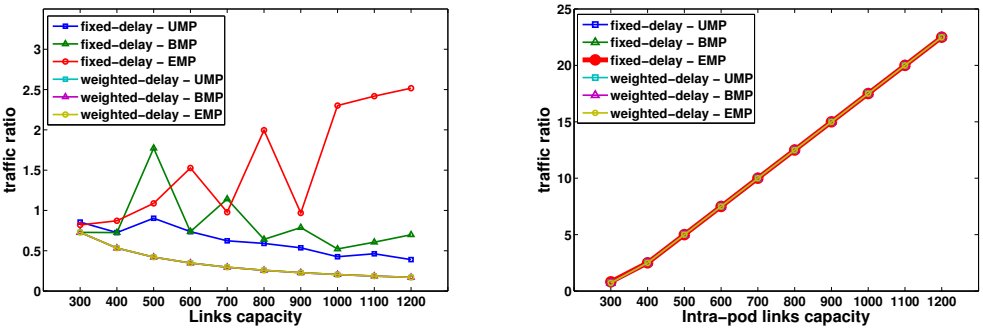
4.3.2 Results

This subsection illustrates the results of the proportional fairness model concentrating the analysis around three key aspects: the throughput allocation, the traffic distribution within and across pods, and the achieved path diversity.



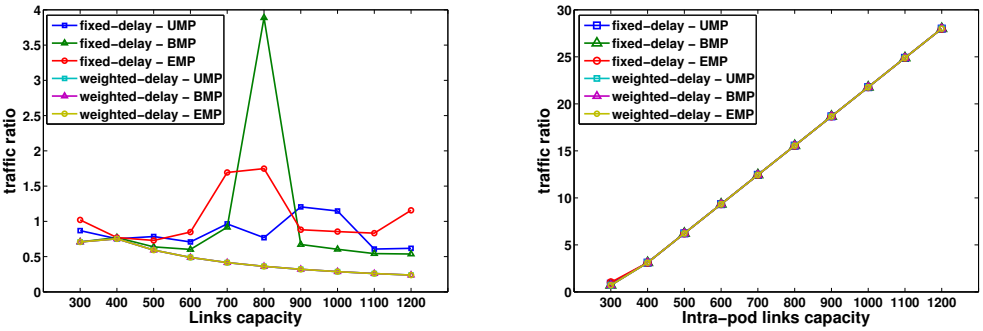
(a) Uniform capacity case (b) Asymmetric capacity case

Figure 4.7: Intra-to-inter pod traffic ratio, fat-tree (n=4).



(a) Uniform capacity case (b) Asymmetric capacity case

Figure 4.8: Intra-to-inter pod traffic ratio, fat-tree (n=6).

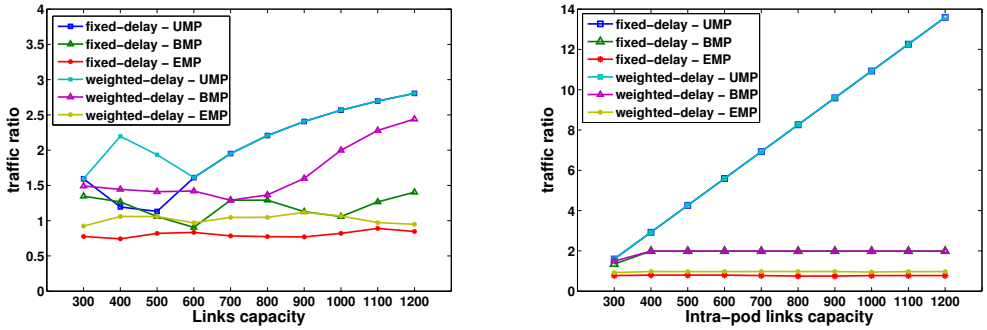


(a) Uniform capacity case (b) Asymmetric capacity case

Figure 4.9: Intra-to-inter pod traffic ratio, fat-tree (n=8).

Throughput allocation

First, we consider the fat-tree topology, regarding to the global throughput when traffic between all pairs of edge switches are allowed. In Figures 4.1, 4.2 and 4.3,



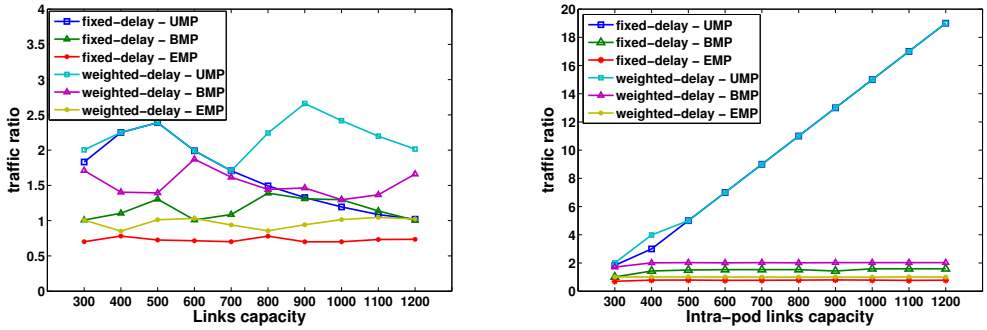
(a) Uniform capacity case

(b) Asymmetric capacity case

Figure 4.10: Intra-to-inter pod traffic ratio, BCube(n=4).

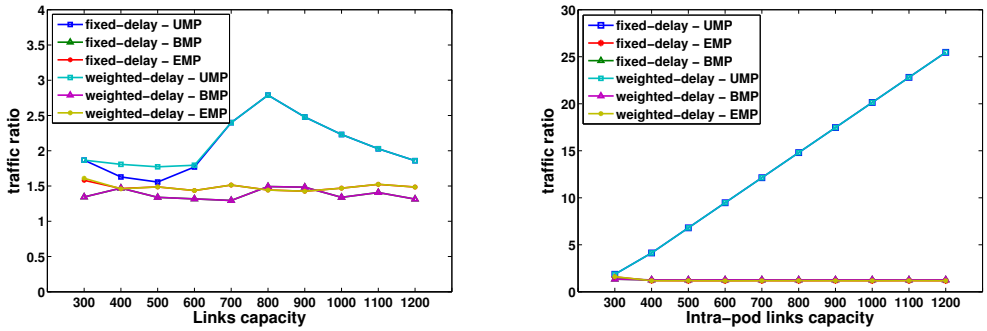
we plot the global throughput as a function of the capacity. We find that it grows linearly as dictated by the capacity of intra-pod links, irrespectively of the capacity of the links connecting aggregation and core switches. More importantly, it is not affected by the multipath case (UMP, BMP, EMP) nor by the type of the utility function (fixed-delay vs. weighted delay). In order to explain this behavior, first let us focus on the asymmetric capacity case. Let us assume that all extra-pod links are dropped, i.e. there are only intra-pod links with a capacity of 300 each, each pod is isolated and there is no inter-pod traffic. Thus, the traffic throughput between the two edge switches in a pod is limited by the capacity of the intra-pod links. Since two links form a path in the pod, for the 4-pod fat-tree topology, we can observe that the throughput within a pod between its two edge switches is 600 units. Thus, we can see that every access link is fully saturated (active). Since there are 4 pods the total throughput is 2400 units. In this case, the traffic allocation between intra-pod and inter-pod is skewed. It is interesting to note that when extra-pod links have a positive capacity, the total throughput still remains at 2400 units as long as the capacity of the intra-pod links are at 300 units each. Similarly, for 6-pod fat-tree topology since two links form a path, further in a pod we have nine links, we can see that the throughput within a pod between its two edge switches is 1350, and the total throughput is equal to 8100. In the same manner, we find that the total throughput is 19200 for 8-pod topology. These results show that the global throughput depends only on the access links where all are fully saturated.

Secondly, regarding the BCube topology, from Figures 4.4, 4.5 and 4.6, we note that the throughput changes with the different dimensioning and multipath cases. Knowing that in $BCube_1$ topology there are two paths between any servers, we can deduce the amount of throughput for a flow in each path at optimality, as an example, when using EMP the flow allocations are equal ($x^*_{d1} = x^*_{d2}$). Hence, we can explain why the two dimensioning cases are different as opposed to the fat-tree



(a) Uniform capacity case (b) Asymmetric capacity case

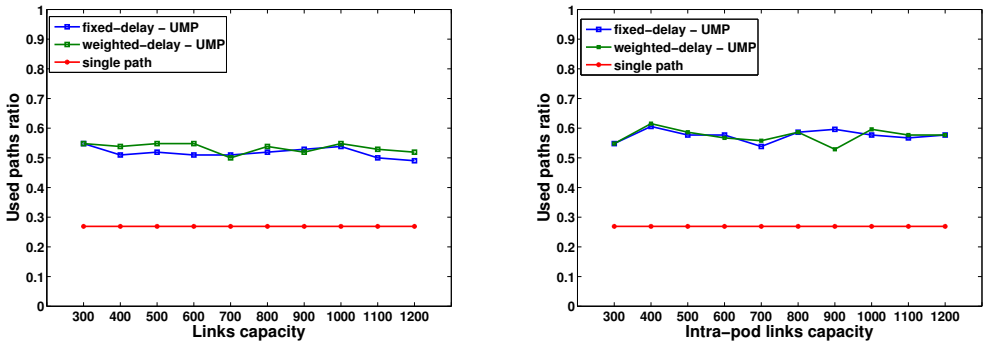
Figure 4.11: Intra-to-inter pod traffic ratio, BCube(n=6).



(a) Uniform capacity case (b) Asymmetric capacity case

Figure 4.12: Intra-to-inter pod traffic ratio, BCube(n=8).

topology, and why the two curves (BMP and EMP) of the asymmetric case are flat. On the one hand, in the BCube topology all links are access links, and thus for the asymmetric case only the intra-pod link loads (only for the links at the $BCube_0$ level) are increased while all the link loads are increased for the uniform capacity case. We get higher global throughput with the uniform capacity case. On the other hand, when multipath is forced each flow has to use two paths: one path uses only the links at the $BCube_0$ level, and the other one use $Bcube_1$ links. Therefore the flow throughput strictly depends on the link with the smallest capacity. That is the reason why for the asymmetric case, when forcing multipath, the throughput strictly depends on the restricted link with the capacity of 300 units, and all the cases are equal. In the EMP case, it gets the worst throughput since it has to share equally the traffic between all the available paths (two in this case: one at the pod level at $BCube_0$ with unrestricted capacity and the other at the extra-pod level at level-1 with restricted capacity). In the BMP case, it is between both paths, since it has to share the throughput between the available paths but also can give slightly



(a) Uniform capacity case

(b) Asymmetric capacity case

Figure 4.13: Used paths ratio, fat-tree (n=4).

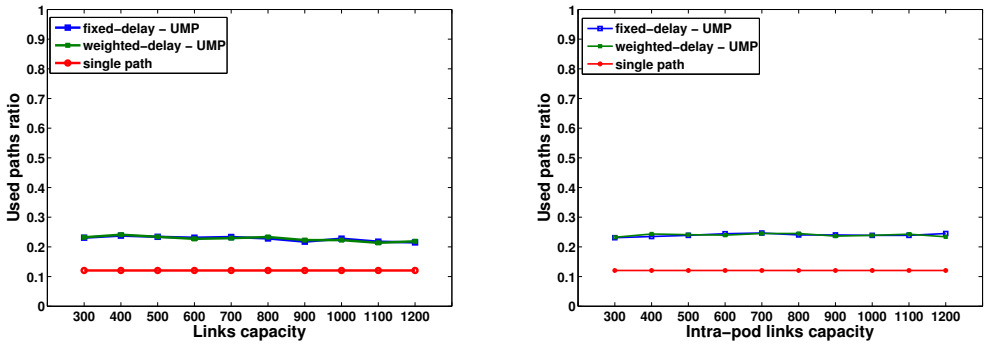
more throughput to one of the two paths (the path belonging to $BCube_0$ level).

Moreover, we can explain why the UMP case (multipath is not forced) gets higher throughput than the other multipath forms, for the uniform and the asymmetric cases. In fact, this is due to the fact that the UMP case can potentially use one or two paths for a flow with any ratio; when using the two paths, it can lead to higher throughput along one path. As opposite to the other multipath mode where they have to use two paths (hence more links are shared by many flows), the global throughput goes down. In order to confirm this guess we observe the link utilization for the uniform case (since for the asymmetric case it can be explained by the dependence on the restricted link as explained before) and we find that for the UMP case all links are fully saturated (active). Nevertheless when forcing multipath forwarding we note that not all the links are active due to the equation (4.5) that forces multipath forwarding, hence forcing multipath forwarding implies the use of the two paths, which leads to share more links and to loss in the global throughput. This phenomenon is further amplified with larger topology size.

Traffic distribution

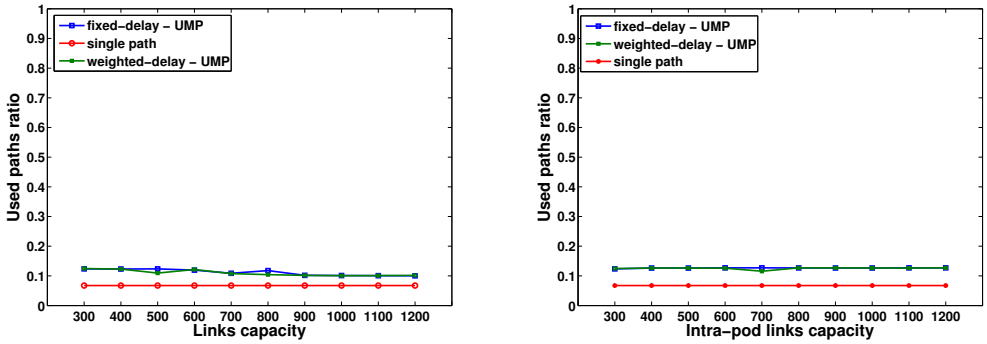
We investigate how traffic distribution is affected between intra-pod and inter-pod edge switches, for which we use the intra-to-inter pod traffic ratio as metric. We characterize the traffic distribution sensibility with respect to the various cases focusing on the intra-to-inter pod traffic ratio.

For the fat-tree topology, in the uniform capacity case with regards to the intra-to-inter pod traffic ratio (Figures 4.7-a, 4.8-a, and 4.9-a), we find that on average the inter-pod traffic has a traffic proportion that is almost twice that of the intra-pod traffic in the weighted-delay situation. This is explained by the fact that the path hop count of an inter-pod demand (4 hops) is twice that of an intra-pod demand



(a) Uniform capacity case (b) Asymmetric capacity case

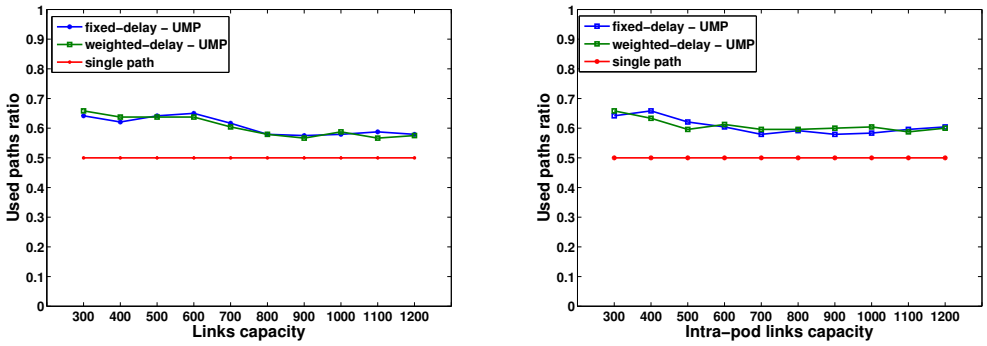
Figure 4.14: Used paths ratio, fat-tree (n=6).



(a) Uniform capacity case (b) Asymmetric capacity case

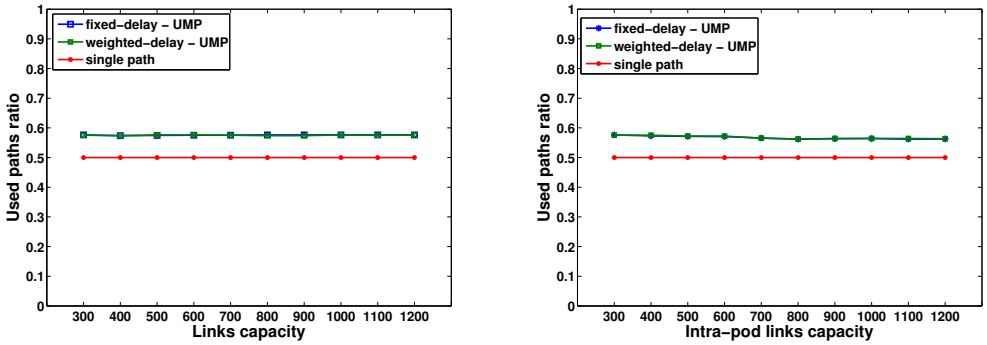
Figure 4.15: Used paths ratio, fat-tree (n=8).

(2 hops) – this is reflected in the weights for the weighted-delay situation. The fixed-delay situation is almost similar: we note the presence of some oscillations that can be explained by the fact that there is no preference set to the inter or intra demands. For the asymmetric capacity case, the observation is strikingly different than for the uniform case. The intra-pod demands have many times more throughput than inter-pod demands (Figures 4.7-b, 4.8-b, and 4.9-b). According to our previous analysis regarding to the global throughput for this case where all the access links are equal to the capacity. Hence, even the capacity of the intra-pod links can reach 1200 units (the extra-pod links capacity was kept fixed at 300 units). Indeed, the global throughput depends on the access links, since in the fat-tree topology all the access links belong to the pod (intra-pod demands), whereas the inter-pod demands see the extra-pod links kept fixed at 300 units. Hence, all the intra-pod demands get the maximum throughput, which grows as the intra-pod links capacity increases, regardless to the inter-pod demands that cannot get more



(a) Uniform capacity case (b) Asymmetric capacity case

Figure 4.16: Used paths ratio, BCube (n=4).



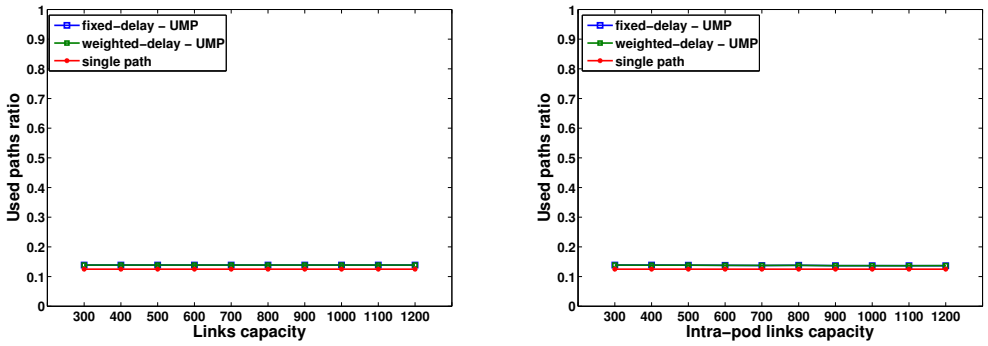
(a) Uniform capacity case (b) Asymmetric capacity case

Figure 4.17: Used paths ratio, BCube (n=6).

than 300 units of throughput.

For the BCube topology, regarding the uniform capacity situation (Figures 4.10-a, 4.11-a, and 4.12-a), we note that the intra-to-inter pod traffic ratio is affected by both the multipath cases and the utility functions. First, for both TCP situations we find that on the average the UMP and BMP cases give slightly higher throughput to the intra-pod demands, while EMP is around 1 and gives the same throughput to the intra-pod or inter-pod demands, which is coherent with our analysis of the global throughput results. Secondly, comparing the two TCP situations, in some cases the intra-pod demands get slightly higher throughput than the fixed-delay situation. This is explained by the weights associated with the TCP. In fact, with the BCube topology the two paths between any access nodes, not belonging to the same pods (inter-pod demands), have the same hop count (equal to 4), while the hop count for any two paths between any intra-pod demands is equal to 2 or 6.

Regarding the asymmetric capacity case, only for the UMP case the intra-pod de-



(a) Uniform capacity case

(b) Asymmetric capacity case

Figure 4.18: Used paths ratio, BCube (n=8).

mands have a largely higher throughput than the inter-pod demands (Figures 4.10-b, 4.11-b, and 4.12-b), due to the fact that UMP can possibly use or not the two paths for a flow with any ratio for each flow, and in the case of using the two paths for a flow it can give more throughput to the Bcube_0 links that have more capacity.

The growth of the UMP curve can be explained as done for the fat-tree topology. In fact, all the intra-pod demands get the maximum throughput, which grows as the intra-pod link capacity increases (it can reach 1200 units), while the inter-pod demands have always the same throughput since the extra-pod links capacity was kept fixed at 300 units. Regarding to the BMP and EMP cases, the two curves are flat. The same throughput is offered to the the intra or inter-demands for the EMP case and more throughput is offered to the intra-pod demands. This confirms our global throughput analysis.

Path Diversity

Figures 4.13, 4.14, 4.15, 4.16, 4.17 and 4.18 illustrate the path diversity for the UMP case, measured as the ratio of the overall used paths to the number of overall available paths. We also plot the line corresponding to the single-path situation. It is worth remembering that for the BMP and EMP cases, all the paths are used (so it would be a top line at a ratio equal to 1).

Any path diversity of the solution does not seem to be affected by the specific utility function (TCP behavior). We can see that although path diversity was allowed, the unconstrained multipath forwarding case did not take full advantage of it. On the contrary, with the BCube topology the UMP case lead to higher global throughput. This implies that path diversity is not always needed to maintain the highest throughput, and can even lead to a decrease to the global throughput.

4.4 Summary

In this chapter, we investigated the fairness issue in the offered throughput to DCN end-points (edge switches and servers) at the equilibrium, using TCP utility functions for modeling elastic demands.

We presented a generalized formulation of the basic proportional fairness model for DCNs allowing multipath forwarding for elastic demands. We also described, analyzed and evaluated our model under two different TCP utility functions: the fixed-delay and the weighted-delay ones. Through a series of scenarios studied on the fat-tree and BCube topologies of various sizes, we discovered a number of interesting results.

Our study, to the best of our knowledge, is the first one to address the impact of DCN topology design, capacity planning and multipath forwarding on traffic fairness in DCN fabrics. We believe our results are interesting and deserve further study, especially grounded on real DCN traffic data.

Chapter 5

Virtual Network Function Placement and Routing

Network Functions Virtualization (NFV) is incrementally deployed by Internet Service Providers (ISPs) in their carrier networks, by means of Virtual Network Function (VNF) chains, to address customers' demands. The motivation is the increasing manageability, reliability and performance of NFV systems, the gains in energy and space granted by virtualization, at a cost that becomes competitive with respect to legacy physical network function nodes. From a network optimization perspective, the routing of VNF chains across a carrier network implies key novelties making the VNF chain routing problem unique with respect to the state of the art: the bitrate of each demand flow can change along a VNF chain, the VNF processing latency and computing load can be a function of the demands traffic, VNFs can be shared among demands, etc. In this chapter, we provide an NFV network model suitable for ISP operations. We define the generic VNF chain routing optimization problem and devise a mixed integer linear programming formulation. By extensive simulation on realistic ISP topologies, we draw conclusions on the trade-offs achievable between legacy Traffic Engineering (TE) ISP goals and novel combined TE-NFV goals.

5.1 Introduction

After about ten years of fundamental research on network virtualization and virtual network embedding, the virtualization of network functions is becoming a reality thanks to huge investments being undergone by telecommunication providers, cloud providers and vendors.

The breaking point sits in 2012, when calls for experimentation and deployment of what was coined as "Network Functions Virtualization (NFV)" [113] lead to the creation of a NFV industry research group at the European Telecommunications Standards Institute (ETSI) [114]. Since then, applied researches and developments have accelerated the investments and the first prototypes are being demonstrated and deployed (leading to commercialization in some cases) since late 2014 [115].

With NFV, network functions historically directly implemented using specialized hardware nodes are run instead as Virtual Machines (VMs). As above mentioned, running routers, firewall, etc, as VMs was a technological step already conceived and experimented well before 2012. However, starting by 2012 the attention of NFV researchers focused on key aspects of NFV systems that were either not considered relevant or not conceived at all before the NFV standardization effort at ETSI (as well as at other Standards Developing Organizations, SDOs, such as the Internet Engineering/Research Task Force, IETF/IRTF). Key aspects that are worth being mentioned and that we address in our work are:

- the so-called NFV chaining, i.e., the problem of allowing a traffic flow passing through a list of NFV nodes;
- the flow orchestration over VNF nodes and chains, i.e., the fact that NFV nodes can be placed at and migrated across virtualization clusters as a function of traffic flow assignment to existing VNF chains or sub-chains;
- the consideration ingress/egress bit-rate variations at VNFs (such as compression as in coding, decompression as in tunneling) due to specific VNF operations;
- the consideration of the VNF switching latency as an important orchestration parameter. It can indeed be exponential with the traffic load on the VNF, or constant up to a maximum bound if computation offloading solutions are adopted, such as direct memory access bypassing the hypervisor as done with Intel/6Wind Data-Plane Development Kit (DPDK) [116], and similar other 'fastpath' solutions are present.

These key aspects make the NFV orchestration problem substantially different from the network embedding problem, as discussed hereafter.

ETSI is de-facto the reference SDO for the NFV high-level functional architecture specification. High-level means that its identified role is the specification of the main functional blocks, their architecture and inter-relationship, whose implementation elements are then precisely addressed by other SDOs such as the IRTF/IETF. So far, ETSI specified three key functional blocks of the NFV architecture: Virtual Network Functions (VNFs), the nodes; the NFV Infrastructure (NFVI), including the logical elements needed to run VNFs such as the hypervisor node architecture; the MANagement and Orchestration (MANO) layer, handling the operations needed to run, migrate, optimize VNF nodes and VNF chains, possibly in relationship with transport network orchestrators.

MANO procedures come therefore to support the economies of scale of NFV, so that physical NFVI virtualization resources (servers and clusters) dedicated to NFV operations are used efficiently with respect to both NFVI operators and edge users. A typical NFV use-case for carrier networks is the virtual Customer Premises Equipment (CPE) that simplifies the CPE equipment by means of virtualized individual network functions placed at access and aggregation network locations, as depicted in Fig. 5.1. Some MANO operations that can be mentioned are where to place VNFs to better meet user's demands, how to route VNF chains over a transport network disposing of multiple NFVI locations, how to share VNFs among active demands, while meeting common Traffic Engineering (TE) objectives in IP transport networks as well as novel NFV efficiency goals such as the minimization of VNF to install. These orchestration operations must take into consideration the special nature of NFV architectures, such as the latency/traffic bounds at both the VNF node and the end-to-end level, the fact that some VNFs can modify the incoming bitrate by compressing or decompressing it, etc. In this context, our contribution is as follows:

- we define and formulate via mathematical programming the VNF placement and routing (VNF-PR) optimization problem, including compression/decompression constraints and two switching latency regimes (with and without fastpath), with both TE and NFV objectives.

- we design a math-heuristic approach allowing us to run simulations also for large instances of the problem within an acceptable execution time.

- we evaluate our solution on realistic three-tier topologies and compare it to alternative algorithms at the state of the art. We draw valuable considerations on NFV deployment strategies.

The chapter is organized as follows. Section 5.2 presents the state of the art on NFV orchestration. Section 5.3 describes the network model and the mixed integer

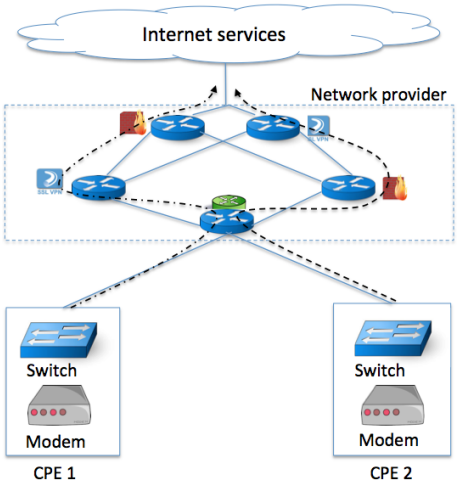


Figure 5.1: VNF chaining with virtualized Customer Premises Equipment (vCPE).

linear programming (MILP) formulation. Discussion of simulation results are given in Section 5.4. Section 5.5 concludes the chapter.

5.2 Background

In the state of the art, preliminary works on NFV orchestration tend to map the problem into a Virtual Network Embedding (VNE) problem. This is for example the case of [117], where VNFs are treated as normal virtual machines to be mapped on a network of VM containers interconnected via physical links which can host logical links of the virtual network. Similarly, authors in [118] propose a VNF chaining placement that combines the location-routing and VNE problems. First, they solve the placement problem and then the chaining problem. In [119] the authors decouple the legacy VNE problem into two embedding problems: service chain embedding and VM embedding, where a service chain is embedded on a VM, and each VM on physical servers. Each service chain has specific requirements as notably an end-to-end latency requirement.

The placement and routing of VNF chains is a problem fundamentally different from the VNE problem. Similarly than in VNE, virtual network nodes need to be placed in an underlying physical infrastructure. However, differently from VNE, in VNF chaining: (i) the demand is not a multipoint-to-multipoint network connection request, but as a point-to-point source-destination flow routing demand and (ii) specific aspects of NFV such as forwarding latency behavior, ingress/egress bit-rate changes, and chaining are not addressed in VNE, and their inclusion would further increase the VNE problem time complexity. In this sense VNF chaining problem is more similar to facility location problems, whereas NVE is a mapping problem.

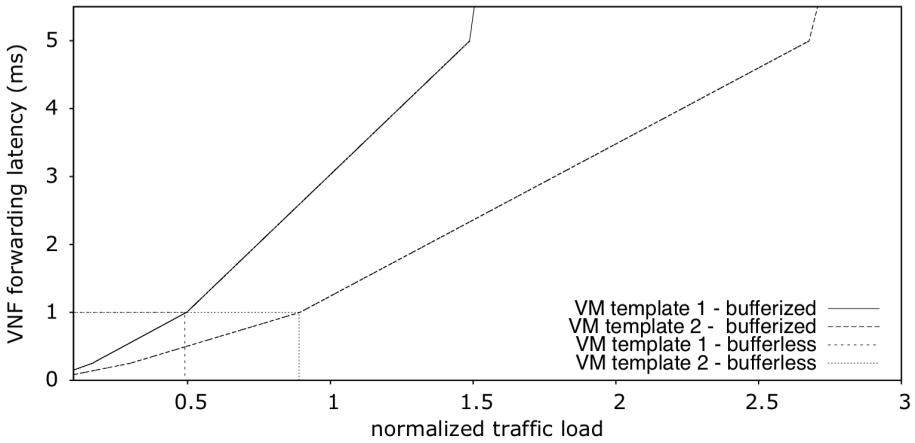


Figure 5.2: Example of VNF forwarding latency profiles.

We believe the appropriate way to deal with NFV MANO decision problems is to define the VNF Placement and Routing (VNF-PR) problem directly tailored to the NFV environment, for the sake of time complexity, modeling precision and practical usability. This is the approach adopted by a few papers in the literature [120–122]. In [120] the authors consider the online orchestration of VNFs, modeling it as a scheduling problem of VNFs and proposing heuristics to scale with the online nature of the framework. In [121] the authors consider a VNF-PR problem for data-center environments with both optical and electronic elements. They formulate the problem as a binary integer programming problem, and propose an heuristic algorithm to solve it. In their work, VNF chains are set as an input to the problem. In [122] the specific Deep Packet Inspection (DPI) VNF node placement problem (with no chaining) is targeted, with a formal definition of the problem and a greedy heuristic algorithm to solve it. Our contribution takes inspiration from these latest works, yet goes beyond being more generic and integrating the specific features of NFV environments mentioned in the introduction.

5.3 Network Model

We provide in the following a definition of the VNF Placement and Routing (VNF-PR) problem, described with our network modeling assumptions, and we provide a mathematical programming formulation, along with a description of its possible customization alternatives.

5.3.1 Problem Statement

Definition (VNF-PR: Virtual Network Function Placement and Routing)
Given a network graph $G(N, A)$, where N is the set of nodes, A the set of arcs

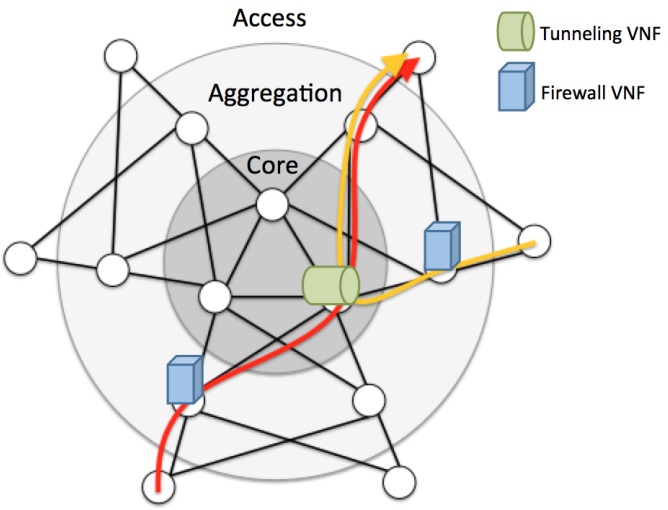


Figure 5.3: Adopted network topology and VNF-PR solution example.

between nodes, $N_v \subset N$ the set of nodes disposing of NFVI server clusters. Given a set of edge demands D, each demand $k \in D$ being characterized by a source o_k, a destination t_k, a bandwidth b_k, and a set of VNFs of different types to be traversed by edge flows, the VNF-PR optimization problem is to find:

- the optimal placement of VNF nodes over NFVI servers;
- the optimal assignment of demands to VNF node chains.

subject to:

- NFVI cluster capacity constraints;
- VNFs' flow compression/decompression constraints;
- VNF forwarding latency constraints.
- VNF node sharing constraints.

The optimization objective should contain both network-level and NFV-level performance metrics. In our NFV network model, we propose as network-level metric the classical TE metric, i.e., the minimization of the maximum link utilization. As NFV-level metric we propose the minimization of computing resources volume. Furthermore, in our network model we assume that:

- Only a subset $N_a \subset N$ of the nodes are source and/or destination of demands.
- Multiple VNFs of the same type (i.e., providing same functionality) can be allocated on the same node, but each demand cannot split its flow on multiple VNF of the same type.

- No sequence is imposed on the VNFs used by each demand.
- There are different Virtual Machine templates, each with a different computing resource consumption and VNF forwarding latency performance.
- The VNF computing resource consumption demand is expressed in terms of live memory (e.g., RAM) and Computing Processing Units (CPUs).
- Latency introduced by a VNF can follow one among the two following regimes (as represented in Fig. 5.2):
  - *Bufferized*: VNFs possess buffers such that the forwarding latency is considered as a convex piece-wise linear function of the aggregate bit-rate at the VNF. This is the case of default VNF functioning, with kernel socket buffers possibly resized.
  - *Bufferless*: VNFs behave without buffers, or with a very small buffer (1-2 packets), so that the forwarding latency is constant up to a maximum aggregate bit-rate (after which packets are dropped). This is the case for Intel/6WIND DPDK fastpath solutions [116].

  Examples of forwarding latency profiles for the two cases are given in Fig. 5.2 (with different VM templates we used for the tests).
- for each demand and each NFVI cluster, only one compression/decompression VNF (referred to as 'comp/dec' in the following) can be installed. This allows us to keep the execution time at acceptable levels, without reducing excessively the VNF placement alternatives.
- for each network node we have a NFVI cluster node attached, with the following simplifications in the notations with a single identifier for collocated network and NFVI nodes.

5.3.2 Mathematical Formulation

Table 5.1 reports the mathematical notations used in the following Mixed Integer Linear Programming (MILP) formation of the VNF-PR problem. We work on an extended graph, in which each access node i is duplicated in a node i'. Arc (i, i') will be added and all arcs (i, j) originating from access node i will be transformed in arcs (i', j). Traffic (demand) originates from original access nodes. This will allow to distinguish between nodes origin/destination of flow and nodes where NVF are allocated.

Table 5.1: Mathematical Notations

	sets
N_a	access nodes
N'_a	duplication of access nodes
N_c	aggregation/core nodes
N_v	nodes where a VNF can be located, $N_v = N'_a \cup N_c$
N	set of all nodes, $N = N_a \cup N_v$
A	set of all arcs, $A \subseteq N \times N$
R	resource types (CPU, RAM, storage)
D	demands
F	VNF types
C_f	set of possible copies of VNF of type f
T	set of VNF configurations
	demand parameters
o_k	origin of demand $k \in D$
t_k	destination of demand $k \in D$
b_k	qt (bandwidth) of demand $k \in D$
m_{kl}	1 if demand $k \in D$ requests v.f. $l \in F$
	network parameters
γ_{ij}	arc capacity
λ_{ij}	link latency
Γ_{ir}	capacity of node $i \in N$ in terms of resource of type $r \in R$
	NFV parameters
rr_{rt}	demand of resource $r \in R$ for a VM of type t
μ_f	compression/decompression factor for VNF $f \in F$
$g^t_{fj}(b)$	j-th latency function of $f \in F$ and aggregate bandwidth b if allocated on VM of type t, linear in requested bandwidth
L	maximum allowed latency for a demand
	binary variables
x^k_{ij}	1 if arc (i, j) is used by demand $k \in D$
z^{kn}_{ift}	1 if demand $k \in D$ uses copy n-th of VNF of type $f \in F$ placed on node $i \in N_c$ on a VM of type t
y^n_{ift}	1 if n-th copy of type of VM t is assigned to VNF of type $f \in F$ hosted by node $i \in N_c$
	continuous variables
ϕ^k_{ij}	flow for demand $k \in D$ on arc (i, j)
ψ^{kn}_{if}	flow for demand $k \in D$ entering in node i and using copy n of VNF of type $f \in F$
l^k_{if}	latency that demand $k \in D$ 'suffers' using VNF of type $f \in F$ hosted by node $i \in N_c$

Binary variables x_{ij}^k represent the per-demand link utilization, and therefore also the path used by the demand. Binary variables y_{ift}^n represent the allocation of a copy of VF on a given node. Binary variables z_{ift}^{kn} represent the assignment of a given demand to a certain copy of a VF. To introduce the possibility of compressing/decompressing flows for some VNFs, we need introduce explicit flow variables ϕ_{ij}^k, and a compression parameter μ_f for each type of VNF. Furthermore, variables ψ_{if}^{kn} represent the flow of demand k entering node i and using copy n of VNF of type f. In the following we present the constraints.
Single path routing flow balance:

$$\sum_{j:(i,j)\in A} x_{ij}^k - \sum_{j:(j,i)\in A} x_{ji}^k = \begin{cases} 1 \text{ if } i = o_k \\ -1 \text{ if } i = t_k \\ 0 \text{ otherwise} \end{cases} \quad \forall k \in D, \forall i \in N \tag{5.1}$$

Flow and compression/decompression balance for NFVI nodes and for each demand:

$$\sum_{j\in N:(i,j)\in A} \phi_{ij}^k - \sum_{j\in N:(j,i)\in A} \phi_{ji}^k = \sum_{f\in F, n\in C_f} (1-\mu_f)\psi_{if}^{kn}$$
$$\forall k \in D, \forall i \in N_v \tag{5.2}$$

Flow balance for access nodes:

$$\sum_{j\in N:(i,j)\in A} \phi_{ij}^k - \sum_{j\in N:(j,i)\in A} \phi_{ji}^k$$
$$= \begin{cases} b^k \quad \text{if } i = o_k \\ -b^k \cdot \prod_{f\in F}^{\mathrm{m}_{kf}=1} \mu_f \quad \text{if } i = t_k \\ 0 \quad \text{otherwise} \end{cases} \qquad \forall k \in D, \forall i \in N_a \tag{5.3}$$

Coherence between path and flow variables:

$$\phi_{ij}^k \leq M_i x_{ij}^k \quad \forall k \in D, \forall (i,j) \in A \tag{5.4}$$

$$x_{ij}^k \leq \frac{1}{b_k \prod_{f\in F:\mu_f \geq 1} \mu_f} \phi_{ij}^k \quad \forall k \in D, \forall (i,j) \in A \tag{5.5}$$

VNF compression/decompression linearization constraints:

$$\psi_{if}^{kn} \leq \sum_{j\in N:(j,i)\in A} \phi_{ji}^k + M_i(1 - \sum_{t\in T} z_{ift}^{kn})$$
$$\forall i \in N_v, k \in D, f \in F, n \in C_f \tag{5.6}$$

$$\psi_{if}^{kn} \geq \sum_{j\in N:(j,i)\in A} \phi_{ji}^k - M_i(1 - \sum_{t\in T} z_{ift}^{kn})$$
$$\forall i \in N_v, k \in D, f \in F, n \in C_f \tag{5.7}$$

$$\psi_{if}^{kn} \leq M_i(\sum_{t\in T} z_{ift}^{kn}) \quad \forall i \in N_v, \forall k \in D, \forall f \in F, n \in C_f \tag{5.8}$$

Only one compression/decompression VNF for each node and demand:

$$\sum_{t\in T} \sum_{f\in F, n\in C_f:\mu_f\neq 1} z_{ift}^{kn} \leq 1 \; \forall k \in D, \forall i \in N_v \tag{5.9}$$

Utilization rate constraints:

$$\sum_{k\in D} b_k phi_{ij}^k \leq U\gamma_{ij} \quad \forall (i,j) \in A \tag{5.10}$$

If there is no VNF, it cannot be used by any demand:

$$z_{ift}^{kn} \leq y_{ift}^n \quad \forall k \in D, \forall i \in N_v, \forall f \in F, \forall t \in T, n \in C_f \tag{5.11}$$

If traffic does not pass by a VNF, it cannot use it:

$$z_{ift}^{kn} \leq \sum_{j:(j,i)\in A} x_{ji}^k \quad \forall k \in D, \forall i \in N_v, \forall f \in F, \forall t \in T : m_{kf} = 1 \tag{5.12}$$

Each demand uses exactly one NVF of each type it asks for

$$\sum_{i\in N_c} \sum_{n\in C_f} \sum_{t\in T} z_{ift}^{kn} = 1 \quad \forall k \in D, \forall f \in F : m_{kf} = 1 \tag{5.13}$$

On each node at most a VM assigned for each VNF copy of a certain type:

$$\sum_{t\in T} y_{ift}^n \leq 1 \quad \forall f \in F, \forall i \in N_c, \forall n \in C_f \tag{5.14}$$

Node resource capacity (VNF utilization):

$$\sum_{f\in F} \sum_{n\in C_f} \sum_{t\in T} rr_{rt} y_{ift}^n \leq \Gamma_{ir} \quad \forall i \in N_v, \forall t \in R \tag{5.15}$$

Latency function linearization:

$$l_{if}^k \geq g_{ft}^j(\sum_{d\in D} \psi_{if}^{dn}) - L_{max}(1 - z_{ift}^{kn}) \quad \forall i \in N_c, \forall f \in F, n \in C_f$$
$$\forall t \in T, \forall k \in D, \forall j \in 1..G \tag{5.16}$$

Maximum latency bound:

$$\sum_{(i,j)\in A} x_{ij}^k + \sum_{i\in N_c} \sum_{f\in F} l_{if}^k \leq L \quad \forall k \in D \tag{5.17}$$

Eq. (5.3) represent flow balance for access nodes. At destination node the quantity of flow is set equal to the demand multiplied for all factors of compression of

all the demanded VNFs. Eq. (5.2) represent the flow balance for a given node that has the possibility of hosting VNFs. We work under the assumption that given a node i, and a demand k, such demand uses at most a VNF f with a factor of compression/decompression $\mu_f \neq 1$. If a demand pass through a VNFs with a factor of decompression μ_f, then the out-flow of the node is proportional to the in-flow:

$$\sum_{j \in N:(i,j) \in A} \phi_{ij}^k = \mu_f \sum_{j \in N:(j,i) \in A} \phi_{ji}^k$$

Using variable z that represents the assignment of demand to VNFs and subtracting the out-flow we get:

$$\sum_{j \in N:(i,j) \in A} \phi_{ij}^k - \sum_{j \in N:(j,i) \in A} \phi_{ji}^k = \sum_{j \in N:(j,i) \in A} \phi_{ji}^k \sum_{n \in C_f} \sum_{t \in T} (\mu_f - 1) z_{ift}^{kn} \tag{5.18}$$

Variable ψ_{if}^{kn} represents the flow of demand k that enters node i and pass by copy n of VNF of type f (non-linear representation)

$$\psi_{if}^{kn} = \Big(\sum_{j \in N:(j,i) \in A} \phi_{ji}^k \Big) \sum_{t \in T} \sum_{n \in C_f} z_{ift}^k$$

Then constraints (5.18) can be linearized using (5.6)-(5.8), with parameter M_i equal to $\sum_{(j,i) \in A} \gamma_{ji}$, that represent the maximum quantity of flow entering node i.

As mentioned before, we consider two objective functions:

- TE goal: minimize the maximum network link utilization:

$$\min U \tag{5.19}$$

- NFV goal: minimize number of cores (CPU) used by the instantiated VNF:

$$\min \sum_{i \in N_v} \sum_{f \in F} \sum_{t \in T} \sum_{k \in D} \sum_{r = \text{`}CPU'} rr_{rt} \; y_{ift}^n \tag{5.20}$$

The former allows taking into consideration the inherent fluctuations related to Internet traffic and therefore minimizing the risk of sudden bottleneck on network links. The latter assumes the fact that today the first modular cost in virtualization servers, especially in terms of energy consumption, is the CPU.

5.3.3 Multi-objective math-heuristic resolution

We are faced to a multi-objective problem: minimizing the maximal link utilization and minimizing the total virtualization cost at the NFVI layer. These two objectives

are in competition; in fact, to obtain a low utilization, a large number of VNFs must be allocated. We decided to prioritize the objectives under two cases. Under the first case, named TE-NFV, we minimize first the maximal link utilization, and then the NFV cost, which reflects the ISP-oriented vision to improve the user quality of experience (strictly related to link congestion, especially for real-time services). Under the second case, named NFV-TE, we minimize first the NFV cost, and then the TE cost, which reflects the priorities of the NFVI provider, which makes sense especially when it is a different entity than the ISP.

In practice, in order to do our best to meet the second objective and allow the largest possible marginal gain, we perform a first step to find the best solution accordingly to the first objective, and then, keeping the best value found in the first step as a parameter, we minimize the second objective in a second step. In fact, for a given optimal value of the first step, different possible configurations are available to the second step, and a large primary cost reduction can be achieved by this second step without losing with respect to the secondary objective.

Furthermore, in order to allow a larger reduction of the total cost, an iterative approach can be used: increasing step by step the value of the first objective until the desired cost level of the second objective is found. Such an iterative procedure can have the advantage of providing to the second step of optimization a feasible solution (warm-start), which in many practical cases can reduce the computational time of the optimization.

5.3.4 Possible modeling refinements and customization

The model we provided above can be possibly refined and customized to meet specific requirements. We list in the following possible variants as well as the corresponding modeling variations.

- *VNF isolation*: if the same VNF cannot be shared between different demands (but two VNFs can share same server if there is enough space), then we need to add:

$$\sum_{k \in K} z_{if}^{kn} \leq 1 \quad \forall i \in N_c, \forall f \in F, \forall n \in C_f \tag{5.21}$$

- *Multiple comp/dec VNFs per NFVI node*: to make presentation simpler, we assumed that in each NFVI node there is at most one VNF that can compress/decompress a flow, i.e. with a factor of compression $\mu_f \neq 1$. This assumption can be relaxed using an extended graph in which each node that can host a VNF (N_v) is expanded in multiple copies, one for each type of

VNF that can be allocated in the node. Otherwise, we can represent all possible combinations of different VNFs allocated to the same node, and adding additional binary variables to represent which combination is chosen.

- *VNF chain ordering*: our formulation does not impose any fixed order of VNFs in a VNF chain. However, it can be easily found heuristically, starting from a (possibly) disordered solution, and proceeding with a basic neighborhood search heuristic swapping VNF places to create a solution with ordered chains.

- *Additional computing constraints*: can be easily included by tuning existing parameters, as far as computing resource requests can be expressed in an additive way (e.g., for storage).

- *Load balancing*: Multiple VNFs of the same type (i.e., providing the same functionality) can be allocated on the same node so that each demand can split its flow on multiple NVF of the same type. This could be done by extending the model allowing multiple paths for each demand, which however is expected to largely increase the execution time.

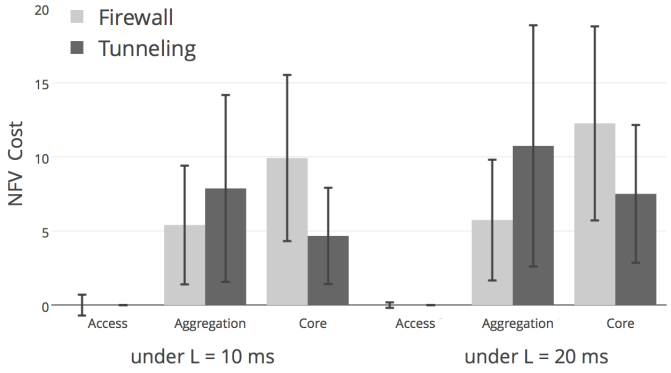
- *Core routing as a VNF*: If also the core routing function is virtualized, i.e., if the NFVI node and the network router can be considered as a single physical node that runs the core routing function, processing the aggregate traffic independently of the demand, as a VNF, then we need to add InFlow plus OutFlow to (5.15):

$$\sum_{k\in D}\sum_{j:(i,j)\in A} r_k\phi_{ij}^k + \sum_{k\in D}\sum_{j:(j,i)\in A} r_k\phi_{ji}^k + \sum_{k\in D}\sum_{f\in F}\sum_{n\in C_f} rr_{fr}y_{if}^n \leq c_{it} \quad \forall i \in N_c, \forall t \in R \tag{5.22}$$

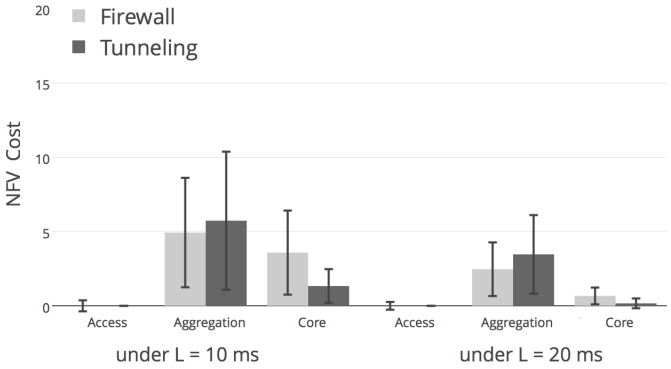
5.4 Results

We implemented our VNF-PR algorithm using CPLEX and AMPL scripting. We limited the execution time to 2 hours for the TE optimization phase and to 3 hours for the NVF optimization phase, which most of the time allowed reaching the optimum for the TE objective and good results for the NVF objective.

We adopted the three-tier topology represented in Fig. 5.3 as a good reference for an Internet Service Provider network. Each access node is connected to two aggregation nodes, each aggregation node is connected to two core nodes, and core nodes are fully meshed. We consider all the nodes as NFVI nodes that can host



(a) TE objective.



(b) TE-NFV cost objective.

Figure 5.4: VNF node distribution across VNFI layers (bufferized case).

VNFs. The demands are created uniformly in the interval $[a, b]$ so that edge demands cannot create any bottleneck at the access links. The aggregation links are dimensioned so that there is a risk of link saturation (i.e., link utilization higher than 100%) if the traffic distribution is not optimized. The core links are such that there is a very low bottleneck risk. Link latencies are set as follows, to cope for the different geographical scopes: 1 ms for access links, 3 ms for aggregation links, and 5 ms for core links (so that routing through core links is facilitated).

VNF processing latencies are set as in Fig. 5.2, for the bufferized and bufferless cases. We use two VM templates, one requiring 1 CPU and 16 GB of RAM, and one requiring 4 CPUs and 64 GB of RAM. We run tests setting for the end-to-end latency bound (L) with strict and loose values (10 and 20 ms, resp.). In order not to introduce unnecessary complexity to capture the differences between the different cases, we limit to two VNF types per demand: a tunneling VNF (requiring decompression) and a firewall-like VNF. The NFVI layer is dimensioned so that there are enough computing resources to satisfy *individually* half of all the demands (i.e., excluding VNF sharing); NFVI access nodes are dimensioned so that they are composed of 5 CPUs, 80 GB RAM at each access node, twice and four times this

quantity at aggregation and core nodes, respectively.

Each case is emulated with 10 random demand matrices. For the bufferized VNF case, we analyze in the following the results shown in Fig. 5.4 (NFV cost distribution), Fig. 5.5 (link utilization distribution), Fig. 5.6 (end-to-end and VNF forwarding latency distribution).

As a general remark, independently of the objective and the latency bound, there is almost no VNF instantiated at access nodes and the quantity of both firewall and tunneling VNFs greatly varies at the aggregation and core segments. Moreover, in all cases but one, there are more tunnelling VNFs than firewall VNFs at the aggregation level, while the inverse holds at the core level, which is likely due to the fact that tunnelling VNFs are pushed toward the destination edge given the corresponding increasing in the traffic rate.

We provide a detailed analysis in the following, with two points of views: what happens when we also consider the NFV cost in the objective function, and what happens when we change the bound on the end-to-end latency. Then, we compare the bufferized case with the bufferless case.

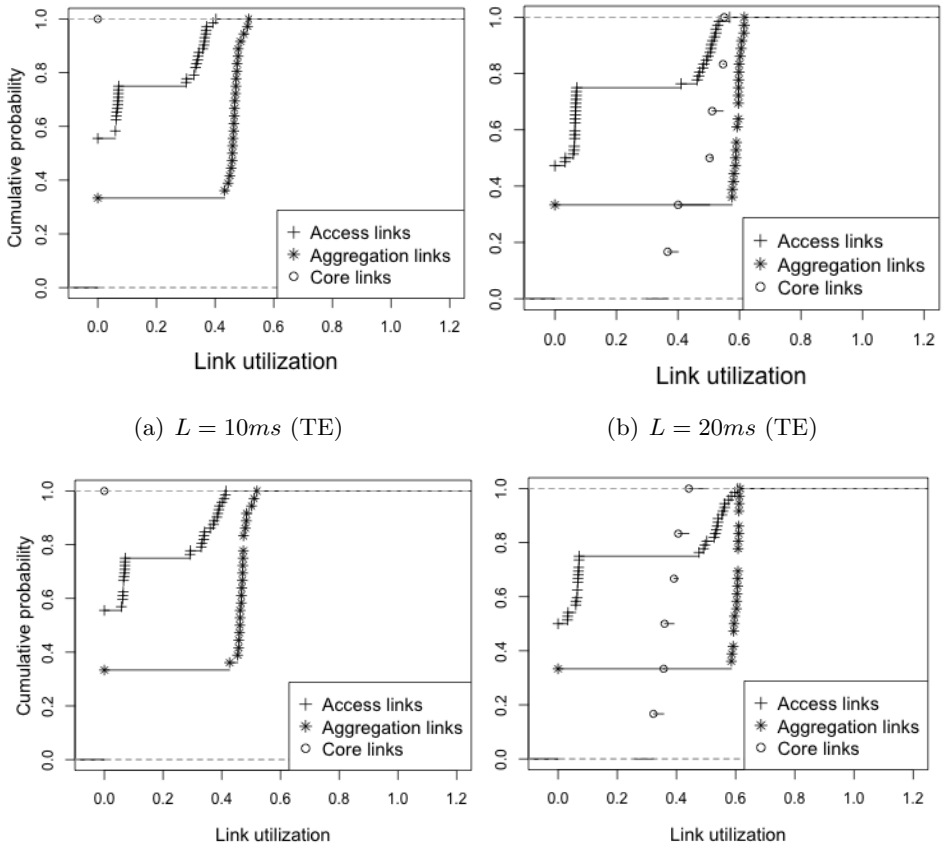
5.4.1 TE vs. TE-NFV objectives sensibility

We analyze the difference between the results with the TE objective and the results with the combined TE-NFV objective, comparing them for an equivalent maximum latency (L).

- NFVI cost (Fig. 5.4): the NFVI cost is significantly reduced with the TE-NFV objective. The relative gap between the two VNF types decreases passing from the TE objective to the TE-NFV objective, for both the aggregation and the core layers. Passing from the TE to the TE-NFV case allows to almost halve the overall number of required CPUs, while maintaining the same TE performance level thanks to our math-heuristic.

- Link utilization (Fig. 5.5): for the aggregation links, there is no major difference. This is likely due to our math-heuristic that first optimizes the TE goal, and then the NFV cost. The core links tend to be less used with the TE-NFV objective and when the latency is set to $20ms$.

- Latency distribution (Fig. 5.6): the total latency distributions are roughly equivalent, while the experienced forwarding latency of the two VNF types increases under the TE-NFV objective. This is likely due to the VNF cost reduction of the second step of the math-heuristic, leading to higher concentration of the flows on fewer VNFs (see Fig. 5.4), whose latency increase exponentially with the traffic load in the bufferized case.



(a) $L = 10ms$ (TE)

(b) $L = 20ms$ (TE)

(c) $L = 10ms$ (TE-NFV)

(d) $L = 20ms$ (TE-NFV)

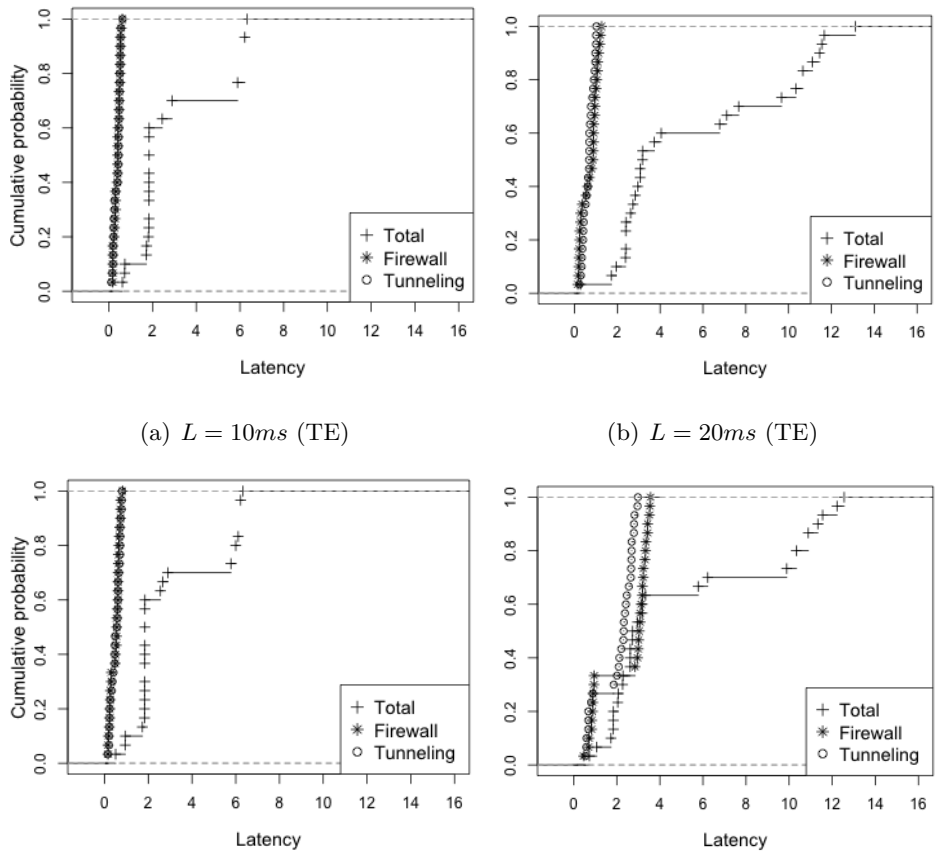
Figure 5.5: Link utilization empirical CDFs (bufferized case).

Independently of the objective, the forwarding latencies of the firewall and tunneling VNFs are close and globally largely lower than the total latency given the non negligible weight of the link latencies.

5.4.2 Sensibility to the latency bound

We analyze in the following the results further insisting on the impact of the VNF chain latency bound L on the results.

- NVFI cost (Fig. 5.4): setting a strict delay bound pushes the NFV cost down and the level of VNF sharing up. While in the TE case the strict latency bound case ($L = 10$ ms) leads to 30% lower NFVI cost than the loose latency bound ($L = 20$ ms), in the TE-NFV case we have an opposite behavior, with a cost doubled passing from the strict to the loose latency bound. This happens because the NFV step under TE-NFV objective let VNFs be better distributed across the different NFVI nodes, almost independently of the layer



(a) $L = 10ms$ (TE)

(b) $L = 20ms$ (TE)

(c) $L = 10ms$ (TE-NFV)

(d) $L = 20ms$ (TE-NFV)

Figure 5.6: Empirical CDFs of latency components (bufferized case).

(aggregation or core) and also be better shared.

- Link utilization (Fig. 5.5): all the links are affected by the latency change. The strict latency bound case ($L = 10$ ms) leads to lower utilization than the loose case - access and aggregation link utilization increases by around 10% while it increases by 30% for core links.

- Latency distribution (Fig. 5.6): obviously, the strict latency bound case leads to the lowest latency distribution: the end-to-end latency and VNF forwarding latency have similar profiles regardless to the objectives. Instead, for the loose case ($L = 20$ ms), the VNF forwarding latency is higher for both types.The median latency ranges from 1 to nearly 3 for the loose case, which indicates that VNF sharing is strongly motivated when the latency bound becomes less stringent. This is confirmed by the distribution of the number of enabled VNFs, which significantly decreases for less stringent latency bounds.

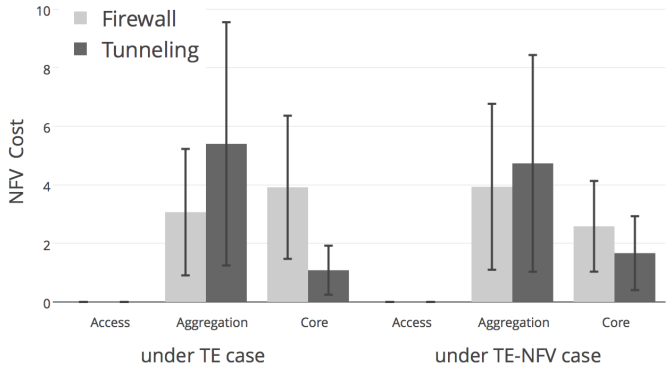
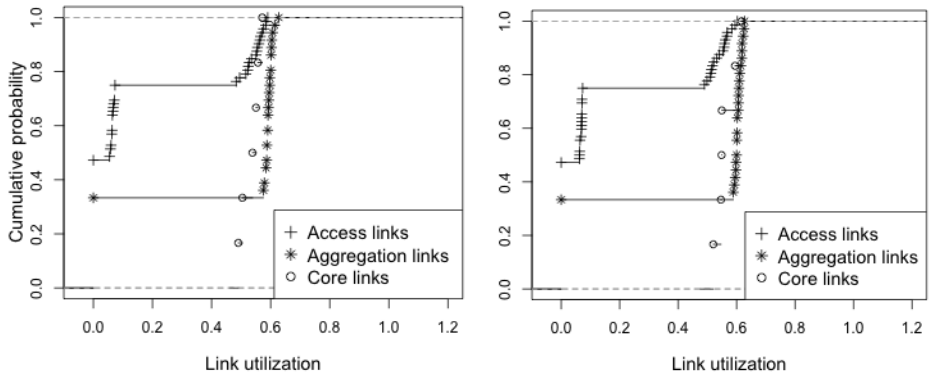


Figure 5.7: VNF node distribution across NFVI layers (bufferless, $L = 20ms$).



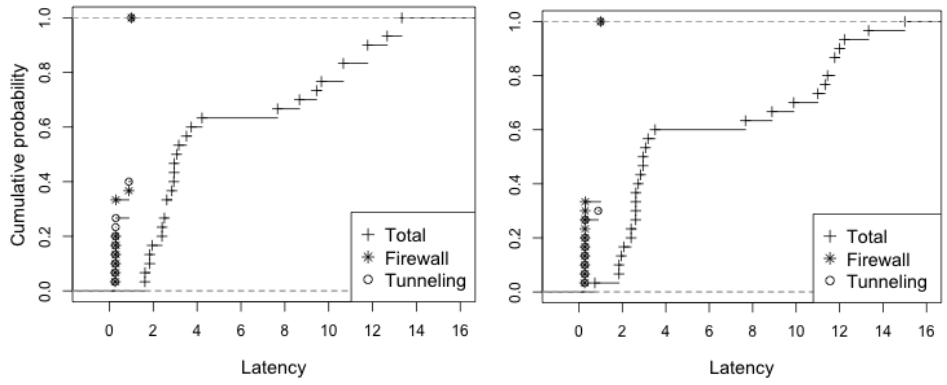
(a) $L = 20ms$ (TE) (b) $L = 20ms$ (TE-NFV)

Figure 5.8: Link utilization empirical CDFs (bufferless case).

5.4.3 Bufferized vs bufferless VNF switching mode

At last, we compare the bufferized case to the bufferless case. We consider only the strict latency bound ($L = 20$ ms) situation in order to better analyze the impact of the VNF forwarding mode (indeed, more stringent latency bounds have a higher impact on the placement and routing). Fig. 5.7, Fig. 5.8 and Fig. 5.9 show respectively the NFV cost, link utilization, latency distributions, to be compared with the corresponding previous ones. We can observe that:

- NFVI cost (Fig. 5.7 vs Fig. 5.4): for the TE case, the number of CPUs, and hence VNFs, in the bufferless mode is less than in the bufferized mode. Instead, for the TE-NFV case, there are slightly more VNFs with the bufferless mode. Associated with the observed increased latency, this implies that when TE is the single goal, the bufferless mode offers a higher VNF sharing among demands. For the TE-NFV case, this gap nullifies and is slightly inverted. We also note that the bufferized mode pushes more VNFs toward the aggregation



(a) $L = 20ms$ (TE) (b) $L = 20ms$ (TE-NFV)

Figure 5.9: Empirical CDFs of latency components (bufferless case).

NFVI nodes compared to the bufferless mode.

- Link utilization (Fig. 5.8 vs Fig. 5.5): a remarkable difference appears only for the TE-NFV case and only for the core links. The median utilization of core links moves from roughly 40% in the bufferized case to roughly 60% in bufferless case.

- Latency distribution (Fig. 5.9 vs Fig. 5.6): the total latency is higher with the bufferless mode. The firewall and tunneling latencies curves are identical for the bufferless mode. Moreover, with the bufferless mode we globally get a lower latency than with the bufferized mode.

5.5 Summary

This chapter proposes a network functions virtualization orchestration model and an algorithm that go beyond recent work at the state of the art. Our model takes into consideration specific NFV forwarding constraints that make the orchestration of edge demands over virtual network function chains unique yet complex. In order to master the time complexity while considering both traffic engineering and network functions virtualization infrastructure cost optimization goals, we adopt a math-heuristic resolution method.

Results from extensive tests qualify the impact of the virtual network function forwarding mode on the orchestration result. We highlighted many interesting aspects. For instance, when the virtualization infrastructure cost is considered irrelevant (i.e., a lot of resources available), using a bufferless forwarding mode (i.e., a virtual network function setting such that it does not bufferize incoming packets more than needed to allow a constant-latency forwarding time up to a maximum

load) allows higher virtual network function resource sharing than using a bufferized forwarding mode (where instead packets can be bufferized, with higher maximum load and piece-wise forwarding latency behavior).

Another important insight of our analysis is that setting strict end-to-end latency bounds on VNF chains can greatly affect the distribution of NFV nodes, increases the virtualization infrastructure cost while decreasing network link utilization because of a higher virtual network function distribution, especially at the aggregation level. With the adopted datasets, access layer tends not to host virtual network functions, which suggests that only extremely low service latency requirements related to edge cloud services, rather than network function virtualization, could justify a so pervasive deployment of virtualization resources into the access network.

Chapter 6

Conclusion and Perspectives

Data Center Networking (DCN) is a challenging field of applications of old and new technologies and concepts. In this dissertation, we investigated various aspects related to data center network optimization. Our research motivation was to understand new arising environments in networking mixing system and network architectures, metamorphosed by software virtualization and new interconnection topologies and softwarized communication modes. We presented in this dissertation three main contributions.

Adopting the point of view of the data center network operator, the first contribution presents the virtual machine placement problem for data-center networks exposing two new key functionalities: layer-2 multipath forwarding and virtual bridging. Once solely based on system level constraints, more recently virtual machine placement takes into consideration not only energy consumption minimization but also network link utilization minimization (traffic engineering goal) to ensure energy, application and network efficiency. In this field, our contribution lead to the formalization of the generic virtual machine placement problem, the proposal of a repeated matching heuristic for its resolution, and an exhaustive sensibility analysis on not so-well understood aspects in data-center networking. We observed counterintuitive behaviors, as for instance that multipath forwarding has a positive impact on energy efficiency when virtual bridging is disabled, and this positive impact becomes negligible when virtual bridging is enabled or energy efficiency is not considered as important. Moreover, we have found that when energy efficiency is the primary goal of the data center network optimization, both multipath forwarding and virtual bridging can be counter-productive, leading to link saturation. Also relevant is the finding that using virtual bridging when traffic engineering is the primary goal is instead definitely interesting. Other important observations allow assessing which data-center network topology better takes profit from specific

data-center network environments.

Furthermore, adopting the point of view of the application and the Infrastructure as a Service (IaaS) user, we investigated the throughput fairness offered at the equilibrium to data-center edge nodes, accounting for the TCP utility and elastic demand behavior. We presented a generalized formulation of the basic proportional fairness model for data-center networks allowing multipath forwarding for elastic demands. We described, analyzed and evaluated our model under two different TCP utility functions: fixed-delay and weighted-delay. Analyzing the behavior with rising data-center network topologies (fat-tree and BCube topologies of various sizes), we observed a number of interesting aspects. In particular, we found out how and when the global throughput is closely correlated to the topology.

Finally, we adopted the Internet service provider carrier network perspective. Major interest has been lately oriented in both industry and academic fora toward the virtualization of network functions in carrier networks. We explored the problem of routing user demands via a chain of virtual network functions that need themselves to be placed across clusters into the carrier network. The modeled problem mixes therefore facility location with multi-commodity flow problems, and integrate unique constraints due to network functions virtualization. Results are interesting and show under which circumstances it can become interesting placing virtual network function and virtualization server clusters into the access, aggregation and core network segments.

The latter is a preliminary work with high potential for further work. As it is expected that network functions virtualization will allow, in the close future, internet service providers to get rid of the vendor lock-in in network hardware, virtualization systems need to prove to be robust against security threads. NFV systems should be at least as reliable as legacy systems. The expectation is that network function virtualization systems could even grant more advantageous reliability-cost and availability-cost trade-offs than legacy systems. The protocols and architectures needed to meet this expectation are yet not fully defined, nor investigated and standardized. Our further research effort will therefore address these open questions, towards the design of advance carrier-grade cloud orchestration platforms and algorithms, the integration of NFV systems in cellular and mobile networks, with particular care toward the consideration of specific industry constraints and needs.

Publications

International Journals with peer review

[1] D. Belabed, S. Secci, G. Pujolle, and D. Medhi. Striking a Balance between Traffic Engineering and Energy Efficiency in Virtual Machine Placement. *IEEE Transactions on Network and Service Management*, to appear.

International conferences with peer review

[2] D. Belabed, S. Secci, G. Pujolle, and D. Medhi. Impact of ethernet multipath routing on data center network consolidations. In *IEEE 34th International Conference onDistributed Computing Systems Workshops (ICDCS)*, pages 57–62, June 2014.

[3] D. Belabed, S. Secci, G. Pujolle, and D. Medhi. Impact of virtual bridging on virtual machine placement in data center networking. In *26th International Teletraffic Congress (ITC)*, pages 1–9, Sept 2014.

[4] D. Belabed, S. Secci, G. Pujolle, and D. Medhi. On traffic fairness in data center fabrics. In *IEEE 3rd International Conference on Cloud Networking (CloudNet)*, pages 40–45, Oct 2014.

Submitted

[5] B. Addis, D. Belabed, M. Bouet and S. Secci. Virtual Network Functions Placement and Routing Optimization. submitted to *IEEE 4th International Conference on Cloud Networking (CloudNet), 2015* .

Part I

Appendix

.1 Matching cost matrix computation

Block 1

Matching two VMs. Given that the matching does not make real sense, its cost is set to infinity.

Block 2

Matching a container pair with a VM. This matching forms a Kit $\phi(cp, \overline{D}^V, \overline{D}^R)$ with $\overline{D}^R = \oslash$ and $\overline{D}^V = \{v\}$. Let cp be the i^{th} element of L_2 and v be the j^{th} element of L_1. With a recursive cp, the matching cost is

$$z_{n_1+i,j} = \sum_{s \in S} K^s_{c^1} / d^s_v. \tag{1}$$

Otherwise, the VM is assigned to the container with the least cost, i.e.:

$$z_{n_1+i,j} = \min\{\textstyle\sum_{s \in S} \frac{K^s_{c^1}}{d^s_v}, \sum_{s \in S} \frac{K^s_{c^2}}{d^s_v}\} + \Theta. \tag{2}$$

For this latter case, Θ is a big constant so that $\Theta \gg \frac{K^s_{c^i}}{d^s_{v_j}}, \forall i, \forall j$; it is meant to discourage the matching over lower cost matchings, given that the created Kit is unfeasible (no route between containers), and it has an unused container.

Block 3: Matching two container pairs. Given that the matching does not make real sense, its cost is set to infinity.

Block 4: Matching an RBridge path with a VM. Given that the matching does not make real sense, its cost is set to infinity.

Block 5: Matching an RBridge path with container pair. Let rp be the i^{th} element of L_3 and cp be the j^{th} element of L_2.

$$z_{n_1+n_2+i,n_1+j} = \begin{cases} 2\Theta & \text{if } c^1 \neq c^2 \\ \infty & \text{otherwise.} \end{cases} \tag{3}$$

In the first case, an unfeasible Kit is formed with both containers unused, while the second case is impossible as it is pointless.

Block 6

Matching two RBridge paths. Given that the matching does not make real sense, its cost is set to infinity.

Block 10

Matching two Kits. Let $\phi_i(cp_a, D^V_a, D^R_a)$ be the i^{th} Kit of L_4 and $\phi_j(cp_b, D^V_b, D^R_b)$ be the j^{th} Kit of L_4. For $i = j$, since an element cannot be matched with itself, the matching cost is equal to the cost of the Kit; otherwise, several cases are possible:

Case 1) All VMs are assigned to cp_a. The matching cost is therefore equal to the cost of the new Kit $\phi'(cp_a, D^V = D_a^V \cup D_b^V, D_a^R)$, i.e., $v_1 = \mu(\phi')$. Assignment of VMs to Kit container is the result of an optimization so that the cost of the Kit is minimized as follows.

$$v_1 = \min\ (1-\lambda) \sum_{c}^{cp} \sum_{v}^{D^V} \left(\sum_{s \in S} \frac{K_c^s}{d_v^s} + \Gamma T_v^c \right) x_v^{c_i} + \lambda U \tag{4}$$

subject to

$$\sum_{c_i \in cp_a} x_v^{c_i} = 1, \quad \forall v \in D^V \tag{5}$$

$$\sum_{v_i \in D^V} \sum_{v_j \in D^V}^{i \neq j} t_{v_i}^{v_j} x_{v_i}^c + l_e \leq K_e U, \quad \forall e \in E_a \tag{6}$$

$$\sum_{v_i \in D^V} \sum_{v_j \in D^V}^{i \neq j} (t_{v_i}^{v_j} x_{v_i}^c + t_{v_i}^{v_j} x_{v_j}^c) = T_v^c, \quad \forall c \in cp_a. \tag{7}$$

where $x_v^{c_i}$ is a binary variable equal to 1 if $v \in D^V$ is assigned to $c_i \in cp_a$ (0 otherwise); $E = \{(c,r),(r,c) | c \in cp, r \in D^R\}$, is the set containing the two links from container c along the route r, and from r to c, and $E_a = \{(c,r),(r,c) | c \in cp_a, r \in D^R\} \subset E$; l_e is the load of the access link $e \in E$ as of last Packing configuration without ϕ_i and ϕ_j; K_e are the link capacities. Constraint (5) guarantees each VM is assigned to only one container; (6) computes U. The objective function also includes cost components inversely proportional to system resources utilization.
Case 2) All VMs are assigned to cp_b. Similarly to the previous case, the new kit $\phi''(cp_b, D_a^V \cup D_b^V, D_b^R)$ is formed with $v_2 = \mu(\phi'')$.
Case 3) The container pairs (cp_a, cp_b) and the set of RB paths (D_a^R, D_b^R) do not change, and VMs can be exchanged between Kits. That is, every $v \in D_a^V \cup D_b^V$ can swap its container.

We then need to find the optimal pair assignment. The cost of the new situation should be less then the previous one. This can be formulated as an integer linear problem. Let us define w_v as a binary variable so that $w_v = 1$ if the $v \in D_a^V$ swaps its current Kit ϕ' for ϕ'', and $w_v = 0$ otherwise; in order to avoid non linearities, we admit that v swaps its current $c^{i^{th}} \in cp_a$ for the $c^{i^{th}} \in cp_b$, i.e., the destination container is in the same indexed position in the container pair than in the source container pair, so $x_v^{c_i}$ is considered as a parameter and no longer as a variable in the following. Also, let z_v be a binary variable equal to 1 if $v \in D_b^V$ swaps its current Kit ϕ'' for ϕ', and 0 otherwise. Finally, g_v and h_v are the marginal costs formed by the pair exchanges (i.e., the difference between the new and the old cost for each $v \in D_a^V \cup D_b^V$), defined as

$$g_v = \mu(\phi'' \cup \{v\}) - \mu(\phi') \tag{8}$$

$$h_v = \mu(\phi' \cup \{v\}) - \mu(\phi''). \tag{9}$$

The swapping problem can be formulated so as to minimize the cost of the Packing:

$$v_3 = \min \sum_{v \in D_a^V} g_v \; w_v + \sum_{v \in D_b^V} h_v \; z_v. \tag{10}$$

Among the cases, we choose the least cost one:

$$z_{n_1+n_2+n_3+i,n_1+n_2+n_3+j} = min\{v_1, v_2, v_3\}. \tag{11}$$

Block 8

Matching a Kit with container pair. Let $\phi_1(cp_a, D_a^V, D_a^R)$ be the i^{th} Kit of L_4 and cp_b be the j^{th} pair of L_2. The cp_b can be seen as a Kit with none assigned $D_a^V = D_a^R = \oslash$: $\phi_2 = (cp_b, \oslash, \oslash) \in L_4$. The matching cost is then the cost of the new kit $\phi(cp_b, D_a^V, D_a^R)$

$$z_{n_1+n_2+n_3+i,n_1+j} = \quad \mu(\phi). \tag{12}$$

Block 7

Matching a Kit with a virtual machine. Let $\phi_1(cp_a, D_a^V, D_a^R)$ be the i^{th} Kit of L_4 and $q(v^1, v^2)$ be the j^{th} pair of L_1. Three cases can be considered.

Case 1) The Kit is as $D_a^V = \oslash$: $\phi(cp_a, \oslash, D_a^R)$. Then D_a^V becomes $D_a^V = \{q\}$. The matching cost for this case is

$$z_{n_1+n_2+n_3+i,j} = \mu(\phi). \tag{13}$$

Case 2) The Kit has $D_a^R = \oslash$, i.e., $\phi(cp_a, D_a^V, \oslash)$. The virtual machine can be assigned to the container cp_a. Then D_a^V becomes $D_a^V \cup \{q\}$. The matching cost for this case is (now $q \in D_a^V$):

$$z_{n_1+n_2+n_3+i,j} = \begin{cases} \mu(\phi) & \text{if } c^1 = c^2 \\ \infty & \text{otherwise.} \end{cases} \tag{14}$$

Case 3) The Kit is as $\phi(cp_a, D_a^V, D_a^R)$. Then D_a^V becomes $D_a^V \cup \{q\}$. Noting that now now $q \in D_a^V$, the matching cost for this case is

$$z_{n_1+n_2+n_3+i,j} = \mu(\phi). \tag{15}$$

Block 9

Matching a Kit with RBridge path. Let $\phi(cp_a, D_a^V, D_a^R)$ be the i^{th} Kit of L_4 and $q(r^1, r^2)$ be the j^{th} pair of L_3. Three case can be considered:

Case 1): The Kit is as $D_a^V = \oslash$: $\phi(cp_a, \oslash, D_a^R)$. The matching for this case is impossible, i.e.,

$$z_{n_1+n_2+n_3+i,n_1+n_2+j} = \infty. \tag{16}$$

Case 2): The Kit is as $D_a^R = \oslash$: $\phi(cp_a, D_a^V, \oslash)$
The RBridge path can be assigned to the container cp_a. Then D_a^R becomes $D_a^R = \{q\}$. This is possible if cp_a is not *recursive*.

$$z_{n_1+n_2+n_3+i,n_1+n_2+j} = \begin{cases} \mu(\phi) & \text{if } c^1 \neq c^2 \\ \infty & \text{otherwise.} \end{cases} \tag{17}$$

Case 3): The Kit is as $\phi(cp_a, D_a^V, D_a^R)$. Then D_a^R becomes $D_a^R \cup \{q\}$. The matching cost for this case is (now $q \in D_a^R$):

$$z_{n_1+n_2+n_3+i,n_1+n_2+j} = \mu(\phi). \tag{18}$$

The least cost case is then selected for this block.

References

[1] Q. Zhang, L. Cheng, and R. Boutaba. "Cloud computing: state-of-the-art and research challenges ". *Journal of Internet Services and Applications*, Vol. 1(1), pages 7–18, 2010.

[2] K. Weins. *Cloud computing trends: 2014 state of the cloud survey.* April 2014.

[3] D. Floyer. Survey results: Vmware dominant in multi-hypervisor data centers. `http://wikibon.org/wiki/v/VMware_Dominant_in_Multi-Hypervisor_Data_Centers`, 2013.

[4] Hawley. "Evaluating server virtualization platforms for the future ". *NET Developers Journal, SYS-CON Media*, 2008.

[5] Extreme Marketing Team. Data center survey. `http://www.extremenetworks.com/data-center-survey-infographic`, 2012.

[6] R. J. Creasy. "The origin of the vm/370 time-sharing system ". *IBM Journal of Research and Development*, Vol. 25(5), pages 483–490, 1981.

[7] G. J. Popek and R. P. Goldberg. "Formal requirements for virtualizable third generation architectures ". *Communications of the ACM*, Vol. 17(7), pages 412–421, 1974.

[8] Cisco. Cisco global cloud index: Forecast and methodology, 2013-2018. `http://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/Cloud_Index_White_Paper.html`.

[9] R. Perlman. "An algorithm for distributed computation of a spanningtree in an extended lan ". In *ACM SIGCOMM Computer Communication Review*, Vol. 15, pages 44–53. ACM, 1985.

[10] APC by Schneider Electric. *Virtualization and consolidation.* 2009. `http://uk.idc.com/downloads/events/di09/8_oconnor.pdf`

[11] Y. Rekhter and T. Li. *A border gateway protocol 4 (bgp-4)*. 1995.

[12] J. Moy. "Open shortest path first (ospf) version 2 ". *IETF: The Internet Engineering Taskforce RFC*, 2328, 1998.

[13] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. "OpenFlow: enabling innovation in campus networks ". *ACM SIGCOMM Computer Communication Review*, Vol. 38(2), pages 69–74, 2008.

[14] Open Networking Foundation Committee et al. *Software-defined networking: The new norm for networks*. 2012.

[15] GS NFV 001 Network Functions Virtualisation (NFV); Use Cases. `http://www.etsi.org/deliver/etsi_gs/NFV/001_099/001/01.01.01_60/gs_NFV001v010101p.pdf`, October 2013.

[16] N. Foster, R. Harrison, M. J. Freedman, C. Monsanto, J. Rexford, A. Story, and D. Walker. "Frenetic: A network programming language ". In *ACM SIGPLAN Notices*, 46, pages 279–291. ACM, 2011.

[17] J. Batalle, J. F. Riera, E. Escalona, and J. Garcia-Espin. "On the implementation of nfv over an openflow infrastructure: Routing function virtualization ". In *IEEE SDN for Future Networks and Services (SDN4FNS)IEEE SDN for*, pages 1–6. IEEE, 2013.

[18] H. Xie, Y. Li, J. Wang, D. Lopez, T. Tsou, and Y. Wen. "vrgw: Towards network function virtualization enabled by software defined networking ". In *21st IEEE International Conference on Network Protocols (ICNP)*, pages 1–2. IEEE, 2013.

[19] M. Kuźniar, P. Perešíni, N. Vasić, M. Canini, and D. Kostić. "Automatic failure recovery for software-defined networks ". In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pages 159–160. ACM, 2013.

[20] S. Song, S. Hong, X. Guan, B. Choi, and C. Choi. "Neod: network embedded on-line disaster management framework for software defined networking ". In *IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pages 492–498. IEEE, 2013.

[21] H. Kim, J.R. Santos, Y. Turner, M. Schlansker, J. Tourrilhes, and N. Feamster. "Coronet: Fault tolerance for software defined networks ". In *20th IEEE International Conference on Network Protocols (ICNP)*, pages 1–2. IEEE, 2012.

[22] H. Kim, A. Voellmy, S. Burnett, N. Feamster, and R. Clark. *Lithium: Event-driven network control*. In *Georgia Institute of Technology*. 2012.

[23] M. Reitblatt, M. Canini, A. Guha, and N. Foster. "Fattire: Declarative fault tolerance for software-defined networks ". In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pages 109–114. ACM, 2013.

[24] P. Fonseca, R. Bennesby, E. Mota, and A. Passito. "A replication component for resilient openflow-based networking ". In *IEEE Network Operations and Management Symposium (NOMS)*, pages 933–939. IEEE, 2012.

[25] Data center architecture overview. In *Cisco Data Center Infrastructure 2.5 Design Guide*, pages 7–16. Cisco, 2011.

[26] M. Al-Fares, A. Loukissas, and A. Vahdat. "A scalable, commodity data center network architecture ". In *ACM SIGCOMM Computer Communication Review*, Vol. 38, pages 63–74. ACM, 2008.

[27] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu. "Bcube: a high performance, server-centric network architecture for modular data centers ". *ACM SIGCOMM Computer Communication Review*, Vol. 39(4), pages 63–74, 2009.

[28] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu. "Dcell: a scalable and fault-tolerant network structure for data centers ". In *ACM SIGCOMM Computer Communication Review*, Vol. 38, pages 75–86. ACM, 2008.

[29] W. Vojdak. Rapid spanning tree protocol: A new solution from an old technology. *Compact PCI systems*, 2008.

[30] P. Thaler, N. Finn, D. Fedyk, G. Parsons, and E. Gray. IEEE 802.1 q. 2013.

[31] IEEE standard for local and metropolitan area networks—virtual bridged local area networks—amendment 4: Provider bridges. *IEEE Std 802.1ad-2005 (Amendment to IEEE Std 802.1Q-2005)*, pages 1–74, May 2006.

[32] "IEEE standard 802.1 ah ". *IEEE Draft Standard for Local and Metropolitan Area Networks, Virtual Bridged Local Area Networks, Amendment 6: Provider Backbone Bridges*, 2008.

[33] Multi-chassis EtherChannel (MEC). In *Cisco Data Center Infrastructure 2.5 Design Guide*, pages 7–16. 2011.

[34] D. Santos, A. de Sousa, F. Alvelos, M. Dzida, and M. Pióro. "Optimization of link load balancing in multiple spanning tree routing networks ". *Telecommunication Systems*, Vol. 48(1-2), pages 109–124, 2011.

[35] A. Iwata, Y. Hidaka, M. Umayabashi, N. Enomoto, and A. Arutaki. "Global open ethernet (goe) system and its performance evaluation ". *IEEE Journal on Selected Areas in Communications*, Vol. 22(8), pages 1432–1442, 2004.

[36] G. Ibanez, A. Garcia, and A. Azcorra. "Alternative multiple spanning tree protocol (amstp) for optical ethernet backbones ". In *29th Annual IEEE International Conference on Local Computer Networks*, pages 744–751. IEEE, 2004.

[37] Provider backbone bridges with traffic engineering. *IEEE Ratifies Computer Society-Sponsored 802.1Qay*, 2009.

[38] D. Papadimitriou, E. Dotaro, and M. Vigoureux. "Ethernet layer 2 label switched paths ". In *IEEE Next Generation Internet Networks*, pages 188–194. IEEE, 2005.

[39] M. Seaman. EEE 802.1aq Shortest Path Bridging. *IEEE std*, 2006.

[40] J. Touch and R. Perlman. Transparent interconnection of lots of links (TRILL): Problem and applicability statement. *RFC 5556*, 2009.

[41] IETF Working Groups. "Network virtualization overlays (NVO3). ". `https://datatracker.ietf.org/wg/nvo3/documents/`.

[42] D. Farinacci, D. Lewis, D. Meyer, and V. Fuller. *The Locator/id Separation Protocol (LISP)*, In The Internet Engineering Task Force, 2013.

[43] T. Sridhar, L. Kreeger, D. Dutt, C. Wright, M. Bursell, M. Mahalingam, P. Agarwal, and K. Duda. *Virtual extensible local area network (vxlan): A framework for overlaying virtualized layer 2 networks over layer 3 networks*, 2014.

[44] M. Sridharan, A. Greenberg, N. Venkataramiah, Y. Wang, K. Duda, I. Ganga, G. Lin, M. Pearson, P. Thaler, and C. Tumuluri. "Nvgre: Network virtualization using generic routing encapsulation ". *IETF draft*, 2011.

[45] B. Davie and J. Gross. "A stateless transport tunneling protocol for network virtualization (stt) ". *draft-davie-stt-04*, 2013.

[46] I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann. "Vnr algorithm: A greedy approach for virtual networks reconfigurations ". In *IEEE Global Telecommunications Conference (GLOBECOM 2011)*, pages 1–6. IEEE, 2011.

[47] J. Inführ and G. R. Raidl. “Introducing the virtual network mapping problem with delay, routing and location constraints ”. In *Network Optimization*, pages 105–117. Springer, 2011.

[48] I. Houidi, W. Louati, W. B. Ameur, and D. Zeghlache. “Virtual network provisioning across multiple substrate networks ”. *Computer Networks*, Vol. 55(4), pages 1011–1023, 2011.

[49] J. Lischka and H. Karl. “A virtual network mapping algorithm based on subgraph isomorphism detection ”. In *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, pages 81–88. ACM, 2009.

[50] J. Lu and J. Turner. “Efficient mapping of virtual networks onto a shared substrate ”. *USA, Washington University: School of Engineering & Applied Science*, 2006.

[51] I. Houidi, W. Louati, and D. Zeghlache. “A distributed virtual network mapping algorithm ”. In *IEEE International Conference on Communications, 2008. ICC’08.*, pages 5634–5640. IEEE, 2008.

[52] I. Houidi, W. Louati, and D. Zeghlache. “A distributed and autonomic virtual network mapping framework ”. In *Fourth International Conference onAutonomic and Autonomous Systems. ICAS 2008.* , pages 241–247. IEEE, 2008.

[53] C. C. Marquezan, L. Z. Granville, G. Nunzi, and Marcus Brunner. “Distributed autonomic resource management for network virtualization ”. In *IEEE Network Operations and Management Symposium (NOMS)*, pages 463–470. IEEE, 2010.

[54] D. Yun and Y. Yi. “Virtual network embedding in wireless multihop networks ”. In *Proceedings of the 6th International Conference on Future Internet Technologies*, pages 30–33. ACM, 2011.

[55] I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann. “Vne-ac: Virtual network embedding algorithm based on ant colony metaheuristic ”. In *IEEE International Conference on Communications (ICC), 2011*, pages 1–6. IEEE, 2011.

[56] M. Yu, Y. Yi, J. Rexford, and M. Chiang. “Rethinking virtual network embedding: substrate support for path splitting and migration ”. *ACM SIGCOMM Computer Communication Review*, Vol. 38(2), pages 17–29, 2008.

[57] Y. Zhu and M. H. Ammar. "Algorithms for assigning substrate network resources to virtual network components ". In *INFOCOM*, 1200, pages 1–12, 2006.

[58] G. Veltri, Q. Huang, G. Qu, and M. Potkonjak. "Minimal and maximal exposure path algorithms for wireless embedded sensor networks ". In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 40–50. ACM, 2003.

[59] T. Guo, N. Wang, K. Moessner, and R. Tafazolli. "Shared backup network provision for virtual network embedding ". In *IEEE International Conference on Communications (ICC)*, pages 1–5. IEEE, 2011.

[60] M. Chowdhury, M. R. Rahman, and R. Boutaba. "Vineyard: Virtual network embedding algorithms with coordinated node and link mapping ". *IEEE/ACM Transactions on Networking (TON)*, Vol. 20(1), pages 206–219, 2012.

[61] M. Gupta and S. Singh. "Greening of the internet ". In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 19–26. ACM, 2003.

[62] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif. "Sandpiper: Black-box and gray-box resource management for virtual machines ". *Computer Networks*, Vol. 53(17), pages 2923–2938, 2009.

[63] S. Lee, R. Panigrahy, V. Prabhakaran, V. Ramasubramanian, Kunal Talwar, Lincoln Uyeda, and Udi Wieder. "Validating heuristics for virtual machines consolidation ". *Microsoft Research, MSR-TR-2011-9*, 2011.

[64] M. Mishra and A. Sahoo. "On theory of vm placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach ". In *IEEE International Conference on Cloud Computing (CLOUD)*, pages 275–282. IEEE, 2011.

[65] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu. "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing ". *Journal of Computer and System Sciences*, Vol. 79(8), pages 1230–1242, 2013.

[66] M. H. Ferdaus, M. Murshed, R.N. Calheiros, and R. Buyya. "Virtual machine consolidation in cloud data centers using aco metaheuristic ". In *Euro-Par 2014 Parallel Processing*, pages 306–317. Springer, 2014.

[67] X. Meng, V. Pappas, and L. Zhang. "Improving the scalability of data center networks with traffic-aware virtual machine placement ". In *IEEE INFOCOM, 2010 Proceedings*, pages 1–9. IEEE, 2010.

[68] H. Jin, T. Cheocherngngarn, D. Levy, A. Smith, D. Pan, J. Liu, and N. Pissinou. "Joint host-network optimization for energy-efficient data center networking ". *IEEE 27th International Symposium on IParallel & Distributed Processing (IPDPS)*, pages 623–634, 2013.

[69] H. Mi, H. Wang, G. Yin, Y. Zhou, D. Shi, and L. Yuan. "Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers ". In *IEEE International Conference on Services Computing (SCC)*, pages 514–521. IEEE, 2010.

[70] J. Xu and J. AB. Fortes. "Multi-objective virtual machine placement in virtualized data center environments ". In *Green Computing and Communications (GreenCom), 2010 IEEE/ACM International Conference on & International Conference on Cyber, Physical and Social Computing (CPSCom)*, pages 179–188. IEEE, 2010.

[71] M. Wang, X. Meng, and L. Zhang. "Consolidating virtual machines with dynamic bandwidth demand in data centers ". In *IEEE INFOCOM Proceedings*, pages 71–75. IEEE, 2011.

[72] B. Li, J. Li, J. Huai, T. Wo, Q. Li, and L. Zhong. "Enacloud: An energy-saving application live placement approach for cloud computing environments ". In *IEEE International Conference onCloud Computing, 2009. CLOUD'09.* , pages 17–24. IEEE, 2009.

[73] M. Rabbani, Rafael Pereira Esteves, Maxim Podlesny, Gwendal Simon, Lisandro Zambenedetti Granville, and Raouf Boutaba. "On tackling virtual data center embedding problem ". In *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, pages 177–184. IEEE, 2013.

[74] O. Biran and et al. "A Stable Network-Aware VM Placement for Cloud Systems ". In *Proceedings of the12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, pages 498–506, 2012.

[75] J. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang. "Joint vm placement and routing for data center traffic engineering ". In *INFOCOM, 2012 Proceedings IEEE*, pages 2876–2880. IEEE, 2012.

[76] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao. "Overview and principles of internet traffic engineering ". *RFC 3272*, may, 2002.

[77] B.d. Fortz and M. Thorup. "Internet traffic engineering by optimizing ospf weights ". In *Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings*, Vol. 2, pages 519–528. IEEE, 2000.

[78] A. Nucci, S. Bhattacharyya, N. Taft, and C. Diot. "Igp link weight assignment for operational tier-1 backbones ". *IEEE/ACM Transactions on Networking (TON)*, Vol. 15(4), pages 789–802, 2007.

[79] P. Casas, L. Fillatre, and S. Vaton. "Multi hour robust routing and fast load change detection for traffic engineering ". In *IEEE International Conference on Communications, 2008. ICC'08.* , pages 5777–5782. IEEE, 2008.

[80] S. Agarwal, A. Nucci, and S. Bhattacharyya. "Controlling hot potatoes in intradomain traffic engineering ". *SPRINT ATL res. rep. RR04-ATL-070677*, 2004.

[81] E. Mannie. "Generalized Multi-protocol Label Switching (GMPLS) architecture ". *Interface (tools.ietf.org)*, 501, pages 19, 2004.

[82] M. Chen, R. Zhang, and X. Duan. "Isis extensions in support of inter-autonomous system (as) mpls and gmpls traffic engineering ". *RFC 5316*, December, 2008.

[83] D. Katz, K. Kompella, and D. Yeung. "Traffic engineering (te) extensions to ospf version 2 ". *RFC 3630*, September, 2003.

[84] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow. "RSVP-TE: extensions to rsvp for lsp tunnels ". *RFC 3209*, December, 2001.

[85] K. Kompella, L. Berger, and Y. Rekhter. *Link bundling in mpls traffic engineering.* 2005.

[86] A. Farrel, J.P. Vasseur, and J. Ash. *A path computation element (PCE)-based architecture.* 2006.

[87] V. Jacobson. "Congestion avoidance and control ". In *ACM SIGCOMM Computer Communication Review*, Vol. 18, pages 314–329. ACM, 1988.

[88] J. Postel. *Transmission control protocol.* In tools.ietf.org, 1981.

[89] R. Srikant. *The mathematics of Internet congestion control.* Birkhauser, 2004.

[90] S. Kunniyur and R. Srikant. "End-to-end congestion control schemes: Utility functions, random losses and ecn marks ". *IEEE/ACM Transactions on Networking*, Vol. 11(5), pages 689–702, 2003.

[91] D.X. Wei, C. Jin, S.H. Low, and S. Hegde. "FAST TCP: Motivation, architecture, algorithms, performance ". *IEEE/ACM Transactions on Networking*, Vol. 14(6), pages 1246–1259, 2006.

[92] S. Low, L. Peterson, and L. Wang. "Understanding TCP Vegas: a duality model ". *J. ACM*, Vol. 49(2), pages 207–235, March 2002.

[93] M. Pióro and D. Medhi. *Routing, flow, and capacity design in communication and computer networks*. Elsevier/Morgan Kaufmann, 2004.

[94] R. Mazumdar, L.G. Mason, and C. Douligeris. "Fairness in network optimal flow control: optimality of product forms ". *IEEE Transactions on Communications*, Vol. 39(5), pages 775–782, 1991.

[95] F.P. Kelly, A.K. Maulloo, and D.K.H. Tan. "Rate control for communication networks: shadow prices, proportional fairness and stability ". *Journal of the Operational Research society*, Vol. 49(3), pages 237–252, 1998.

[96] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar. "Architectural guidelines for multipath TCP development ". *RFC6182 (March 2011), www. ietf. ort/rfc/6182*, 2011.

[97] W. Fang, X. Liang, S. Li, L. Chiaraviglio, and N. Xiong. "Vmplanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers ". *Computer Networks*, Vol. 57(1), pages 179–196, 2013.

[98] H. Van, F. Tran, and J-M Menaud. "Performance and power management for cloud infrastructures ". In *IEEE 3rd International Conference on Cloud Computing (CLOUD)*, pages 329–336. IEEE, 2010.

[99] F. Hermenier, X. Lorca, J. M. Menaud, G. Muller, and J. Lawall. "Entropy: a consolidation manager for clusters ". In *Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pages 41–50. ACM, 2009.

[100] R T. Marler and J S. Arora. "Survey of multi-objective optimization methods for engineering". *Structural and multidisciplinary optimization*, 26(6), pages pages 369–395, 2004.

[101] M.L. Balinski. "On finding integer solutions to linear programs ". In *Mathematica*. May 1964.

[102] M. R.önnqvist, S. Tragantalerngsak, and J. Holt. "A repeated matching heuristic for the single-source capacitated facility location problem ". *European Journal of Operational Research*, Vol. 116, pages 51–68, 1999.

[103] M. Rönnqvist, K. Holmberg, and D. Yuan. "An exact algorithm for the capacitated facility location problems with single sourcing ". *European Journal of Operational Research*, Vol. 113, pages 544–559, 1999.

[104] A. Reinert, B. Sansò, and S. Secci. "Design optimization of the petaweb architecture ". *Transactions on Networking, IEEE/ACM*, Vol. 17(1), pages 332–345, 2009.

[105] M.A. Forbes, J.N. Holt, P.J. Kilby, and A.M. Watts. "A matching algorithm with application to bus operations ". *Australian Journal of Combinatorics*, Vol. 4, pages 71–85, 1991.

[106] M. Engquist. "A successive shortest path algorithm for the assignment problem ". In *INFOR*, Vol. 20, pages 370–384. 1982.

[107] R. Jonker and A. Volgenant. "A shortest augmenting path algorithm for dense and sparse linear assignment problems ". *Computing*, Vol. 38, pages 325–340, 1986.

[108] T. Benson, A. Anand, A. Akella, and M. Zhang. "Understanding data center traffic characteristics ". *ACM SIGCOMM Computer Comm. Review*, Vol. 40(1), pages 92–99, 2010.

[109] T. Benson, A. Akella, and D.A. Maltz. "Network traffic characteristics of data centers in the wild ". In *Proceedings of the 10th annual conference on Internet measurement*, pages 267–280. ACM, 2010.

[110] A. Greenberg and et. al. "Vl2: a scalable and flexible data center network ". *ACM SIGCOMM Computer Communication Review*, Vol. 39(4), pages 51–62, 2009.

[111] L. S. Brakmo and L. L. Peterson. "TCP vegas: End to end congestion avoidance on a global internet ". *IEEE Journal on Selected Areas in Communications*, Vol. 13(8), pages 1465–1480, 1995.

[112] D. Medhi. "Applications with multiple parallel flows, pages Assessing their unfair advantage with proportional fair sharing TCP ". In *Proc. of IEEE International Conference on Communications (ICC'2014)*, Sydney, Australia, June 2014.

[113] M. Chiosi, D. Clarke, P. Willis, A. Reid, J. Feger, M. Bugenhagen, W. Khan, M Fargano, C Cui, H Denf, et al. "Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action ". In *SDN and OpenFlow World Congress*, pages 22–24, 2012.

[114] CenturyLink China Mobile Colt Deutsche Telekom KDDI NTT Orange Telecom Italia Telefonica Telstra Verizon AT&T, BT. *Network functions virtualization-introductory white paper*. October 2012.

[115] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee. Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine*, Vol. 53(2), pages 90–97, Feb 2015.

[116] Intel. "Impact of the intel data plane development kit (intel dpdk) on packet throughput in virtualized network elements ". Intel wite paper

[117] R. Guerzoni, R. Trivisonno, I. Vaishnavi, Z. Despotovic, A. Hecker, S. Beker, and D. Soldani. "A novel approach to virtual networks embedding for sdn management and orchestration ". In *Network Operations and Management Symposium (NOMS), 2014 IEEE*, pages 1–7. IEEE, 2014.

[118] S. Mehraghdam, M. Keller, and H. Karl. "Specifying and placing chains of virtual network functions ". In *IEEE 3rd International Conference on Cloud Networking (CloudNet)*, pages 7–13. IEEE, 2014.

[119] H. Moens and F. Turck. "Vnf-p: A model for efficient placement of virtualized network functions ". In *10th International Conference on Network and Service Management (CNSM)*, pages 418–423. IEEE, 2014.

[120] R. Mijumbi, J. Serrat, J.L. Gorricho, N. Bouten, and F. de Turck. "Design and evaluation of algorithms for mapping and scheduling of virtual network functions ". In *IEEE 1st International Conference on Network Softwarization (NETSOFT)*. IEEE, 2015.

[121] M. Xia, Y. Zhang, H. Green, A. Takacs, et. al. "Network function placement for nfv chaining in packet/optical datacenters ". In *European Conference on Optical Communication (ECOC)*, 2014.

[122] M. Bouet, J. Leguay, and V. Conan. "Cost-based placement of vdpi functions in nfv infrastructures ". In *Network Softwarization (NETSOFT), 2015 IEEE 1st International Conference on*. IEEE, April 2015.

UNIVERSITÉ PIERRE ET MARIE CURIE
LABORATOIRE D'INFORMATIQUE DE PARIS 6
4 PLACE JUSSIEU, 75005 PARIS