Design and Implementation of Resource Allocation Algorithms for a Network Operating

Farah Slim

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THESE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPERIEURE MINES-TELECOM ATLANTIQUE
BRETAGNE PAYS DE LA LOIRE - IMT ATLANTIQUE
COMUE UNIVERSITE BRETAGNE LOIRE

ECOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication

Spécialité : *Informatique*

Par
Farah SLIM

Etude et implémentation d'algorithmes de gestion de ressources pour un système d'exploitation de réseau

Thèse présentée et soutenue à IMT Atlantique – Campus de Rennes, le 13 mars 2018
Unité de recherche : Irisa
Thèse N° : 2018IMTA0071

Rapporteurs avant soutenance :

Pascal LORENZ	Professeur, Université de Haute Alsace
André-Luc BEYLOT	Professeur, IRIT/ENSEEIHT – Toulouse

Composition du Jury :

Président :	Olivier FESTOR	Professeur, Université de Lorraine
Examinateurs :	Pascal LORENZ	Professeur, Université de Haute Alsace
	André-Luc BEYLOT	Professeur, IRIT/ENSEEIHT – Toulouse
	Christine FRICKER	Chargée de recherche, Inria – Paris
	Hind CASTEL	Professeur, Télécom SudParis
	Fabrice GUILLEMIN	Chef de projet recherche, Orange Labs – Lannion
	Yassine Hadjadj-Aoul	Assistant Professor, Université de Rennes 1
Directeur de thèse :	Annie GRAVEY	Directeur d'études, IMT Atlantique

N° d'ordre: 2018IMTA0071

Sous le sceau de l'Université Bretagne Loire

IMT Atlantique Bretagne-Pays de la Loire

En accréditation conjointe avec l'Ecole Doctorale Matisse

Etude et implémentation d'algorithmes de gestion de ressources pour un système d'exploitation de réseau

---

Thèse de Doctorat

Mention : Informatique

Présentée par **Farah Slim**

Département : Informatique

Laboratoire : IRISA - Pôle

Directeur de thèse : Professeur Annie GRAVEY

Soutenance le 15 mars 2018

Jury :

Rapporteurs

André-Luc Beylot	Professeur (ENSEEIHT),
Pascal Lorenz	Professeur (UHA),

Examinateurs

Hind Castel	Professeur (IMT Paris Sud)
Olivier Festor	Directeur de recherche (INRIA)
Christine Fricker	Chargée de Recherche (INRIA)
Fabrice Guillemin	Ingénieur de recherche (Orange),
Yassine Hadjadj-Aoul	Maître de conférence (Rennes 1)

Acknowledgements

"Alone we can do so little; together we can do so much."
Hellen Keller

I have been accompanied and supported by many people during the last three years, I would never have been able to finish my dissertation without their support. I would like to express my gratitude for all of them.

First and foremost, I would like to express my deepest gratitude to my advisors: ***Dr.Fabrice Guillemin***, ***D*r.Annie Gravey** and ***Dr.Yassine Hadjadj-Aoul*** for accepting nothing less than excellence from me. I would like to thank them for their excellent guidance, caring, patience, and especially for providing me with an excellent atmosphere for doing research.

I am also grateful to ***Mr.Gaël Fromentoux*** for giving me the opportunity to live a challenging experience within his team in the Orange Labs and for supporting my work. His support over these three years has been very precious.

I would like to thank ***Mr.Eric Debeau***, ***Mr.Marc Bouillon*** and ***Mr.Nicolas Bihannic*** who were always willing to help and give their best suggestions.

Special thanks goes to all ***ARC*** team members for help, kindness and support. It was an honor to have the opportunity to work with all of you.

Special thanks goes, as well, to the reading committee for their precious time.

Last but not least, my parents ***Abdallah*** and ***Moufida***, without whom nothing would have been possible. I would like to thank them for scarifying everything to see me a happy and successful young woman and for supporting me throughout this thesis and my life in general.

I would like to thank my friends ***Dorra***, ***Maroua*** and ***Mariem***, my brother ***Amine*** and my sister Ghofrane for always supporting me.

Finally, very special thanks goes to you, ***Dali***, my best friend and my everything for always being there cheering me up and standing by me through the good and bad times.

Résumé

Mots clés : Cloud, Infrastructure, Algorithme, Ressources, Réseaux, Virtualisation, Fonction de Réseaux Virtualisée, Blocage

L'un des défis majeurs auquel sont confrontés les opérateurs télécom aujourd'hui est celui de perdre leur place sur le marché face à la multiplication des services proposés par les grands acteurs du Web, tels que les GAFA ou Netflix. En espérant mieux répondre aux besoins de ses clients, le secteur télécom est désormais au cœur d'une transformation digitale. Cette transformation s'appuie sur de nouvelles technologies émergentes telles que la virtualisation, les réseaux définis par logiciels, et le déploiement des services réseaux sur le cloud. Ces paradigmes ont été introduits par différentes initiatives (notamment l'ETSI) qui visent à virtualiser les fonctions réseau en les déployant sur le cloud sous le terme virtualisation des fonctions réseau (NFV). Cette nouvelle approche sur laquelle s'appuient les opérateurs télécom pour accélérer leur transformation numérique impactera non seulement la manière dont les réseaux sont définis mais aussi le rôle principal de l'opérateur qui doit désormais gérer les ressources cloud en combinaison avec les ressources réseaux.

Un deuxième enjeu technique de cette transformation pour l'opérateur est celui de répondre aux contraintes critiques imposées par les fonctions de réseaux virtualisées (NFV) tel que les contraintes temps-réel ou latence. En effet, plusieurs fonctions réseaux présentent de fortes contraintes en termes de latence et doivent par conséquent être déployées près de l'utilisateur final. Ces contraintes ont incité les opérateurs télécom à revisiter leurs infrastructures pour répondre à ces exigences; ceci en distribuant massivement leur data centers pour être présents dans ce qu'on appelle les Points de Présence (PoP) en bordure du réseau. Ceci permet de servir les utilisateurs finaux au plus près et répondre aux exigences strictes de certaines fonctions réseaux, par exemple les fonctions de Radio Access Network (RAN). Ces data centers disséminés sur la bordure du réseaux ont des capacités limitées en termes de ressources (calcul, stockage et ressources réseaux), tout au moins en regard des grands data centers déployés par les géants du cloud tels que Amazon et Google. Le défi majeur auquel est confronté l'opérateur télécom

est celui de la gestion de l'infrastructure qui combine les ressources cloud et réseau ; ceci implique la mise en place d'algorithmes adaptés pour l'allocation de ressources dans ce contexte.

Dans cette thèse, on se propose d'analyser les modifications qui vont affecter l'infrastructure de l'opérateur dans le but d'étudier l'allocation de ressources dans le contexte de data centers distribués en bordure de réseau en tenant compte de nouvelles contraintes qui n'ont pas été considérées dans les plateformes cloud traditionnelles.

Nous proposons d'abord un modèle probabiliste d'estimation de blocage des requêtes dans les plateformes cloud.

Traditionnellement, les ressources dans les plateformes cloud sont supposées infinies et le blocage des requêtes est généralement ignoré à cause de cette hypothèse. Mais avec l'évolution de l'infrastructure de l'opérateur vers une infrastructure massivement distribuée et caractérisée par des capacités finies, le blocage est une métrique clé pour évaluer les algorithmes d'allocation de ressources dans ce contexte.

Pour l'analyse de cette métrique, nous avons proposé un modèle analytique pour l'analyse de blocage dans un système cloud multidimensionnel, qui a été validé dans un premier lieu en utilisant un simulateur à événements discrets écrit en Matlab. Par la suite, nous avons effectué une analyse comparative des stratégies d'allocation de ressources utilisés dans la littérature et par la plateforme de gestion populaire Openstack.

Le modèle proposé ainsi que l'étude comparative, révèlent des résultats pratiques sur l'évaluation de la performance des stratégies d'allocation de ressources ainsi que le dimensionnement des systèmes cloud distribués avec des capacités limitées.

On propose également dans ce travail une stratégie d'allocation de ressources qui facilite la gestion de l'infrastructure pour l'opérateur et qui présente des performances meilleures que dans l'approche utilisée par la plateforme cloud Openstack, fortement recommandée dans le contexte NFV.

La stratégie adoptée par le projet Tricircle, la version Openstack dédiée aux déploiements multi-sites, étant très couteuse en termes d'échange d'informations entre l'orchestrateur global et l'infrastructure, nous proposons un algorithme d'allocation de ressources nommé CLOSE et qui prend en compte les informations locales seulement. Notre stratégie CLOSE propose également un mécanisme de collaboration entre les plateformes cloud avec des capacités limitées. Ce

mécanisme consiste à défléchir les requêtes bloquées à cause de l'insuffisance de ressources vers l'un des premiers voisins dans l'infrastructure avec un déplacement qui respecte la contrainte de latence imposée par la requête.

Pour le choix de voisin, nous avons considéré 2 variantes de l'algorithme:

- aléatoirement parmi les premiers voisins de la plateforme qui a reçu la requête initialement
- en se basant sur un compteur qui enregistre toutes les requêtes redirigées au niveau de chaque plateforme. Ce compteur est une moyenne glissante dans le temps qui reflète le niveau de congestion des data centers. L'idée, c'est de choisir le data center qui a moins défléchi de requêtes dans le passé. Ce compteur est donc défini en se basant sur des informations locales et limite l'échange d'informations entre l'infrastructure et l'orchestrateur global.

Pour l'évaluation de performances de notre stratégie, nous avons défini plusieurs scénarios de simulations pour comparer la performance de CLOSE contre celle obtenue en utilisant le mécanisme proposé dans la littérature, notamment le projet Tricircle dédié pour les déploiements multi-sites dans Openstack. Il s'avère que l'algorithme proposé présente des performances meilleures que dans l'approche utilisée par la plateforme cloud Openstack, fortement recommandée dans le contexte NFV.

En effet, pour accélérer la transition vers la virtualisation, plusieurs projets ont récemment émergé pour orchestrer les fonctions réseaux au sein de l'infrastructure. Le projet ONAP a été créé en fusionnant deux des plus grandes initiatives open source: ECOMP et Open-O.

En prenant des avantages de les deux projets, ONAP repose sur une architecture unifiée pour offrir une plate-forme ouverte permettant aux utilisateurs finaux de créer leurs propres fonctions de réseaux virtualisée. La plateforme vise à automatiser, orchestrer et gérer les fonctions virtualisées et les services réseau.

On se propose dans ce travail d'explorer le mécanisme d'allocation de ressources adopté par la plateforme d'orchestration ONAP, d'identifier les limites de ce mécanisme pour proposer par la suite une solution mieux adaptée au contexte NFV et qui ne nécessite aucune modification au préalable.

Dans l'architecture d'ONAP actuelle, les décisions de placement sont prises d'une manière centralisée par le contrôleur de l'infrastructure. Basé sur un algorithme

heuristique, le planificateur Openstack favorise les serveurs avec la plus grande quantité de ressources disponibles.

Cette mise en œuvre actuelle ne supporte aucune fonctionnalité multi-site, puisque l'allocation des ressources est faite sans prise en compte de la situation géographique. En effet, une fonction de réseau virtualisée est en général composé de plusieurs composants (appelé également sous-fonctions ou microservices), qui exécutent des tâches situés à différents niveaux fonctionnels du réseau, certains faisant partie du plan de données(data plane), tandis que d'autres font partie du plan de contrôle du réseau(control plane).

Les fonctions de type data plane présentent des contraintes strictes en terme de latence tandis que les fonctions control plane sont plus tolérents. Partant de cette observation, nous avons proposé un mécanisme d'allocation de ressources qui favorise le placement des fonctions de type data plane au niveau de la bordure du réseau en déplaçant les fonctions de type control plane vers d'autres niveaux, étant donné qu'elle ne possèdent pas des contraintes strictes en terme de latence. Pour cela, nous avons proposé un mécanisme de déflexion basé sur des seuils d'acceptation des fonctions de type control plane au niveau de la bordure du réseau. Ces seuils sont dynamiquement ajustés au cours du temps en se basant sur les conditions de charges du système.

L'évaluation de performances de notre stratgéie proposée dans le cadre de la plateforme ONAP a été réalisée avec des simulations en comparant les résultats par rapport aux résultats données en utilisant l'approche adoptée par ONAP, à savoir la stratégie d'allocation de ressources dans le projet multi-sites proposé par Openstack. Il s'avère qu'en terme de taux de blocage global, notre stratégie présente de meilleures performances par rapport aux performances obtenues avec la stratégie adoptée par ONAP.

Finalement, dans le dernier volet de ce manuscrit, on se propose d'étudier l'ajustement des seuils utilisés dans la contexte de la plateforme ONAP en se basant sur la technique d'optimisation populaire, à savoir les algorithmes génétiques. La stratégie qu'on propose consiste à définir des seuils d'acceptation de requetes optimaux et qui améliorent la collaboration entre les data centers voisins, un mécanisme que nous avons utilisé pour le placement des fonctions de réseaux virtualisées dans ce travail. Dans plusieurs scénarios, nous avons prouvé avec des simulations que les seuils optimaux données par l'algorithme génétique qu'on a proposé améliore la collaboration et réduit significativement le taux de blocage moyen de notre stratégie d'allocation de ressources.

Abstract

Key words : Cloud, Infrastructure, Algorithm, Resource Allocation, Networks, Virtualization, Virtualized Network Function, Blocking

One of the major challenges for network operators is the proliferation of services offered by the Web players (e.g., GAFA, Netflix, etc.). Trying to satisfy the needs of their customers and to keep their footprint in the digital market, network operators are rethinking their business models by adopting new emerging technologies such as virtualization, software-defined networks, and the deployment of cloud-based network services. The paradigm of Network Function Virtualization (NFV) has been introduced through the initiative by the European Telecommunications Standards Institute (ETSI), which aims at virtualizing network functions by deploying them on the cloud. This new approach on which telecom operators rely to accelerate their digital transformation will impact not only the way networks are defined but also the main role of the operator, who now has to manage cloud resources in combination with network resources.

A critical point in NFV is raised by the intrinsic constraints of virtualized network functions (VNFs) such as real-time or latency. Indeed, several network functions have high latency constraints and must therefore be deployed close to the end user. These constraints prompted telecom operators to revisit their infrastructure to meet these requirements, namely by massively distributing data centers in the so-called Points of Presence (PoPs) at the edge of the network. This allows the stringent requirements of certain network functions to be met; this notably the case for virtual Radio Access Network (RAN), which consists of virtualizing base station functionalities. These data centers geographically spread at the edge of the network have rather limited capacity in terms of resources (computing, storage and network resources) when compared with the huge data centers deployed by cloud giants such as Amazon. The major challenge for a telecom operator is the infrastructure management that combines cloud and network resources. This requires the implementation of appropriate algorithms for the allocation of resources in this context.

In this thesis, we propose to analyze the modifications that will affect the infrastructure of the operator in order to study the allocation of resources by taking into account new constraints that have not been so far considered in traditional cloud platforms. We first propose a probabilistic model for the estimation of blocking in cloud platforms; this point has not been addressed in the literature, always assuming that cloud resources are infinite. This model, which has been validated by several scenarios, will enable the operator to adequately dimension the resources of its infrastructure. In addition, we evaluate the performance of the most popular resource allocation algorithms that have been adopted in the NFV context as in the traditional cloud based on this model.

We then propose a resource allocation strategy that facilitates the management of the infrastructure by the operator. This strategy yields better performance when compared to the approach adopted by the OpenStack cloud platform, highly recommended in the NFV context. Finally, we analyze the management of resources in NFV orchestration platforms by exploring the two basic functions in this process, namely monitoring and scheduling. We also investigate the resource allocation mechanism adopted by the popular ONAP orchestration platform to identify its limits and to propose a solution that is better adapted to the NFV context.

Contents

List of Figures

List of Tables

Abbreviations

5G	Fifth Generation
AAI	Active and Available Inventory
API	Application Program Interface
AWS	Amazon Web Services
BBU	BroadBand Units
CDN	Content delivery network
CO	Central Office
CCO	Core Central Office
COTS	Commercial Off-The-Shelf
DCAE	Data Collection, Analytics and Events
E2E	End-To-End
EPC	Evolved Packet Core
ETSI	European Telecommunications Standards Institute
GAFA	Google Amazon Facebook Apple
HSS	Home Subscriber Server
HPC	High Performance Computing
IT	Information Technology
IaaS	Infrastructure-as-a-Service
IP	Internet Protocol
MANO	Management and Orchestration
MEC	Mobile Edge computing
MCO	Main Central Office
MME	Mobility Management Entity
MSO	Master Service Orchestrator
NFV	Network Function Virtualization

NIC	Network Interface Card
NFVI	NFV Infrastructure
NFVO	NFV orchestrator
NO	Network Operator
ONAP	Open Network Automation Platform
OTT	Over-The-Top
PM	Physical Machine
PoP	Point Of Presence
PDCP	Packet Data Convergence Protocol
RAN	Radio Access Network
RLC	Radio Link Control
SDN	Software Defined Network
SDC	Service Design and Creation
S/PGW	Servicing/Packet Gateway
Telco	Telecom Operator
VM	Virtual Machine
VIM	Virtual Infrastructure Management
VNFM	Virtual Network Functions Manager
vEPC	Virtual EPC

Chapter 1

Introduction

Contents

1.1 Telecommunication Network Evolution

Technology is changing our lives faster than ever before. Communication, health care or entertainment, our daily life is overwhelmed by the use of technology in almost every field. This is made possible by the emergence of smart devices that are becoming more ingrained in today's routine.

The Fortinet White paper indicates that more than a million new smart devices are connected to the Internet every day (Fortinet 2017) [2].

At the same time, with this explosion of the Internet of Things (IOT), the network usage is exploding and the mobile data traffic is expected to increase each year. In fact, a Cisco White Paper (Cisco 2016) reports that mobile data is expected to grow by 11 times in the next four years and that 50 billion IOT devices will be connected by 2021 [3].

This growth creates a challenge for telcos (telecom operators) who have to deal with these new waves of demands and spend huge efforts to not only meet customer demand but also to improve user experience.

Another major challenge facing telcos over the past years and all over the world is the significant growth of Over-The-Top (OTT) providers such as Google, Amazon

and Netflix. Telcos are losing their share of revenue as their primarily role is limited to mere traffic carriers rather than service provisioning. Hence, network operators need to regain their foothold in the market and make major changes in how they build and run their networks, offer new services and interact with customers [4].

As a result, telcos are embracing a new digital transformation that is rethinking their business models by adopting new approaches and emerging technologies such as the Virtualization of Network Functions (NFV).

One of the key initiatives in the communication industry is upgrading networks to virtualization that will free telcos from hardware dependence. NFV aims at decoupling network functions from proprietary hardware appliances so that such functions can run as software on commercial off-the-shelf (COTS) hardware.

NFV is changing the way networks are managed, offering more scalability and flexibility, and helping to create programmable networks for tomorrow's needs. Literally, NFV, this technology developed by the European Telecommunication Standards Institute (ETSI), is a key enabler of the coming 5G infrastructure helping to implement the most discussed concepts for designing 5G networks such as Network Slicing. This concept makes it possible to create several virtual networks on the top of the shared underlying infrastructure [5].

Another technological pillar in the architecture of future 5G mobile networks on which telecom operators rely to accelerate their journey towards virtualization is the Software Defined Network (SDN). SDN provides a programmable interface that controls and orchestrates networks at different levels of abstraction. Thanks to this paradigm, telcos can dynamically reconfigure the network using only software mechanisms. This is the reason why SDN promises to bring more agility and flexibility to the future network.

Network Virtualization is certainly a major revolution that will impact not only the architecture of networks but also the network operators infrastructure. In fact, to meet stringent real-time constraints, some network functions have to be hosted close to end users (e.g. Radio Access Network (RAN) functions, firewalls, deep packet inspection).

These latency requirements incite telcos to massively distribute their cloud infrastructure in order to be closer to end users. This lead to the development of geographically distributed mini data centers at the edge of the network (i.e. typically at Points of Presence (PoPs) level). Therefore, the network operator

infrastructure is evolving towards a distributed architecture of edge data centers with limited capacities deployed close to the end users.

To sum up, the new digital transformation is driven by the evolution of the network operator infrastructure towards a massively distributed architecture coupling cloud and network. However, due to this transformation, there are several challenges facing telcos who want to run in-cloud network.

1.2 Problem Statement

As we have said, the telecommunication infrastructure is going through a major transformation involving many radical changes. In fact, the recent emergence of new bandwidth-intensive and time-constrained services notably Virtualized Network Functions (VNFs) [6] is pushing network operators to geographically distribute mini data centers at the edge of the network. These edge data centers have rather small capacities of storage, compute and networking resources when compared to huge centralized data centers deployed, for instance, by Google[1] or Amazon.

This groundbreaking transformation raises many new challenges for network operators who henceforth have to manage cloud infrastructure in combination with network. The real challenge lies in how to manage cloud platforms in combination with network. In other words, the network resource, namely the bandwidth, must be considered in addition to cloud resources such as storage or compute while instantiating services. In this context, we addressed this issue within centralized cloud platforms as well as massively distributed platforms.

In addition, we evaluated performance of the most popular resource allocation algorithms by paying special attention to the blocking probability metric which has so far not been considered in the cloud literature.

In fact, in the context of telco cloud and given the evolution of the infrastructure that will very likely be composed of small data centers with limited capacities and deployed at the edge of network, blocking is a key performance metric to evaluate algorithms

Another major challenge that telco are facing while moving towards virtualized network functions is to deliver end-to-end network services by instantiating

[1] https://www.google.com/about/datacenters/inside/locations/index.html

VNFs, within the virtualized infrastructure, whilst taking into account their specific requirements. To cope with these challenges, there is a need to have a global orchestration platform, including instantiation logic, life-cycle management, as well as monitoring features. We explored in this thesis the monitoring feature by studying the impact of monitoring traffic and evaluating some monitoring tools that might be useful with regard to resource allocation. Furthermore, we conducted a technical study of the popular orchestration platform, namely the Open Network Automation Platform (ONAP). We finally proposed an appropriate resource allocation strategy to adopt in this context and then presented to ameliorate it using the Genetic Algorithm approach.
This work was done in the scope of Orange Labs research projects.

1.3 Document Structure

In **Chapter 2**, we first discuss the motivations that drive network operators towards virtualization. We then review the context of our thesis by providing a state of the art of cloud computing and virtualization. We focus on resource allocation strategies and the related existing solutions. Finally, we present a testing platform of a telco infrastructure that we have set up to identify hypotheses in this context. Furthermore, this platform enabled us to explore capabilities of some monitoring tools that may be useful for the resource allocation.

In **Chapter 3**, we conduct a study in order to evaluate performance of resource allocation strategies under centralized as well as distributed approaches. First, we propose an analytical model for the blocking analysis in multidimensional centralized cloud system. We lead a comparative analysis of the most popular placement's strategies based on our proposed model. Second, we propose a new strategy for resource allocation in distributed cloud context. We show that our strategy yields better performance when compared to the strategy adopted by the most popular cloud management platform, namely Openstack.

In **Chapter 4**, we present orchestration platforms for NFV by studying its main features including monitoring and scheduling. We first study the impact of monitoring traffic. We then propose an offloading strategy based on thresholds that can be applicable in the context of ONAP. We finally propose to set the optimal threshold for the offloading strategy based on the Genetic Algorithm techniques.

In **Chapter 5**, we point out the main contributions of this thesis and give the research perspectives in relation to the ongoing works.

Chapter 2

State Of the Art

Contents

2.1 Introduction

During the last decade, carrier grade virtualization technologies have been very successful in offering on-line services via on-demand computing and storage capacities in the cloud (cf. EC2 by Amazon, Azure by Microsoft, etc.). While cloud

resources were so far used to run applications owned by end users (residential or business customers), various initiatives in the design of 5G networks, including ETSI NFV and SDN, explore the virtualization of network. There is hence a clear need for carry out a study aiming at understanding keys technologies notably virtualization and cloud computing to diagnose the new challenges leveraged by the emergence of network virtualization in the context of telco cloud.

In this chapter, we first discuss motivations for this work within Section 2.2. Section 2.3 explores the background of the virtualization technology as well as the cloud computing concept, then some prevalent data centers management tools such as Openstack are presented. The main differences between centralized and distributed cloud are discussed in Section 2.4. In Section 2.5, we present an overview for resource allocation techniques in cloud computing with the corresponding related works. An overview of a testbed platform set up to assimilate the context in real case scenarios is presented in Section 2.6. Section 2.7 concludes this chapter.

2.2 Motivations

The convergence of IT and Telecom is a fundamental transformation that will deeply modify the way network operators conceive, produce and operate their services. This transformation incites telcos to redefine their business model by taking inspiration from IT cloud approaches in the manner of major actors such as OTT players. This revolution is now made possible by the emergence of new technological paradigms such as Network Function Virtualization (NFV), Software Defined Network (SDN) and service orchestration. With reference to $5G$ networks, the complementary relation between these technologies is described in [7].

2.2.1 Software Defined Network

As any new technology, there is no single definition of Software Defined Network. Over the last two years, most definitions have emphasized decoupling control logic from the forwarding hardware. In fact, the network intelligence is logically centralized in software-based controllers so called the control plane, and network devices become simple packet forwarding devices (the data plane) that can be programmed via an open interface [8].

The emerging definition of SDN is focusing less on decoupling control logic and more on the ability to provide programming interfaces within network equipments [9]. The main idea is to permit network operators to rely on network resources in the same easy manner as they do on storage and computing resources.

SDN promises to bring to the network what virtualization has brought to the server domains during these last years; better use of network resources, simplification and automation of network provisioning through the use of programmable interfaces and a global abstraction of hardware resources [10].

2.2.2 Network Function Virtualization

In the recent few years, cloud and virtualization have enabled the emergence of virtualization of network functions allowing network operators to decouple network functions from proprietary hardware appliances so that such functions can run in software. The ETSI Industry Specification Group for Network Functions Virtualization (ETSI NFV) is the group charged with designing architecture and developing requirements for various functions for telco networks.

NFV promises telco to deliver agility and flexibility by quickly scale up or down services to address variation of demands and to support innovation by enabling networking services to be running only via software on any industry-standard hardware [11].

Reducing operator CAPEX and OPEX costs is also one of the major benefits of NFV. In fact, NFV reduces equipment costs and power consumption due to the freedom to choose and build the hardware in the most efficient way that suit telcos needs and requirements [12]. It's clear that NFV can address the key trends confronting operators. Therefore, NFV is considered as one of the key technologies on which telcos rely in their digital transformation.

2.3 Virtualization and Cloud Computing

Generally speaking, data centers offer the possibility to residential as well as business customers to run applications by reserving computing and storage (CPU,RAM and Disk storage). However, customers workloads have a dynamic aspect. In fact, resource requirements vary continuously over the time, or are one resource centric which means that they need one resource more than the others (CPU

intensive or memory intensive). In such cases, most of the resources in the physical infrastructure of data centers are vastly underused especially if applications are hosted on dedicated servers which is the case of traditional data centers.

2.3.1 Virtualization As a Paradigm

The revolution of the virtual technology has made possible the virtualization of physical infrastructure in data centers enabling more efficient resource utilization by pooling resources of storage, networking and computing from several formerly siloed data centers to create a central, flexible pool of resources that could be reallocated based on needs [13].

Virtualization allows multiple instances of different Operating Systems (OS) such as Linux or windows to run simultaneously on a single physical host. Each OS is running on a virtual machine and operates as if it was dedicated to a physical computer. The guest OS accesses the hardware architecture underlying via a lightweight system called Hypervisor kernel. The hypervisor acts as a referee between the guest systems; it time-slices the physical processors and resources to each VMs and ensures confinement of guests in their own space [14]. The virtualization technology is a key enabler of cloud computing where applications are not hosted on dedicated servers anymore, but instead in a number of running virtual machines (VMs) and sharing physical resources of the physical machines (PMs) behind [15].

Figure 2.1 shows the server virtualization architecture with the two components core,notably the hypervisor and the VM.

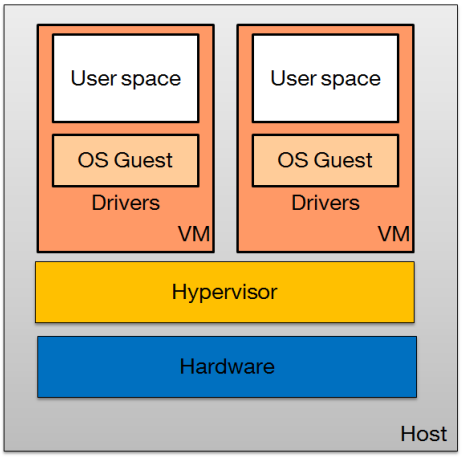

FIGURE 2.1: Server Virtualization architecture.

- **Hypervisor** An hypervisor or VM monitor is a system virtualization software that is running within the operating system of the host. It recreates, through software, a complete runtime environment for a program or a guest system called virtual machine. The hypervisor manages and multiplexes access to the physical resources maintaining isolation between all guests at all time. All guest operations are intercepted and translated to be executed by the host environment, which consumes hardware resources.

- **Virtual machine** The VM is the component core of the virtualization architecture. It can be defined as a software implementation of a computer that runs an operating system and executes programs just like a physical computer. Virtual machine or guest uses virtual hardware resources offered by the physical machine : virtual CPU, memory, hard disk and network interface cards. Guest Operating system sees ordinary hardware devices and is not aware that these devices are virtual.

Virtualization technology is a key enabler of Network Function Virtualization that reveals practical insights into VNFs deployments. Several VNFs could for example run on the top of the guest OS of one physical server. In the context of the use case Cloud RAN, the virtualization intervenes at the BBU level. Figure 2.2 illustrates BBUs running on virtual machines on the top of COTS server. It is clear that virtualization has many benefits, including reduced communication time between BBUs and scalability since that VMs are much easier to turn off or up than PMs [16].

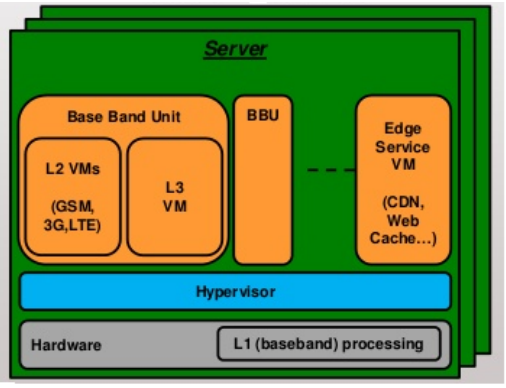

FIGURE 2.2: Server architecture for BBUs Virtualization [1].

2.3.2 Resource virtualization

2.3.2.1 CPU virtualization

When many VMs are running on the same host, hypervisor time-slices the physical processors across all of them so each VM runs as if it has its own number of virtual processors. This feature allows to run different OS in the same server.

2.3.2.2 Memory virtualization

When a VM starts to run, virtualization layer creates a contiguous addressable memory space as the allocated memory for the VM. This allows the hypervisor to run multiple VMs simultaneously while protecting memory of each VM from being accessed by others.

2.3.2.3 Virtual networking

The key virtual components in the virtual networking are virtual Ethernet adapters and virtual switches :

- A virtual machine can be configured with one or more virtual Ethernet adapters.
- A host can contain multiple virtual switches which act just like any physical Ethernet switch and forwards frames at the data link layer. The virtual switch enables VMs on the same host to communicate with each other using the same protocols used over physical switches without the need for additional hardware. It also connects to the external network through physical Ethernet adapters. The virtual switch is capable of binding multiple virtual network cards together, offering greater availability and bandwidth to the VM.

2.3.3 Resource overcommitment

Resource overcommitment is a process ensured by the hypervisor which enables the allocation of more resource for VMs than the host physically has. For example, a host with 6GB of memory available can runs 5 VMs with 2GB for each one, so we can say that the host's machine memory is overcommited. This is

possible because hypervisor is aware of the resource utilization of each VM which mostly don't use its full allocated memory. This comes with benefits especially for platforms developed for cloud computing where latency can be tolerated [17].

However, for latency-sensitive applications such as network functions, overcommitment may affect performance of running VNFs.

Among best practices for performance of NFV recommended by VMWare, overcommitment has to be disabled [18].

2.3.4 Benefits of Virtualization

Virtualization techniques provide several advantages for cloud computing [19], notably:

2.3.4.1 Resource sharing

The major advantage of virtualization is resource sharing. Physical resources of hosts are sharing among the virtual machines. Thus, running on the same host, for each virtual machine is allocated a portion of it's physical resources: CPUs, physical network cards, disk controller and a region of memory. This virtualization feature improves the resource utilization of servers counter to the traditional approach which dedicates a server for each application.

Resource sharing can reduce communication time between VNF sub-functions by deploying them in multiple VMs on top of the same physical host.

2.3.4.2 Server Consolidation

In traditional data centers, each application runs on a dedicated server,so most of the resources in the physical infrastructure are vastly underused. But with virtualization, multiple applications can run on a single server, which reduces the number of servers.

Consequently, energy consumption is significantly reduced which is very advantageous for Network operators looking to save network energy and to build green network which became a critical requirement with the expanding of the network size **[20].**

2.3.4.3 Reducing costs

For the cloud provider, hardware is always the most expensive component in the datacenter. Because virtualization reduces the number of servers, cost also goes down.

As a matter of fact, Capex and Opex also go down for Network Operators which is line with NFV goals.

2.3.4.4 Agility

One of the greatest advantage offered by virtualization and not available on traditional data centers is live migration which enables to easily move a virtual machine from one host to another for several needs. This feature makes the infrastructure more agile and improves the resource utilization in the data center.

Virtualization offers great opportunities for telcos by ensuring agility, the big promise of NFV.

2.3.4.5 Faster backup

Thanks to what we call a snapshot of a virtual machine which can provide informations about the state of several machines at a single point in time, full backup of what is running in the infrastructure can be created easily.

This feature eases the monitoring for the infrastructure management which is considered as a critical requirement for NFV [21].

2.3.5 Cloud Management Platforms

In order to extend the server virtualization environment with an Infrastructure-as-a-Service (IaaS) cloud, there is a need for cloud management platforms. These deployment platforms are in charge of setting up virtual infrastructure and the virtual management on top of that infrastructure.

There are many cloud deployment solutions that include all many components necessary for the the management and the control of a virtualized infrastructure: deployment, resource orchestration and application monitoring. Goals of these platforms are the same, but architecture and strategies differ from a solution to another.

2.3.5.1 OpenNebula

OpenNebula is an open-source Cloud Management tool that enables several features like security, virtualization, storage, and network solutions deployed in the data center. These features also facilitate its integration with any product and service in the cloud ecosystem, and management tool in the data center. OpenNebula provides an abstraction layer independent from underlying services to support all these features.

2.3.5.2 Eucalyptus

Eucalyptus is an open-source software for deploying private and public cloud. Due to its distributed architecture, Eucalyptus is scalable and compatible with other technologies like Amazon Web Services (AWS). The architecture of this software is made out of 6 components grouped into 3 layers:

- The first layer consists of the cloud controller and the storage service Walrus. The cloud controller offers a web interface which is the management tool that performs the resource scheduling. The Walrus component manages storage for all VMs in the cloud environment.
- The second layer consists of the cluster controller, the storage controller, and the VMWARE broker. The cluster controller acts as an intermediate communicator between a cluster and the other controllers. It handles the execution of the VMs. The storage controller manages storage within a cluster. Furthermore, the VMWARE broker transforms Eucalyptus images to VMWARE disks allowing an AWS compatible interface for VMWARE environments.
- The last layer holds the node controller that in turn hosts the VMs. It acts as a networking manager as well.

2.3.5.3 Openstack

"Network Functions Virtualization (NFV) is now synonymous with OpenStack. When people say NFV, there is an implication that they are talking about OpenStack."[1]

[1] "Dimensioning OpenStack Neutron for NFV Systems", Mark Lambe, SDx Central, September 2014.

Both ETSI and the Linux Foundation collaboration project, namely OPNFV, have defined reference platforms for NFV that include Openstack as the Virtualization Infrastructure Manager (VIM). This motivates our choice of Openstack as a benchmark in this work.

An architectural as well as conceptual overview of this platform was presented in [22] where authors enumerated the benefits of this platform. Openstack[2] is a set of open-source software for cloud management. This platform is based on a modular architecture composed of several separate projects where each project implements necessary functions needed to build an IaaS cloud. Multiple services that Openstack components offer are reachable through API-requests exposed as RESTFUL web services. Thus, Openstack is the most commonly used solution in data centers because of the flexible and scalable properties it offers.

Architecture : The different components of the Openstack architecture are:

- **Dashboard(Horizon)**: This is the web portal serving to control the virtualized infrastructure. It allows users to manage VM instances and related resources. Horizon is a web-based graphical interface where users specify their needs in terms of VMs by communicating through Openstack API.

- **Identity(Keystone)**: This is the manager of authentication and access rights. Keystone provides a token-based authentication and high level authorization. Users specify login and password in order to get a valid token that allows them to fetch a specific resource for a time period.

- **Image service(Glance)**: Provides services for discovering, registering and retrieving VM images. Glance stores images that could be used as template to launch instances.

- **Compute(Nova)**: Nova compute is the core component of the Openstack project. It is the manager of instances that is to say the creation, modification or removal of VMs. To ensure this function, NOVA is based on three tools:

[2]https://docs.openstack.org/pike/

1. *NOVA- API*: It supports API calls from the user. It initiates the boot of VMs and verifies that certain rules are respected.

2. *NOVA- COMPUTE*: It runs on the host servers and manages the life-cycle of VMs via the API hypervisor.

3. *NOVA- SCHEDULER*: It receives the VMs requests creation from the queue and determines on which machine host the VM will be placed.

- **Object storage(Swift)**: Swift contains cluster of servers to store large amount of data. Every time a new node is added, the cluster scales horizontally and in case of a node failure, Openstack works by moving content to other active nodes.

- **Block storage(Cinder)**: Cinder's feature are to create more volume for images.

- **Networking(Neutron)**: Provides Network As A Service between device interfaces managed by other Openstack services. Neutron relies on keystone for authentication and authorization for all API requests. When creating a new instance, nova-compute communicates with the neutron API to plug each virtual network interface card (NIC) on the instance into a particular neutron network through the use of a virtual switch: Open-Vswitch.

- **Orchestration(Heat)**: This service implements an orchestration engine for managing the entire lifecycle of resources within the infrastructure.

Basic Request flow in OpenStack: Figure **??** illustrates the request flow when creating a VM in OpenStack. The required steps are described below:

1. The Dashboard is the access point of the request flow, it gets the credentials of the connected user and, through a REST call, it gets an authentication token from Keystone.

2. Horizon uses the token in order to connect to Nova.

3. Nova interacts with the database in order to store the state of the request.

4. The VM creation request is sent to the message queue and the status of the request is updated.
5. Nova uses a two-steps algorithms to find the best compute node to host the instance.
6. The compute node asks Nova for information to prepare the creation of the VM.
7. The compute node fetches the image to start the instance by interacting with Glance.
8. Glance fetches the image stored in Swift and transmits it to the compute node.
9. The node asks neutron to create the virtual interfaces of the instance and establish the connectivity.
10. If the user wants a persistent storage, the node will ask Cinder to attach a storage volume to the instance.
11. The compute node interconnects all previous informations, transmits it to the hypervisor and start the instance.

2.4 Centralized vs Distributed Cloud Infrastructure

This section analyses cloud computing systems from a design perspective. A comparison between centralized and distributed cloud data centers topologies has been made in [23]. Based on locations, authors have distinguished two types of data center topologies:

- Centralized cloud data center topology defined as a topology with one big datacenter to service all clients requests around the globe [24].
- Distributed cloud infrastructure defined as multiple small data centers spread across a large geographical area and interconnected with high capacity WAN leased lines [25].

Hence, in distributed cloud systems, requests can be serviced from locations closet to them. Being closer to user has many advantages; this reduces the need

for network capacity especially for high bandwidth applications and improves access latency.

Extend datacenters across geographical locations became common nowadays for cloud service providers. For example, Google[3] has invested significantly in constructing largescale datacenters across the world, to host their services. The company has deployed at least 15 datacenters across 3 continents. Another example is VSP.NET, a leading cloud services provider that is offering today cloud hosting to over $10,000$ clients, in more than 180 different countries with 22 deployed datacenters across 5 continents[4].

Amazon Web Service (AWS) uses at least 13 locations all over the world with 35 availability zone spread over 13 locations all over the world.[5]

This will very likely be the case of network operators who plan to deploy data centers at the points of presence of their network. Indeed, depending on the geographical span of a network, several hundreds of PoPs could be deployed by a medium-size network operator.

2.5 Related Work on Resource Allocation Strategies

Resource allocation is one of the most challenging problem in virtualized infrastructure management. In the recent few years, many research works have addressed this problem considering the large number of possible optimization criteria and different formulations that could be studied.

2.5.1 Problem Statement

It is an important decision where to allocate resources, in other words, where to place required virtual machines of user request in cloud computing system. Several works in the existing literature have addressed virtual machine placement issue.

In this operation, first a set of virtual machines is given with some resource requirements like CPU, memory, disk, etc. There are two keys approaches for defining VM requirements based on taking provider-defined and user-defined views:

[3]https://www.google.com/about/datacenters/inside/locations/index.html
[4]https://www.vps.net/cloud-datacenter-locations
[5]https://aws.amazon.com/fr/about-aws/global-infrastructure/

- In the provider-defined view, cloud providers like Amazon EC2 predefine limited number of types of VM configuration of resources requirements for cloud users[6].
- In the user-defined view, cloud providers such as IC cloud allow their cloud users to define VM configuration based on their needs [26].

Then, a set of physical machines (PM) is given with different resources capacities (CPU, memory and disk) on which those VMs are to be placed [27].
The placement process is to decide on which PM to place the different VMs based on one of two goals :

- Load balancing: Placing the new VM in such a manner that it helps in load balancing of resource utilization within the PM.
- Server Consolidation: Placing the new VM such that it helps in server consolidation and that means favoring the PM with high load.

Basically, the problem of VM placement can be illustrated as in Figure 2.3 where we have a set of virtual machines arriving at the scheduler that must make the decision of where to fill the placement among 2 PMs.

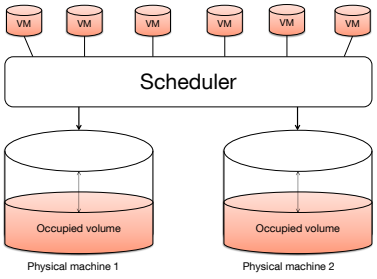

FIGURE 2.3: Virtual machine placement.

Due to the multidimensional characteristics of both VMs and PMs, the mapping VM-to-PM is an NP-hard problem which has been extensively studied in cloud computing literature [28]. The problem is traditionally described as a Bin Packing Problem where different size items are to be packed in bins of fixed size. Although, here the problem translates into a Vector Packing Problem where PMs are bins and VMs are items to be placed. Dimensions of vectors are the number of resource required like CPU and memory.

In the following, we review algorithms for virtual machine placement.

[6]https://aws.amazon.com/fr/ec2/instance-types/

2.5.2 Resource Allocation Algorithms

Several works have addressed virtual machine placement problem, but most methods focus on resource allocation mechanisms in centralized cloud platforms. In this approach, the placement is fulfilled in centralized fashion and the possible traffic that can be considered is only between VMs inside the same data center. In such a context, a popular approach is to adopt an optimization formulation: Given a demand for resources in terms of storage and computing, the problem is to find the optimal request placement. This leads in general to Mixed Integer Linear Programming (MILP) problems.

2.5.2.1 MILP

The problem is classically posed as a Mixed Integer Linear Programming problem where objective function aims to reduce the number of physical hosts used or costs due to different placement, hosts capacities with other restriction like power consumption are specified as constraints.

Authors in [29] defined two constraints that need to be satisfied during the placement process:

- **Capacity constraint**: For each dimension of a given PM, the sum of the resource requirements of all VMs placed on it should be less than or equal to the total available capacity.
- **Placement Guarantee constraint**: All VMs should be placed.
- **Objective Function**: Aims to minimize the number of PMs.

Using a linear programming tool, a solution of VM-to-PM mapping is generated. Authors conducted some experiments which revealed that the time required to generate a placement plan for 20 VMs reached more than 8.5 hours which is undesirable in a practical scenario.

To achieve optimal resource placement, an exact formulation that aims at finding the best placement of resources by maximizing the revenue and minimizing the corresponding costs is proposed in [30]. The authors have also noted that the ILP formulation suffers from scalability problems.

A similar approach was adopted in [31] to resolve the VNF orchestration problem by providing a MILP formulation. Although the authors have proved that the

time needed to evaluate the model increases more or less linearly with the number of service to be placed, this approach requires all requests to be known in advance and seems to be unable to manage rapid fluctuations of demand.

Thus, although the MILP formulation gives an exact placement plan, it does not scale well with the increase in the number of virtual or physical machines. Optimization approaches can be very time consuming, several works addressing resource allocation in cloud systems proposed alternative approaches providing solutions in more reasonable time.

2.5.2.2 Heuristic-based approaches

To reduce time complexity, several heuristics are introduced to give placement plan that is close to the optimal solution.

2.5.2.2.1 FFD-based heuristics The typical approach for one-dimensional bin packing problem is FFD (First Fit Decreasing). The principle is to order the items and the bins by size. Starting with the first bin, it iterates over the items placing any item it can into it. Once the first bin is filled, it proceeds to the second bin from the ordered list, repeating the same process [32].

If the problem is constrained by a single resource CPU for example, then FFD can be applied where the size of items is the number of required Cores by the VM and the size of bins is the remaining Cores in the PM. But, generally the placement problem is constrained by more than a single resource (CPU, memory, disk). Thus, a generalization of the FFD is needed. Classically, to generalize FFD to a multidimensional scenario,the multidimensional vector of PM capacities is mapping into a single scalar called metric, then FFD for single resource can be performed based in this metric. In existing literature, this approach has been well explored but it stills not clear what formula to use to generate this metric so that one-dimensional FFD can be performed. In what follows, we will investigate some of existing methods in literature that calculate this scalar so-called metric for FFD multidimensional problem .

a) Weighted sum : OpenStack scheduler [7]

[7] https://docs.openstack.org/nova/pike/user/filter-scheduler.html

The main role of the OpenStack scheduler called Nova-scheduler is to make decision on where a new instance should be created according to its resources requirements. The algorithm of nova scheduler only consider CPU speed, memory capacity and hard drive capacity when scheduling an instance. During its work, the scheduler iterates over all physical servers evaluating them with a metric called score calculated thanks to a two-phases algorithm. Based on this score, the one-dimensional FFD can then be performed [33]. First, a filtering process is used to determine which hosts are eligible for consideration and an eligible host list is created. Many filters are available for this end so that users can specify which filters to use when performing the scheduling. Some of available filters:

- **RAM filter**: Ensures that only nodes with sufficient RAM are selected for the eligible hots list. If the RAM is not used, the nova scheduler may over-provision a node with insufficient RAM resources. By default, the filter is set to allow over-commitment on RAM by 50 percent.
- **Core filter**: Ensures that only nodes with sufficient CPU cores are chosen for the eligible host list. If the core filter is not used, the nova scheduler may over-provision a node with insufficient physical cores. BY default the filter is set to allow over-commitment based on a ratio of 16 virtual cores to one physical core.

Then, a second process is applied against the list to determine which host is optimal for fulfilling the request. It applies one or more cost function to get numerical score for each host. The score is a way to select the suitable host and is calculated this way:

$$score_{host} = (w_1^{multiplier} * norm(w_1)) + (w_2^{multiplier} * norm(w_2)) + ... + (w_n^{multiplier} * norm(w_n))$$

where w_n is the normalized value of the amount of the resource n, and $w_n^{multiplier}$ is the weight associated to it. Multipliers can be negative or positive. If multipliers are negative then we are favoring the host with largest available resource. With the nova scheduler, client can make its own filters and specify with which type of resource calculate the score. Thus, we can say that this scheduler is flexible and can be adapted to the user needs. But, this single metric calculated for different resources is not aware about the availability of each type of resource. We can imagine a very loaded host in terms of memory capacity will be chosen for the placement because of its large number of available cores. Hence, we can

say that resources all over the cluster are non used in a balanced manner, and even after this placement, the situation will be worst.

b) Volume approach : Sandpiper

Another metric, referred to as $sand_{volume}$ is defined in [34] as:

$$sand_{volume} = \frac{1}{1 - w_{cpu}} . \frac{1}{1 - w_{mem}} . \frac{1}{1 - w_{net}},$$

$$\phi_t = \alpha.n + (1 - \alpha).\phi_{t-1}, \alpha = 0.14$$

$$C_v = \frac{\sigma}{\bar{\rho}},$$

w_{res} represents the corresponding normalized utilization ratio of the resource res. Let's consider a $3D$ unit cube where each dimension represents the normalized capacity of each type of resource offered by the PM. The total capacity of the server is then obtained with the volume of the unit cube, this metric provides informations about the exploitable volume of a PM.

If the problem were constrained by one dimension, then comparing PMs by using the above metric is the natural choice for the FFD algorithm since its value is inversely proportional to the amount of available resources. However, in the case of multidimensional resource allocation, this algorithm may have some shortcomings. In fact, the volume does not completely reflect the amount of resources on each dimension.

As an illustration, consider two PMs with three types of resources (RAM, CPU and Disk). Respective normalized utilization ratio as illustrated in Figure 2.4 :

- $PM1$ has $(0.4CPU, 0.45RAM, 0.5Disk)$
- $PM2$ has $(0.2CPU, 0.7RAM, 0.2Disk)$

The $sand_{volume}$ metrics of the two PMS are :

- $sand_{volume}$(PM1) = 6.06
- $sand_{volume}$(PM2) = 5.20

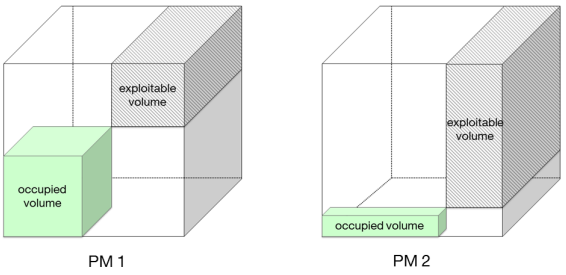

FIGURE 2.4: Server state before the placement

With sandpiper approach, $PM2$ which has the highest exploitable volume (the lowest metric) will be chosen as a target for placing any VM request regardless of its resources requirements.

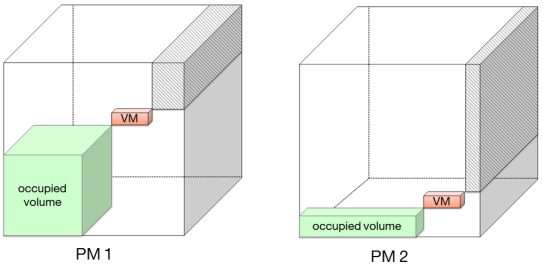

FIGURE 2.5: Server state after the placement

If we consider a VM requiring $(0.1CPU, 0.2RAM, 0.1Disk)$, the remaining capacities of both PM after the placement of this VM are illustrated in Figure 2.5 and given by :

- $PM1 = (0.5CPU, 0.35RAM, 0.4Disk)$
- $PM2 = (0.7CPU, 0.1RAM, 0.7Disk)$

It is clear as illustrated in Figure 2.5 that after the placement of the VM on $PM2$ which is recommended by the sandpiper approach, resources of $PM2$ are not utilized in a balanced manner and the exploitable volume left is less than that when it is placed on $PM1$. This may lead to blocking of further requests.

2.5.2.2.2 Multidimensional aware heuristics We just proved that information about resource utilization may be lost with a single metric. Another set of heuristics was proposed in the existing literature to achieve a better host

utilization. These heuristics are called multidimensional-aware heuristics. The basic idea is to represent various resources as a d-dimensional vector. As each VM demand is a combination of different type of resources for example CPU, memory and disk, it is represented by a 3-dimensional vector where each dimension represents a single type of resources. Let' s call this vector $\vec{R}i$ which is the

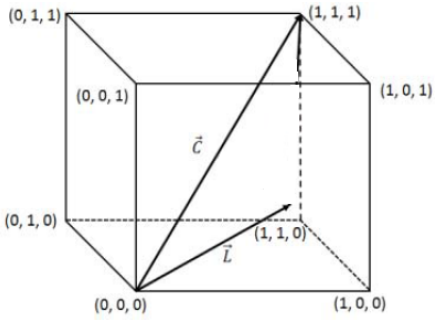

FIGURE 2.6: Normalized Resource Cube

vector of normalized resources required by the VM_i.
The same representation is adopted to represent the PM resources capacities. A 3-dimensional vector $\vec{C_j}$ of normalized resources capacities is generated for each PM_j. Finally, resource utilization of a PM_j is also represented by a three-dimensional resources vector $\vec{L_j}$. Thus, all resources are normalized and all information are defined as vectors. The typical approach proposed in many related works [35][36] consists on representing all these information vectors within a unit cube called Normalized Resource Cube (NRC) as shown in Figure 2.6.

a) Imbalance heuristic

Imbalance is a metric that calculates how much the utilization of resources of a single PM is imbalanced. This metric is calculating by defining a new vector $\vec{I}$ which is the vector difference between the projection of the resource utilization vector $\vec{L}$ and on the resource capacities vector $\vec{C}$ and $\vec{L}$.

If $\vec{C}$ of a PM is exactly aligned with the imbalance vector $\vec{I}$, thus we can say that resources of this PM are utilized in a balanced manner.
Admit that the resource capacities vector $\vec{C}$ is defined by $\vec{C} = \vec{i} + \vec{j} + \vec{k}$ where $\vec{i}, \vec{j}, \vec{k}$ are the unit vectors along the three resources CPU, memory and disk.
The resource utilization of a PM is given by : $\vec{L} = c * \vec{i} + m * \vec{j} + d * \vec{k}$ where c,m and d are the normalized values of CPU, memory and disk utilized in this PM. Hence, the projection of the utilization vector $\vec{L}$ on the principal diagonal of the NRC is given by:

$(\frac{1}{\sqrt{3}}c+\frac{1}{\sqrt{3}}m+\frac{1}{\sqrt{3}}d)*(\frac{1}{\sqrt{3}}*\vec{i}+\frac{1}{\sqrt{3}}*\vec{j}+\frac{1}{\sqrt{3}}*\vec{k}) = \frac{c+m+d}{3}*\vec{i}+\frac{c+m+d}{3}*\vec{j}+\frac{c+m+d}{3}*\vec{k}$

Therefore, imbalance vector $\vec{I}$ is given by:
$\vec{I} = (c-\frac{c+m+d}{3})*\vec{i}+(m-\frac{c+m+d}{3})*\vec{j}+(d-\frac{c+m+d}{3})*\vec{k}$ The $\vec{I}$ vector appears in the NRC like shown in Figure 2.7. Every time we have a VM to place, we simulate a

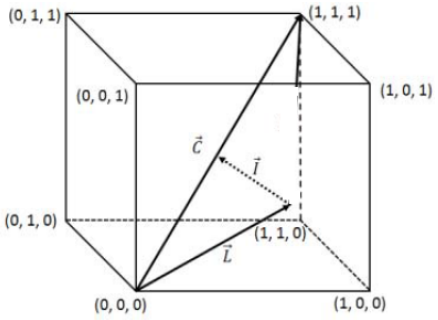

FIGURE 2.7: Imbalance degree vector

placement on each PM in order to calculate the new values of imbalance degree of each server and choose the server which minimizes this value.

Although this approach takes into account the multidimensional aspect of the problem, it does not consider the resource requirements of the VM when scheduling an instance, it just considers resource utilization of the PM regardless to what it is required by the instance.

To rectify this, authors in [35], used the imbalance heuristic combined with another approach that takes into account the complementary aspect when placing a VM. The idea is to find a VM which is more utilized in CPU compared to memory to place in a PM that has more resource utilization along the memory axis the imbalanced degree is minimized. To achieve this goal, PMs are grouping into $d!$ groups according to the resource utilization if each PM and where d is the dimension of the problem.

Let's illustrate this approach with an example of a 3-dimensional problem with 3 types of resources(CPU, memory and disk); we then have $3! = 6$ groups. For the group CMD, the CPU is the most utilized resource and the disk is the least utilized one. The memory is between both($CPU \geq Memory \geq Disk$). Its complementary group in terms of resource utilization will be DMC($Disk \geq Memory \geq CPU$).

When scheduling a VM, we identify into each group it will fall. Hence, the target PM will be the one that minimize the degree of imbalance the most from the complementary group for the VM. The mechanism is illustrated in figure 2.8.

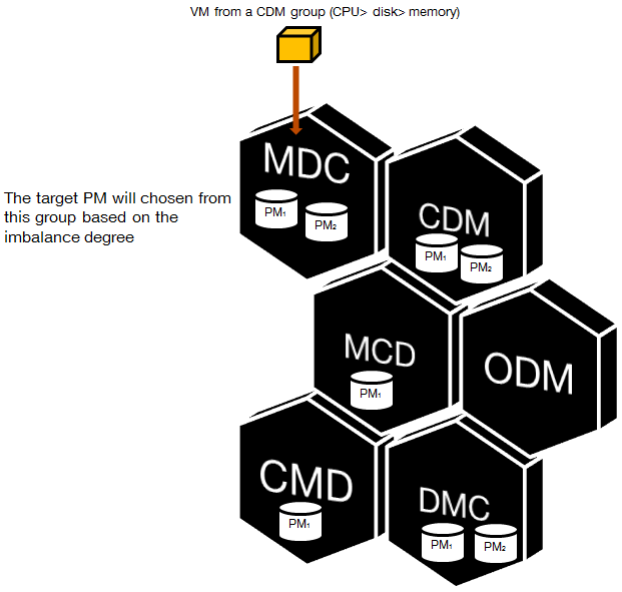

FIGURE 2.8: Grouping PMs for scheduling

b) Dot-product heuristic

In this approach like adopted in [36], we take the dot product of resource requirements vector $\vec{R}$ of the VM to be placed and the resource utilization vector $\vec{L}$ of each PM. The PM who gives lower dot product is chosen. The idea behind is to place the VM in a complementary PM. The intuition is that a small dot product means a large angle between the VM vector and the resource utilization vector of the PM, and a large angle means that VM and PM are complementary. In 2-dimensional space and as illustrated in Figure 2.9, it is clear that the lower dot product is given by PM1 which is a better choice to balance the resource utilization where the VM to place is asking for more CPU than RAM and PM1 has more resource utilization in terms of RAM than CPU.

The dot approach recommends to choose $PM1$ as a target. It is clear that $PM1$ is the better choice because in this case, both of PMs have almost the same utilization vector $\vec{L}$. Hence, the dot product value gives a correct information about the angle. In fact, as the dot-product is defined by :
$\|\vec{R}\|*\|\vec{L}\|$ * $\cos(\vec{R}, \vec{L})$ and this will be calculated for each PM_i, the length of $\vec{L}$ plays a role in deciding the target PM. Thus, this approach does not really take into account the angle by calculating the dot product.

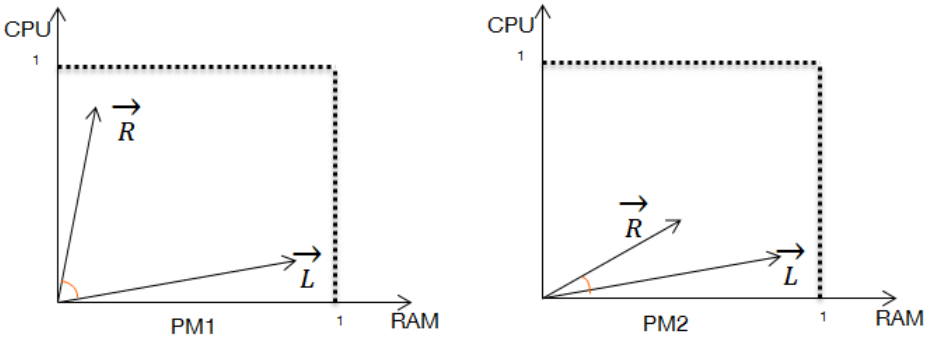

FIGURE 2.9: Dot product approach

2.6 Testbed Platform

Generally speaking, the orchestrator, also called the scheduler is the entity in charge of taking placement decision. To this end, the orchestrator has to maintain an updated view of the underlying infrastructure, notably the amount of available resources.

In the following, we describe some experimentations that we did in a real cases scenarios to validate a major hypothesis of this work, notably that the orchestrator can have a global vision and be aware about the resource occupation in the infrastructure.

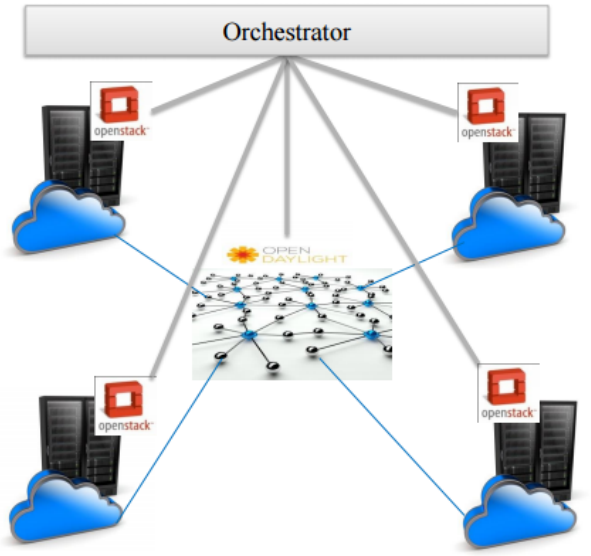

FIGURE 2.10: Integrated IT and Network test-bed Platform.

We present in this section an integrated IT and network test-bed platform. This platform consists on :

1. A distributed infrastructure composed of 4 data centers represented by 4 deployments of different instances of Openstack. We used the 11th release of the open source software, namely the Openstack Kilo[8]. For the end-to-end inter data centers connectivity, we use an SDN network emulated by the Mininet Emulator. On any laptop, Mininet can create a realistic virtual network based on the Openflow protocol [37]. As the SDN controller, we choose the Opendaylight Controller. Opendaylight is a platform ensuring the network programmability and the interaction with the network resources through the APIs that it exposes[9].

2. A bottom layer for the orchestration that is interacting with the openstack and the opendaylight controller. To this end, we implement an orchestration platform consisting on several python scripts that consumes the APIs exposed by opnetsack as well as opendaylight through http request calls.

The considered platform is illustrated in Figure 2.10.

Through interaction with infrastructure controllers, the Orchestrator is able to collect information and keep a global vision of the state of the infrastructure. Thanks to these relevant information, the orchestrator can be aware of the state of the infrastructure when allocating resources.

As a proof of concept, the orchestrator in our implementation rely on Openstack and Opendaylight to collect informations and to have a global view of the infrastructure by performing some actions. These actions are described in Figure 2.11.

2.6.1 Network Topology Discovery

The orchestrator fetches first informations about the network topology by calling the function `getTopology` which interacts with the topology discovery module within the network controller Opendaylight. OpenDaylight Controller uses the LLDP messages to discover the topology of the connected OpenFlow Devices and provides a view of the physical network topology. An example of informations provided about a 3 nodes topology is illustrated in Figure 2.12. This topology under consideration in this example is composed of:

[8] https://www.openstack.org/software/kilo/
[9] https://www.opendaylight.org/

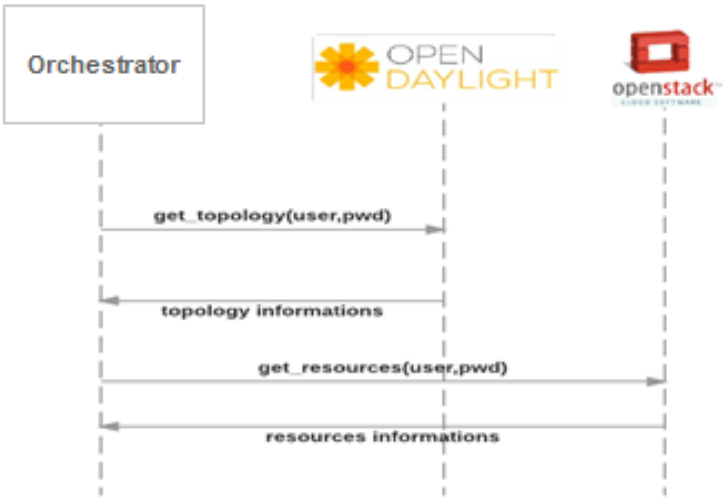

FIGURE 2.11: Network topology informations.

```
zkth9555@l-zkth9555:~/scripts$ sudo python ./topology.py
== topology ==
== node ==
host:8a:f4:a5:ba:4d:ea
termination-point:
host:8a:f4:a5:ba:4d:ea
== node ==
openflow:1
termination-point:
openflow:1:LOCAL
openflow:1:1
openflow:1:2
== node ==
host:7e:7b:44:3b:89:5e
termination-point:
host:7e:7b:44:3b:89:5e
==link==
openflow:1:2/host:7e:7b:44:3b:89:5e
destination:
host:7e:7b:44:3b:89:5e
source:
openflow:1
==link==
host:7e:7b:44:3b:89:5e/openflow:1:2
destination:
openflow:1
source:
host:7e:7b:44:3b:89:5e
==link==
openflow:1:1/host:8a:f4:a5:ba:4d:ea
destination:
host:8a:f4:a5:ba:4d:ea
source:
openflow:1
==link==
host:8a:f4:a5:ba:4d:ea/openflow:1:1
destination:
openflow:1
source:
host:8a:f4:a5:ba:4d:ea
zkth9555@l-zkth9555:~/scripts$
```

FIGURE 2.12: Network topology informations.

1. 3 nodes identified by physical addresses which represents an OpenFlow device as well as 2 data centers.

2. 4 logical links which represents 2 physical links connecting data centers to the virtual switch.

2.6.2 Resource Utilization Level Collector

With regard to the resource utilization level in the data centers, the orchestrator interacts with the openstack instances managing the cloud infrastructure to collect informations by calling the function `getResources`. Openstack provides informations about the amount of the used resources in the data centers. These provided informations are about the several types of resources, notably the memory, compute and storage resources. An example of these informations of a specific data center returned by openstack to the controller is illustrated in the Figure 2.13.

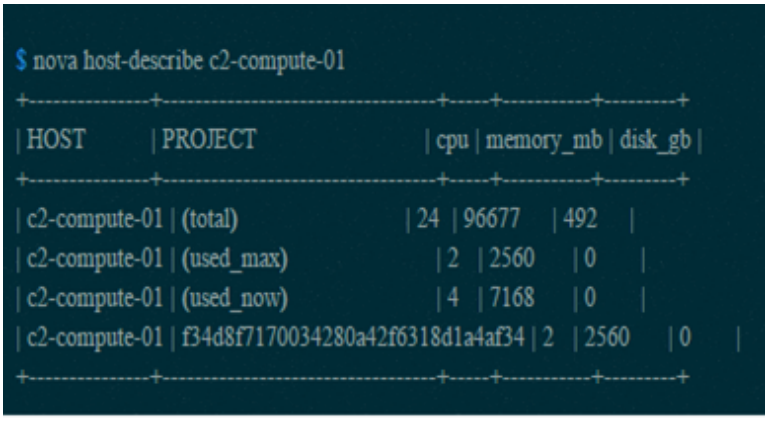

FIGURE 2.13: Resources utilization level.

Thanks to our test-bed platform and the orchestration platform implementation, we have shown that the orchestrator is able to have a global view of the underlying infrastructure comprising the network topology as well as the resource utilization level. We have proven that with the right monitoring tools, the orchestrator can collect all the necessary informations and metrics. These informations can ameliorate the scheduling decisions taken by the orchestrator.

2.7 Summary

Given the transformation of network towards virtualization, cloud computing and resource allocation continue to draw immense attention from researchers in both industry and academia, and from cloud as well as network fields.

This chapter discussed first motivations driving network operators to accelerate the transformation towards NFV. Second, we focused on virtualization aspects and it included an overview of resource allocation in cloud computing, notably concerning placement algorithms.

Chapter 3

Performance Evaluation Of Resource Allocation Algorithms

Contents

3.1 Introduction

In recent years, cloud and virtualization have enabled the emergence of new applications and services by significantly reducing costs and, above all, providing unprecedented agility. These advantages contributed to the convergence between Information Technology and telecommunications networks with notably the development of NFV, which clearly makes it possible to overcome the ossification of traditional networking techniques based on hardware [38].

The virtualization of network functions is certainly a major revolution that will impact not only the architecture of networks but also the Network Operators (NO) infrastructures. In fact, to meet stringent real-time constraints of some VNFs [39], some network functions have to be hosted close to end users (e.g. Radio Access Network (RAN) functions, firewalls, deep packet inspection). This has led to the development of geographically distributed mini data center, also referred to as cloudlets [40], at the edge of the network (i.e. typically at Points of Presence (PoPs) level). See for instance [41] where authors describe Edge Network examples.

Capacities in terms of storage, compute and networking resources provided by edge data centers are considered as infinite when compared to the infinite capacity assumption of huge centralized data centers deployed for instance by Google.

Therefore, the network operator infrastructure is evolving towards a distributed architecture of edge data centers with limited capacities deployed close to the end users.

All these radical changes in network operators' infrastructures raise new issues, specifically in terms of resource allocation, which have so far not been considered in the cloud literature. Traditionally, resources in cloud platforms are considered as to be infinite and request blocking is most of the time ignored when evaluating resources' allocation algorithms, because of this infinite capacity assumption [42]. However, if we assume that the NO's infrastructure will be very likely composed of small data centers with limited capacities, and deployed at the edge of network, the congestion of such a system may occur specifically if the demand is sufficiently high and exceeds what the infrastructure can handle at a given time.

Generally speaking, VNFs will be implemented in both big centralized data centers and smaller one distributed at network edge to improve response time. There is hence a clear need for analyzing blocking of requests for such infrastructure and for finding algorithms to allocate resources to requests.

In this Chapter, we evaluate resource allocation algorithms by paying special attention to request blocking which has so far not been considered in the cloud literature and presents a key metric in the context of telco cloud.

We start by giving some background and review existing work on blocking analysis in cloud literature in Section 3.2.

In section 3.3, we first propose an analytical model for the blocking analysis in a centralized cloud system, which was validated using discrete events' simulations. Second, we conducted a comparative analysis of the most popular placement's strategies. The proposed model, as well as the comparative study, reveal practical insights into the performance evaluation of resource allocation and capacity planning for distributed edge cloud with limited capacities.

In section 3.4, we investigate placement and offloading strategies of constrained services in distributed cloud. We set design principles of future distributed edge clouds in order to meet application requirements. We precisely introduce a costless distributed resource allocation algorithm, named *CLOSE*. We compare via simulations performances of *CLOSE* against those obtained by using mechanisms proposed in the literature, notably the Tricircle project within OpenStack.

Section 3.5 concludes this chapter.

3.2 Background

3.2.1 Virtual Network Function Placement

According to the NFV group specification [43], End-to-End (E2E) network services can be mapped into a forwarding graph composed of several VNFs.

Each VNF is composed of one or more virtual machines performing a set of specific functional tasks. Based on VNF requirements, the VNF descriptor is a package that describes a list of needed resources (e.g. storage, computation, etc.) mapped to VMs. This descriptor is considered as a template of a service specifying the resource infrastructure to allocate in order to instantiate the VNF [44].

The E2E network service placement consists of the instantiation of the set of VNFs corresponding to its forwarding graph in the different geographical locations. This can be expressed by allocating the resources required by virtual

machines in different points of presence from the underlying infrastructure. This constraint is referred to as the *Multi-Site* constraint.

Generally speaking, each VM requested to instantiate a VNF has its own combination of resource requirement (for example CPU, memory and disk) that can be represented by a multidimensional vector where each dimension represents the desired amount of a single type of resources. Group of VMs with the same vector requirements can be mapped into class. Let this constraint be referred to as the *Multi-Class* constraint apart from the *Multi-Dimension* one constraining the resource allocation problem in this context.

The shortage of one or more of the resource types required may cause the request to be blocked. However, VNFs has to perform as expected when the network service is requested. This gives rise to new issues for network operators, which now have to characterize service availability that may be affected by request blocking when allocating resources.

There is hence a clear need for finding an appropriate model considering both the *Multi-dimension* and the *Multi-class* constraints that can quantify blocking in cloud infrastructure where capacities are limited. This model can reveal practical insights into performance evaluation of resource allocation algorithms and capacity planning so that edge data centers constrained by finite capacities will be correctly dimensioned.

3.2.2 Analysis of Blocking in Cloud Computing

Performance evaluation of resource allocation strategies has been studied in the cloud literature, but only few works have addressed cloud performance problems in terms of blocking probability since cloud resources have always been considered to be infinite. See for instance [45] [46] [47] [48].

An analytical model to estimate blocking probability as well as the waiting time of requests was introduced in [49]. Although this model takes into account multiple critical cloud features such as batch arrival of requests, only the memory resource was considered in the model. Several blocking estimation models from the existing literature are subject to the same limitation; notably the single-resource case. See for instance [50] [51].

In [52], two policies of resource allocation to handle both data center and network resources were proposed. Based on those policies, authors evaluate several

allocation strategies in terms of blocking probability but no analytical model was proposed.

In [53], a topology-aware virtual machine placement is proposed. The algorithm proposed handling several types of arrival requests is then compared to two other algorithms based on blocking rate and energy consumption. However, the placement decision of the introduced policy is made only based on computational resource requirements. It is worth noting also that no theoretical modeling was introduced in this work.

In [54], a queuing model is employed to optimize resource allocation for multimedia cloud based on two metrics namely response time and resource allocation cost. However, blocking was not quantified and the analysis was limited to a single-resource case.

A general analytical model for evaluating task blocking probability in cloud computing system is proposed in [55]. The proposed model considers the concept of virtualization as well as heterogeneous server pools but this study is also limited to single-resource dimension.

Analysis of blocking in cloud data centers requires relevant modeling counting a large number of parameters such as several types of advent request at the cloud system and heterogeneous resources of the system. To the best of our knowledge, these features all together are not available in any of the existing models of blocking analysis in cloud systems.

3.3 Resource Allocation Algorithms In Centralized Cloud Platforms

We propose in this section an analytical model for the blocking analysis in a multidimensional centralized cloud system. We assume in the following that a request cannot be split in the sense that either a request can be hosted by a PM or else the request is rejected. Splitting requests offers an additional degree of freedom for the placement algorithm but induces additional traffic inside the data center. Contrary to previous studies on resource allocation in cloud platforms, we analyze in this thesis system performance in a probabilistic context and we pay special attention to blocking of requests.

3.3.1 Model Settings

We consider a data center composed of N servers. Each server comprises J types of resources (CPU, RAM , disk, bandwidth , etc.). We assume that all servers are identical. The capacity of a server is denoted by c_j for resource j. The data center capacity is then Nc_j in resource j.

The data center accommodates resource requests of K classes. The demand in resource j of a class k request is denoted by A_j^k. We assume that requests of class k arrive according to a Poisson process with rate λ_k. The mean holding time of resources by a request of class k is denoted by $1/\mu_k$. To simplify the analysis we assume that the nominal request A_j^k has integer values. Moreover, we assume that the greatest common divisor of A_j^k for $k = 1, \ldots, K$ and fixed j is equal to 1.

The load of the system in resource j is

$$\rho_j = \frac{1}{Nc_j} \sum_{k=1}^{K} \rho_k A_j^k,$$

where $\rho_k = \lambda_k/\mu_k$.

The model under consideration is illustrated in Figure 3.1

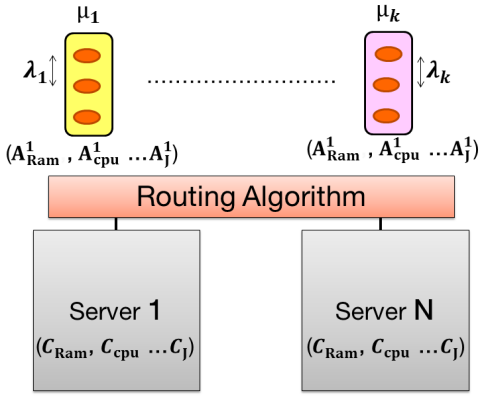

FIGURE 3.1: Model settings

In practice, upon the arrival of a request, the scheduler, aware of the occupation of different servers in the data center, forwards the request to one of the available servers that can accommodate the request. The selection of the server is made according to a given algorithm. If the requested amount of resources is not available in all servers, the request is then blocked.

Due to the fragmentation of the resources among servers, there is a potential loss of efficiency. For a given request, the claimed amount of resources may be globally available but since the resources are fragmented it may happen that none of the server can accommodate the request.

The routing algorithm may have a major impact on the performance of the system. To study the impact of the routing algorithm on the global blocking of the system, we consider that the N servers are grouped into a unique big server, since there is no loss. Then we analyze the blocking for this system in order to evaluate subsequently the efficiency of different routing algorithms by comparing global blocking rates.

3.3.2 Blocking in a grouped data center: One Big Server

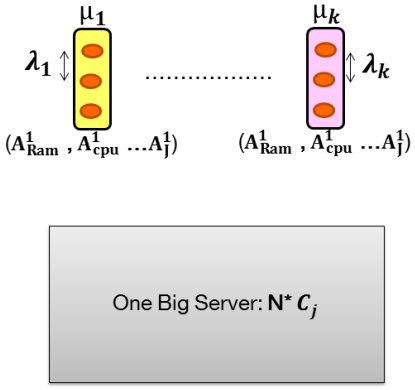

FIGURE 3.2: One Big Server Model settings

If we assume that the N servers can be grouped into a unique big Server, with capacity Nc_j for resource j as shown in Figure 3.2, then we obtain a classical blocking system with heterogeneous resources.

If we consider the system in equilibrium, let $\mathbf{n} = (n_1, \ldots, n_K)$ denotes the occupation of the server when there are n_k customers of class k in the system.

The resource constraints translate into

$$\sum_{k=1}^{K} n_k A_j^k \leq N c_j$$

for $j = 1, \ldots, J$.

The probability of being in state $\mathbf{n}$ is

$$\mathbb{P}(n) = \frac{1}{\mathcal{G}} \prod_{k=1}^{K} \frac{\rho_k^{n_k}}{n_k!},$$

where $\mathcal{G}$ is the normalizing constant given by

$$\mathcal{G} = \sum_{\mathbf{n} \in \mathcal{S}} \prod_{k=1}^{K} \frac{\rho_k^{n_k}}{n_k!},$$

the state space $\mathcal{S}$ being defined by

$$\mathcal{S} = \left\{ \mathbf{n} \in \mathbb{N}^K : \sum_{k=1}^{K} n_k A_j^k \leq N c_j,\ j = 1, \ldots, J \right\}.$$

The state space $\mathcal{S}$ is delimited by J hyper-planes. We easily note that if $A_k^j \leq A_k^{j'}$ and $C_j \geq C_{j'}$ then the condition $\sum_{k=1}^{K} n_n A_j^k$ is dummy. In the following we assume that this situation does not occur. Otherwise, we have to consider smaller number of resources $J' < J$ but the analysis is similar.

In the following we assume that N is large and we take N as a scaling factor. In other words, we replace λ_k by $N\lambda_k$ for the arrival rate of class k customers.

The classical asymptotic methods developed in the context of circuit-switched for large networks (see for instance [56]) is a way of looking at our model.,The aims of the study conducted in [56] is to estimate the blocking probability that an arriving request will not find enough bandwidth on a route to its destination.

On the basis of this study, we deduce the following estimates for the blocking probability β_k for class k customers under 3 different load conditions:

- **Underload conditions** If $\sum_{k=1} \rho_k A_j^k < c_j$, for large N

$$\beta_k = \frac{1}{\sqrt{2\pi N}} \sum_{j, A_j^k \neq 0} \frac{e^{-N.I_j}}{\sqrt{\Gamma_j}} \frac{1 - e^{-y_j A_j^k}}{1 - e^{1 - e^{-y_j}}} \left[1 + O\left(\frac{1}{\sqrt{N}} \right) \right]$$

with

$$\begin{aligned} I_j &= \sum_k \rho_k \left(1 - e^{-A_j^k y_j} \right) - C_j y_j, \\ \Gamma_j &= \sum_k \rho_k A_j^{k\,2} e^{-A_j^k y_j} \end{aligned}$$

and $y_j < 0$, $j = 1, \ldots, J$ being the solution to the linear system

$$\sum_{k=1}^{K} \rho_k A_j^k e^{-A_j^k y_j} = C_j, j = 1, \ldots, J; \tag{3.1}$$

- **Critical conditions:** If $\sum_{k=1} \rho_k A_j^k = c_j$,

$$\beta_k = \frac{1}{\sqrt{2\pi N}} \sum_{j, A_j^k \neq 0} \frac{A_j^k}{\sqrt{\Gamma_j}} \left[1 + O\left(\frac{1}{\sqrt{N}}\right)\right];$$

- **Overload conditions:** When $\sum_{k=1} \rho_k A_j^k > c_j$ and the linear system (3.1) has positive solutions,

$$\lim_{N \to \infty} \beta_k = 1 - \prod_{j, A_j^k \neq 0} e^{-y_j A_j^k}.$$

3.3.3 Numerical Validation of the proposed model

We propose in this section a quantitative evaluation of the blocking probability estimation obtained with our model. For comparison, results are provided via simulation where we consider 2 scenarios:

- a two-dimensional system (RAM and CPU resources) and two classes of requests (Mice and Elephant)
- a three-dimensional system (RAM, CPU and Disk resources) and two classes of requests (Mice and Elephant)

Two-dimensional system We consider in this scheme a two-dimensional system offering 2 types of resources (RAM and CPU) and two classes of requests. One class is composed of requests with small requirements in terms of CPU and RAM (referred to as mice). The requests of the second class (referred to as elephants) have high requirements in terms of CPU and RAM.

The arrival process of class 1 requests is assumed to be Poisson with rate λ_1. A class 1 customer requires $c_1 = 2$ units of CPU and $r_1 = 3$ units of RAM. The holding time of resources is assumed to be exponential with mean $1/\mu_1 = 1$. Similarly, class 2 requests arrive according to a Poisson process with rate λ_2, require $c_2 = 17$ units of CPU and $r_2 = 65$ units of RAM, and hold the resources for exponentially distributed duration with mean $1/\mu_2 = 1$.

We consider a data center with two identical servers with capacity C in terms of CPU and R in terms of RAM. We assume that the two servers are grouped into a unique big server with capacity $2R$ in terms of RAM and $2C$ in terms of CPU.

The load of the system is then

$$\rho_{CPU} \stackrel{def}{=} \frac{c_1\rho_1 + c_2\rho_2}{2C}$$

in terms of CPU and

$$\rho_{RAM} \stackrel{def}{=} \frac{r_1\rho_1 + r_2\rho_2}{2R}$$

in terms of RAM, where $\rho_1 = \lambda_1/\mu_1$ and $\rho_2 = \lambda_2/\mu_2$.

We obtain via simulation blocking probabilities for both mice and elephant classes. Simulation results are averaged to obtain confidence intervals with a 95% confidence level.

Figure 3.3 displays the blocking probability versus traffic intensity under three different regimes: underload, critical and overload. This figure shows that the Mices have quite different blocking probabilities when compared to Elephants but in both cases, the blocking probability given by our model is comprised between the upper and lower value given by simulations.

Results illustrate the good accuracy of the proposed analytical model through blocking probability estimation for both classes especially under the critical regime.

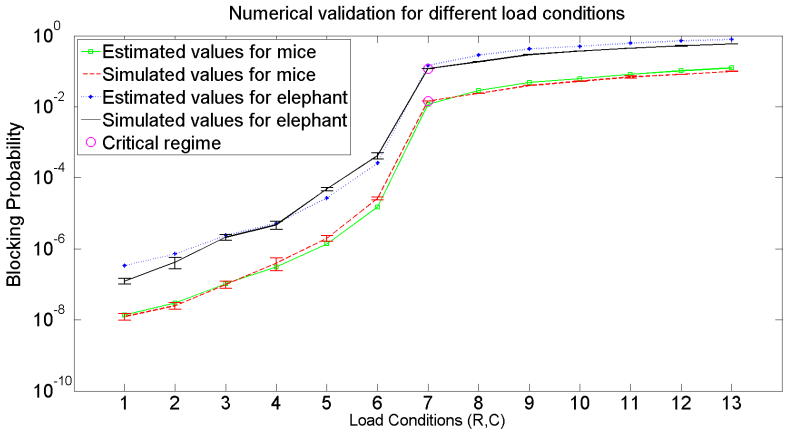

FIGURE 3.3: Numerical Validation Of the Model with two-dimensional system

.

Three-dimensional system We consider in this section that the system offers 3 types of resources (namely CPU, RAM and Disk). As in the previous scenario,

requests arrive according to a Poisson process with different rates; λ_1 for the first class and λ_2 for the second class. The Mice class requires $c_1 = 4$ units of CPU, $r_1 = 1$ units of RAM and $d_1 = 30$. Similarly; the Elephant class requires $c_2 = 16$ units of CPU, $r_2 = 64$ units of RAM and $d_2 = 70$. The holding time of resources is assumed to be the same for both classes ($1/\mu_1 = 1/\mu_2 = 1$).

We consider a data center with two identical servers with capacity C in terms of CPU, R in terms of RAM and D in terms of Disk storage. We assume that the two servers are grouped into a unique big server with capacity $2R$ in terms of RAM, $2C$ in terms of CPU and $2D$ in terms of storage. We obtain via simulation blocking probabilities for both mice and elephant classes.

The load of the system is then

$$\rho_{CPU} \stackrel{def}{=} \frac{c_1\rho_1 + c_2\rho_2}{2C}$$

in terms of CPU,

$$\rho_{Disk} \stackrel{def}{=} \frac{d_1\rho_1 + d_2\rho_2}{2D}$$

in terms of storage and

$$\rho_{RAM} \stackrel{def}{=} \frac{r_1\rho_1 + r_2\rho_2}{2R}$$

in terms of RAM, where $\rho_1 = \lambda_1/\mu_1$ and $\rho_2 = \lambda_2/\mu_2$.

Simulation results obtained with a 95% confidence level.

Box plots in the Figure 3.4 displays the blocking probability obtained via simulations for both classes under two different regimes: underload conditions (where $\rho_{RAM} = 0.80$, $\rho_{CPU} = 0.80$ and $\rho_{DISK} = 0.85$) and overload conditions (where $\rho_{RAMA} = 1.10$, $\rho_{CPU} = 1.45$ and $\rho_{DISK} = 1.247$). Analytically estimated values of blocking probabilities are marked with a red star. Results illustrate the good accuracy of the proposed analytical model through blocking probability estimation for both classes especially under the underload as well as the overload regime.

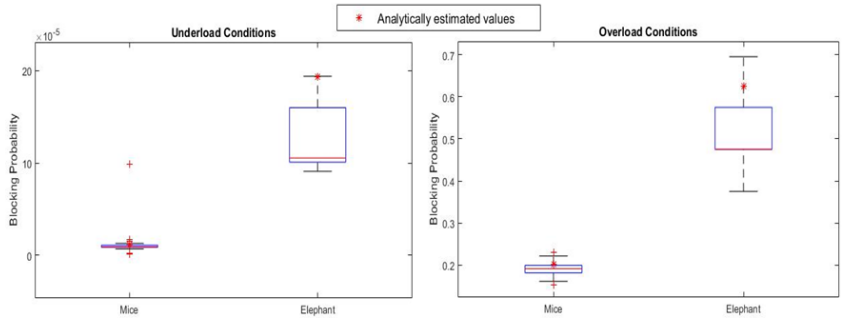

FIGURE 3.4: Numerical Validation Of the Model with three-dimensional system .

3.3.4 Evaluation of resource allocation algorithms

In this section, we evaluate via simulation the efficiency of some placement algorithms presented in Chapter 2 by comparing the blocking probabilities values to those obtained in the case where we consider servers grouped into one big server where there is no efficiency loss due to the fragmentation. Depending on the number of arrival classes, we have studied two scenarios.

3.3.4.1 Two-Class System

Performed Simulations The system we have first considered is a data center composed of 2 servers with identical capacities offering two types of resources(RAM and CPU) and intercepting 2 arrival classes of requests; one class is composed of requests with small requirements in terms of resources referred to as Mice and the second class Elephants has hight requirements as in the previous section.

To study the performance of the system, we have considered three regimes:

- Underloaded system: $\rho_{CPU} < 1$ and $\rho_{RAM} < 1$;
- Critical load conditions: $\rho_{CPU} = 1$ and $\rho_{RAM} = 1$.
- Overloaded system: $\rho_{CPU} > 1$ and $\rho_{RAM} > 1$.

a) Underloaded System

Simulations Settings To study the performance of the system, we have considered first the underloaded conditions. Parameter values for underloaded conditions are given in Table 3.1. The loads are $\rho_{CPU} = 0.81$ and $\rho_{RAM} = 0.88$.

TABLE 3.1: Parameter values for underloaded conditions.

parameter	value
$(\lambda_1, \mu_1, c_1, r_1)$	(70,1,2,4)
$(\lambda_2, \mu_2, c_2, r_2)$	(30,1,17,32)
(C, R)	(400,700)

Simulations Results The blocking probabilities for the two classes of requests and the various algorithms are given in Figure 3.5. We can verify that there is no significant difference between the various algorithms for both classes; mice as well as elephant.

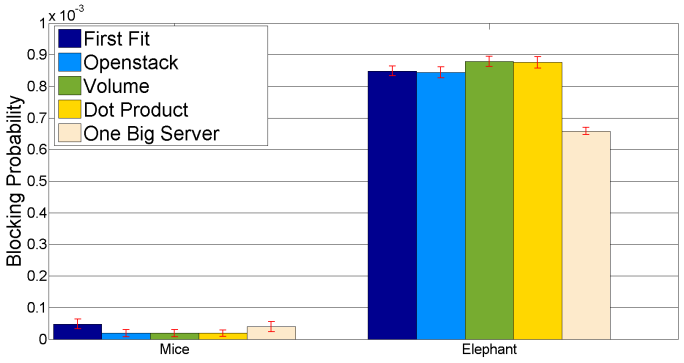

FIGURE 3.5: Blocking Rates in an Underloaded System .

b) Critical load conditions

Simulations Settings We consider in this scenario critical load conditions with parameters given in Table 3.2. The loads are $\rho_{CPU} = 0.99$ and $\rho_{RAM} = 1$.

TABLE 3.2: Parameter values for critical load conditions.

parameter	value
$(\lambda_1, \mu_1, c_1, r_1)$	(70,1,2,4)
$(\lambda_2, \mu_2, c_2, r_2)$	(30,1,17,32)
(C, R)	(328,620)

Simulations Results The blocking probabilities given by the different algorithms and for both classes are given in Figure 3.5. We verify that the blocking rates for elephants are greater than those for mice but the blocking rates are not significantly different from one algorithm to the other; there are variations but not by an order of magnitude. To go further in the analysis of the system

in terms of blocking, we compare the system with two servers against a unique big server with capacity equal to the sum of the capacities of the two servers. We observe that the blocking rate are quite similar and in the same order of magnitude.

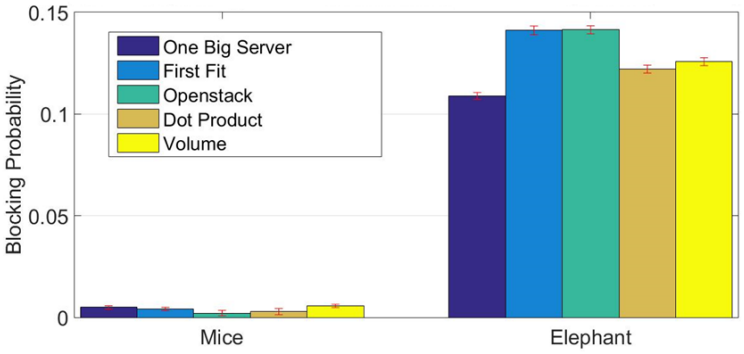

FIGURE 3.6: Blocking Rates under critical load conditions .

c) Overloaded System

Simulations Settings In this section, we confirm the results by considering an overloaded system. Parameter values for overloaded conditions are given in Table 3.3. The loads are $\rho_{CPU} = 1.16$ and $\rho_{RAM} = 1.26$.

TABLE 3.3: Parameter values for overloaded conditions.

parameter	value
$(\lambda_1, \mu_1, c_1, r_1)$	(70,1,2,3)
$(\lambda_2, \mu_2, c_2, r_2)$	(30,1,17,35)
(C, R)	(280,500)

Simulations Results The blocking probabilities for the two classes of requests and the various algorithms are given in Figure 3.7. As in the other cases, we can verify that the blocking rates are slightly different but on the same order of magnitude especially when compared to values given by the case of one big server.

Results Interpretation We compared the system with two servers against one big server with capacity equal to the sum of the capacities of the two servers. We observed that the blocking rates are sightly different but of the same order of magnitude. These observations hold for all the simulation experiments we have performed for this kind of system. Hence, to qualitatively analyze the system

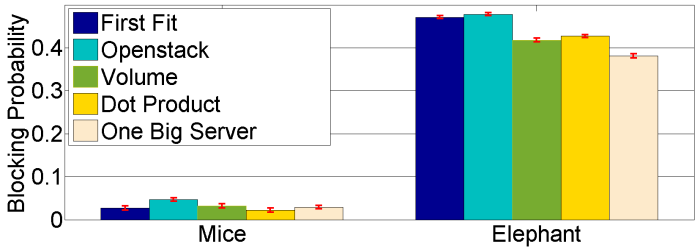

FIGURE 3.7: Blocking Rates in an Overloaded System .

with two servers of capacity C, it suffices to consider a system with a unique server of capacity $2C$. This means that the fragmentation of resources into various servers has no impact as long as individual requests are small when compared to server capacities. This is a key fact for qualitatively estimate blocking in cloud platforms because the analysis of large multidimensional systems can be analyzed by using classical methods used in the context of circuit switched networks as we proposed in Section 3.3.1.

3.3.4.2 Four-Class System

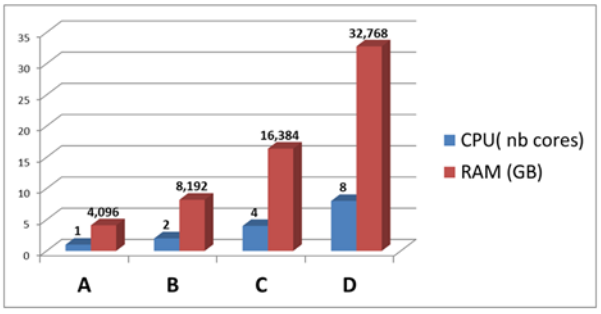

FIGURE 3.8: Resource requirements of VM classes .

Simulations Settings Based on class specification of a popular cloud platform, we have defined 4 arrival types of requests with different resource requirements. Resource requirements of each class are shown in Figure 3.8.

Arrival rate as well as holding times are obtained from the proportion of each class in the system. Figure 3.9 illustrates these proportions on our system.

We have performed extensive simulations under different load conditions. We have varied the number of servers composing the system ($N = 2, ..., 10$) while conserving the same load conditions; the system is first underloaded with load values $\rho_{CPU} = 0.9444$ and $\rho_{RAM} = 0.8897$. Then, we considered an overloaded system with load values $\rho_{CPU} = 1.37$ and $\rho_{RAM} = 1.15$.

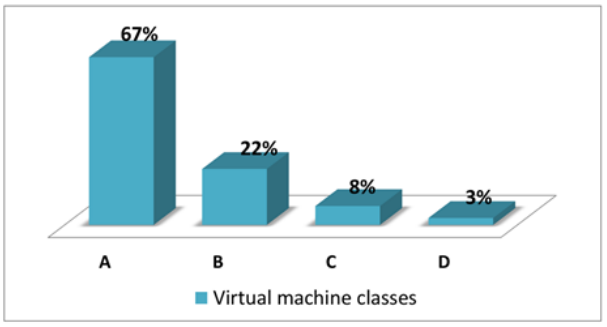

FIGURE 3.9: Cloud load per VM class .

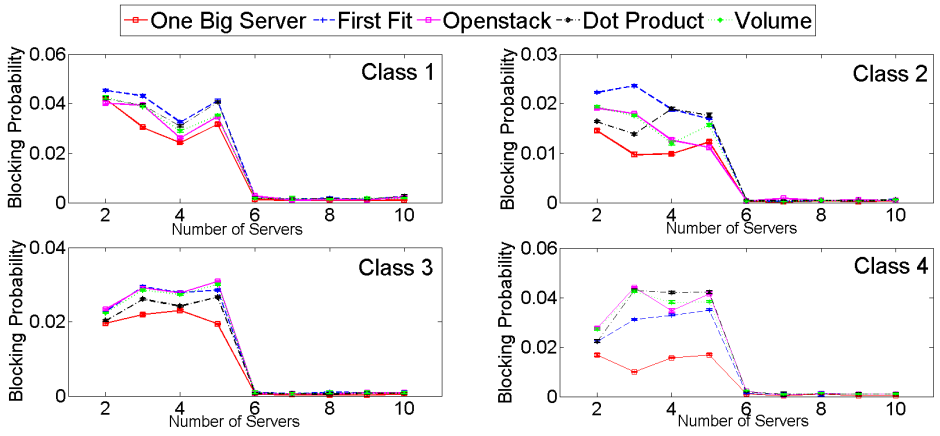

FIGURE 3.10: Blocking Rates in an Underloaded System .

Simulations Results Figure 3.10 displays the blocking probabilities for each class under the underloaded conditions. In Figure 3.11, we show the blocking rates under an overloaded regime. Results are qualitatively the same for both regimes. As in the previous scenarios, we can verify that there is no significant difference between the various algorithms. We also note that as before, comparing these values against those obtained via simulation of a big unique server validates our proposed model in the sense that the blocking rates are similar. This opens the door to the analysis of a system composed of many servers by considering a unique big server whatever be the resource allocation algorithm in the multi-server system. Such an analysis is sufficient for dimensioning purposes; in practice only a rough estimates of blocking rates are sufficient.

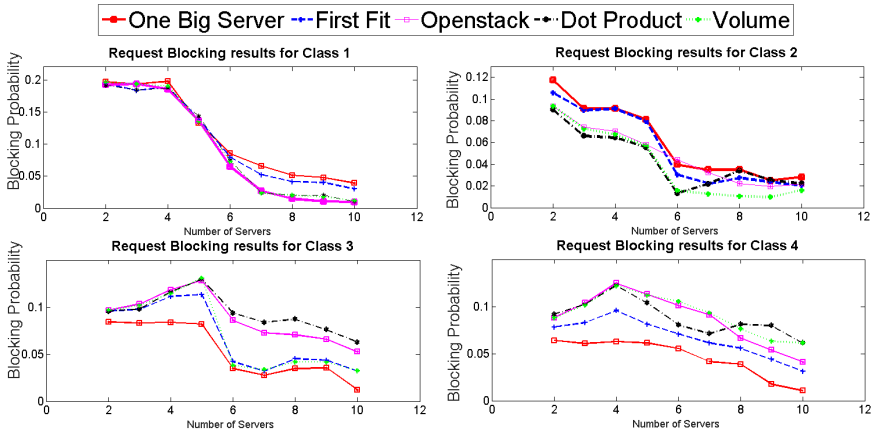

FIGURE 3.11: Blocking Rates in an Overloaded System .

3.4 Resource Allocation Algorithms In Distributed Cloud Platforms

We present in this section an overview of resource allocation in distributed cloud from the existing literature. We then investigate placement and offloading strategies of constrained services in distributed cloud. We finally propose for this context a costless distributed resource allocation algorithm for, named *CLOSE*.

3.4.1 Related Work on resource allocation in Distributed Cloud

In the recent few years, many research works have addressed the problem of resource allocation in distributed cloud environments [57]. However, most contributions have considered configurations in which the placement decision runs in a centralized platform. In such a context, a popular approach is to adopt an optimization formulation: Given a demand for resources in terms of storage and computing, the problem is to find the optimal request placement. This leads in general to Mixed Integer Linear Programming (MILP) problems. In addition to the classically considered resources such as CPU, RAM and disk, bandwidth can also be included for virtual machines placement inside a single data center [58].

In [59], an optimization problem is proposed for the placement of VNFs across a distributed cloud. It can be observed from that paper that MILP takes less than one second to run for an infrastructure comprising 5 data centers, while it needs several tens of minutes for only 20 data centers. Hence, this approach will hardly scale with the size of a distributed data center system, in particular, for systems composed of hundreds of data centers, as it might be the case for network operators who plan to deploy a data center at PoP of their network.

Indeed, depending on the geographical span of a network, several hundreds of PoPs could be deployed by a medium-size network operator.

To achieve optimal resource placement, an exact formulation that aims at finding the best placement of resources by maximizing the revenue and minimizing the corresponding cost is proposed in [60]. The authors have also noted that the ILP formulation suffers from scalability problems. They then proposed an alternative approach via dynamic resource placement by representing the resource allocation problem by a directed graph and by using a minimum cost maximum flow algorithm for resource placement. To compute the minimum cost maximum flow in the graph, the Edmonds-Karp algorithm is used. This approach does not consider, however, the latency constraints of requests.

Since optimization approaches can be very time-consuming and may suffer from scaling issues, several works propose alternative approaches. In [61], an algorithm for network-aware allocation of virtual machines in distributed cloud systems is studied. By representing the distributed cloud system as a complete graph, where vertices represent data centers, weights represent the number of available virtual machines or data center capacities, edges represent links between data centers and labels represent the number of hops or distance, the proposed algorithm selects first the relevant data centers to serve a user request and then the physical machines to run the virtual machines. Even if this selection aims at minimizing the maximum distance among virtual machines running the request and therefore the bandwidth usage, this algorithm applies only if the total amount of network traffic between virtual machine is known.

A set of greedy algorithms for VNFs scheduling was proposed in [62]. Network mapping was also taken into account but physical links were not considered, which means that latency constraints were not handled by the proposed algorithms.

In [34], a cloud management middle-ware is proposed in order to reduce web application response time by migrating virtual machines closer to end users. In [63] a high locality scheduling for an edge cloud environment that reduces the networking costs is presented. In [64] a resource allocation algorithm for distributed cloud system is proposed with the primary objective of minimizing the overall operating cost, which is a trade-off between energy cost and WAN cost, when energy price is not the same across different geographical locations.

Authors in [65] enumerates required steps to build a massively distributed OpenStack-based architecture. Multiple challenges are addressed to adapt Openstack to the new Telco context.

Last but not least, the Openstack community has created the Tricircle project, the new edition to cope with distributed cloud architecture[1]. The main objective of the project is:

- to allow cloud capacity expansion, by adding new instances
- to improve reliability and availability through supporting a geo-distributed cloud architecture
- to reduce bandwidth usage by allocating resources close to end users on each site

Figure 3.12 illustrates the Tricircle architecture where we have 3 nodes for 2-regions Openstack deployment: The Tricircle central as a controller and 2 regions representing 2 datacenters. The Tricircle Central provides networking automation across Neutron in the two regions and each region includes its own Nova and Cinder. Compute and storage resources are exposed to end user through an exposed service endpoint from Nova and Cinder of each region. With regard to resource allocation, the data center with the maximum amount of available resources from a user's availability zone (sub-list of data centers) is selected to accommodate a request. The selection of the most appropriate physical machines to place the request, within a data center, is made locally by the Nova and Cinder schedulers managing the data center. Since one availability zone could include several groups of data centers, if one data center reaches the limit of the resource utilization, the request will be rerouted to another data center but in the same zone.

It should be emphasized that most of the above-mentioned works consider a global knowledge about resource utilization. Additionally, some of the contributions cannot be applied in realistic use cases with a dynamical arrival of user requests. Finding a strategy, which is based only on local knowledge with no signaling overhead, represents the main focus of the next section.

[1] https://wiki.openstack.org/wiki/Tricircle

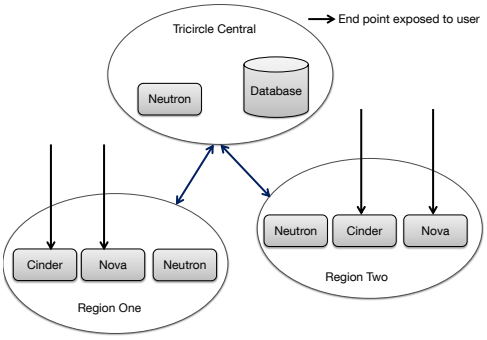

FIGURE 3.12: Tricircle Architecture .

3.4.2 CLOSE: A Services' Offloading Algorithm for Distributed Edge Cloud

3.4.2.1 Preliminary considerations

Network function placement is a complex problem depending on several parameters. This problem is all the more difficult when one considers the dynamics of the allocation of resources. In this context, it is natural to consider geographical aspects related to the origin of requests, especially for constrained functions (i.e., latency, overhead, bandwidth, security ...). Indeed, some functions such as firewall, deep packet inspection, etc. have to be placed close to end users (e.g., to prevent users from sending confidential data through the Internet). Some others such as authentication, IP address allocation, etc. with looser time constraints can be placed in a distant data center [66].

Generally speaking, requests may be made of components (i.e., sub-functions) having various requirements in terms of latency. Some of them may have stringent requirements, for instance a response time of the order of a few milliseconds, while some others may be more tolerant with regard to delay. The placement of sub-functions could be in principle distributed over several data centers as long as the global response time requirements are met. In this work, we assume, however, that a function with strict latency requirements, possibly composed of several sub-functions, shall be instantiated on the same edge data center or be rejected. This leads us to claim our first design statement.

Design principle 1. **Instantiate a function with latency constraints on the same edge data center instead of spreading it on many data centers.**

Generally speaking there is a design choice between optimization and dimensioning. This latter task consists of assessing the amount of resources needed to accommodate a resource demand with a prescribed very small rejection rate. Optimization may lead to better acceptance rate of requests but also to send traffic to a sub-function hosted by an edge data center and then back for further treatment. To avoid this traffic "tromboning" effect, we state that a complete function shall be hosted by an edge data center or rejected.

Design principle 2. **Avoid traffic "tromboning" in the network due to function splitting.**

To achieve a request acceptance rate objective, edge data centers have to be dimensioned according to the demand in terms of resources (compute, storage, bandwidth). We thus transform an optimization problem into a dimensioning problem.

Design principle 3. **Dimension edge data centers instead of optimizing function placement.**

Dimensioning edge data centers differ from the traditional transmission link dimensioning problem in telecommunications networks. On the one hand, we have to take into account more parameters beyond the sole bandwidth resource. A multidimensional Erlang formula can, nevertheless, be expected [67]. On the other hand, a request can be displaced as long as response time requirements are met. This introduces some flexibility in the acceptance of a request. An Erlang loss formula taking into account potential migration of requests among a possible set of servers, capable of meeting request requirements, is still an open problem.

3.4.2.2 Algorithm principles

To allocate resources in a distributed data center system by taking into account the location of requests, we clearly have two possibilities. In the first one, there is a centralized entity which has a view of the resources available in the various data centers and depending upon the location a request is issued from, this central dispatcher can select those data centers, which can accommodate the request while respecting the constraint on the maximum displacement of the request; once this set of data centers is known, the dispatcher can pick up a data center at random or one among those with the maximum amount of available resources or with the better score [68]. If all data centers able to respect the displacement constraint are occupied, then the request is simply rejected. It is worth noting

that this approach is in line with the current Tricircle approach of OpenStack, which in addition uses the concept of area. This approach is refereed to as the centralized approach. We assume in this thesis that edge data centers are operated up to a certain overbooking limit of resources. If an edge data center is too loaded, a request is blocked.

For the second possibility, a user request can be intercepted by the first data center on the data path. The Distributed resource allocation algorithm that we propose is as follows:

1. When a request arrives in the system, the request is intercepted by the first data center along the data path.

2. If the request cannot be accommodated by this edge data center (i.e., when the function `IsAvailable` returns **False**), then, it is forwarded to one of its neighbors, which may respect the time constraints of the service. A Time To Live (TTL) field can, also, be considered to limit the displacement of the request.

3. To forward the request, the edge data center takes into account the number of redirections from its neighbors and the time constraints[2]. Specifically, an edge data center maintains a counter, which records the moving average number of redirections (deflected requests) from its neighboring edge data centers. The edge data center with the smaller number of deflected requests is chosen. The request is forwarded with the label of the deflecting data center in order to avoid loops.

4. The redirected request is examined by the edge data center, the request is forwarded to, and the TTL value is decreased accordingly. If the request can still not be accommodated and if the TTL field is non null and time constraint can be met, then the previous step is repeated otherwise the request is discarded using the `Discard` function.

The pseudo-code of the proposed mechanism is illustrated in Algorithm 1 by using the notation summarized in Table 3.4.

The major difference between the proposed algorithm and the centralized one is the amount of information to be exchanged between the various edge data centers and the central dispatcher. For the centralized algorithm, each time a request is accepted by or leaves an edge data center, then this data center has

[2]The current data center selects a sub-list of data centers respecting the request criterion using the `GetNeighborsIdx`, which returns the set of possible data centers.

TABLE 3.4: Notation used in the distributed algorithm.

Variable	Description
N	Number of requests
DC_i	Data center i
$l_{i,j}$	Latency between $DC_i and DC_j$
$d_{i,j}$	Number of deflected requests from DC_i to DC_j
Rq	A request including: the amount of requested resources, the maximal hops in terms of TTL and latency l_{max}, and the cumulative latency

Algorithm 1 CLOSE algorithm for services' offloading

```
1: procedure FORWARD(DC_cur, DC_ori, Rq)
2:     if IsAvailable(DC_cur, Rq) then
3:         Allocate(DC_cur, Rq)
4:     else
5:         Rq.latency ← Rq.latency + l_{cur,ori}
6:         Rq.TTL ← Rq.TTL − 1
7:         J ← GetNeighborsIdx(DC_cur, Rq) \ {DC_ori}
8:         if J ≠ {} then
9:             dst ← arg min_{j∈J} {d_{j,cur}}
10:            FORWARD(DC_dst, DC_cur, Rq)
11:        else
12:            Discard(Rq)
13:        end if
14:    end if
15: end procedure
16:
17: while True do
18:    Rq ← getRequest()
19:    DC_cur ← getClosestDC(Rq)
20:    FORWARD(DC_cur, DC_cur, Rq)
21: end while
```

to send an update of available resources to the central dispatcher so that this latter maintains accurate information about the occupancy of the system. In big systems, with several hundreds of distributed data centers, this may represent a significant overhead. For the distributed algorithm, such information has not to be exchanged between edge data centers since only local information is used. The counterpart is that the forwarding of a request is performed with less information. This algorithm is hence less accurate but more efficient in terms of overhead.

There is clearly a trade-off between the accuracy of the placement of requests and the cost to maintain accurate information. This is a classical issue in telecommunications networks and has been so far solved by using monitoring tools and

regular upgrade of network capacities. In the present case, we use an additional degree of freedom by allowing the displacement of requests up to a certain limit; this is impossible with bandwidth in classical networks, even for elastic traffic, which can severely suffer from congestion (e.g., flows with very small bit rates leading to very poor quality of experience). In the following, we shall see that displacement allows local congestion to be absorbed by the system.

3.4.3 Performance Evaluation of the CLOSE Algorithm

To assess the performance of our proposal, we have different strategies. In a first one, we do not consider any offloading. In other words, edge data centers does not collaborate, which means that a request is rejected if there is not enough resources to accommodate it. We refer to this algorithm as "Isolation" since an overloaded edge data center cannot take benefit of the possible available capacity in the global system.

A second strategy relies on a central dispatcher which is aware of the occupation of all edge data centers and selects the one with the most available capacity and respecting the constraints of the service (without considering latency), even if the latter data center is not the closest to the origin of the request. This algorithm is referred to as "Full Sharing".

We have, also, considered the algorithm used by Openstack, where the centralized dispatcher has to maintain an updated view of the infrastructure topology as well as the resource utilization level. Respecting the maximum displacement constraint of a request, the dispatcher has to select those data centers, which are eligible to accommodate it. Those candidate data centers are mapped onto a geographically area from which the request can be serviced. This area can be mapped to an "Availability Zone" according to the Openstack terminology. Technically, each Openstack project implements it differently – with regard to resource allocation – in a manner to enable logical subdivision of resources.

Furthermore, we have considered requests with two types of latency requirements. The first class has stringent latency requirements (namely, a response time less than 4 ms) while the other class is more delay tolerant (10 ms). Data centers take local decisions and we evaluate the global blocking rate.

To highlight how these strategies perform, we have devised several simulations, in which arrival requests are created using a probability distribution.

3.4.3.1 Simulation settings

To study the performance of a distributed cloud system, we have considered a realistic network of $N = 21$ data centers with different capacities located at the edge of an Autonomous System, as illustrated in Figure 3.13. We have considered the structure of the network of Orange, in which the distance between small data centers, located at Main Central Offices (MCO), is 100 km from Core Central Offices (CCOs), equipped with bigger data centers. CCOs are connected to a big centralized data center at a distance of 300 km[3]. Latencies are computed by using the speed of light in fiber.

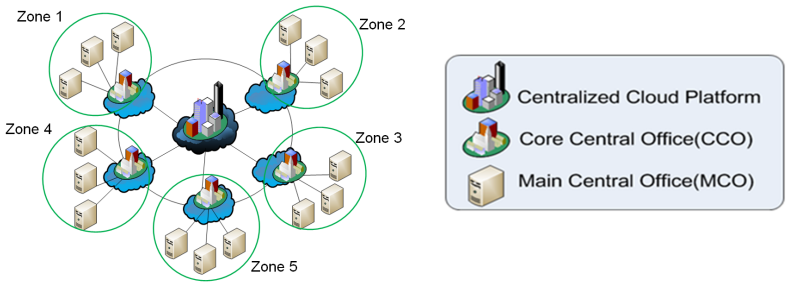

FIGURE 3.13: Network topology under consideration.

Request arrivals are assumed originating from one of the considered regions according to a Poisson process, which has proven realistic for a number of real traffic arrival processes [69]. For the sake of clarity, only one type of resource is considered , typically the CPU. Similar results can be obtained using multiple resources as we demonstrated in Section **??**. As mentioned above, we have considered two profiles of requests. Profile 1 is assumed to have strong requirements in terms of latency (4ms), while Profile 2 is more delay tolerant (10 ms).

The global load of the system (data centers) is defined as:

$$\rho \stackrel{def}{=} \sum_{j=1}^{N} \frac{\rho_j}{NC_j}$$

where $\rho_j \stackrel{def}{=} \lambda_j/\mu_j$ is the load of Data Center j (for short, DCj) with λ_j and $1/\mu_j$ representing the arrival rate and the mean holding time of resources at DCj, respectively and where C_j is the capacity of DCj. Data centers (DCs) are

[3]For the sake of confidentiality, the DC locations and real distances used in the simulations are not given in this work

unevenly loaded; we only consider the global load ρ of the system given that some DCs are overloaded while some others are underloaded.

In order to compare the various allocation schemes, we introduce the average blocking rate defined as the fraction of requests, which are eventually rejected by the system:

$$\overline{\beta} = \frac{1}{\Lambda} \sum_{j=1}^{N} \lambda_j \beta_j$$

where $\Lambda = \sum_{j=1}^{N} \lambda_j$ is the global arrival rate and β_j is the blocking rate of requests originally arriving at DCj. More precisely, β_j is the fraction of requests which are originally arriving at DCj but eventually not accepted by the system.

3.4.3.2 Simulation Results

Blocking Probabilities The average blocking rates of the system under the various allocation strategies are given in Figure 3.14. This figure displays the blocking probability versus traffic intensity under three different regimes: underload ($\rho < 1$), critical ($\rho = 1$) and overload ($\rho > 1$). Simulation results are averaged to obtain confidence intervals with a 95% confidence level.

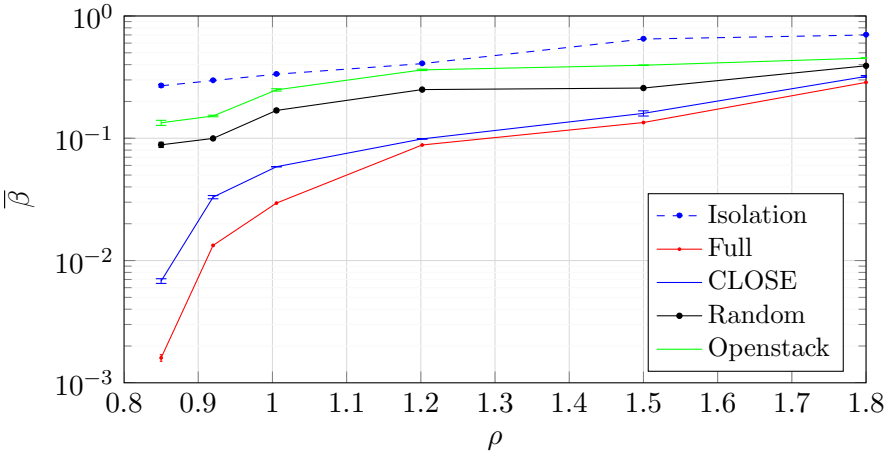

FIGURE 3.14: Blocking rates under different load conditions.

Results show that collaboration between data centers significantly reduces blocking of requests. We verify that the *Isolation* scheme yields the worst performance in terms of rejection. Blocking is obviously minimal with the Full Sharing algorithm when the global dispatcher has a full view of the occupancy of the system, but this strategy does not take into account the latency constraints.

It is worth noting that the proposed policy *CLOSE*, which counts the average number of deflections of each neighboring data center, significantly reduces blocking when compared against the *Isolation* allocation and yields a performance comparable to that obtained with the *Full* Sharing strategy. We clearly see that deflecting requests can significantly reduce blocking and share load on data centers. Moreover, the proposed scheme with no exchange of information between data centers performs well and even better than the Tricircle approach of OpenStack, where load is shared only within a geographical area.

Latency To further evaluate the performance of the *CLOSE* algorithm, let us emphasize another key difference when compared to the *Full* Sharing strategy. The displacement of requests for the Full Sharing algorithm may be larger than that for the distributed one, since the centralized algorithm only takes into account resources and not displacement constraints.

To illustrate this latter point, we have studied the Latency Distribution of accepted requests for both algorithms. In Figure 3.15, we have plotted the latency Cumulative distribution for an overloaded System where the system load is $\rho = 1.2592$. We see that *CLOSE* leads to the lowest latency distribution since most of requests are serviced from the edge data center closest to the end user. Hence, the proposed algorithm performs better with regard to displacement than the *Full* Sharing algorithm while offering comparable blocking.

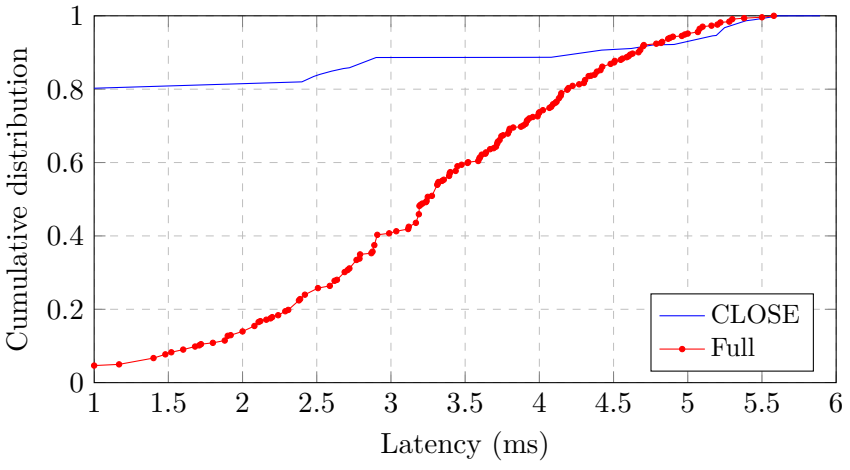

FIGURE 3.15: CDFs of Latency: CLOSE vs Full Sharing.

For the same experiment, we have also plotted in Figure 3.16 the latency distribution function for both profiles introduced in the previous section to show how the *CLOSE* algorithm respects request constraints in terms of latency.

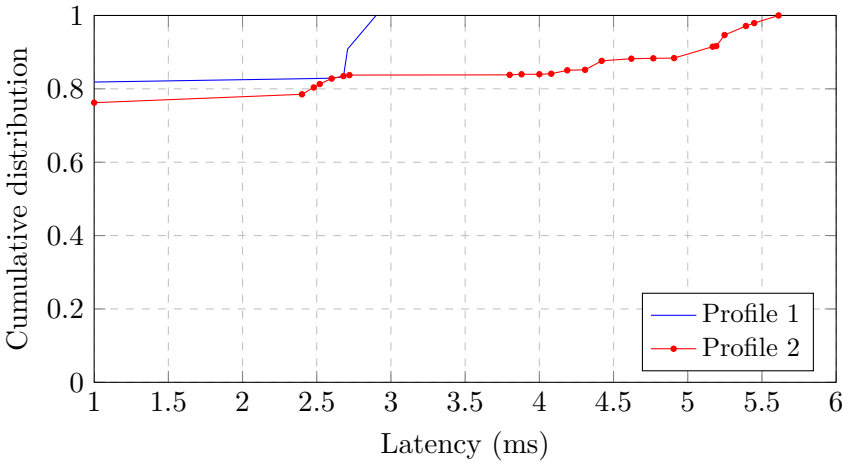

FIGURE 3.16: Per-profile CDFs of Latency for the *CLOSE* algorithm.

Fairness: Load Balancing Another key metric for evaluating the performance of the proposed algorithm is the load balancing index calculated on the basis of the Jain's fairness index. Figure 3.17 illustrates the fairness index for load balancing under the different strategies. Except the *Isolation* scenario, which leads to a totally unfair system, results are slightly different and comparable to that obtained with the *Full* sharing strategy considered as the most fair strategy.

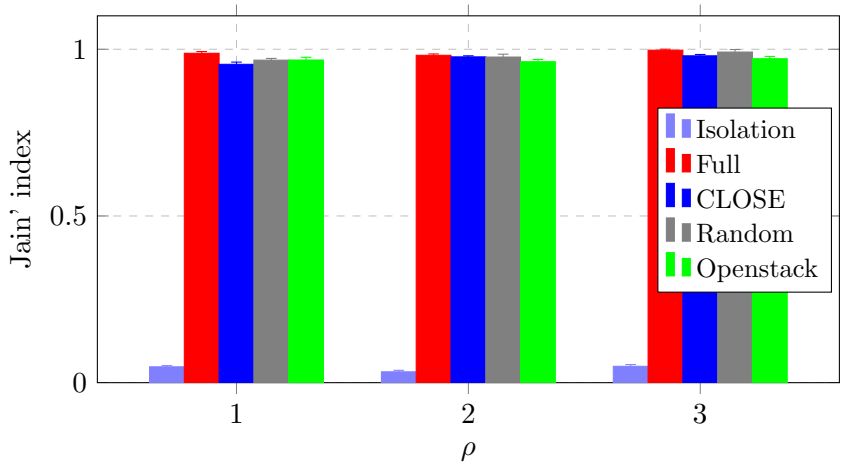

FIGURE 3.17: Jain's index for resource utilization level.

The impact of load In Figure 3.18, we have evaluated how the squared coefficient of variation (CV^2) of the loads of data centers. The coefficient of variation CV indicates the extent of variability in relation to the mean of the distribution. This can reflects how the degree of homogeneity of the loads between the DCs

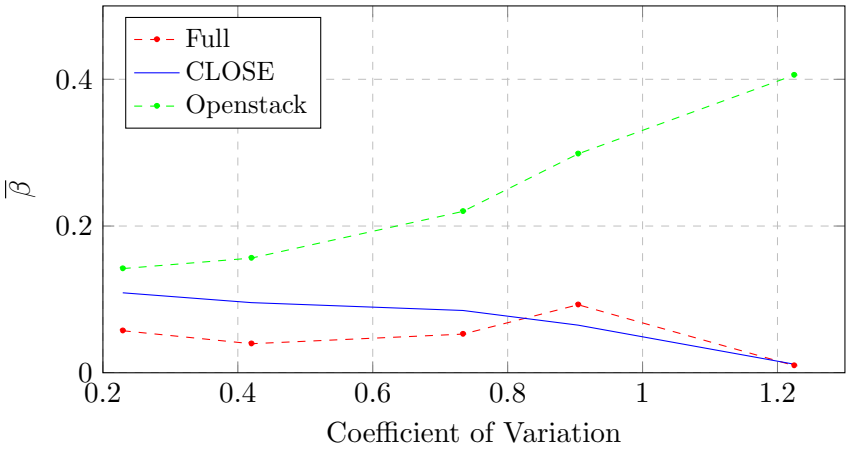

FIGURE 3.18: Algorithm Behavior under different load conditions.

can impact the blocking probability. The bigger is the coefficient, the more heterogeneous is the load between DCs. As the Openstack strategy is based on load sharing within the same zone (area), the performance of this strategy strongly depends on how the system is loaded, as it can be seen in Figure 3.18. In other words, if the zones are homogeneously loaded, this strategy ensures good load balancing in each zone by choosing the less loaded edge data center, which leads to a fair system and yields to good performance. In contrast with this latter, the *CLOSE* algorithm, which is able in some cases to distribute loads far from the origin request, yields to performance comparable to that obtained with *Full* Sharing.

3.5 Summary

In this chapter, we have described an overview regarding resource allocation performance in cloud systems with finite capacity.

First, we provided under the centralized approach an analytical model to analyze the blocking in such system. We have evaluated resource allocation performance in cloud systems with finite capacity by paying special attention to blocking of requests in a probabilistic context. The key observation is that with regard to request blocking there is no noticeable difference between the various placement algorithms so far considered in the literature. In fact, we show that blocking rates are similar to that obtained when considering a global data center with a capacity equal to the sum of servers capacities. This model is able to accurately estimate the blocking probability in multidimensional cloud systems. It turns out that if we have several data centers disseminated in the network and if it is

possible to know their occupancy upon each request, then everything happens as if the network had a unique big data center with a unique server with capacity equal to the sum of all capacities. This gives a means of estimating request blocking at a network scale and eventually a simple method of dimensioning a system of data centers for a given demand.

Second, we have presented the specific requirements of VNF placement in geographically distributed cloud system. We have devised basic design principles for handling requests with displacement constraints in a large system of distributed edge data centers. Our claim is that traffic "tromboning" and request splitting (especially for applications or virtualized network functions with stringent response time requirements) should be avoided. Moreover, adequately dimensioning edge data centers is preferable to optimizing request placement. Furthermore, we have proposed an algorithm for placing requests in a large distributed cloud platform with minimal exchange of information. Contrary to the OpenStack approach, notably the Tricircle project, this algorithm is based on local information only. In spite of minimal information, this algorithm can mitigate overload at some data centers by using a simple redirection principle by exploiting deflection information between data centers. This is a very promising result as the proposed algorithm allows a network operator to easily manage a large distributed infrastructure.

Chapter 4

Orchestration Platforms For NFV

Contents

4.1 Introduction

Moving towards virtualized network functions (VNFs) allows the introduction of more flexibility and scalability in the network's infrastructure, while reducing capital and operational expenditures (CAPEX and OPEX) [70]. However, many challenges remain as the NFV paradigm aims at delivering end-to-end network

services by instantiating VNFs, within the virtualized infrastructure, whilst taking into account their specific requirements.

To cope with these challenges, there is a need to have a global orchestration platform, including instantiation logic, life-cycle management, as well as monitoring features.

In this Chapter, we give some background on orchestration platforms for NFV and review some NFV orchestration projects. We investigate in particular two of the most important features of an orchestrator, notably the monitoring and the scheduling features.

In section 4.2, We introduce first the Orchestration and Management Framework (MANO) delivered by ETSI NFV from an architectural perspective. Second, we present an overview of some orchestration projects compliant with the MANO architecture.

Section 4.3 we present the monitoring feature; essential for infrastructure management.

In Section 4.4, we investigate the scheduling feature by studying a use case, notably the most popular orchestration platform named ONAP.

Section 4.5 presents concluding remarks.

4.2 Background

4.2.1 MANO: Management and Orchestration for NFV

The ETSI NFV is the group in charge of the development of standards for NFV. To this end, they have defined reference architectures in particular for management and orchestration [71].

The reference architecture for Management and Orchestration (MANO) illustrated in figure 4.1 is delivered by ETSI NFV.

The Mano architecture consists of three functional blocks [72]:

1. Virtualized Infrastructure Manager (VIM): The main role of this component is to control the compute, storage and network resources within the operator's infrastructure. The VIM is also responsible of collecting and forwarding performance measures such as the level of resource utilization.

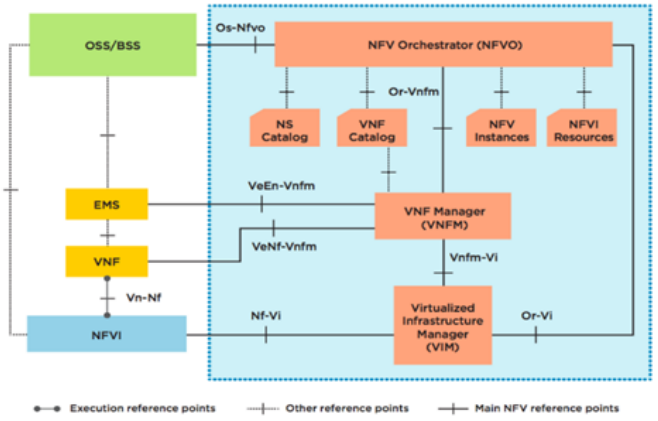

FIGURE 4.1: NFV Management and Orchestration Architecture

2. Virtual Network Function Manager (VNFM): VNFM manages life cycle of VNFs. This component creates, maintains and terminates VNF instances deployed on the Virtual Machines (VMs) created by the VIM.

3. NFV Orchestrator (NFVO): The NFVO is in charge of the network service life-cycle management from the top level. This component is in charge of the global resource management and the resource coordination from different VIMs especially when there are multiple VIMs from different PoPs. This is precisely the challenge that the network operator faces in adopting the distributed multi-site cloud.

4.2.2 Related Work on Orchestration Platforms

Several NFV orchestration projects have been initiated with the idea of being compliant with the MANO (Management And Orchestration) reference architecture delivered by ETSI NFV [73].

The main objective of the CORD (Central Office Re-architected as a Data Center) orchestration solution for NFV environment is to bring elasticity and cloud agility to the telco Central Office [74]. From an architectural point of view, CORD has defined its own architecture but most of the architectural blocks might be mapped to the MANO reference architecture. CORD is based on the ONOS SDN controller to manage network resources [75]. With regard to resource allocation, the CORD platform delegates VNF placement to the infrastructure controller, the so-called VIM (Virtual Infrastructure Manager), notably based on the Openstack platform.

Gigaspaces Cloudify project[1] was originally introduced to orchestrate application

[1]https://www.gigaspaces.com/cloudify-overview

deployment in a cloud similarly to the Openstack's Heat orchestrator. Later, with the emergence of NVF, a Telecom Edition was delivered including NFV-related use cases.

Inspired by ETSI MANO, OpenMANO [76] is an opensource platform led by the network operator Telfonica and based on opensource ecosystems such as Openstack as a VIM. One of the selling points of NFV is the ability to scale resources dynamically which is a very limited functionality within the current implementation of OpenMANO.

4.3 Monitoring feature

4.3.1 What is Monitoring ?

With Virtualization being a key driver to the new digital transformation of telcos, the need for managing virtualized infrastructure is consistently increasing. The critical requirements of NFV raise potential issues as it became necessary to implement a more robust orchestration platform able to anticipate and resolve issues before any user impact. A key requirement throughout this orchestration platform is the infrastructure monitoring feature.

A monitoring system enable the network operator to identify and detect any potential issues before they become critical and affect the high availability of VNFs.Having a monitoring system in place also means gain visibility on the integrity of resource within the infrastructure as well as the occupation level which is critical to fulfill intelligent placement decisions.

For the virtualized infrastructure, there are several monitoring approaches based on different design patterns but the ambition is the same: evolve to collect, analyze and have a global supervision of the infrastructure.

Figure 4.2 illustrates the Agent-Based Monitoring architecture which can be described as follow:

- A monitoring agent present in different VMs deployed for applications. The role of this agent is to collect different data within the operating system(CPU usage, RAM usage, bandwidth usage) or within the application (number of queries for examples) every slot of time.
- A Monitoring platform which role is to receive these informations via various protocols (SNMP, MQ,etc), process and store them[2].

The size of traffic occasioned by this data exchange basically depends on the number of information to be collected, the predefined monitoring period and especially the number of agent present in the infrastructure.

This architecture has been adopted by the orchestration platform Cloudify which is, as we have said, an NFV orchestrator implementing the management and orchestration component within the ETSI standard [77]. The monitoring platform is a module directly integrated in the orchestrator which installs en each deployed VM a plugin called diamond, this latter is the agent responsible for the monitoring [78].

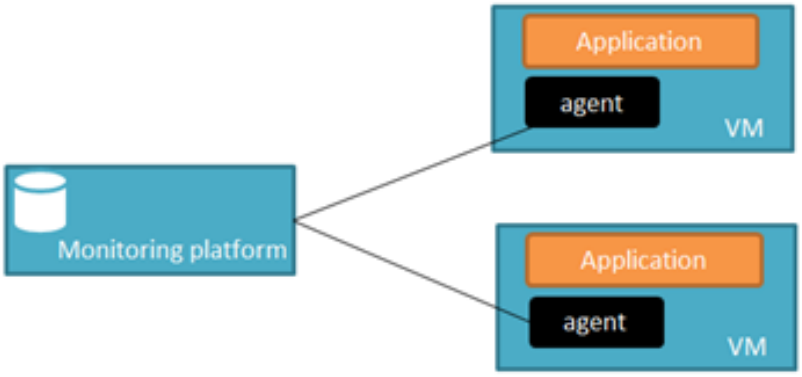

FIGURE 4.2: Agent-Based Monitoring .

VM-based Monitoring Solution Figure 4.3 illustrates the VM-based Monitoring architecture which can be described as follow:

- A dedicated VM for monitoring which can be realized by simple port mirroring which can inject some traffic into this dedicated VM performing some monitoring tasks such as performance, or by using an external monitoring software like Nagios [79] for supervision and Cacti for metrology [80].

[2]https://wiki.monitoring-fr.org/supervision/snmp

- A Monitoring platform which role is the same of the first solution, notably to collect and store informations.

In this case, the size of traffic generated by this data exchange depends on the number of information to be collected and the predefined monitoring period, and especially the number of monitoring VMs present in the infrastructure.

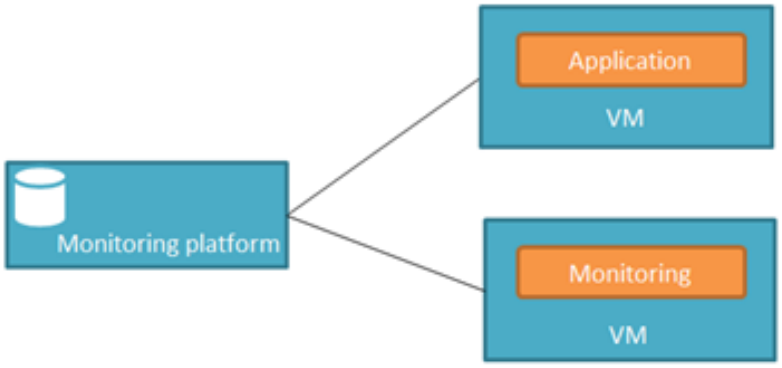

FIGURE 4.3: VM-Based Monitoring .

Under the second approach, we describe in the following the impact of monitoring VMs on the peering link connecting the server to the network, in particular on the perturbation experienced by other VMs.

4.3.2 The impact of monitoring traffic:

The motivation for studying the impact of monitoring traffic elaborated in [81] stems from the important role of monitoring in NFV dedicated infrastructure.

If we assume that a data center hosts monitoring probes embedded in virtual machines or containers, these probes will very likely permanently send traffic (e.g., analytics or traffic reports to supervision platforms). Monitoring traffic will in turn impact traffic generated by applications. We assume that the bandwidth of the link connecting the data center to the network is shared by flows generated by applications according to the processor sharing discipline. This is a classical hypothesis, when information is transmitted by using TCP. We then obtain a processor sharing system, where permanent customers correspond to monitoring flows and jobs are application (regular) flows.

Processor Sharing (PS) is a classical queuing discipline introduced by Kleinrock in the 60' [82] (see also the standard book [83]) to model resource sharing in computer networks. The $M/G/1$-PS queue has subsequently been extensively

studied in the queuing system literature, notably by Ott [84], Schassberger [85] and Yashkov [86], who derived the distribution of the sojourn time conditioned on the service time request of a tagged customer.

In the analysis driven in [81], we consider an $M/M/1$-PS system with permanent customers, denoted, for short, by $M/M/1$-PS-K.

A classical modeling assumption consists of stating that regular flows appear according to a Poisson process. To simplify the system, we further assume that the volume of data to transmit per regular flow is exponentially distributed. These assumptions lead us to consider an $M/M/1$-PS-K queue, where K is the number of monitoring flows in the system. The problem investigated in this study is to understand the impact of permanent flows on the duration of regular flows, notably the potential degradation in terms of quality due to monitoring.

The $M/M/1$-PS-K exhibits the remarkable fact that the decay rate of the sojourn time of a customer in the system depends on the number K of the permanent customers up to a certain threshold and then becomes identical to that of the sojourn time of a customer in a regular $M/M/1$-PS queue. From a practical point of view, we have noticed that the presence of permanent customers is especially sensitive for moderate loads; their presence has a lesser impact for high loads.

4.4 Scheduling feature: use case ONAP

We presented in Section 4.2 several orchestration solutions for NFV. Each orchestration solution has defined its own architecture and objectives, but the main technical challenge is the same: Provisioning an end-to end network service that involves the creation of an IT infrastructure followed by the instantiation of all its necessary components. Hence, there is a clear need for finding algorithms for resource allocation to be encapsulated within the scheduler engine of the orchestration platform. In the following, we focus on the scheduling feature or, in other words and from orchestration point of view, on resource allocation and algorithms within orchestration framework.

The scheduling functionality is the core component of an orchestrator in charge of the scheduling of requests to instantiate VNFs. To better understand this feature, we propose to study a use case in this section, notably the Open Network Automation Platform (ONAP). Under the Linux Foundation, the newly formed project ONAP was created by merging two of the largest open source networking initiatives: ECOMP and Open Orchestrator project (Open-O). By taking

benefits of both projects, ONAP is based on a unified architecture and implementation to deliver an open platform enabling end users to create their own VNFs. The platform aims at automating, orchestrating and managing VNFs and network services.

4.4.1 Architecture of ONAP

The ONAP project has defined its own unified architecture and communication logic across components. The main visible advantage of the ONAP platform is the flexible and extendable architecture which supports the addition of new components. From a general point of view, the complete ONAP architecture can be split into two basic groups.

The main role of the design time environment is to describe the design functions through a graphical studio, the so-called the ONAP Portal. Basically, this component defines recipes for instantiating, monitoring and managing VNFs and services. It is also responsible of the distribution of these specific design rules into the execution time component. The execution time framework contains metadata driven modules enabling VNF configuration and instantiation and delivers real-time view of available resources and services.

To further analyze the ONAP architecture, we take a closer look at the two components described above. The design time framework consists of the following subcomponents:

1. Service Design and Creation (SDC): The SDC is considered as the ONAP design tool. Based on meta data description, SDC is an environment that entirely describes how VNFs or services are managed. It also describes multiple levels of assets, including resources such as cloud or network resources (compute, storage, network connectivity functions, etc.) and services described by means of resource requirements.
2. Policy Creation: This subsystem of ONAP contains a set of rules defining control, orchestration and management policies. VNF placement rule is the policy that specifies where VNFs should be placed respecting some constraints such as affinity rules.

Concerning the execution time framework, the core subsystems are:

1. Active and Available Inventory (AAI): The main role of this component is to provide an updated global view of inventory and network topology. As

changes are made within the cloud, AAI is continually updated to provide a real-time view of the topology and the underlying available resources.

2. Controllers: A controller is an entity in charge of managing the state of a single resource that can be network, application or cloud resource. ONAP uses multiple controllers to execute resource's configuration and instantiation such as the Network Controller for configuration of the network and management of VNFs or the Application Controller for management of more complicated VNFs and services. Both controllers are based on the Opendaylight platform. An additional controller is used for infrastructure's orchestration, in particular, to manage resources within the cloud's infrastructure (compute, storage, etc.). The latter orchestration is based on the orchestration engine of the cloud provider's management platform, namely the Openstack platform.

3. Master Service Orchestrator (MSO): From the top level, the MSO handles capabilities of end-to-end service provisioning. This master orchestrator is based on the underlying controllers described above.

4. Data Collection, Analytics and Events (DCAE): The primary role of the DCAE is to collect telemetry from VNFs and deliver a framework for analytic applications to detect network anomalies and publishes corrective actions.

Figure 4.4 illustrates the simplified architecture of the ONAP project with the main core components described above.

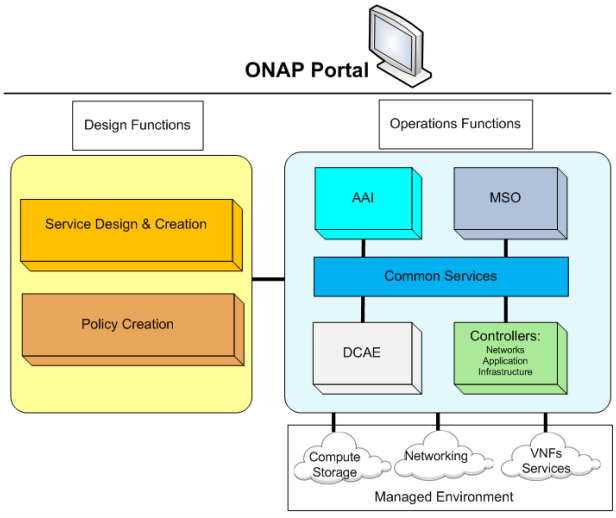

FIGURE 4.4: ONAP Simplified Architecture.

4.4.2 Resource allocation principle under ONAP

Resource allocation consists in making placement decisions, in particular, determining where VNFs should be placed. In the current ONAP architecture design, placement decisions are made by the single infrastructure controller, namely Openstack. Based on a heuristic algorithm, Openstack scheduler favors those servers with the largest amount of available resources. To perform a placement request, the Openstack controller first collects informations stored in the AAI component in order to take the appropriate placement decision. Once this decision is made, the placement execution is also done by the same subsystem, notably the execution time component.

It is worth noting that the ONAP architecture handles the VNF placement and the resource allocation in a centralized fashion. The current implementation does not support any multi-site features, since resource allocation is made without taking geographical location into account. Furthermore, Multi-VIM is not supported; this means that no constraints on which cloud and which site the elements of a VNF should be placed can be specified.

The ONAP community is aware of the fact that both multi-site and multi-VIM are fundamental requirements of service provisioning. To cope with these challenges, the community propose an optimization framework to replace resource allocation functions. The optimization framework will allow designers to specify placement constraint considering VNF's needs such as geographical constraint meeting latency requirements or level of reliability that only some cloud providers can meet. The objective function linked to the service model can be also stated aiming for example to minimize costs or latency.

Once constraints and objective function specified, an optimization problem characterizing the placement demand would be generated, and then automatically solved by the execution time module. Optaplanner[3] is considered as a likely candidate for a pluggable solver by the ONAP community; it could be used to implement the last step of the above procedure.

We believe that ILP system may be very time consuming to solve even for a very small network which does not correspond to the reactivity and agility aspects of NFV.

[3] https://www.optaplanner.org/

To revisit the scheduling logic, the present work proposes a solution to allowing each placement demand to be characterized with geographical and administrative constraints to support multi-site and multi-VIM frameworks.

4.4.3 Proposal: Dynamic Adaptive Placement for ONAP

4.4.3.1 Preliminary considerations

A VNF is in general composed of several components (also called sub-functions or microservices), which execute tasks located at different functional levels of the network, some being part of the data plane (i.e., manipulate user's packet data streams), while some others are part of the control plane of the network. This can be illustrated for a virtual Evolved Packet Core (vEPC): Mobility Management Entity (MME) and Home Subscriber Server (HSS) are in the control plane while Servicing/Packet Gateway (S/PGW) are part of the data plane [87] .

Considering all these requirements and the future "in network" cloud computing infrastructure, we propose in this Section a potential solution for dynamic placement of VNFs, which can be directly used in the context of ONAP without any adjustment. We begin in the following with the considerations that have been taken into account when developing our solution.

Taking into account the performance requirements Data plane components may present stringent requirements in terms of latency in order not to introduce unacceptable delays in the delivery of data. These components should preferably be instantiated along the data path, which is fixed by the routing algorithm. This is typically the case for RAN functions, which could be placed at different data centers but definitely along the data path (say, in a BBU hostel for encoding/decoding functions) and in a regional data centers for RLC and PDCP functions. Components with similar goals (e.g., firewalls) could be co-located to prevent unnecessarily delays by placing them at geographically distant locations; this could be required in order to meet global latency objectives.

Control plane components may be more tolerant to delays and could be placed in more centralized cloud platforms. This is notably the case of some functions of the mobile core (e.g., HSS, AAA, etc.).

Taking into account the network architecture The distributed data centers deployed by network operators could be organized into three levels. The

first level (edge level) is close to end-users, typically providing the IP edge of the network. This level could be installed within MCOs in order to host data plane functions, such as Deep Packet Inspection, Firewalls, S/P Gateways, some RAN functions, etc.

Note that the most recent advances in optical technology enables the migration of the current Central Offices (COs), notably hosting Optical Line Terminations, higher in the network, namely in MCOs. The same also applies for radio access, via the separation of Remote Radio Head and Base band Unit functions. Higher concentration levels enable better coordination of resources between access areas and are made possible by the ever growing capacities of High Performance Computing (HPC) platforms.

The second level (regional level) could be installed within the CCOs and would be equipped with important storage and computing capacities in order to host service platforms, CDN servers and some control plane VNF components that would benefit from being distributed (e.g. mobility support). The CCOs are already the current location of the regional PoPs that are deployed at the current edge of a nationwide IP network used by fixed access end-users.

The third level (nationwide level) would be centralized very high in the backbone IP network. Its data centers could host non delay-sensitive applications, regular cloud applications and control plane VNF components that would benefit from being centralized (e.g., HSS).

The distributed data centers architecture under consideration is illustrated in Figure 4.5.

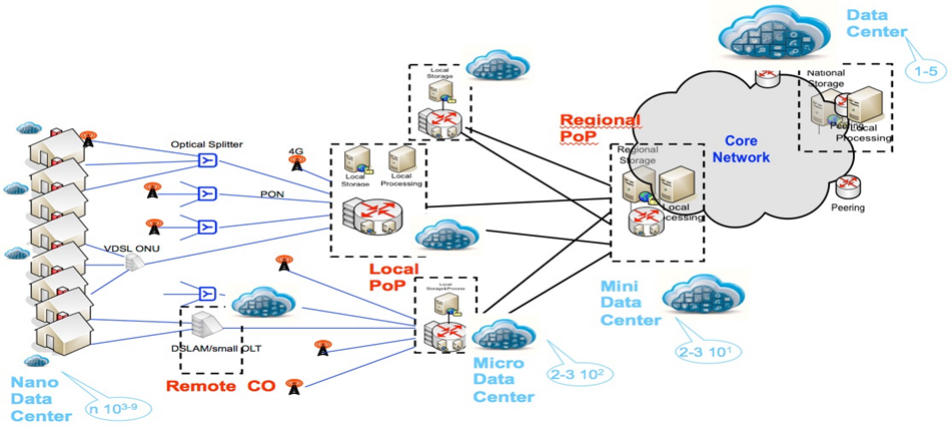

FIGURE 4.5: Three levels network architecture

As mentioned above, MCOs will host data plane VNFs but are also good candidates for hosting MEC(Mobile Edge Computing) applications. Note that it is also possible to envisage the implementation of MEC capacities even closer to end users, say in COs or even in eNodeBs but at a much higher cost [88]. Another trend regarding the future architecture of 5G networks is the so-called Cloud RAN or Centralized RAN (C-RAN). In this architecture, the fronthaul links BroadBand Units (BBUs) and the Radio Units co-located with the antennas. BBUs would be grouped within "BBU hostels" located deeper in the network than the antennas, typically at MCOs. These elements are, thus, to be considered as likely candidates to host BBU hostels, MEC host platforms and edge level data centers.

As long as the end-user does not move from the area controlled by a given MCO, a data plane VNF component should not be displaced as it would increase the consumed bandwidth in the network and increase the latency of the global VNF.

MEC applications on the the contrary could be displaced up to a certain limit depending on the application's latency requirements. Displacement could also be applied to control plane VNF components, which could also be hosted either in CCOs or central data centers.

4.4.3.2 Resource allocation scheme

In view of the previous section, we can reasonably suppose that at the scale of a nationwide network (typically an Autonomous System of a Tier 2 IP network), we have a system with three hierarchical levels of data centers: MCOs, CCOs and centralized data centers. While it is natural to suppose that centralized data centers have huge capacity, CCOs and MCOs may have more limited capacities (because of their number) and will certainly have to implement resource allocation schemes. Moreover, as discussed above MCOs and CCOs will have to cope with two kinds of requests:

- VNFs, which can be split between MCOs (for data plane functions) and CCOs or even centralized data centers (for control plane functions);
- MEC applications, which can be displaced while respecting possibly tight time constraints.

We then propose a resource allocation algorithm, which favors the placement of data plane VNF, while possibly offloading MEC applications (by exploiting the

fact that they can be displaced up some certain limits). For this purpose, we introduce a *target* threshold, which is automatically adjusted according to the arrival rates of the different type of requests. When the occupancy of an MCO or a CCO is above a given threshold, a request is deflected to neighbors as specified in Algorithm 2.

The Distributed resource allocation algorithm that we propose is as follows:

1. When a request arrives in the system using the `getRequest` function, the central dispatcher selects and redirects it to the edge data center, which is the closest to the origin of the request, using the `getClosestDC` function.
2. If the request cannot be accommodated by this edge data center (i.e., when the average resources obtained with `getResources` exceeds the *target*), then it is forwarded to one of its neighbors, which may respect the services' time constraints.
3. To forward the request, the edge data center takes into account the number of redirections received from its neighbors and the time constraints of the request[4]. Specifically, an edge data center maintains a counter, which records the moving average number of deflected requests from its neighboring edge data centers and its own requests' deflection. The edge data center with the smaller number of deflected requests is chosen. If the data center selects itself, it handles the request (using the `Allocate` function).
4. The redirected request is examined by the edge data center, the request is forwarded to. If the request can still not be accommodated, then the previous step is repeated otherwise the request is discarded using the `Discard` function.

The target threshold is continuously adjusted by using a classical hysteresis principle between a maximum and a minimum value.

Indeed, whenever the average load of the data center exceeds the maximal threshold, the target is reduced to augment deflections. Similarly, when the average load is below the minimal threshold, the target is increased to reduce deflections.

Algorithm2 describes the offloading strategy that we propose.

[4]The current data center selects a sub-list of data centers respecting the request criterion using the `GetNeighborsIdx`.

Algorithm 2 Dynamic Services' Offloading

```
1: procedure FORWARD(DC_cur, DC_ori, Rq)
2:     if getResources(DC_cur) < target then
3:         if isAvailable(DC_cur, Rq) then
4:             Allocate(DC_cur, Rq), return
5:         end if
6:     end if
7:     Rq.latency ← Rq.latency + l_{cur,ori}
8:     Rq.TTL ← Rq.TTL − 1
9:     J ← GetNeighborsIdx(DC_cur, Rq) ∪ {cur}
10:    dst ← arg min_{j∈J}{d_{j,cur}}
11:    if dst = cur then
12:        if isAvailable(DC_cur, Rq) then
13:            Allocate(DC_cur, Rq)
14:        else
15:            Discard(Rq)
16:        end if
17:    else
18:        FORWARD(DC_dst, DC_cur, Rq)
19:    end if
20: end procedure
21:
22: while True do
23:    DC_cur ← getClosestDC(getRequest())
24:    FORWARD(DC_cur, DC_cur, Rq)
25: end while
```

4.4.4 Performance Evaluation

The performance evaluation of our strategy has been done via simulations using a discrete event simulator implemented in MATLAB. We compared results obtained from our strategy to those obtained from the Openstack strategy currently adopted by the ONAP platform.

4.4.4.1 Simulation settings

To simulate the decentralized cloud, we have considered the realistic network of Orange that we described in Chapter3. The infrastructure under consideration consists of three level: Edge level (MCO), regional level (CCO) and nationwide level (Centralized Cloud Platform). MCO is about 100 km from CCO that is connected to a big centralized data center at a distance of 300 km. We have considered $N = 21$ data centers with different capacities deployed into the three levels described above. The system under consideration as well as Availability Zones defined for the Openstack strategy are illustrated in Fig. 3.13.

We have considered 3 types of requests, which arrive according to Poisson processes and require to be executed at different functional levels of the network. Data centers hosted at the MCO level might intercept the 3 different types of requests: (1) Data plane functions that must be installed only within MCOs; (2) Control plane functions and MEC applications that are more delay tolerant and might be displaced if needed.

Apart from the data plane functions, centralized cloud platform and data centers deployed at the CCO level intercept control plane functions and MEC applications considered very volatile when compared with VNF, with larger arrival rates of requests and shorter holding times of resources. Requests are expressed in terms of CPU only. Results can be generalized to multiple resources scenario as we demonstrated in our previous work [67].

The global load of the system (data centers) is defined as

$$\rho \stackrel{def}{=} \frac{1}{N} \sum_{j=1}^{N} \frac{\rho_j}{C_j}$$

where $\rho_j \stackrel{def}{=} \lambda_j/\mu_j$ is the load of Data Center j (for short, DCj) with capacity C_j and λ_j and $1/\mu_j$ represent the arrival rate and the exponentially distributed holding time of resources at DCj, respectively. Data centers (DCs) are unevenly loaded; we only consider the global load ρ of the system given that some DCs are overloaded while some others are underloaded. In the simulations, we have taken the capacities of MCOs equal to 200 CPU units, that of CCOs to 500 CPU units and that of the centralized data center equal to 800 CPU units.

In order to compare our strategy against that currently used in ONAP, basically the Openstack strategy, we introduce the average blocking rate of requests defined as the fraction of requests, which are eventually rejected by the system:

$$\overline{\beta} = \frac{1}{\Lambda} \sum_{j=1}^{N} \lambda_j \beta_j$$

where $\Lambda = \sum_{j=1}^{N} \lambda_j$ is the global arrival rate and β_j is the blocking rate of requests originally arriving at DCj. More precisely, β_j is the fraction of requests, which are originally arriving at DCj but eventually not accepted by the system (even after deflection).

4.4.4.2 Results and Discussion

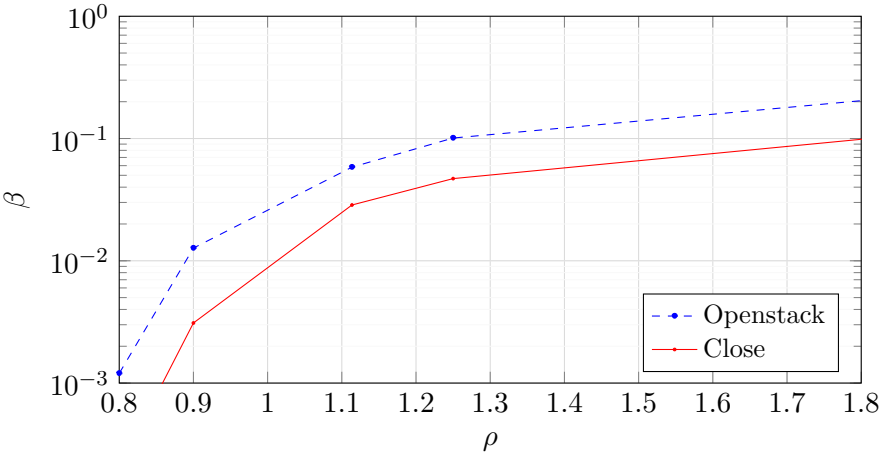

FIGURE 4.6: Blocking rates For Control Plane Functions.

Blocking probabilities of Control Application Figure 4.6 compares the average blocking rates for control plane functions under our strategy while setting static and dynamic adaptive threshold against those obtained with the Openstack strategy. Static thresholds are set equal to 90 % of the capacity of the data center. This figure displays the blocking probability versus traffic intensity and simulation results are averaged to obtain confidence intervals with a 95% confidence level. The results show that our strategy significantly reduces blocking of requests when compared to the Openstack one specifically when the system is overloaded. The more the system is loaded, the better performance is obtained.

Blocking probabilities of MEC Application Figure 4.7 shows that results are qualitatively the same for MEC applications. Static threshold setting yields a performance comparable to that obtained with the dynamic threshold. It is noteworthy that both strategies do not take into account informations about the occupancy of the system when placing requests which requires less overhead.

Blocking probabilities of Data plane functions To further evaluate the system in terms of blocking, we compare the average blocking rates for data plane functions under the different strategies.

Figure 4.8 shows that our strategy with dynamic threshold yields the best performance for data plane functions. This is explained by the fact that thresholds are limiting the acceptance of MEC and control plane applications in data centers hosted at the CCO level and favors the placement of data plane functions at this level. We can also verify the impact of the adaptive threshold when compared

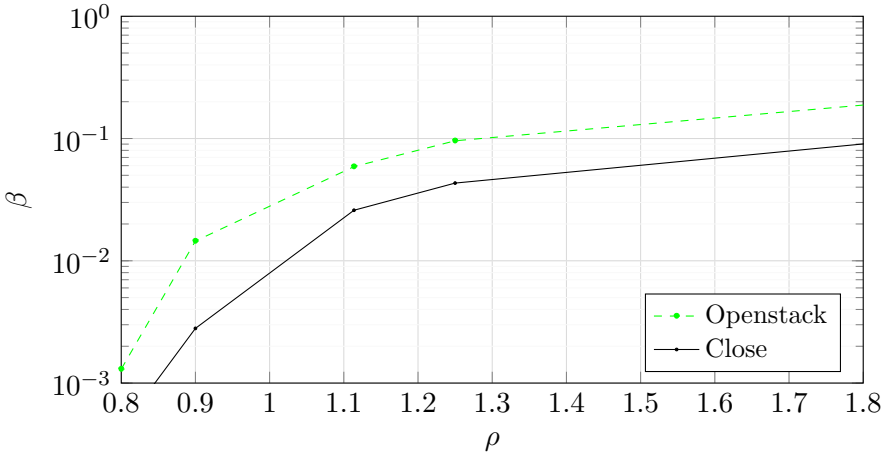

FIGURE 4.7: Blocking rates For Mec Applications.

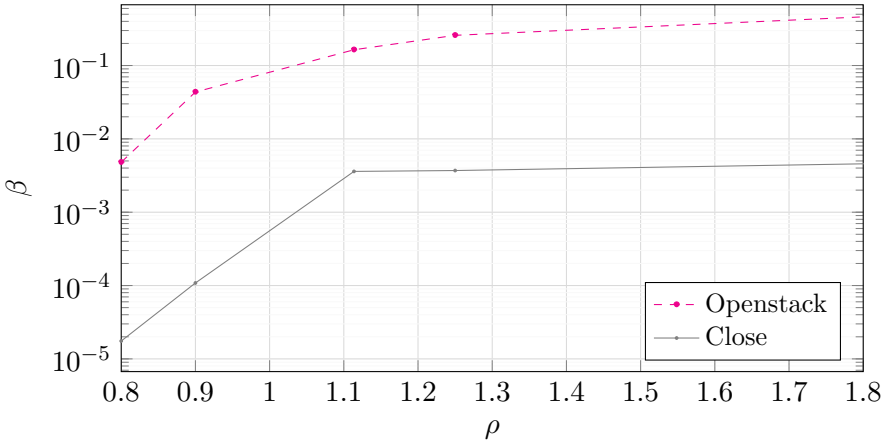

FIGURE 4.8: Blocking rates For Data Plane Functions.

to static settings which confirms that thresholds must be set according to the system load.

Adaptive threshold variation Figure 4.9 shows the variation of the load within a data center as well as the variation of the corresponding target threshold under our dynamic adaptive offloading strategy. We verify that the threshold value is adapting dynamically as a function of the load conditions during the simulation time. Furthermore, We verify that the threshold tends to decrease as soon as we have a maximum load and vice versa. Hence, we can say that thresholds limiting the acceptance of requests are dynamic in this variant of algorithm.

In the next section, we investigate another way to use thresholds. We prove that thresholds based on optimization criteria, namely the global blocking rate of the

algorithm ameliorate the collaboration between data centers. Furthermore, We use Genetic Algorithm (GA) to get over this threshold problem.

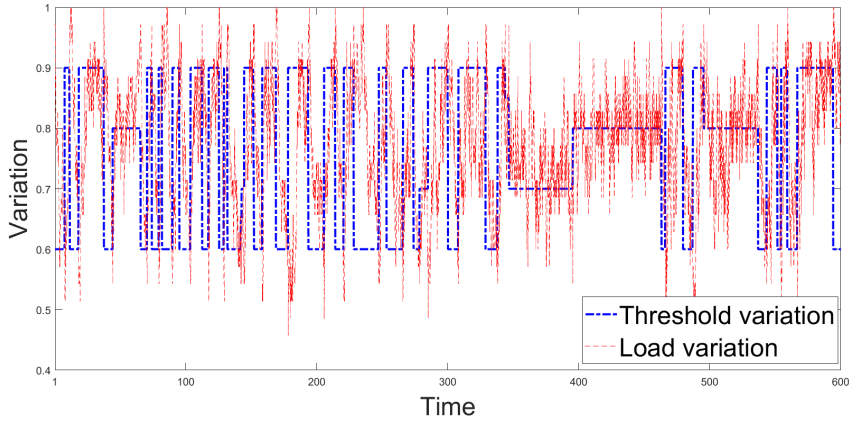

FIGURE 4.9: Adaptive threshold variation according to the current load

4.4.5 Threshold Optimization by Genetic Algorithm

We present in this section another way to use thresholds under the offloading approach that we introduced in the previous section. We propose to set an adjustable threshold for each data center limiting the acceptance of the originally arrived requests in order to better collaborate with the neighbors. We use the GA approach to set optimal offloading threshold.

4.4.5.1 Genetic Algorithm

Genetic algorithms are optimization strategies based on techniques which are derived from genetics and evolution mechanisms of nature such as crosses, mutations, selections, etc ... These algorithms belong to the larger class of evolutionary algorithms (EA)[89].

GA is a good approach to resolve the VM placement problem which enables the optimization of several behaviors. Authors in [90] proposed a genetic algorithm to resolve the VM placement while minimizing the energy consumption in physical machines as well as the communication network in a data center.

A multi-objective genetic algorithm is proposed and compared to other approaches in [91]. Contrary to the majority of research works, this work focuses on minimizing two criteria, notably the total resource wastage and power consumption.

In [92] Parallel Genetic Algorithm was proven more efficient and faster than the simple GA to resolve the VM placement problem. The discussed optimization criterion is the utilization rate of the resource which has to achieve the maximum value.

Generally speaking, the GA aims at optimizing a given function based on the evolution of a population of solutions. Each candidate solution is represented by a chromosome formed by a set a genes and evaluated based on a score given by the fitness function.

By applying one of the following genetic operators*selection*, *mutation* or *crossover*, the GA creates a new population of solutions based on the previous one.

- The *Selection* is a key operator for the GA. According to the scores obtained by each individual of the population, the most pertinent candidates are selected for the next generation.
- The *Mutation* operators aims at introducing random changes in several genes in a chromosome in order to create a new individual.
- The *Crossover* operators creates 2 new individuals from 2 old ones. These old chromosomes ares referred to as the parents.

The population created in the previous step based on the genetic operators is next evaluated based on the given fitness function. Parents are then selected based on the fitness in order to create the new population.

The principle of a basic genetic algorithm are illustrated in Figure 4.10.

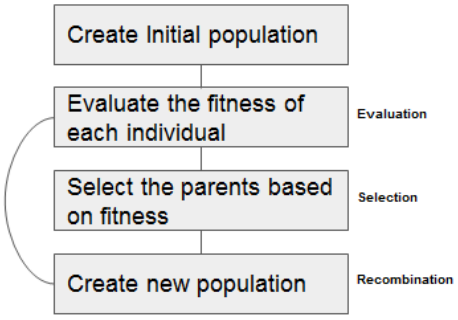

FIGURE 4.10: Principle of a basic GA.

4.4.5.2 Optimal Offloading Threshold by Genetic Algorithm

Methodology The resource allocation strategy that we proposed in the Section 4.4.3.2 is based on collaboration between small data centers. This collaboration is ensured by the offloading strategy between neighbors which relies on deflexion informations.

By limiting the acceptance of the originally arrived requests at each data center, collaboration can be ameliorated which is reflected by the global blocking rate that we will evaluate later. To this end, we set a threshold for each data center which, based on the local load conditions, accept or forward the originally arrived requests to its neighbors even if there is enough resources to accommodate it. This scenario is illustrated in Figure 4.11 where each color represents a flow of requests arriving initially at a given data center, and forwarded or not based on the corresponding threshold.

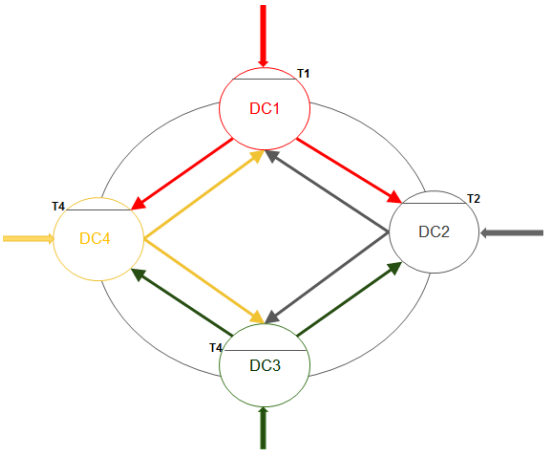

FIGURE 4.11: Methodology of Threshold Optimization by GA .

In order to set optimal offloading thresholds, we propose to use the GA method. The same approach was discussed in [93] which aims to fix optimal interference threshold setting in cognitive radios based on GA implementation.

The goal of the algorithm that we propose is to select the optimal threshold among all of the possible thresholds $T \in]0, 1]$ for each data center. A similar strategy for Image segmentation threshold was introduced in [94].

We describe in the following the basis of our GA proposal; notably the score and how it is calculated.

a) Fitness Let β be the global blocking rate of our collaboration strategy, the proposed fitness function is defined as:

$$F = 1 - \beta$$

The optimization criterion is hence to maximize the fitness function F.

b) Evaluation Function

To evaluate a potential solution of thresholds, we conduct a simulation of our offloading strategy with these threshold as parameters in order to calculate the global blocking rate and evaluate the score of this candidate solution, namely the fitness. The simulation is performed for one million arrival requests at each data center.

This evaluation function allows the GA to create a new generation that will be evaluated in the next iteration.

Steps of our proposed GA are described in the Figure 4.12.

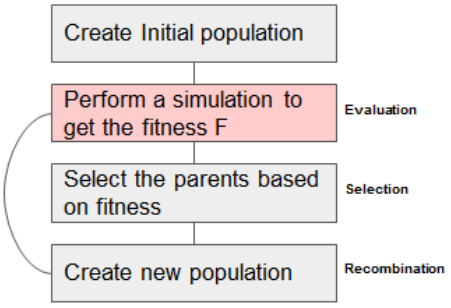

FIGURE 4.12: GA for Optimizing Thresholds.

Implementation The implementation of our approach involves two steps:

1. **GA**
   For the GA implementation, we used the *TCL* programming language which is a scripting language suitable for testing in many fields such as networking, administration...etc.[5] In addition to this language, we used an extension that enables the GA processing in TCL. This extension consists on the *Genetic Algorithm Utility Library (GAUL)* which is a library dedicated to ease the GA development[6]. This library implements the most

[5] https://www.tcl.tk/
[6] http://tcl-gaul.sourceforge.net/

common of the genetic operators, notably *Crossover*, *Mutation* and *Evaluation*. Our GA implementation calls these operators from the *TCL GAUL* :

- **Seed**: This procedure is called before the evolution starts in order to initialize all the members of the population. In our case, the population is the set of thresholds which are randomly initialized (values generated are between 0 and 1).
- **Generate**: This procedure is executed at the end of each generation.
- **Evaluate**: This procedure determines the fitness of each member of the population. In our case, the simulation that we will describe later is performed to evaluate the fitness.
- **Mutate**: This procedure is executed to mutate a single member.
- **Crossover**: This procedure is invoked to breed two members.

2. **Simulation**

The scenario of simulation consists of a mesh network composed of N data centers located at the edge of an Autonomous System. We assume that clients request only one type of resource and data centers are with identical capacity C. The analysis of blocking could easily be extended to multiple resources. We focus only on one resource (typically CPU resources).

The jth data center DCj receives a flow of requests for a fixed amount of resources, taken as unity. Requests at data center DCj are assumed to arrive according to a Poisson process with rate λ_j and the holding time of resources is with mean $1/\mu_j$. The system under consideration is illustrated in Figure 4.13

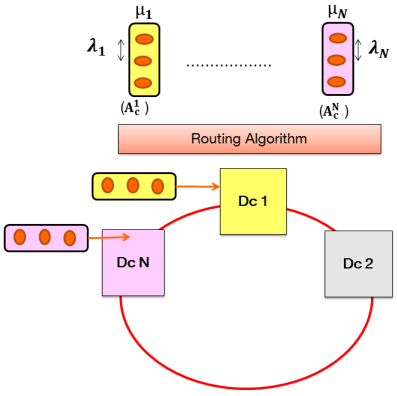

FIGURE 4.13: Simulation Model.

The global load of the system is defined as

$$\rho \stackrel{def}{=} \sum_{i=1}^{N} \frac{\rho_i}{NC}$$

where $\rho_j \stackrel{def}{=} \lambda_j/\mu_j$ is the load of data center DCj.

In order to evaluate each potential solution while iterating the GA, we introduce the average blocking rate defined as the fraction of requests which are eventually rejected by the system and given by

$$\overline{\beta} = \frac{1}{\Lambda} \sum_{j=1}^{N} \lambda_j \beta_j,$$

where $\Lambda = \sum_{j=1}^{N} \lambda_j$ is the global arrival rate and β_j is the blocking rate of requests originally arriving at data center DCj.

We implement our simulation based on the C programming language.

4.4.5.3 Results and Discussion

To evaluate the performance of our proposed strategy, we performed multiple steps:

1. We fix load conditions of the data centers and perform simulation without considering the threshold ($T = 1$ for each data center) in order to obtain the global blocking rate that we will compare to the one we obtain with the optimal threshold given by the GA.
2. We perform the GA which calls the simulation with the same load conditions of the previous step. We obtain then optimal threshold for these load conditions.
3. We finally perform the simulation with the threshold given by the GA as parameters in order to compare the blocking rate with the one given by the first step, notably where there is no threshold.

Results In order to compare blocking rates, we considered a system composed of $N = 4$ data centers under several scenarios where we changed the load conditions.

1. **Scenario 1** : We consider in this scenario mixed load conditions(underload and overload load conditions) which are given in Table 4.1. Optimal thresholds given by the GA for these load conditions are given in the same Table.

TABLE 4.1: Load Conditions and Optimal Thresholds.

parameter	value
(ρ_1, T_1)	(2,1)
(ρ_2, T_2)	(0.9,0.72)
(ρ_3, T_3)	(0.8,1)
(ρ_4, T_4)	(0.75,0.34)

We compare the global blocking rate given by simulation where considering thresholds given by the GA against those given where we do not consider threshold. Simulation with optimal thresholds is referred to as Optimal Offloading. Results are given in Table 4.2.

TABLE 4.2: Blocking Rates.

Offloading	Optimal Offloading
0.107118	0.072961

2. **Scenario 2** : Load conditions for this scenario and the optimal thresholds, results of the GA, are given in Table 4.3.

TABLE 4.3: Load Conditions and Optimal Thresholds.

parameter	value
(ρ_1, T_1)	(0.88,0.97)
(ρ_2, T_2)	(1.37,0.66)
(ρ_3, T_3)	(0.83,0.96)
(ρ_4, T_4)	(0.12,0.12)

The comparison of blocking rates is given in Table 4.4.

TABLE 4.4: Blocking Rates.

Offloading	Optimal Offloading
0.000588	0.000216

3. **Scenario 3** : For this scenario, we consider load conditions that are given in Table 4.5, in addition to the optimal thresholds obtained with the GA.

TABLE 4.5: Load Conditions and Optimal Thresholds.

parameter	value
(ρ_1, T_1)	(2,1)
(ρ_2, T_2)	(0.9,0.53)
(ρ_3, T_3)	(2.5,1)
(ρ_4, T_4)	(0.75,0.69)

As in the previous scenarios, we compare blocking rates given with the optimal thresholds against the classical offloading strategy. Results are given in Table 4.6.

TABLE 4.6: Blocking Rates.

Offloading	Optimal Offloading
0.202098	0.165314

4. **Scenario 4** : For this last scenario, we also consider mixed load conditions with parameters given in Table 4.7. Optimal thresholds are given in the same Table.

TABLE 4.7: Load Conditions and Optimal Thresholds.

parameter	value
(ρ_1, T_1)	(2.5,1)
(ρ_2, T_2)	(1.37,0.24)
(ρ_3, T_3)	(0.8,1)
(ρ_4, T_4)	(0.75,0.88)

As in the previous scenarios, we perform simulation first without considering threshold in order to compare blocking rates to those given with optimal thresholds set by the GA. Results are given in Table 4.8.

TABLE 4.8: Blocking Rates.

Offloading	Optimal Offloading
0.167588	0.122392

Discussion In the previous section, we compared global blocking rates of our offloading strategy with optimal thresholds given by the GA against the simple offloading strategy with no thresholds under 4 settings for load conditions.

As expected, the policy with optimal thresholds is much better than that without considering limitation and reduce the blocking significantly. This can be noticed for the several load conditions that we have considered. In fact, with the combination of thresholds given by the GA, our offloading strategy yields better performance.

We observe that the adjustment of thresholds has a real impact on the blocking rates and significantly ameliorate the collaboration between neighbors. We also conclude that the GA approach presents a good technique revealing practical insights into parameterizing distributed cloud infrastructure for network operators as it gives solutions to reduce global blocking of the system in a reasonable time.

4.5 Summary

To accelerate the transition to NFV, several projects have recently emerged to orchestrate the deployment of VNFs inside a network. In this Chapter, we presented an overview of the NFV MANO framework delivered by ETSI as well as several orchestration projects compliant with this framework.

We then investigated two of the core functionalities that have to be encapsulated within the orchestration framework notably the monitoring as well as the scheduling feature.

To explore the monitoring feature, we presents an overview of a theoretical analysis conducted to study the impact of the monitoring on the network traffic.

Finally, we presented the ONAP software, a solution dedicated to manage NFV infrastructure, as a use case to explore the scheduling feature in the context of NFV. We highlighted resource allocation's issues within the current release of this platform. It turns out that the way VNFs's placement is handled does not fit with critical requirements of NFV environments, notably, geographical location and latency constraints. Hence, we proposed a VNF's placement strategy that can be used in the context of ONAP. Contrary to the current approach of VNF's placement within ONAP, our solution takes into account latency and geographical constraints. This makes it possible to manage distributed cloud infrastructures with ONAP. Our results show that the proposed scheme improves

the performance in terms of VNFs acceptance while requiring less overhead when compared to the Openstack-based approach adopted in the current version of ONAP.

Chapter 5

Conclusion

Contents

5.1 Main Contributions

The aim of this thesis was to study resource allocation in the future telco infrastructure.

First of all, we have identified the new challenges of resource allocation raised by the transformation of the network operator infrastructure driving by virtualization.

After reviewing the state of the art, that covered virtualization paradigm as well as resource allocation strategies in traditional cloud, we proposed an analytical model to estimate blocking of cloud requests since that we claim that blocking is a key metric in the telco cloud context. We then evaluate under a centralized approach the most popular strategies of resource allocation based on our model. The second proposal consists of the proposal of an offloading strategy for NFV service in the context of distributed cloud.

In the second part of this thesis, we presented an overview of orchestration platforms for NFV dedicated infrastructure. We reviewed some critical features such as monitoring and scheduling. Furthermore, we proposed a resource allocation strategy to adopt in the context of the most popular orchestrator, namely ONAP.

These resulted in the following contributions.

1. **Blocking Estimator for limited-capacity cloud data centers.**

   We provided an analytical model to estimate blocking probabilities of VNFs requests under Multiple constraints, notably the Mutli-dimension and the multi-class constraint.

   The strength of our model is the ability to consider multiple parameters as it can be generalized to handle multiple resources that may include computing, storage and networking resources. Our model is also able to consider several types of requests which is line with NFV requirements. Our proposed model reveals practical insights into capacity planning of cloud infrastructure and allows network operators to well dimension edge data centers that will be deployed for NFV needs.

2. **Performance analysis of the most used strategies in the technical literature.**

   Thanks to our blocking analysis model, we provided in this thesis a performance evaluation of the most popular strategies of resource allocation adopted in the cloud literature as well as strategies used in the context of VNFs placement.

3. **The proposal of a Costless Resource Allocation strategy for distributed Edge cloud.**

   In the context of telco cloud, we have shown that new challenges raise when compared to traditional cloud platforms, especially with regard to the critical requirements of NFV. Hence, there is a clear need to adapt resource allocation algorithms to this new context constrained by latency and geographical constraints. We propose in this context a well adapted strategy of resource allocation for NFVs deployment which takes into account multiple aspects of the new network infrastructure, notably the massive distribution of data centers as well as its limited resource capacities.

4. **Setting up a testbed platform for exploring monitoring and scheduling features.**

   We have set a test environment for cloud infrastructure involving cloud data centers as well as network topology. This environment enabled verifying hypothesis that we made to study resource allocation, notably the ability of the orchestrator to be aware of the level of the utilization of resource within the underlying infrastructure.

5. **Technical study of resource allocation under the platform ONAP and the proposal of a dynamic Adaptive placement in this context.**

   The technical study of the resource allocation strategy under the orchestration platform ONAP enabled us to identify issues that may affect VNFs performance as this strategy does not take into account critical requirements such as latency. Hence, we proposed an adapted strategy that we prove more efficient when compared to the adopted one. It is noteworthy that our proposed strategy can be used in the context of ONAP without further modification. Furthermore, we propose another way to use optimal threshold within our strategy. We described steps to set optimal thresholds based on the Genetic Algorithm Approach.

5.2 Research perspectives

The study done in the context of this thesis is a good starting point for research concerning optimized resource allocation for telco cloud infrastructure. In this Section, we highlight some open questions and perspectives for future work.

1. **Performance evaluation of resource allocation under another approach of Virtualization: containerization (container-based virtualization).**

   Container virtualization has many benefits when compared to virtual machine virtualization that we only consider in this work [95]. To further validate our contributions, more tests should be performed under a container-based approach.

2. **Implementation of a dimensioning tool for capacity planning of data centers.**

   Based on our blocking analysis theoretical model, a dimensioning tool that can be very useful to dimension massively distributed cloud infrastructure can be implemented.

3. **Test the integration of the proposed resource allocation strategy into ONAP.**

   The performance evaluation of the strategy that we proposed in the context of ONAP can be also conducted by testing its integration in this platform.

5.3 Publications

1. International conferences and workshops (3)

    - Farah Slim, Fabrice Guillemin, and Yassine Hadjadj Aoul,"On Virtual Network Functions' Placement in Future Distributed Edge Cloud", CloudNet , Prague Czech Republic, September 2017
    - Farah Slim, Fabrice Guillemin, Annie Gravey and Yassine Hadjadj Aoul, "Towards a Dynamic Adaptive Placement of Virtual Network Functions under ONAP", SDN/NFV, Berlin Germany, November 2017
    - Farah Slim, Fabrice Guillemin, and Yassine Hadjadj Aoul, "CLOSE: A Costless Service Offloading Strategy for Distributed Edge Cloud", CCNC, Las Vegas United States, January 2018

2. Journals (1)

    - Fabrice Guillemin and Farah Slim, "Sojourn time in an M/M/1 processor sharing queue with permanent customers",STOCHASTIC MODELS

Bibliography

[1] Enabling Cloud RAN Virtualization . 6WINDGate.

[2] UNDERSTANDING THE IOT EXPLOSION AND ITS IMPACT ON ENTERPRISE SECURITY. Fortinet White Paper.

[3] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update . 2016–2021 White Paper.

[4] Joshi Sujata, Sarkar Sohag, Dewan Tanu, Dharmani Chintan, Purohit Shubham, and Gandhi Sumit. Impact of over the top (ott) services on telecom service providers. *Indian Journal of Science and Technology*, 8(S4):145–160, 2015.

[5] Panagiotis Demestichas, Andreas Georgakopoulos, Dimitrios Karvounas, Kostas Tsagkaris, Vera Stavroulaki, Jianmin Lu, Chunshan Xiong, and Jing Yao. 5g on the horizon: key challenges for the radio-access network. *IEEE Vehicular Technology Magazine*, 8(3):47–53, 2013.

[6] Linh Thi Xuan Phan. Real-time network function virtualization with timing interfaces. *CRTS 2016*, page 47, 2016.

[7] Faqir Zarrar Yousaf, Michael Bredel, Sibylle Schaller, and Fabian Schneider. Nfv and sdn—key technology enablers for 5g networks. *IEEE Journal on Selected Areas in Communications*, 35(11):2468–2478, 2017.

[8] Nick McKeown. Software-defined networking. *INFOCOM keynote talk*, 17 (2):30–32, 2009.

[9] Sakir Sezer, Sandra Scott-Hayward, Pushpinder Kaur Chouhan, Barbara Fraser, David Lake, Jim Finnegan, Niel Viljoen, Marc Miller, and Navneet Rao. Are we ready for sdn? implementation challenges for software-defined networks. *IEEE Communications Magazine*, 51(7):36–43, 2013.

[10] Chung-Sheng Li and Wanjiun Liao. Software defined networks. *IEEE Communications Magazine*, 51(2):113–113, 2013.

[11] Yoshihiro Nakajima, Tomoya Hibi, Hirokazu Takahashi, Hitoshi Masutani, Katsuhiro Shimano, and Masaki Fukui. Scalable high-performance elastic software openflow switch in userspace for wide-area network. *USENIX Open Networking Summit*, pages 1–2, 2014.

[12] Daniel King, Adrian Farrel, and Nektarios Georgalas. The role of sdn and nfv for flexible optical networks: Current status, challenges and opportunities. In *Transparent Optical Networks (ICTON), 2015 17th International Conference on*, pages 1–6. IEEE, 2015.

[13] Rich Uhlig, Gil Neiger, Dion Rodgers, Amy L Santoni, Fernando CM Martins, Andrew V Anderson, Steven M Bennett, Alain Kagi, Felix H Leung, and Larry Smith. Intel virtualization technology. *Computer*, 38(5):48–56, 2005.

[14] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *ACM SIGOPS operating systems review*, volume 37, pages 164–177. ACM, 2003.

[15] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy H Katz, Andrew Konwinski, Gunho Lee, David A Patterson, Ariel Rabkin, Ion Stoica, et al. Above the clouds: A berkeley view of cloud computing. Technical report, Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, 2009.

[16] Cloud RAN: Basics, Advances and Challenges. A Survey of C- RAN Basics, Virtualization, Resource Allocation, and Challenges.

[17] Mehiar Dabbagh, Bechir Hamdaoui, Mohsen Guizani, and Ammar Rayes. Efficient datacenter resource utilization through cloud resource overcommitment. In *Computer Communications Workshops (INFOCOM WKSHPS), 2015 IEEE Conference on*, pages 330–335. IEEE, 2015.

[18] Best Practices for Performance Tuning of Telco and NFV Workloads in vSphere . TECHNICAL WHITE PAPER.

[19] Lakshay Malhotra, Devyani Agarwal, and Arunima Jaiswal. Virtualization in cloud computing. *J Inform Tech Softw Eng*, 4(2):136, 2014.

[20] Chuang Lin, Yuan Tian, and Min Yao. Green network and green evaluation: mechanism, modeling and evaluation. *Jisuanji Xuebao(Chinese Journal of Computers)*, 34(4):593–612, 2011.

[21] Georgios Gardikis, Ioannis Koutras, George Mavroudis, Socrates Costicoglou, George Xilouris, Christos Sakkas, and Akis Kourtis. An integrating framework for efficient nfv monitoring. In *NetSoft Conference and Workshops (NetSoft), 2016 IEEE*, pages 1–5. IEEE, 2016.

[22] Rakesh Kumar and Bhanu Bhushan Parashar. Dynamic resource allocation and management using openstack. *Nova*, 1:21, 2010.

[23] Maurice Bolhuis. A comparison between centralized and distributed cloud storage data-center topologies. 2013.

[24] Ian Foster, Yong Zhao, Ioan Raicu, and Shiyong Lu. Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE'08*, pages 1–10. Ieee, 2008.

[25] Praveen Khethavath, Johnson Thomas, Eric Chan-Tin, and Hong Liu. Introducing a distributed cloud architecture with efficient resource discovery and optimal resource allocation. In *Services (SERVICES), 203 IEEE Ninth World Congress on*, pages 386–392. IEEE, 2013.

[26] Yi-Ke Guo and Li Guo. Ic cloud: Enabling compositional cloud. *International journal of automation and computing*, 8(3):269–279, 2011.

[27] Nicolo Maria Calcavecchia, Ofer Biran, Erez Hadad, and Yosef Moatti. Vm placement strategies for cloud scenarios. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 852–859. IEEE, 2012.

[28] Deepal Jayasinghe, Calton Pu, Tamar Eilam, Malgorzata Steinder, Ian Whally, and Ed Snible. Improving performance and availability of services hosted on iaas clouds with structural constraint-aware virtual machine placement. In *Services Computing (SCC), 2011 IEEE International Conference on*, pages 72–79. IEEE, 2011.

[29] Dhaval Bonde. Techniques for virtual machine placement in clouds. *Department of Computer Science and Engineering, Indian Institute of Technology, Bombay, Mumbai*, 2010.

[30] Makhlouf Hadji and Djamal Zeghlache. Minimum cost maximum flow algorithm for dynamic resource allocation in clouds. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 876–882. IEEE, 2012.

[31] Hendrik Moens and Filip De Turck. Vnf-p: A model for efficient placement of virtualized network functions. In *Network and Service Management*

(CNSM), 2014 10th International Conference on, pages 418–423. IEEE, 2014.

[32] Rina Panigrahy, Kunal Talwar, Lincoln Uyeda, and Udi Wieder. Heuristics for vector bin packing. *research. microsoft. com*, 2011.

[33] Oleg Litvinski and Abdelouahed Gherbi. Openstack scheduler evaluation using design of experiment approach. In *Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC), 2013 IEEE 16th International Symposium on*, pages 1–7. IEEE, 2013.

[34] Barnaby Malet and Peter Pietzuch. Resource allocation across multiple cloud data centres. In *Proc. of the 8th Int. Workshop on Middleware for Grids, Clouds and e-Science*, page 5. ACM, 2010.

[35] Sijin He, Li Guo, Moustafa Ghanem, and Yike Guo. Improving resource utilisation in the cloud environment using multivariate probabilistic models. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 574–581. IEEE, 2012.

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

[37] Rogério Leão Santos De Oliveira, Ailton Akira Shinoda, Christiane Marie Schweitzer, and Ligia Rodrigues Prete. Using mininet for emulation and prototyping software-defined networks. In *Communications and Computing (COLCOM), 2014 IEEE Colombian Conference on*, pages 1–6. IEEE, 2014.

[38] Bo Han, Vijay Gopalakrishnan, Lusheng Ji, and Seungjoon Lee. Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine*, 53(2):90–97, 2015.

[39] Philipp Schulz, Maximilian Matthe, Henrik Klessig, Meryem Simsek, et al. Latency critical iot applications in 5g: Perspective on the design of radio interface and network architecture. *IEEE Communications Magazine*, 55 (2):70–78, 2017.

[40] Ranjan Pal, Sung-Han Lin, and Leana Golubchik. The cloudlet bazaar dynamic markets for the small cloud. *arXiv preprint arXiv:1704.00845*, 2017.

[41] Richard Cziva and Dimitrios P Pezaros. Container network functions: bringing nfv to the network edge. *IEEE Communications Magazine*, 55(6):24–31, 2017.

[42] Shuai Zhang, Shufen Zhang, Xuebin Chen, and Xiuzhen Huo. Cloud computing research and development trend. In *Future Networks, 2010. ICFN'10. Second International Conference on*, pages 93–97. Ieee, 2010.

[43] NFVISG ETSI. Gs nfv-man 001 v1. 1.1 network function virtualisation (nfv); management and orchestration, 2014.

[44] Alcatel Lucent. Alcatel-lucent virtualized epc delivering on the promise of nfv and sdn.

[45] Patricia Takako Endo, Andre Vitor de Almeida Palhares, Nadilma Nunes Pereira, Glauco Estacio Goncalves, Djamel Sadok, Judith Kelner, Bob Melander, and Jan-Erik Mangs. Resource allocation for distributed cloud: concepts and research challenges. *IEEE network*, 25(4), 2011.

[46] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53 (4):50–58, 2010.

[47] Ewa Deelman, Gurmeet Singh, Miron Livny, Bruce Berriman, and John Good. The cost of doing science on the cloud: the montage example. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, page 50. IEEE Press, 2008.

[48] S Brintha Rajakumari and C Nalini. An efficient cost model for data storage with horizontal layout in the cloud. *Indian Journal of Science and Technology*, 7(3S):45–46, 2014.

[49] Hamzeh Khazaei, Jelena Misic, and Vojislav B Misic. A fine-grained performance model of cloud computing centers. *IEEE Transactions on parallel and distributed systems*, 24(11):2138–2147, 2013.

[50] Tulin Atmaca, Thomas Begin, Alexandre Brandwajn, and Hind Castel-Taleb. Performance evaluation of cloud computing centers with general arrivals and service. *IEEE Transactions on Parallel and Distributed Systems*, 27(8):2341–2348, 2016.

[51] Hamzeh Khazaei, Jelena Misic, and Vojislav B Misic. Performance analysis of cloud computing centers using m/g/m/m+ r queuing systems. *IEEE Transactions on parallel and distributed systems*, 23(5):936–943, 2012.

[52] Shuyi Yan, Xue Wang, Miguel Razo, Marco Tacca, and Andrea Fumagalli. Data center selection: A probability based approach. In *Transparent Optical Networks (ICTON), 2014 16th International Conference on*, pages 1–5. IEEE, 2014.

[53] Rodrigo AC da Silva and Nelson LS da Fonseca. Algorithm for the placement of groups of virtual machines in data centers. In *Communications (ICC), 2015 IEEE International Conference on*, pages 6080–6085. IEEE, 2015.

[54] Xiaoming Nan, Yifeng He, and Ling Guan. Optimal resource allocation for multimedia cloud based on queuing model. In *Multimedia signal processing (MMSP), 2011 IEEE 13th international workshop on*, pages 1–6. IEEE, 2011.

[55] Hamzeh Khazaei, Jelena Mišić, Vojislav B Mišić, and Nasim Beigi Mohammadi. Availability analysis of cloud computing centers. In *Global Communications Conference (GLOBECOM), 2012 IEEE*, pages 1957–1962. IEEE, 2012.

[56] Ravi R Mazumdar. *Performance modeling, stochastic networks and statistical multiplexing*. Morgan & Claypool, 2013.

[57] Abdul Hameed, Alireza Khoshkbarforoushha, Rajiv Ranjan, Prem Prakash Jayaraman, Joanna Kolodziej, Pavan Balaji, Sherali Zeadally, Qutaibah Marwan Malluhi, Nikos Tziritas, Abhinav Vishnu, Samee U. Khan, and Albert Zomaya. A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems. *Computing*, 98 (7):751–774, 2016.

[58] Xiaoqiao Meng, Vasileios Pappas, and Li Zhang. Improving the scalability of data center networks with traffic-aware virtual machine placement. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9, 2010.

[59] Zahra Abbasi, Meral Shirazipour, and Attila Takacs. An optimization case in support of next generation NFV deployment. In *7th USENIX Workshop on Hot Topics in Cloud Computing*, 2015.

[60] Makhlouf Hadji and Djamal Zeghlache. Minimum cost maximum flow algorithm for dynamic resource allocation in clouds. In *Cloud Computing (CLOUD), 2012 IEEE 5th Int. Conf. on*, pages 876–882. IEEE, 2012.

[61] Mansoor Alicherry and TV Lakshman. Network aware resource allocation in distributed clouds. In *Infocom, 2012 proc. IEEE*, pages 963–971, 2012.

[62] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck, and Steven Davy. Design and evaluation of algorithms for mapping and scheduling of virtual network functions. In *Network Softwarization (NetSoft), 2015 1st IEEE Conference on*, pages 1–9. IEEE, 2015.

[63] Albert Jonathan, Abhishek Chandra, and Jon B. Weissman. Awan: Locality-aware resource manager for geo-distributed data-intensive applications. In *2016 IEEE International Conference on Cloud Engineering, IC2E 2016, Berlin, Germany, April 4-8, 2016*, pages 32–41, 2016.

[64] Kuan-yin Chen, Yang Xu, Kang Xi, and H Jonathan Chao. Intelligent virtual machine placement for cost efficiency in geo-distributed cloud systems. In *2013 IEEE Int. Conf. on Communications (ICC)*, pages 3498–3503, 2013.

[65] Ayoub Bousselmi, Jean François Peltier, and Abdelhadi Chari. Towards a massively distributed iaas operating system: Composition and evaluation of openstack. In *Standards for Communications and Networking (CSCN), 2016 IEEE Conference on*, pages 1–6. IEEE, 2016.

[66] Shanhe Yi, Cheng Li, and Qun Li. A survey of fog computing: concepts, applications and issues. In *Proceedings of the 2015 Workshop on Mobile Big Data*, pages 37–42. ACM, 2015.

[67] Farah Slim, Fabrice Guillemin, and Yassine Hadjadj Aoul. On virtual network functions' placement in future distributed edge cloud. In *6th IEEE International Conference on Cloud Networking, CloudNet 2017, Prague, Czech Republic, September 25-27, 2017*, pages 12–15, 2017.

[68] Openstack Home. docs.openstack.org/developer/nova/filterscheduler.html/.

[69] Richard G Clegg, Carla Di Cairano-Gilfedder, and Shi Zhou. A critical look at power law modelling of the internet. *Computer Communications*, 33(3): 259–268, 2010.

[70] Hassan Hawilo, Abdallah Shami, Maysam Mirahmadi, and Rasool Asal. Nfv: state of the art, challenges, and implementation in next generation mobile networks (vepc). *IEEE Network*, 28(6):18–26, 2014.

[71] Rashid Mijumbi, Joan Serrat, Juan-luis Gorricho, Steven Latre, Marinos Charalambides, and Diego Lopez. Management and orchestration challenges in network functions virtualization. *IEEE Communications Magazine*, 54 (1):98–105, 2016.

[72] CAROLINE Chappell. Nfv mano: What's wrong & how to fix it. *Heavy Reading*, 13(2), 2015.

[73] Mehmet Ersue. Etsi nfv management and orchestration-an overview. In *Proc. of 88th IETF meeting*, 2013.

[74] Ali Al-Shabibi and L Peterson. Cord: Central office re-architected as a datacenter. *OpenStack Summit*, 2015.

[75] Pankaj Berde, Matteo Gerola, Jonathan Hart, Yuta Higuchi, Masayoshi Kobayashi, Toshio Koide, Bob Lantz, Brian O'Connor, Pavlin Radoslavov, William Snow, and Guru Parulkar. Onos: Towards an open, distributed sdn os. In *Proc. of the Third Workshop on Hot Topics in Software Defined Networking*, HotSDN '14, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2989-7.

[76] DR Lopez. Openmano: The dataplane ready open source nfv mano stack. In *IETF Meeting Proceedings, Dallas, Texas, USA*, 2015.

[77] Nicolas Ferry, Alessandro Rossini, Franck Chauvel, Brice Morin, and Arnor Solberg. Towards model-driven provisioning, deployment, monitoring, and adaptation of multi-cloud systems. In *Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on*, pages 887–894. IEEE, 2013.

[78] Carlos Eduardo Fernandez Afonso. An elasticity controller for applications orquestrated with cloudify. 2016.

[79] nagios. https://www.nagios.org//, 2016.

[80] Liu Ying Liu Jing Zheng Haiyan. The application of cacti in the campus network traffic monitoring [j]. *Computer & Telecommunication*, 4:004, 2008.

[81] Fabrice Guillemin and Farah Slim. Sojourn time in an $m/m/1$ processor sharing queue with permanent customers. volume 0, page 1–23, 2017.

[82] L. Kleinrock. Time-shared systems: a theoretical treatment. *J. Assoc. Comput. Mach.*, 14(2):242–251, 1967.

[83] L. Kleinrock. *Queueing Systems*, volume 2. Wiley, New York, 1976.

[84] T. J. Ott. The sojourn-time distribution in the M/G/1 queue with processor sharing. *J. Appl. Prob.*, 21, 2:360–378, 1984.

[85] R.A. Schassberger. A new approach to the M/G/1 processor sharing queue. *Adv. Appl. Prob.*, 16:202–213, 1970.

[86] S.F. Yashkov. Explicit formulas for the moments of the sojourn time in the M/G/1 processor sharing queue with permament jobs. Technical report, arXiv:math/052281, 2005.

[87] Arsany Basta, Wolfgang Kellerer, Marco Hoffmann, Klaus Hoffmann, and Ernst-Dieter Schmidt. A virtual sdn-enabled lte epc architecture: A case study for s-/p-gateways functions. In *Future Networks and Services (SDN4FNS), 2013 IEEE SDN for*, pages 1–7. IEEE, 2013.

[88] Salah-Eddine Elayoubi and James Roberts. Performance and cost effectiveness of caching in mobile access networks. In *Proc. of the 2Nd ACM Conf. on Information-Centric Networking*, ACM-ICN '15, pages 79–88, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3855-4.

[89] Lawrence Davis. Handbook of genetic algorithms. 1991.

[90] Maolin Tang and Shenchen Pan. A hybrid genetic algorithm for the energy-efficient virtual machine placement problem in data centers. *Neural Processing Letters*, 41(2):211–221, 2015.

[91] Yongqiang Gao, Haibing Guan, Zhengwei Qi, Yang Hou, and Liang Liu. A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. *Journal of Computer and System Sciences*, 79(8):1230–1242, 2013.

[92] Zhongni Zheng, Rui Wang, Hai Zhong, and Xuejie Zhang. An approach for cloud resource scheduling based on parallel genetic algorithm. In *Computer Research and Development (ICCRD), 2011 3rd International Conference on*, volume 2, pages 444–447. IEEE, 2011.

[93] Aravind Narayanan Krishnamoorthy, Arun Shivaram Pasupathy, Maheshkumar Mani, Santhoshkumar Krishnamurthi, Sathiesh Kumar Leelakrishnan, and Kotheneth Achuthan Narayanankutty. Optimization of threshold for energy based spectrum sensing using differential evolution. *Wireless Engineering and Technology*, 2(3):130–134, 2011.

[94] Ahmad EL Allaoui and M'barek Nasri. Threshold optimization by genetic algorithm for segmentation of medical images by region growing. *International Journal of Emerging Trends and Technology in Computer Science (IJETTCS)*, 1(2):161–166, 201 2.

[95] Jason Anderson, Hongxin Hu, Udit Agarwal, Craig Lowery, Hongda Li, and Amy Apon. Performance considerations of network functions virtualization

using containers. In *Computing, Networking and Communications (ICNC), 2016 International Conference on*, pages 1–7. IEEE, 2016.

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : Etude et implémentation d'algorithmes de gestion de ressources pour un système d'exploitation de réseau

Mots clés : Cloud, Infrastructure, Algorithmes, Ressources, Réseaux, Virtualisation, Fonction de Réseaux Virtualisée

Résumé : L'automatisation et la capacité de rendre les réseaux totalement programmables est un enjeu majeur dans l'évolution des systèmes de télécommunications.

Rendre les réseaux plus flexibles et proches du système d'information est une question récurrente dans le monde des réseaux depuis de nombreuses années, mais l'émergence de nouvelles initiatives dans le cadre de l'Internet du futur, notamment avec le développement d'outils de gestion associés comme OpenStack, ont permis de développer de nouvelles approches pour la commande et la gestion des équipements de réseau.

Grâce à ces nouveaux paradigmes, le secteur télécom est désormais au cœur d'une transformation digitale. Cette transformation s'appuie sur de nouvelles technologies émergentes telles que la virtualisation, les réseaux définis par logiciels, et le déploiement des services réseaux sur le cloud.

L'objectif de la thèse est d'analyser les modifications qui vont affecter l'infrastructure de l'opérateur dans le but de concevoir des algorithmes de gestion de ressources adaptés au contexte de la virtualisation des fonctions réseaux.

Title : Design and Implementation of Resource Allocation Algorithms for a Network Operating System

Keywords : OpenStack, Virtual Network Function, Virtualization, Network Architecture, Performance Evaluation

Abstract: Network programmability is a major issue in the evolution of telecommunication systems.

However, the emergence of new initiatives in the context of the internet of the future, such as network function virtualization and specialized tools such as Openstack makes it possible to develop new approaches to control and manage the network infrastructure.

This new approach on which telecom operators rely to accelerate their digital transformation will impact not only the way networks are defined but also the main role of the operator, who now has to manage cloud resources in combination with network resources.

The main goal of the thesis is to analyze the modifications that will affect the infrastructure of the operator in order to design resource allocation algorithms adapted to the context of the virtualization of network functions.