SDN/NFV-based Network Slicing Management

Amal Kammoun

▶ **To cite this version:**

Amal Kammoun. SDN/NFV-based Network Slicing Management. Computers and Society [cs.CY]. Université Paris-Nord - Paris XIII, 2019. English. NNT : 2019PA131102 . tel-03385223

HAL Id: tel-03385223

https://theses.hal.science/tel-03385223v1

Submitted on 19 Oct 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE

pour obtenir le grade de

Docteur

délivré par

l'Université Paris 13 & l'Ecole Supérieure des Communications de Tunis

Discipline : Doctorat d'Ingénierie en Informatique

présentée et soutenue publiquement par

Amal Kammoun

le 19 Décembre 2019

SDN/NFV-based Network Slicing Management

Directeur de thèse : **Prof. Nabil Tabbane**
Directeur de thèse : **Dr. Gladys Diaz**
Co-directeur de thèse : **Dr. Nadjib Achir**

JURY

Composition du jury

Lamia Chaari,	Professeur, ISIMS, Université de SFAX , Tunisie	Président
Michelle Sibilla,	Professeur, Université Toulouse III, France	Rapporteur
Kaouther Sethom,	Professeur, ENICarthage, Tunis, Tunisie	Rapporteur
Tijani Chahed,	Professeur, Télécom SudParis, Paris, France	Examinateur
Slim Rekhis,	Professeur, Sup'Com, Tunis, Tunisie	Examinateur
Nabil Tabbane,	Professeur, Sup'Com, Tunis, Tunisie	Directeur de thèse
Gladys Diaz,	MC HDR, Université Paris 13, France	Directeur de thèse
Nadjib Achir,	MC HDR, Université Paris 13, France	Co-Directeur de thèse

Dédicace

Je dédie ce travail à mes chers parents qui ont toujours cru en moi.

Je dédie ce travail à mon cher mari qui m'a toujours soutenu.

Acknowledgments

I would like to express my deepest gratitude to Mr. Nabil TABBANE from SUP'COM (TUNISIA), Mrs. Gladys DIAZ and Mr. Nadjib ACHIR from University Paris 13 (FRANCE) for agreeing to supervise my PhD. I sincerely thank you for all your contributions, recommendations, priceless advice, and encouragement.

I greatly appreciate the support of Mr Abdulhalim DANDOUSH. I am grateful for his help and valuable support.

Besides my advisors, I would like to thank the rest of my thesis committee for accepting to evaluate my work.

I am grateful endlessly for the encouragement and the support of my parents and my husband. I am thankful for their patience and belief on me. This accomplishment would not have been possible without them.

Last but not least, I would like to thank my brothers and my sisters, my wider family, my friends who were a great source of motivation throughout the process of my thesis.

I thank each and every person from whom I received help, without whom this work would have not have been possible.

Abstract

Fifth Generation of Mobile Networks (5G) is proposed to address the future network context. This context is characterized by an everywhere-connected society where not only persons will be connected to the network but also clothes, vehicles and smart objects. Therefore, 5G has to serve an expanding number of users and terminals and thus an exponentially increasing traffic. This traffic will be generated from several emerging services such as e-health, autonomous driving, IoT, etc. These services request different and stringent constraints mainly in terms of *latency*, *reliability*, and *availability*. For instance, for the smart city applications, current network architectures are not able to handle neither the increased number of sensors and IoT devices nor the large amount of data exchanged over the network.

5G networks try to cope with the limitations of current network implementations by proposing a new system aiming to meet new challenges. Indeed, unlike previous technologies, 5G will not only enhance the network system but will also provide an end-to-end infrastructure that will support emergent services and respond to stringent user requirements. The key concepts for the 5G vision are the Software Defined Networking (SDN), Network Function Virtualization (NFV) and Network Slicing technologies. Those paradigms allow the network to provide services for various scenarios under different requirements. They permit to achieve higher performance and flexibility for the network through the introduction of the network programming and the *one size per service* approach instead of the traditional *one size fits all* approach.

In this context, this thesis aims first to propose a new architecture for network slices provisioning and management. We propose the multi-level Delegation Architecture for Network Slicing Orchestration (DANSO) architecture which considers the SDN, NFV and network slicing technologies in order to present a programmable and flexible framework for services provisioning. The second aim is the optimization of the process

of the network slices creation and the admission control of users' request. For this purpose, we propose heuristic algorithms in order to either map the users' demands to existing slices or to create a new network slices. Our algorithms consider the *reliability*, *availability* and *latency* requirements as well as the offered quality by the underlying infrastructure. The third aim is related to the management aspect. In fact, we interest on the management of sudden events that occur in the slice during its running time. In this regard, we study the congestion of the slices and users' mobility events. We propose a fuzzy logic-based algorithm that considers the actual and the predicted load state of the slice in order to perform auto-scaling actions. The future load values are determined using the Support Vector Regression (SVR) technique. Finally we interest on the problem of vertical handover management in the context of network slices. We propose an algorithm that decides when to perform the handover and selects the target slice.

Keywords

SDN, NFV, Optimization, Management, Network Slicing, Reliability, Availability, Prediction, Handover, Mobility, Fuzzy Logic

Résumé

Aujourd'hui, les applications et les services de l'Internet sont devenus de plus en plus complexes et exigeants. Ces nouveaux services, comme la e-health, la communication inter-véhicules et la vidéo à la demande, nécessitent différents modèles pour leurs mises en place et aussi pour la gestion de leurs qualités de service. Plusieurs avancées technologiques se succèdent dans le but de permettre aux opérateurs d'augmenter la capacité de leurs réseaux et aux fournisseurs de services d'introduire des nouveaux services. Bien que l'Internet ait été très efficace en termes de performances du réseau et elle ait été adoptée à grande échelle, elle rend la mise en place et le déploiement de nouveaux services réseaux difficiles et coûteux à cause de son architecture physique. De plus, Le service "best effort" de l'Internet ne permet pas de garantir une qualité de service de bout en bout pour les nouveaux services réseaux.

Les dernières directions lancées dans le contexte de l'Internet du futur reposent sur la mise en place d'une couche logicielle programmable permettant une vue globale du réseau et une gestion dynamique des ressources. Trois nouveaux paradigmes sont aujourd'hui au cœur de cette tendance : le SDN (Software Defined Networking), le NFV (Network Function Virtualization) et le Network Slicing. Ces approches visent à introduire une certaine agilité pour le déploiement et l'évolution des services réseau.

Afin de répondre aux interrogations présentées, nous nous intéressons dans la première partie de la thèse au processus de déploiement des services réseaux dans un contexte évolutif en termes de technologies et d'usages. En effet, nous nous intéressons au processus d'orchestration permettant l'automatisation du processus de création, de monitoring et de gestion continue des services réseaux au niveau de l'architecture proposée. Ce système d'orchestration offre une vue complète et un contrôle total sur l'infrastructure réseau. Il gère dynamiquement les fluctuations de la demande et les événements soudains au niveau du réseau. Dans ce contexte de recherche nous proposons

une nouvelle architecture pour le provisioning des services. Il s'agit d'appliquer les concepts proposés dans le cadre du paradigme SDN/NFV et la technologie « Network Slicing » pour la délivrance et la continuité du service réseau.

Dans la deuxième partie de nos travaux de recherche, nous proposons un nouveau module au sein de notre framework pour l'automatisation et l'optimisation des opérations de création des slices réseaux. En effet, ce module exécute des algorithmes en se basant sur les informations liées au réseau ainsi qu'aux utilisateurs pour avoir les décisions optimales pour la création des slices réseaux. Dans cette partie, notre objectif est de respecter les contraintes des services de 5G en termes de latence, sureté de fonctionnement et disponibilité pour le déploiement des services dans les nœuds du réseau.

La troisième partie de nos travaux est consacrée pour l'étude de la gestion des événements qui occurrent dans les slices tels que la congestion des slices, la dégradation de la qualité de service, la mobilité des utilisateurs, etc. Pour Pallier ces problèmes, nous proposons un algorithme basé sur la logique floue. Cet algorithme analyse la valeur actuelle de la charge ainsi que sa future valeur prédite. Pour la prédiction de l'état de la charge du slice nous utilisons l'algorithme Support Vector Regression (SVR). Nous étudions aussi la mobilité des utilisateurs entre les slices. Nous proposons un algorithme pour la gestion du handover des utilisateurs d'un slice à un autre. L'objectif est de garantir la continuité du service avec la meilleure qualité de service possible.

Mots Clés

SDN, NFV, Optimisation, Gestion, Découpage du Réseau, Fiabilité, Disponibilité, Prévision, Handover, Mobilité, Logique floue

Contents

Abstract v

Résumé vii

Contents ix

List of Figures xv

List of Tables xix

Acronyms xxi

1 Introduction 1

1.1 Context and Problem Statement 1

1.2 Research Challenges 2

1.2.1 Network Slices Isolation 3

1.2.2 Network Resources Optimization 4

1.2.3 Management, Orchestration and Control of Network Slices 4

1.3 Considered Research Aspects 5

1.3.1 Framework Architecture 5

1.3.2 Service Provisioning 5

1.3.3 Network Slicing 5

1.4 Our Contributions 6

1.4.1 New Architecture for Network Slices Deployment 6

1.4.2 Optimization of the Network Slice Deployment 6

1.4.3 Management of End-to-End Network Slices 7
1.5 Organization of the Thesis 7

2 State of the Art on Dynamic Deployment and Management of End-to-End Network Slices 9

Introduction 9
2.1 Related Concepts 9
2.1.1 5G Use Cases Requirements 9
2.1.2 SDN/NFV Paradigms 11
2.1.2.1 Network Function Virtualization (NFV) 12
2.1.2.2 Software Defined Networking (SDN) 12
2.1.3 SDN/NFV Architecture 14
2.1.4 Network Service Chaining 16
2.1.5 Network Cloudification 16
2.1.6 Network Slicing 17
2.2 Related Work to Framework Architectures for Network Slices Implementation 18
2.2.1 5G Platform Design 18
2.2.2 5G Infrastructure Public Private Partnership (5G-PPP) Projects 19
2.2.2.1 5G SliceNet 20
2.2.2.2 Sonata 20
2.2.2.3 5GEx 20
2.2.2.4 5G NORMA 21
2.2.2.5 Discussion 21
2.2.3 Novelty 22
2.3 Related Work to Optimization Algorithms for Network Slices Deployment 23
2.3.1 Exact Solutions 24
2.3.2 Heuristic Solutions 25
2.4 Related Work to Management Algorithms of Network Slices 26

2.4.1 Failure Prevention . . . 27
2.4.2 Failure Detection . . . 27
2.4.3 Failure Remediation . . . 29
Conclusion . . . 31

3 DANSO Architecture for Network Slices Implementation 33

Introduction . . . 33
3.1 Motivation for DANSO Architecture . . . 33
3.2 Actors and Use Cases for *DANSO* Architecture . . . 34
3.3 DANSO Architecture . . . 36
3.3.1 Network Store Layer . . . 37
3.3.2 Control System Layer . . . 39
3.3.3 Slice Service Layer . . . 40
3.3.4 Virtual Infrastructure Layer . . . 41
3.3.5 Controllers: Cross-Layer Entities . . . 41
3.4 DANSO Operations . . . 42
3.4.1 Slice Creation . . . 43
3.4.2 Slice Management . . . 46
3.5 Vehicular Network Use Case . . . 48
Conclusion . . . 51

4 Optimization of Network Slice Deployment 53

Introduction . . . 53
4.1 Problem Statement . . . 53
4.2 System model . . . 54
4.2.1 Virtual Networks Modeling . . . 55
4.2.2 Network Slice Modeling . . . 57
4.2.3 Resources Modeling . . . 58
4.2.4 User Requests Modeling . . . 58

4.3 Network slice Deployment Process 58
4.3.1 Problem Formulation 58
4.3.2 Proposed Algorithm 60
4.3.3 Numerical Results for Network Slice Deployment 62
4.4 Admission Control and Request Mapping Mechanism 65
4.4.1 Problem Formulation 66
4.4.2 Overload Cost and Requirements based Algorithm for Admission Control Mechanism 67
4.4.2.1 Overloading Cost of Virtual Machine 67
4.4.2.2 Links Overloading Cost 68
4.4.2.3 Switches Overloading Cost 68
4.4.2.4 Slice Overloading Cost 69
4.4.3 Numerical Results for Admission Control Algorithm 69
Conclusion 72

5 Dynamic Handler Framework for Network Slices Management 73

Introduction 73
5.1 Presentation of the Dynamic Handler Framework 74
5.2 Management of Network Slices Resources 76
5.2.1 Information Gathering Phase 76
5.2.1.1 Load Ratio 77
5.2.1.2 Resource usage Prediction 78
5.2.1.3 Predicted Load-Time Fairness Index 80
5.2.2 Management Decision Phase 80
5.2.3 Management Execution Phase 80
5.2.4 Proposed Algorithm 80
5.2.4.1 The Fuzzification 81
5.2.4.2 The Fuzzy Inference System (FIS) 81

5.2.4.3 The Defuzzification 84
5.2.5 Validation of the Resource Usage Prediction Module 85
5.2.6 Validation of the RMD Algorithm 87
5.3 Inter Slice Mobility Management 90
5.3.1 Mobility Management Process 91
5.3.2 Simulations and Results 92
Conclusion 97

6 General Conclusion and Perspectives **99**

List of Publications **103**

Bibliography **113**

List of Figures

1.1 Problem statement 2

1.2 Emerging technologies 3

1.3 Our contributions 6

2.1 High level NFV framework [10] 12

2.2 Simplified view of an SDN architecture [11] 13

2.3 Positioning SDN controllers in the NFV architecture [3] 15

2.4 Service chain 16

2.5 Network slicing concept 18

2.6 Different actions for auto-scaling [68] 30

3.1 Use cases diagram for DANSO system 35

3.2 Use cases diagram for slice creation 35

3.3 Use cases diagram for slice management 36

3.4 DANSO architecture overview 38

3.5 Request analysis algorithm executed by the *Control System Orchestrator* 43

3.6 Slice description algorithm executed by the *Slice Controller* 44

3.7 Slice implementation algorithm executed by the *Slice Controller* 45

3.8 Request negotiation algorithm executed by the *Slice Controller* 46

3.9 Slice management algorithm 47

3.10 Diagram of sequence of a new slice creation 52

4.1 An illustration of a Network Slice 54

4.2 Optimization framework 55
4.3 Network slice deployment topology 56
4.4 The Abilene topology (11 nodes, 28 directed links) 64
4.5 The Geant topology (44 nodes, 142 directed links) 65
4.6 Slice deployment score in Abilene topology 66
4.7 Slice deployment score in Geant topology 66
4.8 Slice overloading cost 70
4.9 Conformity of the SLA description 71

5.1 Modules of the Dynamic Handler Framework (DHF) 75
5.2 Network slice management process 77
5.3 Network slice management algorithm 78
5.4 Structure of the fuzzy logic controller 81
5.5 Membership graph for load index 82
5.6 Membership graph for predicted load-time fairness index 82
5.7 RMD Algorithm 84
5.8 Membership graph of management strategy 85
5.9 Prediction of the CPU utilization rate of machines during execution time 86
5.10 Load state of the considered network slice 86
5.11 Progress in time of the processing delay 87
5.12 Load state of each network slice 88
5.13 (a) The template of the processing delay-CPU Utilization Rate (b)The template of the processing delay-Memory Utilization Rate 89
5.14 The management decision selected for each slice 89
5.15 Progress in time of the processing delay for each slice 90
5.16 VANET architecture 91
5.17 Main interactions of the handover process 94
5.18 Users' mobility from one slice to another 95

5.19 Slice performance when cars are moving from one slice to another in terms of throughput 96

5.20 Slice performance when cars are moving from one slice to another in terms of end-to-end delay 97

List of Tables

2.1 5G use cases classification 11

2.2 Architectures comparison 22

5.1 Example of fuzzy decision rules 85

5.2 Virtual machines characteristics 88

5.3 Simulation parameters 94

Acronyms

3GPP 3rd Generation Partnership Project.

5G Fifth Generation of Mobile Networks.

5G-PPP 5G Infrastructure Public Private Partnership.

BSS Business Support System.

DANSO multi-level Delegation Architecture for Network Slicing Orchestration.

DHCP Dynamic Host Configuration Protocol.

DHF Dynamic Handler Framework.

DNS Domain Name Service.

E2E End-to-End.

eMBB enhanced Mobile Broadband.

ETSI European Telecommunications Standards Institute.

IaaS Infrastructure as a service.

INC Infrastructure and Network Controller.

IoT Internet of Things.

ISG Industry Specification Group.

MANO NFV Management and Network Orchestration.

mMTC massive Machine Type Communication.

MTBF Mean Time Between Failures.

MTTR Mean Time To Recover.

NAT Network Address Translation.

NF Network Function.

NFV Network Function Virtualization.

NFVI Network Function Virtualized Infrastructure.

NFVO NFV Orchestrator.

NGMN Next Generation Mobile Networks.

NSC Network Service Chaining.

NStC Network Store Controller.

ONF Open Networking Foundation.

OSS Operations Support System.

PaaS Platform as a Service.

PNF Physical Network Function.

QoE Quality of Experience.

QoS Quality of Service.

RMD Resource Management Decision.

SaaS Software as a Service.

SAL Service Availability Level.

SC Slice Controller.

SDN Software Defined Networking.

SP Service Provider.

SVR Support Vector Regression.

V2I Vehicle to Infrastructure.

V2V Vehicle to Vehicle.

VI Virtual Infrastructure.

VIM Virtualized Infrastructure Manager.

VM Virtual Machine.

VMC VM Controller.

VNF Virtual Network Function.

VNFC VNF Component.

VNFD VNF Descriptor.

VNFM VNF Manager.

VNFs Virtual Network Functions.

VNS Virtual Network Service.

XaaS Everything-as-a-Service.

Chapter 1

Introduction

1.1 Context and Problem Statement

Nowadays, services are on the top of digital transformation, involving new uses cases, such as e-health, autonomous vehicles and smart traffic light. Also, several kinds of users (e.g., end-user, enterprises, IT systems, etc.) are involved in this transformation and request different constraints in the utilization of these emergent services. These new services characterized by stringent constraints in terms of *latency*, *reliability*, and *availability* cannot be guaranteed by current network implementations. For instance, some emergent use cases require less than the LTE delay. As an example, the tactile internet applications need at most 1ms in order to move an object on the touch screen in real time [1]. Moreover, the Vehicle-to-Vehicle (V2V) and the Vehicle-to-Infrastructure (V2I) use cases require less than 10ms latency for the communications between cars and between cars and the infrastructure.

In fact, as shown in Fig 1.1, the major dilemmas of the existing technologies are the vertical implementation of network functions and the strong coupling between data and control planes. These are the main causes of the lack of flexibility, extensibility and also scalability of network solutions. Therefore, *service providers* and *network operators* are not able to efficiently implement emerging services and to satisfy all user requirements.

In the context of 5G, SDN, NFV and network slicing are introduced as the key

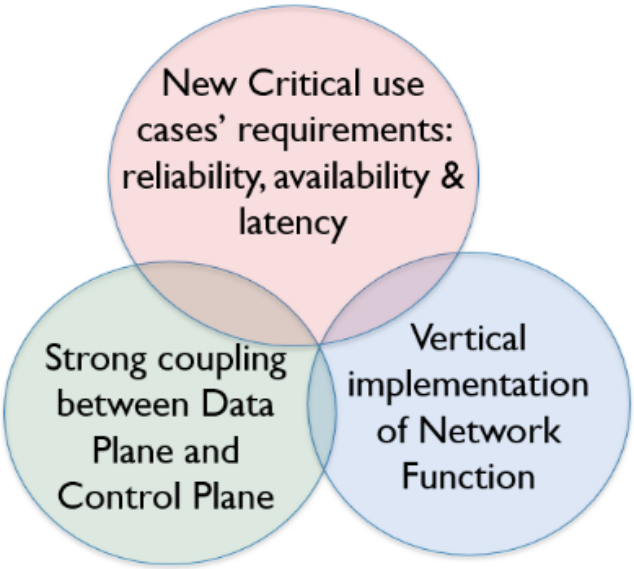


Figure 1.1: *Problem statement*

technologies to support the new requirements of network services as presented in Fig 1.2. A network slice is a customized virtual network running on top of a shared infrastructure. Each slice is composed of a set of connectivity links and programmable resources. Slice resources will embed specific *Virtual Network Functions (VNFs)* to meet the needs of network applications and use-cases.

In this thesis, our work is related to the presented context. We focus on the problematic presented in Fig. 1.1. Using *SDN*, *NFV* and *Network Slicing* approaches, we aim to study the provision of network slices in an SDN/NFV environment. Those slices will be provision on-demand and each slice will tailor only functions needed by the targeted service in order to better satisfy user requirements. In the next sub-section we will present the different challenging aspects for the creation of network slices.

1.2 Research Challenges

Several issues and challenges are surrounding the coupling of the SDN/NFV and network slicing technologies. In fact, to create and provide on-demand network slices, Virtual Network Function (VNF) and Virtual Network Service (VNS) will be mapped on several network resources. This requires a complete aware of the slice information such as networking, storage, compute resources, network functions, service elements, etc. Thus, the creation of network slices in a SDN/NFV environment introduces several

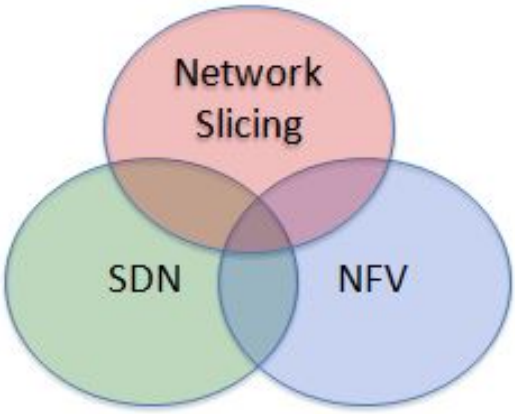


Figure 1.2: *Emerging technologies*

interrogations:

- how to deploy network slices in an SDN/NFV environment?
- How the slice will be shared between users and how to satisfy each user demand?
- How to optimize dynamically the slice creation?
- How to avoid failures and congestion in network slices and how to manage them dynamically?

All these challenging questions need to be raised. In this thesis, we interest especially on the network resources optimization and network slices management challenges. First of all, network and service providers have to satisfy the requirements of the user request. Thus, they should select the suitable network resources during the slice deployment phase. They need to not only provide slices to users but also to consider an optimization of their resources. Then, they have to survey the different changes on each slice. In fact, while slices are running, a dynamic management system has to ensure their well execution in order to avoid slices failures. Sub-sections *1.2.1*, *1.2.2* and *1.2.3* presents in more details these challenges that surround the network slices creation.

1.2.1 Network Slices Isolation

Network slices technology requires a complete isolation between the different created slices. In fact, the deployed slices share a common underlying infrastructure and they are executed at the same time. The isolation between slices is not only a security issue, but it allows a more efficient management and control for each slice. Moreover, the

isolation between slices insures that if a failure or an overloading issue occurs in one slice, it will not be propagated to other slices in the network.

1.2.2 Network Resources Optimization

For the network slicing technology, challenges of the admission control and resources allocation arise. In fact, the deployment of End-to-End (E2E) network slices requires the consideration of the available network resources and their real time performances in order to guarantee the required Quality of Service (QoS). Target resources have to be chosen according to the requirements of each use case. In fact, if the network functions are not well placed, the performance of the entire slice and the performance of the entire network will be significantly impacted. Moreover, a user may use an already deployed slice. Thus the appropriate slice has to be selected in order to guarantee the QoS of the old users and also the QoS of the new user [2].

1.2.3 Management, Orchestration and Control of Network Slices

One of the most important challenges for the network slices is their control, life cycle management and E2E orchestration. The management of the interconnected group of physical and virtual network resources as well as network functions is a challenging aspect. The management of a slice includes its creation, activation/deactivation and maintenance. In fact, the network resources of each slice have to be managed efficiently in order to optimize the resources consumption. The optimization of slice operations needs a continuous monitoring of the slice and a continuous autonomic configuration of the resources allocated to the slice. Aspects of load balancing, charging policies, security, and QoS have to be considered as well. A monitoring system has to monitor continuously the state of all the network slices components. Moreover, the coordination of several resources from multiple domains and also the reconfiguration and the control of all the network resources and network functions are a challenging aspects too. Thus, an end-to-end orchestration is required in order to meet the required QoS for all the deployed slices.

1.3 Considered Research Aspects

In this thesis, we deal with the problem of network slices provisioning in SDN/NFV environment. Therefore, our researches are related to three research aspects. The first aspect is related to the network architecture that will combine SDN, NFV, and network slicing paradigms. The second aspect is the provisioning of network services in future networks. Our third research aspect is related to the management and optimization of network slices. Each of these aspects is concisely presented below.

1.3.1 Framework Architecture

A network framework architecture is a structure used for the provisioning of network services for end users. It comprises several methods, tools and modules to accomplish several goals. For instance, algorithms could be integrated in order to perform the optimization of network resources or to manage users requests. In this thesis we consider the NFV Management and Network Orchestration (MANO) architecture as a reference architecture for the SDN and NFV paradigms [3]. We integrated additional modules and algorithms in order to enhance the flexibility and extensibility of slices creation.

1.3.2 Service Provisioning

Service provisioning has always been the focus of network operators and over-the-top third parties. Several architectures were proposed in order to enhance the manner that the services have been provided. With the introduction of 5G, promising technologies were presented in order to enhance the services provisioning in future networks.

1.3.3 Network Slicing

5G proposes the network slicing technology that provides for each type of service a dedicated network slice. This technology presents a promising opportunity for the provisioning of emerging services with several consideration. In our work we interest on the network slices deployment and management in order to provide the requested services to the users while considering their requirements.

1.4 Our Contributions

To meet the requirements of emerging network services, this thesis proposes three contributions. We first proposed a new framework for the provisioning of network services. Secondly, we studied the optimization of network resources when a network slice will be deployed. Finally, we examined the management of the deployed network slices in order to ensure the well execution of network slices. Our contributions are interrelated as depicted in the following figure:

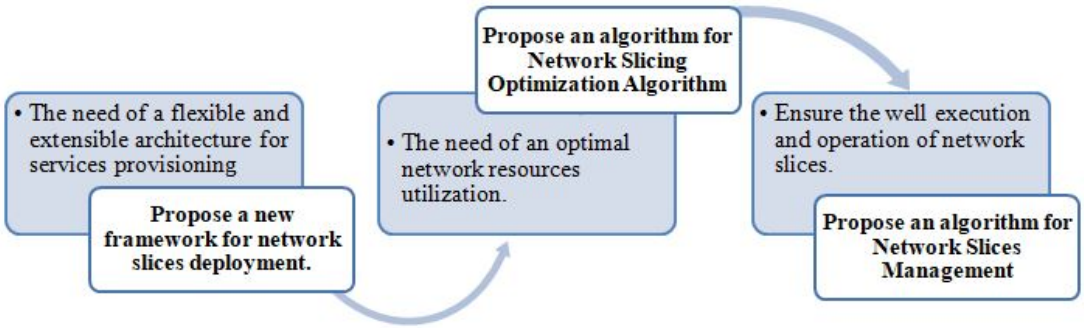


Figure 1.3: *Our contributions*

1.4.1 New Architecture for Network Slices Deployment

Our first contribution was the proposition of a multi-level SDN/NFV architecture that aims to create network slices and ensure their management during the run-time phase. This architecture takes advantage of several proposals in the literature. It presents a global vision of the needed network modules to deal with the slice creation phase and the slice management phase. It is composed of i) forwarding elements that constitute the Virtual Infrastructure (VI), ii) controllers, managers and orchestrators that constitute the distributed control system and iii) a network store maintaining a catalog of pre-defined services ready to be deployed for network slices.

1.4.2 Optimization of the Network Slice Deployment

We propose an optimization algorithm for the creation of a new slice in a given network according to a user request description. The purpose of our proposal is to select the best target network resources among available configuration scenarios in order to satisfy the slice requirements. Our proposed algorithm considers the values of the E2E *availability*, *reliability*, *CPU* and *memory* in order to provide the convenient slice to the users'

requests.

1.4.3 Management of End-to-End Network Slices

We interest on the management of the deployed network slices in order to ensure their well execution while considering the virtual resources status. To this end, we propose a Dynamic Handler Framework (DHF) that collects information about slice performances and then determines if the slice resources need to be modified or not. For this purpose, we propose a reactive and dynamic management policy which is based on a fuzzy logic system that interests essentially on the load situation of each slice.

1.5 Organization of the Thesis

This thesis is divided into four chapters.

Chapter 2 presents a background on dynamic deployment and management of end-to-end network slices. It focuses on the main related work to proposed architectures for network slices implementation, optimization and slices management. It also underlines the shortcomings of the existing solutions.

Chapter 3 introduces our proposed architecture for network slices implementation. It details the different layers of this architecture as well as each component of each layer. Moreover, it details some operations of this architecture specially for network slices creation and management.

Chapter 4 is dedicated to the presentation of our proposed algorithm for the optimization of network slices deployment. We consider several network criteria in order to select the best scenario for slice implementation.

Chapter 5 presents our proposed mechanism for network slices management based on fuzzy logic system. This mechanism aims to perform dynamically the management action that the slice needs during its running time.

Finally, the conclusion summarizes our contributions, discusses their results and determines their limits. Besides, it suggests our perspectives for future work aiming to implement the proposed architecture in a real platform to prove its efficiency.

CHAPTER 2

State of the Art on Dynamic Deployment and Management of End-to-End Network Slices

Introduction

In order to deploy and manage *Network Slices* in *SDN/NFV* environment, we have to analyze these concepts and examine the interactions between them. Furthermore, we have to survey the existing researches that worked on the same problematic and highlight their shortcomings.

For this purpose, this chapter presents the state of the art of related concepts to the deployment of E2E Network Slices. First, we begin our survey by the presentation of the 5G use cases and we discuss their requirements. Second, we define the *NFV*, *SDN* and *Network Slicing* technologies. We present next the reference architecture for the integration of SDN/NFV concept. Finally, we analyze the existing works dealing with the deployment and management of E2E network slices.

2.1 Related Concepts

2.1.1 5G Use Cases Requirements

Several industries, research groups and standardization bodies such as *Ericsson*, *Next Generation Mobile Networks (NGMN) Alliance* and *3rd Generation Partnership Project*

(3GPP) have been focusing recently on the feasibility of some use cases of the 5G technology [4–6]. They have set the requirements that the 5G network has to achieve for each use case. Those use cases were classified according to their requirements into several categories. For instance, the Ericsson white paper [4] presents three families for the 5G use cases which are i) *massive Machine Type Communication (mMTC)*, ii) *Critical Communications*, and iii) *enhanced Mobile Broadband (eMBB)*.

According to the literature, the key use case requirements are the reliability, the availability and the latency. In the following, those parameters are defined.

- Reliability: the reliability is the ability to deliver a service correctly according to its requirement without interruption. The reliability rate required for each 5G use case depends on its classification. Some numerical values of reliability rate for different 5G use cases are presented in the Ericsson white paper [4]. For instance, the 5G network should provide a reliability rate that is equal or higher than 99.999% for the ultra-reliable communication use case. For the *"non-critical reliability"* use cases, such as pervasive video, the reliability rate may be less than 99%. The value of the reliability rate is measured through the Mean Time Between Failures (MTBF) [7].

$$\boxed{Reliability(t) = e^{\frac{-t}{Uptime}} = e^{\frac{-t}{MTBF}}} \tag{2.1}$$

- Availability: as the 5G will be used for the public safety and emergency communications, the network has to insure a required level of availability. In fact, the availability is the ability to deliver a service when requested in order to fulfill the required functions. It is measured considering the MTBF and Mean Time To Recover (MTTR) as given by the following equation[7]:

$$\boxed{Availability = \frac{Uptime}{Uptime+Downtime} = \frac{MTBF}{MTBF+MTTR}} \tag{2.2}$$

- Latency: the latency metric depends on the E2E delay and the data plane latency. 5G networks have to insure in general 10 ms E2E latency for the non-critical latency use cases and at most 1 ms for the ultra-low latency use cases [5].

Table 2.1 presents for each 5G use case family the required values of availability,

reliability and also latency. It presents a classification of the 5G use cases according to their Service Availability Level (SAL) based on the European Telecommunications Standards Institute (ETSI) recommendations [7].

Table 2.1: *5G use cases classification*

Use Case Family	Example of use cases	Requirements: Latency, Reliability, and Availability	$Latency_{max}$	Service Availability Level (SAL)	$Availability_{min}$	$Reliability_{min}$
Enhanced Mobile Broadband (eMBB)	Pervasive video, Dense Urban Society, Video streaming in stadium Ultra-low cost network High Speed train, Moving Hot spots, etc.	There is no accuracy need for the reliability and availability for this use case family. For most use cases, the latency should be very low.	Between 5 and 10 ms	SAL 3: Available in normal conditions only	95%	95%
Massive Internet of Things (IoT)	Smart wearables, Sensor networks	This family is relatively tolerant to the delay and it requires a normal to high reliability and availability.	10 ms	SAL 3: Available in normal conditions only	95%	95%
Critical Communications: Ultra high reliability and ultra low latency use cases	Automated traffic control/driving, collaborative robots, remote object manipulation	Use cases of this category are characterized by a very low latency, a very high reliability and a high availability.	1 ms	SAL 2: Available in normal and overloaded conditions.	99,9%	99,999%
Critical Communications: Ultra low latency use cases	Tactile Internet	Use cases of this category are characterized by a very low latency, a high reliability and availability.	1 ms	SAL 2: Available in normal and overloaded conditions	99,9%	99,9%
Critical Communications: Ultra high reliability and availability	eHealth (Extreme life critical), Public safety	Use cases of this category are characterized by a very high reliability and a very high availability with a low latency	5 ms	SAL 1: Always available	99,999%	99,999%

2.1.2 SDN/NFV Paradigms

As we presented in the *Introduction* chapter, a new network vision is required in order to satisfy the aforementioned requirements of emerging use cases. The fundamental paradigms for the future network vision are the SDN and the NFV [8]. Those two paradigms are complementary and they will together achieve 5G goals and ensure high flexible networks.

2.1.2.1 Network Function Virtualization (NFV)

NFV [8] softwarizes network functions so that they can be running on virtual resources on top of servers and network resources. It decouples the network functions from hardware appliances so they can be replaced by a software running in a container (e.g., Docker, LXC) or in Virtual Machine (VM). The functions and behaviors of a VNF are expected to be identical to the functions and behaviors of the non-virtualized Network Function (NF). Some examples of VNFs are firewalls, Dynamic Host Configuration Protocol (DHCP) servers, Network Address Translation (NAT) servers, handoff control, intrusion detection algorithms, Domain Name Service (DNS), etc. Each NFV is composed of one or more VNF Component (VNFC) which are designed by a *Service Provider (SP)* and it has one VNF Descriptor (VNFD) which defines the connections between the VNFCs [9]. This approach is applicable to any data plane processing or control plane function and allows the deployment of an E2E *service chain* for a *network slice*. A high level view of the NFV Architecture is presented in Fig. 2.1.

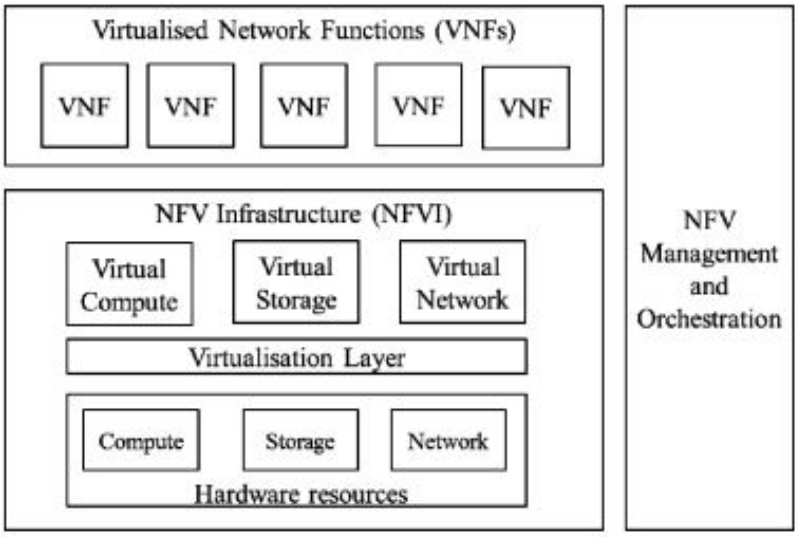


Figure 2.1: *High level NFV framework [10]*

2.1.2.2 Software Defined Networking (SDN)

SDN [11] paradigm is standardized by the Open Networking Foundation (ONF). It consists on the separation of the control logic from the forwarding network resources. It provides new ways to configure and build networks [12]. In fact, SDN aims to facilitate the deployment of virtual networks, the programming of network resources and the configuration of forwarding devices.

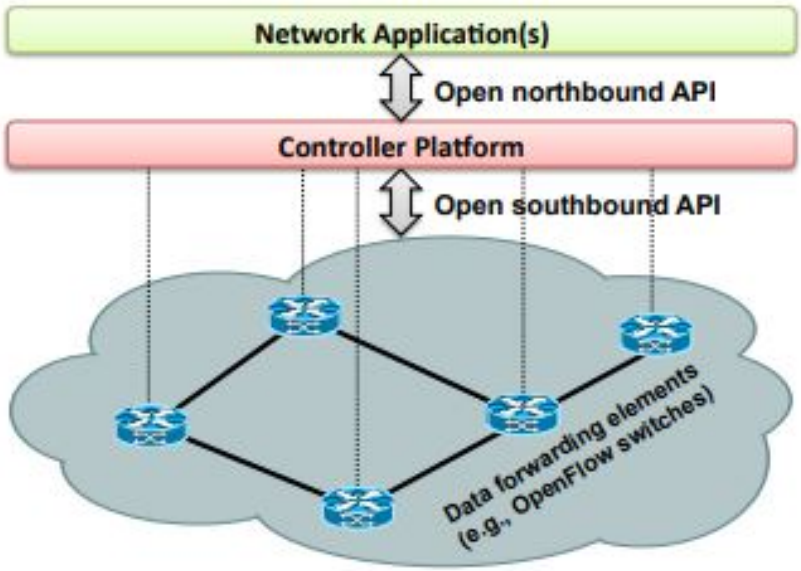


Figure 2.2: *Simplified view of an SDN architecture [11]*

SDN architecture, as presented in Fig. 2.2, contains three layers and two interfaces which are presented as the following:

- the infrastructure layer: it is defined also as the forwarding plane. It contains the forwarding devices (e.g. switches), connected through wired and wireless links. This layer will be then programmed by the controller and it will execute the specified actions on the incoming packets.
- The control layer: it contains the SDN controller such as OpenDaylight, ONOS, etc. This controller has the intelligence of the underlying infrastructure. It oversees and manipulates the forwarding devices. Its role is to implement the forwarding rules that will be executed on incoming packets on each device.
- The application layer: the application layer contains several network applications such as routing, firewall, load balancing, etc.
- The northbound interface: the northbound interface communicates the policies of network applications to the *SDN* controller.
- The southbound interface: the southbound interface ensures the communication between the controller and the forwarding devices. It uses a communication protocol such as OpenFlow for these interactions.

In summary, *SDN* separates the control plane from the underlying infrastructure

and moves the control logic to a centralized controller. In his turn, *NFV* decouples *NF* from hardware resources to be implemented on virtual machines which are running on top of servers, switches or routers. The controller will then program network elements and will steer flows through network functions. Thus, *SDN* and *NFV* provide together the network programming and break vertical software integration.

2.1.3 SDN/NFV Architecture

ETSI proposes a framework for the integration of *SDN* and *NFV* architectures presented in Fig. 2.3 [3]. This framework is composed of two centralized SDN controllers which are the *Infrastructure SDN Controller* (*IC*) and the *Tenant SDN Controller* (*TC*). The *IC* is responsible for the control and management of the underlying network infrastructure. It controls the behavior of the infrastructure and changes it according to the VIM requests. The *TC* is a *VNF* which is instantiated in the tenant domain in order to control and manage the *VNFs* constituting the network service of the tenant. Concerning management and orchestration aspects, *Management and Network Orchestration working group (ETSI SG)* [13] proposes a standard for the management and the orchestration of clouds and data centers resources: the *NFV MANO*.

NFV-MANO system deals with the lifecycle of VNFs and orchestrates the resources of the underlying infrastructure in order to deploy VNFs. The NFV-MANO is composed of essentially three functional blocks which are the *NFV Orchestrator (NFVO)*, *VNF Manager (VNFM)* and the *Virtualized Infrastructure Manager (VIM)*. Moreover, it communicates with the *Operations Support System (OSS)* and the *Business Support System (BSS)* of the operator. Details of the *NFVO*, *VNFM* and *VIM* are presented in the following parts:

1. VNF Manager (VNFM): the *VNFM* role is to manage and supervise the lifecycle of *VNFs*. For instance, it i) creates a *VNF*, ii) updates its software and its configuration and iii) terminates it by releasing the allocated resources. One *VNFM* can serve multiple *VNFs*. It is also possible to deploy *VNFMs* as much as existing *VNFs*.

2. Virtualized Infrastructure Manager (VIM): *VIM* controls and manages Network Function Virtualized Infrastructure (NFVI) resources (i.e. compute, storage and network) of the operator infrastructure domain. It is capable of handling

all resources types. It manages the virtual resources capacities and reports the capacity and the usage of each *NFVI* resource. It can be specialized in a single resource management (e.g. compute-only, storage-only or network-only). Resources management functions include the allocation of *NFVI* resources and the association of virtual resources to physical resources. Therefore, VIM holds a *NFVI* Resources repository where information about the allocated and available hardware (i.e. compute, storage, and networking) and software (e.g. hypervisors and virtual machines) resources is stored.

3. NFV Orchestrator (NFVO): is responsible for the orchestration of the network resources, network services and *VNFs*. It exposes the network service catalogue, which is a repository that contains the registered network services. The main functions of *NFVO* include registration of network services in a catalogue and management of their lifecycle. *NFVO* takes in charge also the registration of *VNFs* in a VNF catalogue and the instantiation of the corresponding *VNFM* when needed. Moreover, it checks the consistency and viability of the provided network service and *VNF*. Finally, it authorizes *VNFM* access to *NFVI* resources.

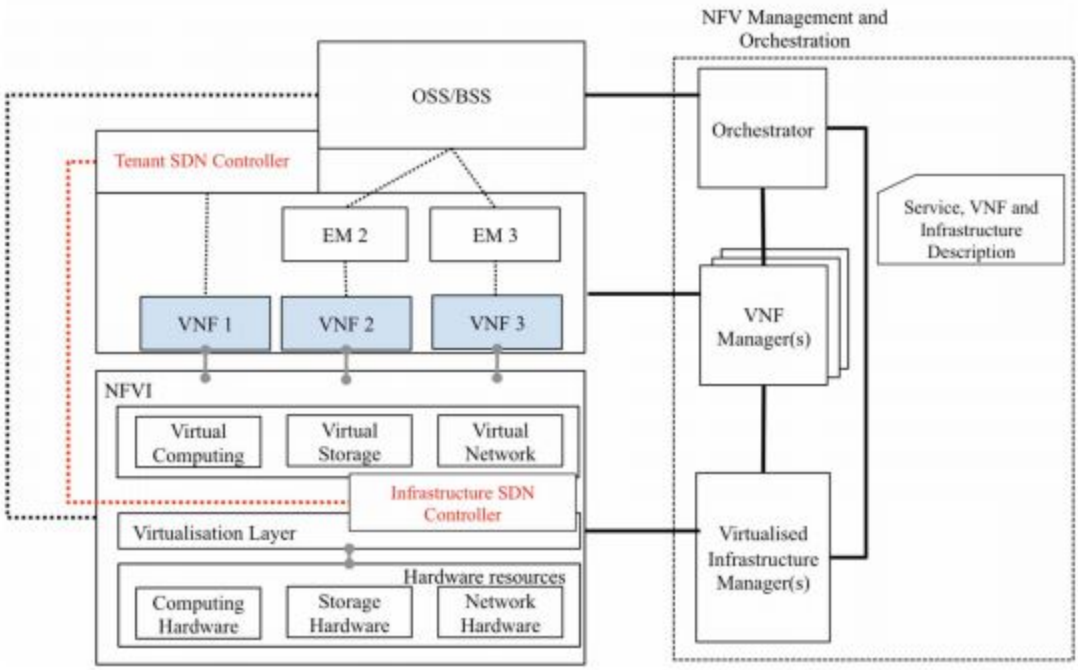


Figure 2.3: *Positioning SDN controllers in the NFV architecture [3]*

As examples of projects which provide an implementation of the MANO software according to the ETSI specifications we cite Open Source MANO (OSM) [14], Open

Baton [15], ONAP [16], etc.

2.1.4 Network Service Chaining

SDN and *NFV* enable the flexible composition of network functions which is known as *Network Service Chaining (NSC)* concept. This allows the provisioning of service chains and the introduction of new services. Thus, due to *NSC* concept, future networks will be more flexible with a lowest *CAPEX* and *OPEX* [17].

A network service chain is the composition of an ordered and chained set of functions with pre-defined parameters as shown in Fig. 2.4 [18].

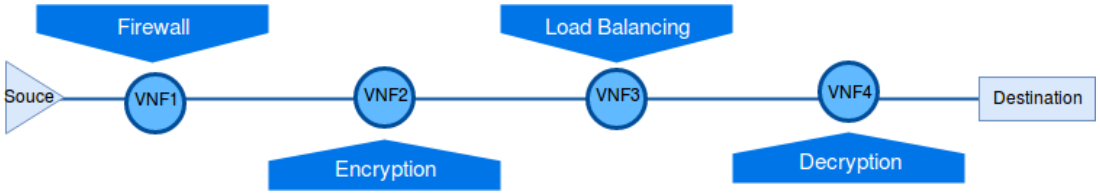


Figure 2.4: *Service chain*

2.1.5 Network Cloudification

According to the *National Institute for Standards and Technologies (NIST)*, *Cloud Computing* model permits the access and the provision of computing resources (i.e. processing, storage and networking) according to specific properties [19]. Initially, *Cloud Computing* has defined three service models which are namely *Software as a Service (SaaS)*, *Platform as a Service (PaaS)* and *Infrastructure as a service (IaaS)*. For several years now, researchers have been proposing several other models such as Radio-Access-Network-as-a-Service (RANaaS), Data-and-Knowledge-as-a-service (DKaaS), Computing-as-a-Service (ComaaS), Network-as-a-Service (NaaS), etc. This leads to the definition of a new concept: Everything-as-a-Service (XaaS) [20].

NaaS or *Network Cloudification* is an example of XaaS and it intends to the deployment of network slices. This concept involves the application of *Cloud* model to networks. It considers not only the pay-as-you-go business model but also the *"as a Service (aaS)"* model for network applications, network services chains and network resources (whether hardware, platform, or software) [21]. Every component of the network is viewed as a service which enables the *Service Providers* to compose several service chains for several use cases.

2.1.6 Network Slicing

In our work, we define a network slice as a customized virtual network running on top of a shared infrastructure that may cross several domains. A network slice, as shown in Fig. 2.5, is a set of interconnected network functions and programmable resources. It is composed of a set of connectivity links, storage, and computing resources dedicated to one application. It comprises mainly several virtual network functions and services dedicated to one customer to meet the needs of a specific application or use case. According to the literature, a network slice may include one or more network service. Each network service is composed of one or more Physical Network Function (PNF) or VNF such as routing, load balancing, and security functions. A slice tenant can be a network operator, end users, etc. For an operator a network slice is a complete network infrastructure that uses only some network resources in order to meet services and applications requirements.

Network slicing technology receives the attention of several research groups and standardization bodies such as NGMN [5], ONF [22], ITU-T [23] and 3GPP [6]. The current works and initiatives in standardization diverge in their approaches at various viewpoints which leads to a large and fragmented landscape. In fact, organization and industries deal differently with the network slicing concept.

For instance, the NGMN defines the network slices as a combination of network functions and radio access resources deployed over a shared physical infrastructure that boost business innovation in a programmable and multi-tenant environment.

For the 3GPP, a network slice is a logical network created and customized by operators in order to provide specific network capabilities and resources.

For ITU-T, network slices are considered as logical isolated network partitions composed of isolated and programmable virtual resources.

Network slicing technology introduces the *"one size per service"* approach instead of the traditional *"one size fits all"* approach. Using this approach, slices will be provision on-demand and each slice will tailor only functions and services that the business model needs. For example, when a slice will be deployed for the management of fixed sensors, there is no need to include the mobility function into the slice implementation. According to the ONF, the SDN paradigm is well aligned with the network slicing; it enables to implement specific logic control to each slice which is a key aspect for mobile

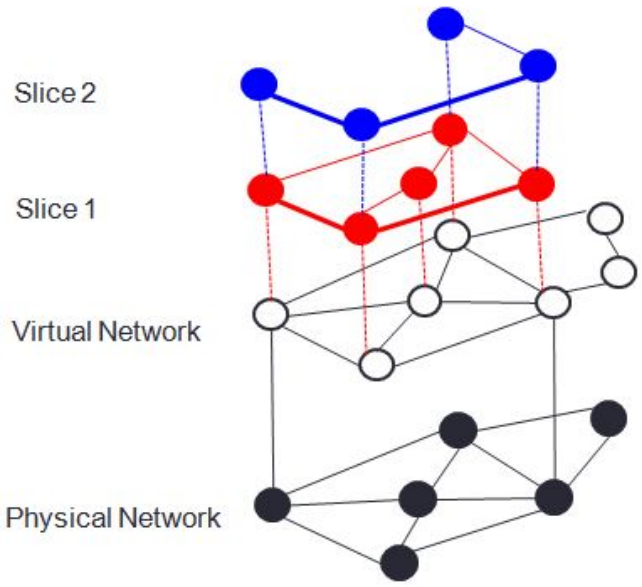


Figure 2.5: *Network slicing concept*

applications where network adaptation should be more agile.

2.2 Related Work to Framework Architectures for Network Slices Implementation

As mentioned previously in chapter 1, the integration of network slicing technology to the future network architecture is required in order to respond to the requirements of emerging use cases. However the network slices creation is a challenging aspect for many reasons as we presented in section 1.2.

In this section, we present some works that have been done in the context of the control, management and orchestration of network slicing. We provide a classification guideline for the projects that interest on future network architecture. We review also some of the most important approaches regarding slices deployment in the context of SDN/NFV environment.

2.2.1 5G Platform Design

In [24], authors present a 5G architecture design for the network slices management based on NFV, SDN, and cloud-computing paradigms. This architecture is composed of three layers. The first is the *business layer* which provides virtual network applications (VNA)

and virtual network functions (VNF) in order to create network slices description. The second is the *service layer* which creates slices according to their description. Finally, the third is the *infrastructure layer* which supports programmable cloud and virtualization technology. According to their proposal, the cloud will be re-configurable and the network will be dynamically sliced essentially through the concept of the *network store*. As an implementation of the proposed architecture, authors present a proof-of-concept prototype of the *LTE as a Service* slice. They present a list of network functions such as *virtual eNB* and a list of network applications such as *spectrum sharing* required for the creation of an LTE slice. In this work, authors interest only in the implementation of LTE slices. The presented Network Store focuses essentially on physical functions and applications. Thereby, it needs to be extended in order to support 5G use cases like IoT applications and virtual and augmented reality. Also, roles of some entities of the architecture are not detailed. Moreover, this proposal lacks information for the functionality of the network control.

In [25], authors present a framework for the management and orchestration of network slices. The aim of their proposal is the end-to-end automation of the design of network slices. Their proposed framework extends the reference framework of 3GPP with the automation of end-to-end network slicing management and orchestration.

In [26], authors present their design for a unified control and optimization framework which is based on the SDN and NFV technologies. This Framework aims to control and maximize the system performances, the resource utilization and the service provisioning. Their proposed control module is composed of an NFV orchestration system which is in charge of the assignment and the provision of the virtual network functions and SDN controller which determines rules of packet forwarding, implements them in the forwarding tables and controls the NFV orchestration system. The major dilemma of this paper is the centralization of the control and orchestration mechanisms. Their proposed framework cannot be implemented in a large scale network and especially if the network slicing technology is integrated to the network.

2.2.2 5G-PPP Projects

As part of the 5G-PPP program [27], several project have been interested in the SDN/NFV paradigm such as Sonata [28], 5GEX [29], NECOS [30], 5G TRANS-

FORMER [31], 5G NORMA [32] and 5G SLICENET [33]. In the following, we present the major aim of some of those projects

2.2.2.1 5G SliceNet

5G SLICENET proposes an E2E architecture for cognitive network-slicing in a virtualized multi-domain multi-tenant network. The project addresses the E2E network-slices provisioning, control, management and orchestration across multiple administrative domains. It interests essentially on three use-cases which are the *5G Smart Grid Self-Healing*, the *5G eHealth* and the *5G Smart City*. This project considers a layered architecture that aims to ensure a modularity, extensibility and scalability of the proposal. The major aim of the project is to facilitate the creation of network slices for verticals in order to satisfy the requirements of different use cases and guarantee the Quality of Experience (QoE) of the services of each slice.

2.2.2.2 Sonata

Sonata project focuses on the development and orchestration of services in virtual networks. The aim of this project is to reduce the time to market of network services and also the optimization of the deployment cost and resource utilization. The most important contribution of this project is the integration of service composition and orchestration. Thus, the Sonata project interest on coupling networks programming, flexibility and the optimisation of software deployments. Sonata Supports the Hierarchical Service Provider (HSP) policy for the orchestration of network services. The HSP aims to orchestrate different orchestrators in a multi-domain context. Sonata project supports slices deployment, management and orchestration over a multi-operator and multi-MANO environment.

2.2.2.3 5GEx

5GEx project interests on the multi-domain network service orchestration over multi-domain and multiple administrations. The main goal of this project is to enable multi operator collaboration and to build a unified European infrastructure service market for services provisioning. Thus, 5GEx project will allow the collaboration between several operators.

2.2.2.4 5G NORMA

5G NORMA is a new architecture for mobile networks that proposes multi-service and multi-tenancy support. It is based on the composition and decomposition of network functions which allows the support of different requirements of 5G services.

2.2.2.5 Discussion

Regarding slices deployment and as denoted in table 5.1 most of the existing projects does not provide an optimization mechanism for the deployment of network slices. In addition, current proposals in the context of 5G-PPP, does not propose any negotiation mechanism between the users and slice providers in case of slice request drop (e.g., due to capacity limitation). A negotiation can lead to an agreement between the two actors which can lead to an increase of the accepted services. Among few works concerning the negotiation aspect, we can highlight the architecture proposed by Orange in [34].

5G NORMA and Sonata projects take into account the life-cycle operations concerning the network slice creation and operation (e.g., re-allocation of resources re-composition of network functions, etc.). In Sonata, life-cycle management operations are divided into service-level and function-level operations. Other works such as 5Gex proposes full life-cycle management, spanning from design, release, order, and operations to dispose services for NSaaS concept (network slicing as a service). Similarly, SliceNet project adopts JOX (Juju-based orchestrator), which supports life-cycle management of network slices and the mobile network orchestration. Other works such as [34] focus on negotiation aspect for 5G. These concerns specially the *Service Order Management (SOM)* definition and protocol for a dynamic service parameter. In [34] the *Connectivity Provisioning Negotiation Protocol (CPNP)* is defined. This protocol carries information exchanged between client and servers during negotiation phase. The life-cycle of network slices is a primary goal for the dynamic network architecture, this includes slice deployment, resources and quality of service updating, suspending and deletion, etc. The life-cycle is supported only for the VNF but the dynamic deployment and suppression of the whole slice is still an open issue in the most of the mentioned works.

The *recursion* of network slices is an important aspect for the future network architecture that is also not considered by several projects [35]. In fact, the future architectures have to allow a hierarchical slicing architecture with a parent-child relationship between slices. The support of the recursion in the network system allows the deployment of new

Table 2.2: *Architectures comparison*

Project	NORMA	Sonata	5GEx	SliceNet	Orange Lab
Recursiveness	Yes	Yes	Yes	No	Yes
Multi-Domain	Yo	Yes	Yes	Yes	Yes
Life-cycle management of network slices	Yes	Yes	Yes	Yes	Yes
Environment optimization	No	Yes	No	No	Yes
Management and orchestration	Yes	Yes	Yes	Yes	Yes
Negotiation	No	No	No	No	Yes

business models. For instance, Sonata Supports the hierarchical service provider policy for the orchestration of network services in order to orchestrate different orchestrators in a multi-domain context.

2.2.3 Novelty

The aforementioned projects are based on the MANO architecture presented in section 2.1.3. As we presented, this architecture is designed for the management of VNF life-cycle and the orchestration of cloud and data centers resources. However, MANO architecture focuses only on the management and orchestration of VNF. In order to deploy, manage and orchestrate dynamically E2E slices, the whole system resources has to be managed and orchestrated.

Moreover, those projects consider centralized solutions for the management and orchestration of network resources and functions. This poses some limitations mainly with the reaction response time and the scalability while dealing with several administrative domains (i.e., implication of several cloud, mobile or broadband network operators). In other words, due to the centralization, the delay as well as the overhead communication will increase in the system. As an example, we consider the *tactile Internet* use case which is a critical communication use case characterized by its stringent requirements in terms of delay [6]. In this case, the system has to ensure that the delay will not exceed the fixed threshold during the service execution. The centralized approach generates a lag between the analysis of network and slices states and the processing of

reconfiguration actions. However, with a multi-level delegation approach, the system is able to quickly identify and react when the delay is increasing.

To overcome these issues, we present in this thesis a new vision for the network slices implementation through a multi-level delegation architecture. Our architecture presents and details two important modules which are the optimization and the management of network slices. We detail the interaction between these modules, the controllers and the orchestrators of the network system.

2.3 Related Work to Optimization Algorithms for Network Slices Deployment

As we presented in the previous section, each *Network Slice* is the composition of several *Network Services* and each *Network Service* is composed of chained *VNFs*. The deployment of a *Network Slice* requires that data traffic passes through the set of its *VNFs* in a predefined order. For each deployment the system has to select the resources that will host network functions and steers the traffic among them. One of the main challenges to deploy a *Network Slice* is to achieve dynamic composition and allocation of network functions. The question that raises is how to allocate and how to schedule *VNFs* of each *Network Slice* onto a Substrate Network.

The resource allocation in NFV is performed through three stages:

1. *VNFs Chain Composition (VNFs-CC)* or Service Function Chaining: this phase consists on the composition of *VNFs* chains in order to create the requested service [36].

2. *VNF Forwarding Graph Embedding (VNF-FGE)* [37]: the composed chain of *VNFs* for the network service is called *VNF-FG*. This graph is the input for the embedding algorithm in order to find where to place the *VNFs* in the substrate network. This action is accompanied with an optimization algorithm to maximize the *QoS*, minimize the energy consumption, etc.

3. *VNFs Scheduling (VNFs-SCH)* [38]: the final step of the resource allocation in the *NFV* problem is the scheduling process. This stage determines how to execute each function of the chain in order to minimize the total execution time while

maintaining the required level of the service performance.

In our thesis we interest mainly on the second phase of the process of the resource allocation in *NFV*: *VNF-FGE*. In the *NFV* environment, the resource allocation and the embedding of virtual resources require efficient algorithms to determine on which resources *VNFs* will be placed. In the literature, several algorithms have been proposed. The goal of these proposals differs from one algorithm to another. For instance, they aim to enhance the user's satisfaction, minimize the network deployment CAPEX and OPEX, improve resources utilization and ensure energy saving.

According to the literature, the optimization of the process of resource allocation in virtual environment is known to be a NP-hard problem [39], which means that the execution time to solve the problem optimally is so important for medium and large instance size. This problem can be solved by using exact, heuristic or metaheuristic optimization strategies. In this thesis, we interest on exact and heuristic solutions.

Exact techniques determines the optimal solution for small size problem. This solution is considered as a baseline solution and optimal bound for heuristic-based strategies. Heuristic-based solutions aim to find a good solution for the problem, but not necessary the optimal one, while keeping the running time as low as possible. In the following, we present some related work in the context of NFV and network slices resource allocation problem with exact and heuristic solutions.

2.3.1 Exact Solutions

In [40], authors interest on the placement of *VNFs* across the physical resources. They present a queuing-based model which considers the computational capabilities of physical hosts and *VNFs* requirements in order to share the *CPU* between all *VNFs* running on the same host.

In [41], authors propose an algorithm that considers the resource specifications, the requirements of packet processing and the bandwidth limitation in order to determine the best implementation of VMs. No interest on the user's requirement in terms of the availability, reliability E2E latency was considered.

In [42] authors propose a unified approach based on the SDN paradigm that aims to optimize the provisioning of both virtual machines and network bandwidth. The proposed algorithm minimizes the cost of the over-provisioning of cloud resources when

they are requested by the users. This work interests essentially on the optimization of the network bandwidth provisioning without considering other parameters that may impact the offered QoS.

In [43], authors propose a mechanism for both the network resource provisioning and the admission control in *Wireless Virtualized Networks*. The aim of their work is the maximization of the total data rate of the slices through an admission control algorithm. The proposed algorithm adjusts dynamically slice requirements according to the channel state but they do not consider the global resource utilization of network resources as well as their reliability and availability.

2.3.2 Heuristic Solutions

In [44] authors propose two heuristic algorithms in order to solve the network slicing problem in the case of simultaneous processing and routing of service requests. The aim of their work is to optimize the implementation of *Service Function Chains* according to the link and node capacities constraints. No interest on the user's requirements was made. In [45] authors propose a multi-constraints based algorithm for the virtual data center embedding problem. The major aim of their proposal is the minimization of the utilization rate of network resources while maximizing the reliability and also the revenue of infrastructure providers.

In [46], authors propose an *Integer Linear Program (ILP)* for *VNF-FG* placement problem when *VNFs* are shared across multiple tenants. In their work, they aim to optimize resource usage and maximize provider revenue. In order to solve the issue of scalability and the size problem in linear programming, authors consider the selection process to reduce the number of candidate resources.

In [2] authors propose a heuristic algorithm for the admission control mechanism of user's requests and the dynamic allocation of network resources to the implemented slices. This Algorithm allocates dynamically network resources to each slice while maximizing the slice user's QoE based on RAN slice prioritization. The paper considers only the maximization of the user data rate for the network resources allocation when the network slice is created.

In [17] authors study the problem of the network services decomposition and embedding. They propose two algorithms for the mapping of network service chains to the

network infrastructure. The first proposed algorithm is an Integer Linear Program (ILP) while the second one is a heuristic algorithm. Both algorithms try to minimize the cost of the mapping while considering the requirements of the network service chaining and also the capabilities of the network infrastructure. The proposed embedding cost is based on the CPU, memory and storage capacities of physical nodes, the cost of bandwidth in a physical link and the resources utilization rate. In this paper, authors consider only the bandwidth and the delay for the required QoS. Regarding the requirements of the emergent use cases, other QoS parameters need to be considered.

In summary, network resources allocation and slicing issues were studied for different contexts and several decision algorithms were proposed. They exploit multiple objective optimization techniques. However, the emergent 5G use cases are characterized by stringent requirements in terms of availability, reliability, and latency. Thus, additional parameters have to be guaranteed in order to satisfy the users' requests. Therefore, the decision of the slices allocation needs to consider additional constraints in order to guarantee the offered QoS and QoE to users. In our work, we present an optimization algorithm for the creation of a new slice in a given network according to a user request description. The purpose of our algorithm is to select the best target network resources among available configuration scenarios in order to satisfy the slice requirements. Our proposed algorithms consider the values of the E2E availability and reliability as well as computational capacities in order to provide the convenient slice to the users' requests.

2.4 Related Work to Management Algorithms of Network Slices

While deploying network slices, the management of their resources is a crucial aspect which allows keeping the required QoS for users. In [7], Industry Specification Group (ISG) NFV presents some mechanisms required for the management of NFV environment. According to their paper, a fault management system involves three features which are failures prevention, failures detection and failures remediation. Each module should cooperate with the rest of the network entities in order to report and diagnose performance and real-time resource consumption.

2.4.1 Failure Prevention

Failure prevention is the ability to block the occurrence of failures in the system. It takes effect through the control of the quality offered by the system. Several mechanisms are used for the prevention of VNF failure such as overload prevention, prevention of single point of failure and failure prediction.

Online failure prediction is considered as the adequate approach to enhance the reliability and availability in NFV environment [7]. In [7], authors present some requirements for the prediction module in NFV. They propose the utilization of the false alarm filtering and anticipating the evolution of the system to an unhealthy state. In [47] and [48], authors propose a mathematical algorithm for the faults prevention problem which aims to minimize the probability of failures in network resources.

In [49] authors propose an energy-effective prediction algorithm for the cloud environment in order to identify future requirements of cloud resources based on *Multilayer Perceptron (MLP)* model. In [50] authors propose *Wavelet Support Vector Machine (WSVM)* algorithm for the prediction of data centers behaviour. In [51], authors interest on application-aware SDN solutions. They propose an *Autoregressive Integrated Moving Average (ARIMA)* based model to forecast large data transfers. Their model integrate an automated parameters estimation module and a module for parameters verification and re-estimation.

2.4.2 Failure Detection

In an NFV environment, both hardware and software levels must be considered for the detection of failures. On the hardware level authors in [52] consider a periodically evaluation of each resource status in order to detect hotspots. On the software level, failures detection can be achieved through health checking techniques such as heartbeat [53] and watchdog [54]. The heartbeat technique consists on the delivery of "heartbeat messages" to a responsible entity which will restart the VNF that exceeds the response delay. The watchdog technique defines a timer to detect the existence of the VNF and recover it from failures.

In the following, we present briefly some algorithms and techniques applied in the case of failure detection in NFV and network environments.

- *Self Organizing Maps (SOMs)*: in [55] and [56], authors propose a VNF failures detector module based on *Self-Organizing Map*. This module analyses information related to CPU utilization, memory utilization, disk I/O and network interface I/O in order to detect abnormal events. Self Organizing Map is considered as a dimension reduction technique. It permits the detection of anomalies without previous learning. Authors demonstrate that their algorithm detects successfully the memory overhead and network congestion failures better than the clustering approach.

- Anomaly Detection Technique [57]: in networking, this technique is used mainly in self-healing to detect abnormal behaviors. Its main goal is to identify resources that present anomalies and unusual system behavior. These algorithms will compare the actual performance of network resources to their initial state. In the context of NFV, [58] proposes an architecture for faults detection in processes and links. The system monitors the processes by sending "liveness" messages with a response timeout. According to the response time, the state of the process is identified. The link state is identified through the technique of topology discovery executed by the switches.

- Markov Models [59]: this model is used in the case of randomly changing systems that follow the Markov property. The Markov property is a memoryless property. It is independent of all previous states of processes. Thus the future state of a process depends only on its current state. In networking, Markov models are mainly applied to resource optimization and fault detection.

- Fuzzy Logic Controllers (FLC) [60]: unlike normal strategies which use Boolean logic (0 or 1), FLC considers multiple levels of input parameters which enable them to analyze unclear data behavior. Fuzzy controller is characterized by three modules: i) the fuzzifier which translates the input variables to fuzzy logic language such as low, normal, high, ii) inference engine system which applies a set of rules to the input variables and iii) the defuzzifier which returns a quantifiable result after the application of the rules. In networks, fuzzy controllers are used for load balancing, resource optimization, fault detection, etc.

2.4.3 Failure Remediation

Failure remediation in *NFV* environment comprises the *VNF* level and the virtual resources level. The remediation of *VNFs* is performed either independently by the *VNF* or by the *VNFM*. The *VNFM* performs several actions in order to avoid the degradation of *VNFs* such as migrating *VNFs*, reducing *VNFs* capacity, removing the unhealthy hardware, etc. Given the constraints required by 5G services in terms of latency, reliability and availability, remediation mechanisms have to adapt the offered QoS to not be violated.

1. VNF migration: the resource migration strategy has to determine which VM should be selected for migration and where it should be deployed. For the management of virtual machines resources, authors in [61] propose an SDN based orchestration system which considers the temporal network information in order to perform the VMs migration and minimize the network cost. In [62], authors propose a network status sensing algorithm for VM migration in order to reduce the cost of network communication.

2. Load balancing [63, 64]: the load balancing aims to eliminate hot-spots in order to improve resource utilization and energy efficiency. When a VNF gets overloaded, new VNFs are instantiated and the traffic will be shared between the old and the new instances. Thus, the network has to forward the traffic to the old as well as to the new VNF instances. This traffic between instances should not be overlapped.

3. Auto-scaling: the auto-scaling is the ability to dynamically scale the resources according to the current demand [19]. As reported by the ETSI, a NFV system has to support scaling mechanisms [13]. Two types of auto-scaling are defined in the literature which are the vertical and horizontal auto-scaling. For vertical auto-scaling, the virtual machine needs to be more or less powerful and thus the defined actions are the increase (scaling-up) or the decrease (scaling-down) of virtual machine resources. While vertical auto-scaling focuses on making one machine more efficient, horizontal auto-scaling consists on the addition (scaling-out) and removal (scaling-in) of virtual machines [65]. In Fig. 2.6, the different actions for auto-scaling mechanism are presented. In [66], authors propose a fuzzy-based auto-scaling mechanism which is a lightweight mechanism that provides a scalable

and elastic fog environment. In [67], authors propose a dynamic learning strategy based on a fuzzy logic system. They propose a self-adaptive controller which is combined with two reinforcement learning approaches: Fuzzy SARSA learning and Fuzzy Q-learning. In the context of NFV, most of the projects do not provide a complete management solution in terms of capabilities and functionalities. Open Baton project [15] proposes an autoscaling module however the decision adopts only the linear threshold method. This method is not suitable for rapid changeable service context.

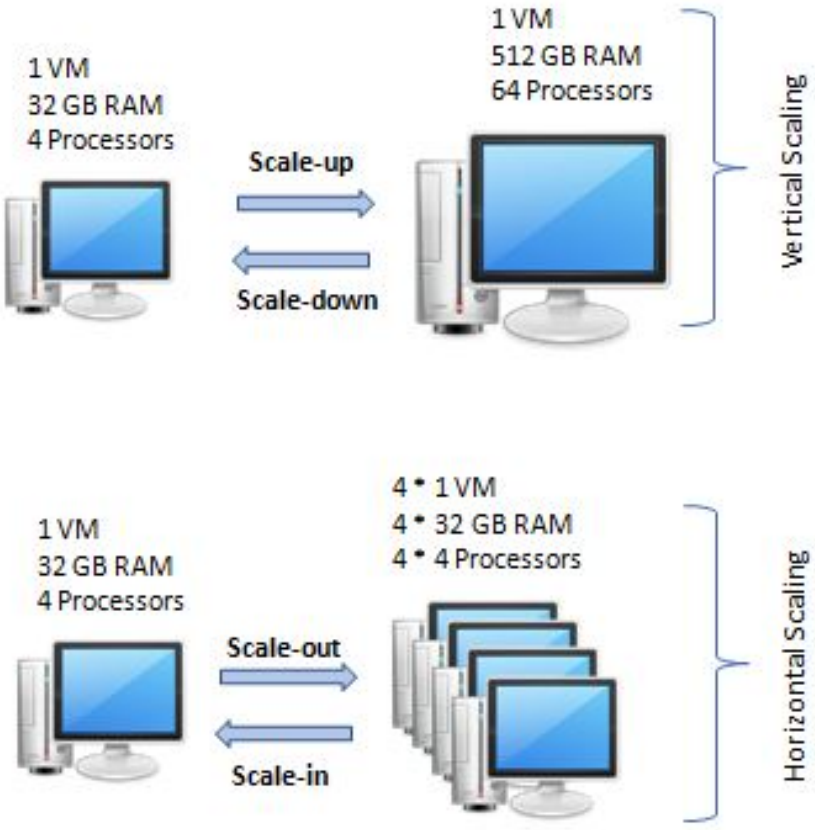


Figure 2.6: *Different actions for auto-scaling [68]*

In our work, we interest on the management of network slices. We propose a framework that aims to ensure their well execution while considering the virtual resources status. For this purpose, we propose a reactive and dynamic management policy which is based on fuzzy logic system that interests essentially on the load situation of each slice.

Conclusion

In this chapter, we have presented the principal concepts needed to enhance the flexibility and the programming of service provisioning in future networks. Indeed, we have discussed the requirements of emerging use cases and the importance of SDN, NFV and network slicing technologies in service provisioning. We have also presented the main related work to raise these challenges. Based on this literature review, we can conclude the main shortcomings of the existing works. As a matter of fact, the major limitation is the centralisation of the control and the orchestration of virtual network resources. Moreover those works do not focus on the management and the optimization of network slices. For this purpose, we propose in our work a new multi-level delegation architecture for network slices creation and management.

Moreover, we have explored the main existing works related to network resources optimization. We have remarked that the majority of contributions interest on computational capacities of network resources and do not consider their reliability and availability. Afterwards, we have studied the management in the context of network slices. We highlighted the main features of a management system and the main existing works.

CHAPTER 3

DANSO Architecture for Network Slices Implementation

Introduction

In this chapter we focus on the implementation and the management of network slices and we propose an architecture for the creation and management of end-to-end slices. Our proposal takes advantage of the SDN/NFV and the network slicing paradigms. In the following, we detail the design of this architecture, we describe its modules and we present the network slices creation and management operations.

3.1 Motivation for DANSO Architecture

As discussed in the *Introduction* chapter, new network architecture models are required in order to respond to emerging services requirements especially in terms of reliability, availability and latency. The integration of SDN, NFV and network slicing technologies seems to be a promising solution for this issue. However, as we presented in 1.4, several challenges surrounds this integration.

We present in this chapter the *DANSO* architecture. Several aspects are considered by our architecture such as the optimization, the management and the negotiation of network slices. In fact, regarding slices deployment, the architecture has to provide an

optimization mechanism. An efficient mapping of network slices functions and services on the underlying infrastructure is an important aspect that may allow an optimal use of resources. Moreover, a complete control and orchestration of the slice operations enhance the treatment of users demand and slice management. Furthermore, the centralization of the management and orchestration of network resources and functions cause some issues essentially with the response time and the scalability.

For this reason, the key idea of this architecture is the multi-level delegation for slice creation and management. Indeed, *DANSO* is a layered architecture that provides a control system and a network store for the management and provision of network services. The control system handles all the user requests, optimizes the system performances and programs the underlying virtual infrastructure with network functions and services. The network functions and services are stored in the network store layer and they are enclosed by *Over-The-Top third* parties and network operators. Finally, *DANSO* aims to maintain a control in the creation phase in order to make the best correspondence between the network slice and the user demand.

3.2 Actors and Use Cases for *DANSO* Architecture

The introduction of the network slicing technology to the network ecosystem involves new business model and thus new network actors. In this thesis, we consider three kinds of actors for our *DANSO* framework for the creation and management of network slices:

- *Network Provider*: it owns and manages physical network resources. Those resources will be then virtualized and proposed to be programmed by *Service Providers*.

- *Service Provider* or *Slice Provider*: it leases virtual resources from one or more *Network Providers* in order to create its own virtual network and slices. It will then create, manage and provide network slices to its users. It is also responsible of designing and developing network services that will be exposed to the end user. Furthermore, it collects, stores and analyzes information about slice execution.

- *End User*: it requests a service from the catalogue proposed by the *Service Provider* and then consumes the offered services.

The cited actors interacts differently in order to create and manage network slices. For each use case, each actor has a specific role and actions to execute.

The use cases diagram for the *DANSO* system is presented in Fig. 3.1. The *end user* accesses the catalogue of services proposed by the *Service Provider* and selects a service. A request for the service provisioning will be sent to the *Slice Provider* which requests the creation of the corresponding slice from the *Network Provider*. The *Network Provider* will then create the requested slice and manage it. The information about the *end users* is managed by the *Slice Provider*.

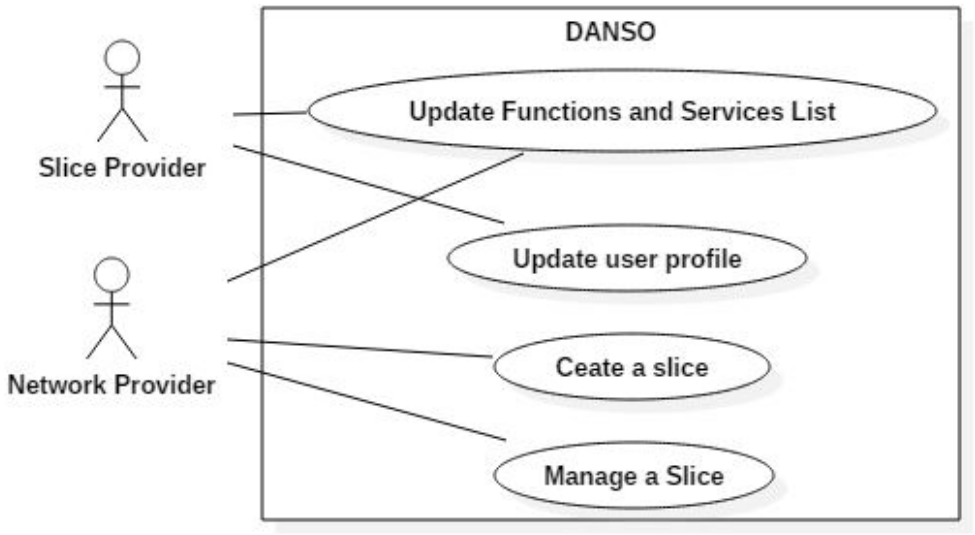


Figure 3.1: *Use cases diagram for DANSO system*

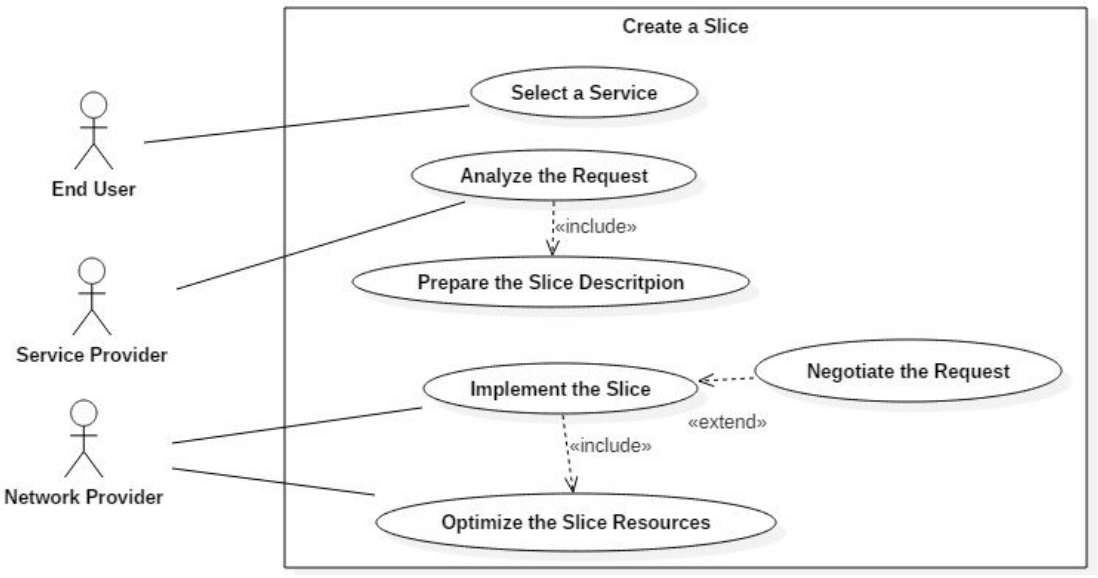


Figure 3.2: *Use cases diagram for slice creation*

The diagram of use cases for the slice creation is presented in Fig. 3.2. After receiving the user request, the *Slice provider* analyzes it and then prepares the slice description. When the description is sent to the *Network Provider*, the *Network Provider* will implement the slice while optimizing this action. If the creation is not possible a request negotiation could be generated.

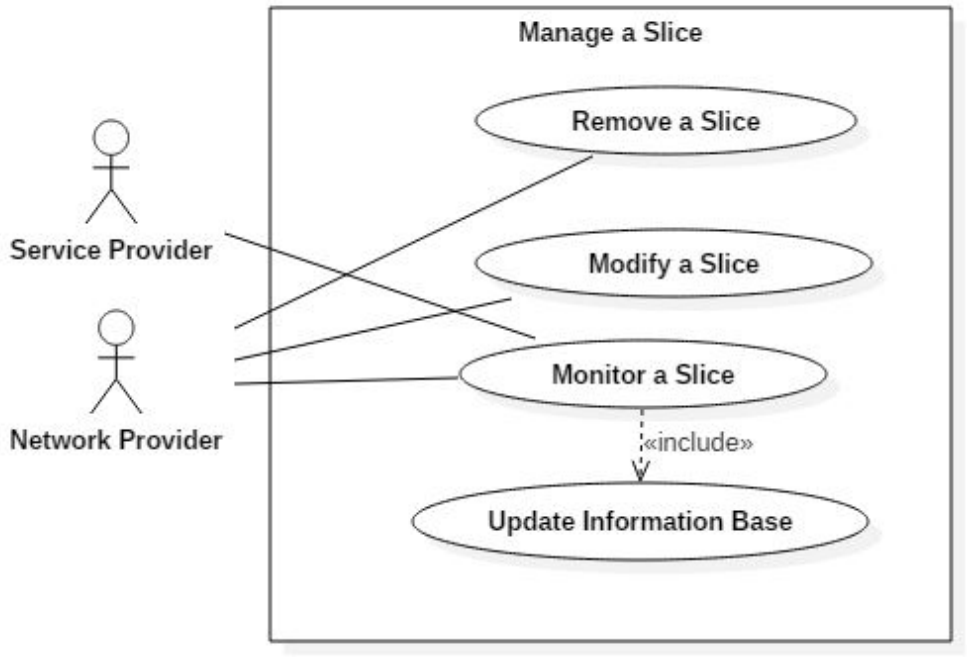


Figure 3.3: *Use cases diagram for slice management*

The diagram of use cases for the slice management is presented in Fig. 3.3. When the Slice is created, a performance monitoring process is executed in order to examine its execution. The network provider may remove or modify a slice. It will also update periodically the information base after monitoring the slice.

3.3 DANSO Architecture

DANSO proposes a multi-level architecture which is composed of:

1. a network store that contains a catalog of several network services ready to be used by the users.

2. Several modules to ensure the optimization and the dynamicity for slice deployment and management, and also to update the needed information (e.g. data bases about network topologies, resources and functions states, etc.).

3. A set of slices that forward users' flows.

4. Servers and forwarding elements that constitute the virtual infrastructure.

Orchestration and management of all the actions in the system is delegated to several components. Thus, for each level a manager and an orchestrator are associated in order to create and manage network slices. Also, the controllers are defined as being the cross-layer entities. The general role of these entities are as follows:

- *Orchestrator:* it plays the role of northbound interface for each layer. It is a decision entity which receives commands and information request. It translates the demand into configuration tasks. Then it forwards them to be treated at the adequate controller. To deal with an E2E treatment, the orchestrator coordinates also the communications between entities at different layers.

- *Manager:* it is responsible for the supervision of operations executed at its layer. It ensures correctness of the creation and management processes. It handles the life-cycle of processes and services involved in its layer.

- *Controller:* it plays the role of executor and programming entity. When creating a network slice, it executes the associated process (e.g., optimization of slices placement, mapping of network components into virtual infrastructure, VM allocation, install forwarding rules, etc.) according to the orders coming from the *Control System Orchestrator*.

Fig. 3.4 presents the proposed architecture. We present in the following parts more details about the different modules of our proposal.

3.3.1 Network Store Layer

The *Network Store Layer* provides a catalog of use cases which are exposed to end users. It has the same utility as the applications store for the software platform (e.g. App Store, Play Store, etc.). In fact, it provides a marketplace of Network Services (NS) and Network Functions (NF). Thanks to this marketplace, use cases will be supplied. Thus, following a user's request, suitable VNF and VNS are chosen from this catalog to build the chain of NFV (combining VNF(s) and/or VNS(s)). As an example of *Network Store* we can mentioned the project T-NOVA [69], which provides a marketplace for VNFs. In

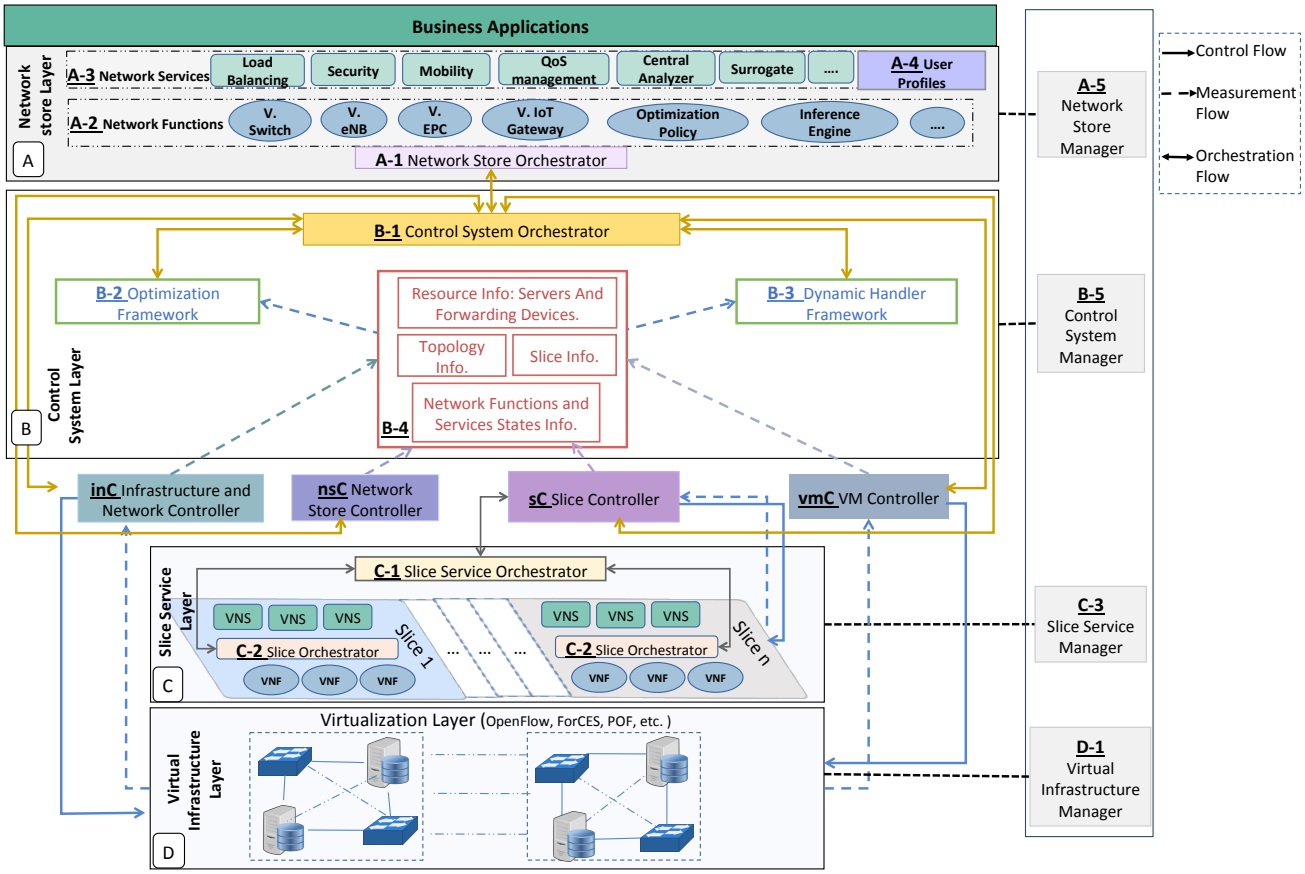


Figure 3.4: *DANSO architecture overview*

the rest of our work, we consider that both VNF and VNS are performed by NFVs. In the following, we describe each element of the Network Store layer:

- **Business applications**: this sub-layer encompasses several use cases ready to be deployed by the proposed architecture upon users request, and by referring for example, to Ericsson [4], NGMN [5] and 3GPP [6]. As examples of those applications, we can list smart wearable, sensor networks, and drones connectivity.

- **Network Store Orchestrator (A-1)**: it coordinates the communications between the *Network Store Layer* and other layers. It receives demands from the *Control Orchestrator*, orchestrates them, and then provides, from the *Network Store Layer*, the needed information (functions or services).

- **Network Functions Store (A-2)**: it serves as a database that contains the metadata needed for the function provisioning. This metadata describes the required resources for a network function.

- **Network Services Store (A-3)**: this store provides to end-users value-added ser-

vices. They are implemented on the fly by the network system on the executed slices in order to extend the executed application. As examples of network services we can list *load balancing* and *localization services*.

- **User Profiles (A-4)**: this component is a database that encodes users policies. The *System Control Orchestrator* uses this component to apply policies to each user's request according to their profile. In our architecture, we define several classes of policies depending on user profiles.

- **Network Store Manager (A-5):** it is responsible of the management of the *Network Services* and *Functions Stores*. It is in charge of on-boarding new NF and NS to the *Network Store Layer*, deleting, modifying and updating functions and services of this level.

3.3.2 Control System Layer

We define in this level the fundamental modules supporting the optimization of the deployment and the management of events introducing changes in the behavior of deployed slices. This layer defines also data about network capacities, services and function states, network slices, etc. *Control System Layer* communicates with different controllers to forward the actions to be realized in other layers.

- **Control System Orchestrator (B-1)**: receives the user's request, translates it through its translator module and then communicates it to the controllers set. In addition, it coordinates the communications between the controllers as well as between controllers and the *Control System Manager*. Once the message is received, the *Control System Orchestrator* decides to which controller it should be delivered.

- **Control System Manager (B-5)**: defines management functions based on two fundamental modules: the *Optimization Framework* and the *Dynamic Handler Framework*:

  - ***Optimization Framework (B-2)***: in a virtual environment, resources can host and execute several functions and virtual machines simultaneously. Since the capacities of servers and network devices are limited, efficient optimization strategies should be implemented. For this purpose, the *Optimization*

Framework takes into consideration user requirements as well as the network system state in order to optimize the service chain creation, the VM allocation, functions assignment to VMs and the deployment of the slices over the network infrastructure. Also, this component determines whether the current slice is still able to execute additional requests or not.

- ***Dynamic Handler Framework-DHF (B-3)***: in order to deal with events that occur in the system such as the congestion of a slice, the failure of a VM implementation, the user mobility, etc., we need to monitor in real time the deployed slices. The *DHF* analyzes periodically the information database and detects any events that can trigger the deletion or the modification of slices, VMs and VNFs. For example, after analyzing the occurred event, the DHF asks the *Slice Controller* to create another slice with the same description if the trigger event was slice congestion, slice failure, etc.

- **Information Base (B-4)**: this database gathers the information collected by the controllers. This information relates to several network aspects and it is classified into four categories. The first database (*Resources information: servers and forwarding device information*) exports information related to the state of the servers and forwarding devices, such as the current processing capacity of the servers, the available link bandwidths, the number of VMs per server and also the list of implemented VNF at the VMs. If any change occurs in this information, controllers have to update the database with new measurement. The second database (*Topology information*), exports the network topology which is reported by the forwarding devices. The third database (*Virtual Network Functions states information*) export the state of the VNFs. Their states could be "running", "interrupted", "migrated", etc. Finally, the last database (*Slice Information*) contains the slice template description, the state of the slices and the utilization rate of each slice.

3.3.3 Slice Service Layer

The *Slice Service Layer* contains the set of the deployed slices over the underlying infrastructure. Each network slice is isolated from other slices and it is dedicated to a specific type of service. The orchestration over this layer is ensured due to the *Slice*

Service Orchestrator module.

This level defines several components as follows:

- **Slice Service Orchestrator**: this orchestrator manages the set of slices orchestrators. It focuses on the management of the network services life-cycle. Basically, it manages the network service view associated with each network slice. During the operational phase, it communicates with the *Slice Controller* to identify the functions and services to be added or removed from slices. Finally, since it communicates directly with each *Slice Orchestrator*, it ensures the isolation among the network slices.

- **Slice Orchestrator**: it is in charge of the end-to-end life-cycle and management of each slice. Basically, it communicates its slice information and requests to the *Slice Service Orchestrator*. It also implements controller commands and decisions (e.g. packet forwarding rules) into slices forwarding devices.

- **Slice Service Manager**: it oversees the deployed network slices as well as their life-cycle management. It has also an overall coordination role between *Slice Service Layer* elements and *Control System Layer*.

3.3.4 Virtual Infrastructure Layer

The virtual infrastructure level encompasses all software and hardware resources that form the virtualized environment. It includes the connectivity between data centers and clouds systems as well as the computing, storage and network capabilities. Within this level, we find the ***Virtual Infrastructure Manager*** that controls and monitors the virtual infrastructure and communicates with the virtual infrastructure provider in order to ask for extra resources.

3.3.5 Controllers: Cross-Layer Entities

- **Network Store Controller (NStC)**: it is in charge of receiving the user's request description. Then, it communicates with the *Network Store Layer* in order to transform this demand into a service chain. The service chain is composed of logical resources set which are the VNFs and it is created while considering the information related to the corresponding user in the *User Profiles* database. This

information identifies the policies that will be applied to this request.

- **Slice Controller (SC)**: it is responsible to execute several tasks :
  - it maps a new slice request to an already created slice, if it finds a deployed slice for this type of service, or it will ask for the creation of a new slice.
  - It assigns a dedicated *Slice Orchestrator* for every created slice.
  - It interacts with the *Network Store Controller* in order to request the instantiation of VNFs from the *Network Store Layer*.
  - It communicates with the *VM Controller* to ask for the setting up of a new virtual machine for a VNF or for the release of the allocated slice resources.

- **Infrastructure and Network Controller (INC)**: it is responsible for the control and the supervision of the entire underlying infrastructure (concerning computer and network resources). It is in charge of determining which resources will be allocated for a slice while taking into account the description of logical resources as well as the state of the virtual infrastructure. It works in collaboration with *Virtual Machine Controller* to implement network components into *VM* when a new slice will be created.

- **VM Controller (VMC)**: it is responsible for the programming of virtual machines with different network functions and services. It controls also the deployment, the update and the re-provisioning of virtual machines in the system. This controller follows the methodology of the infrastructure manager proposed by the *ETSI NFV* and *SDN* mechanisms [3].

3.4 DANSO Operations

DANSO architecture considers several network aspects like the end-to-end slice creation and slice management. Therefore, several modules and algorithms are executed in order to provide for the end users the best QoS. In the following, we focus on the slice creation and management and we provide the related process models.

3.4.1 Slice Creation

Several steps are required for the slice deployment. Those steps start by the analysis of the user's request, then the creation of the slice description according to the user's requirements and finally the slice implementation. In the following we detail each of these steps.

1. **Request analysis**: when the *Control System Orchestrator* receives the user's demand, it starts by defining the set of its requirements and also the information about the user profile in order to have a complete description of the user's request. Following this step, the *Control System Orchestrator* prepares a slice description ready to be implemented. Different constraints can be associated with this description such as mentioned in [70]. The sequence of those actions is schematized in Fig. 3.5.

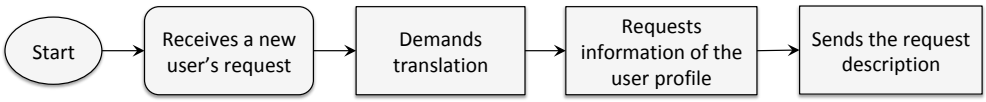


Figure 3.5: *Request analysis algorithm executed by the Control System Orchestrator*

2. **Slice description**: after analyzing the request, the *Slice Controller* searches for an existing deployed slice that responds to this demand. Two cases can be considered:

   (a) first, the *Slice Controller* may find some deployed VNFs that can be reused for the user's request. However, only some types of VNF, such as routing, can be shared by several users because of the isolation constraint. In this case, the *Slice Controller* will study the performances of the VNF which will be reused. In [71], we have proposed an algorithm for the performance evaluation in terms of reliability, availability and latency. Therefore, if the performances of the founded VNFs are less than a fixed threshold then an additional demand may be served by those VNFs. Else, the *Slice Controller* will try to enhance the performances of the selected VNFs. To do so, we have two possibilities:

   (i) to migrate some VNFs to another VM or container with more resources than the current ones,

   (ii) to duplicate some VMs and balance the traffic between those two instances.

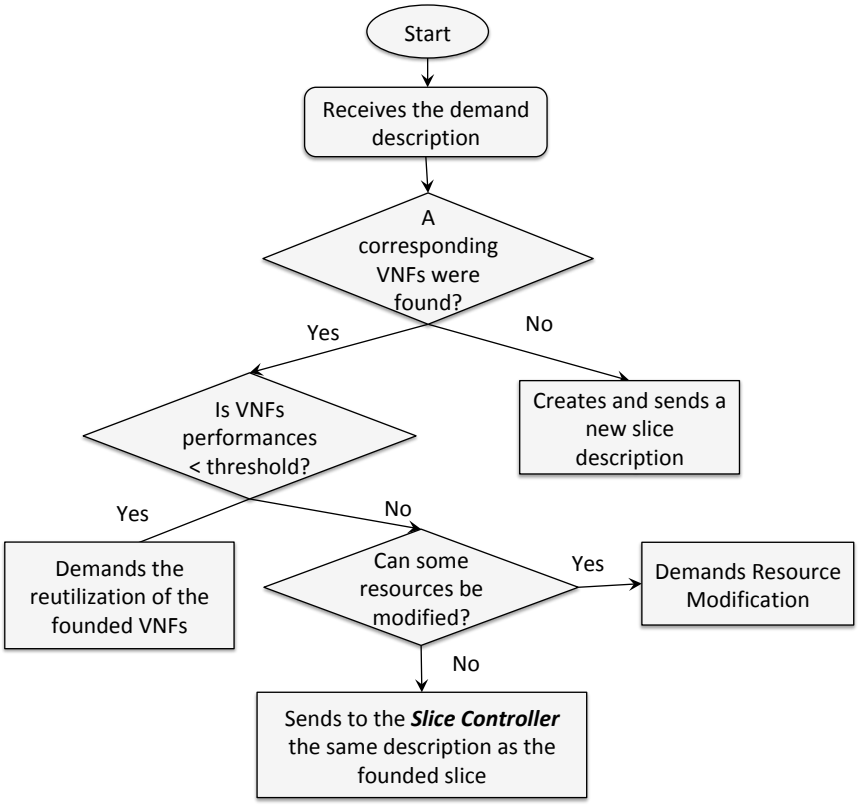


Figure 3.6: *Slice description algorithm executed by the Slice Controller*

(b) Second, the *Slice Controller* will create a new slice description while specifying the needed information for its implementation.

Fig. 3.6 presents the sequence of the described actions for the creation of the slice description.

3. **Slice Implementation**: after defining the required operation for the user's request, three actions are possible as shown in the Fig. 3.7.

   (a) If the defined action is the reuse of some already implemented VNFs, then the *Slice Controller* will create a slice which is composed of the reused VNFs and the new created ones. Then, it will inform the *Slice Orchestrator* and the *Slice Service Orchestrator* about this new operation.

   (b) If the action is the modification of the slice, then the *Slice Controller* will modify the identified VNFs and will map the request into the created slice.

   (c) Finally, if the action is to create a new slice, then the *Slice Controller* studies the *Virtual Infrastructure (VI)* performances to determinate if it is able to support another slice implementation or not. So, the *Infrastructure and*

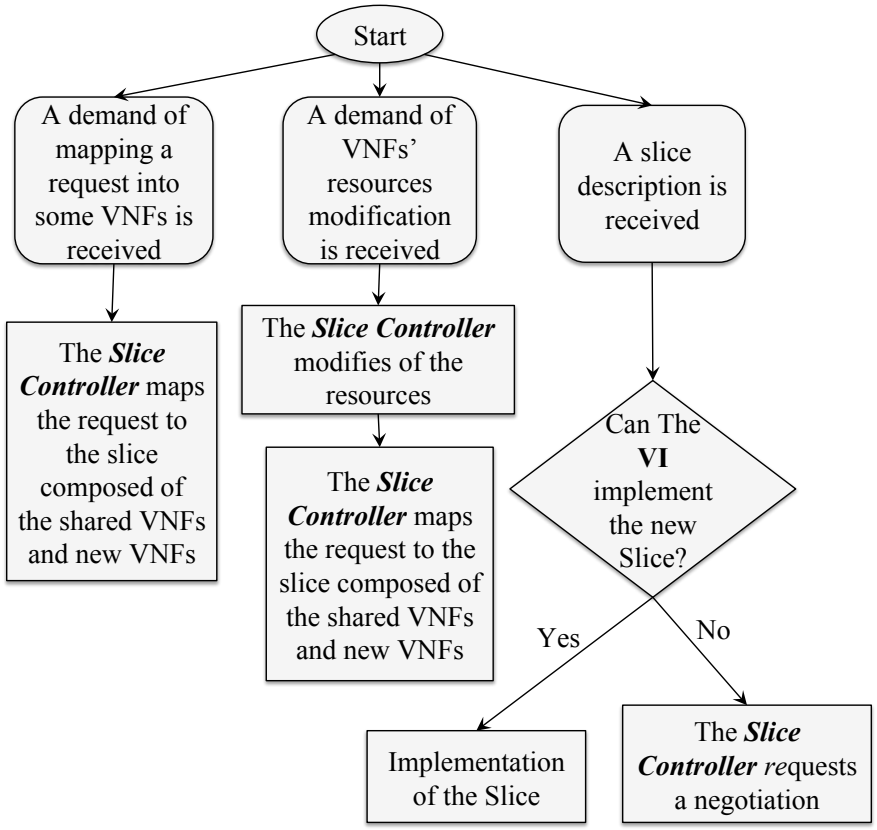


Figure 3.7: *Slice implementation algorithm executed by the Slice Controller*

Network Controller reserves the needed network resources and implements the list of VNF defined in the already prepared slice description. Otherwise, a **request negotiation** will be executed.

- **Request Negotiation**: in the case of request negotiation, the *Control System Orchestrator* proposes to the user a new description for its demand and a negotiation will take place in order to find a compromise between them. If the user's request can be modified then the new request will be sent and treated by the *Control System Orchestrator*. If the request cannot be modified, then the *Virtual Infrastructure Manager* may ask other cloud providers for additional resources, as in [72]. However, this case is not considered in this thesis and it can be the object of future work. Therefore, if the user refuses to modify its request, his demand will be rejected. The sequence of the mentioned actions is presented in Fig. 3.8.

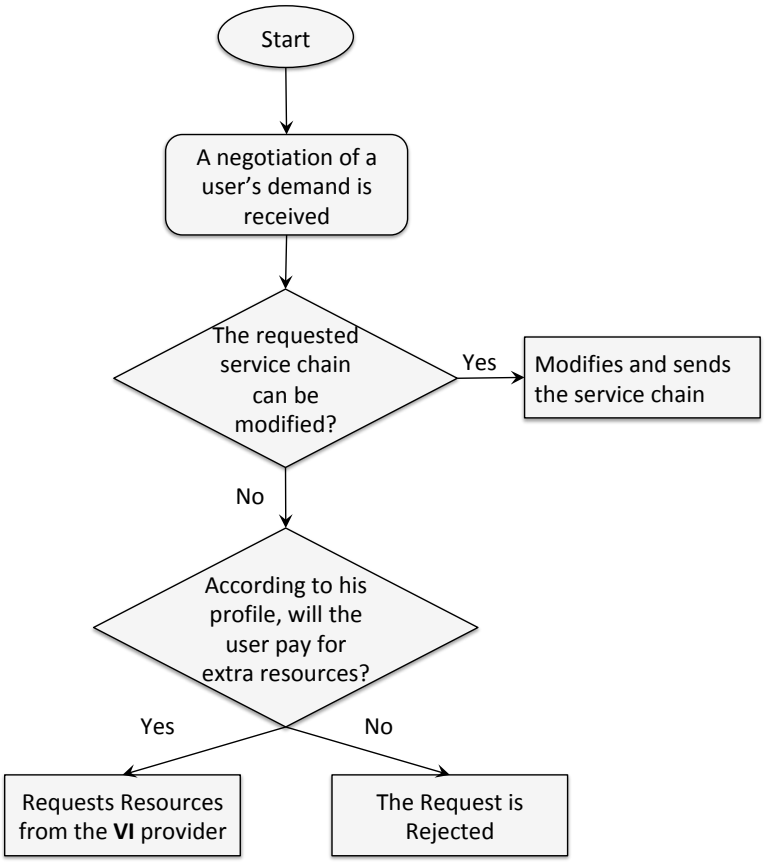


Figure 3.8: *Request negotiation algorithm executed by the Slice Controller*

3.4.2 Slice Management

During slice execution and network operations, many unexpected events can occur. To deal with those sudden events, we propose the ***Dynamic Handler Framework (DHF)*** that monitors the network state, analyzes it and informs the controllers in case of any dysfunction. The *DHF* may detect several types of events and for each event it triggers the adequate action. Fig. 3.9 presents the sequence of actions to be triggered for two events: slice modification and slice suppression. In the following we detail some of possible actions.

- **Slice Suppression**: a slice will be removed if it does not respect the established QoS agreement or if the VI is congested. In the case of a slice removal, the *DHF* notifies the *Slice Controller* which will ask the *Slice Service Orchestrator* to remove the identified slice. This action will take place through an exchange between the *Slice Service Orchestrator* and the *Slice Orchestrator* of the concerned slice. Finally, the *Slice Controller* notifies the *VM Controller* to release all the

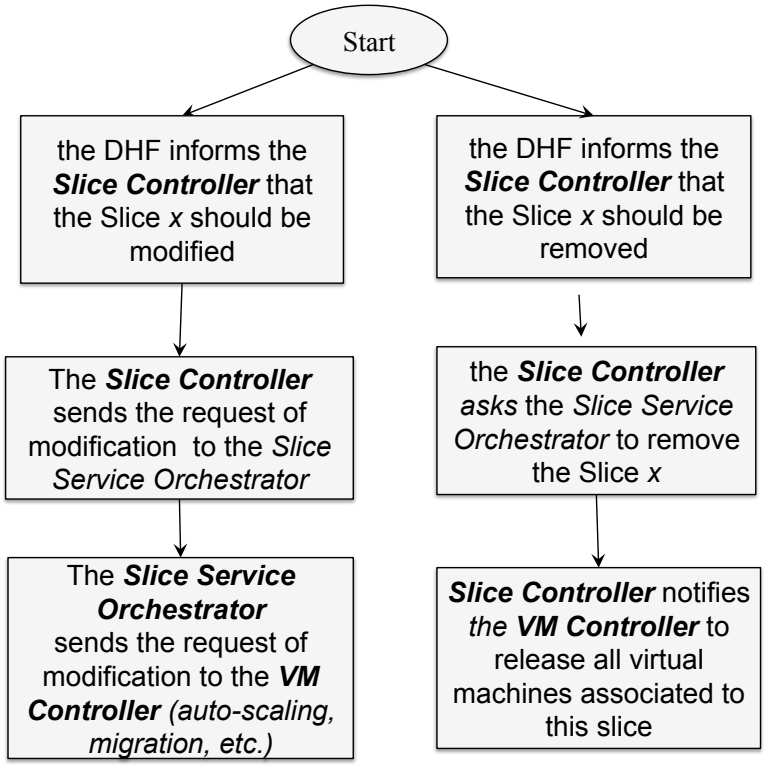


Figure 3.9: *Slice management algorithm*

virtual machines associated to this slice.

- **Slice and Resources Re-provisioning**: resources re-provisionning can be envisaged to deal with several situations, for example:
  - Slice congestion: in this case, *DHF* asks the *Slice Controller* to either increase the performance of this slice (through the migration or the auto-scaling of some VNF) or to create another slice with the same description and move some users' flow to this new slice.
  - User's mobility: in this case some resources need to be re-provisioned in other locations in order to guarantee the same quality of service. In this case, the *DHF* demands the creation of another slice or the migration of services and functions. If a slice needs to be recreated, *Slice Controller* will proceed in the same way as described for the slice creation and suppression.
- **Slice Modification**: it concerns all changes involving the current chain of services (e.g. addition, removal or migration of NFVs). User may modify its slice by requesting new functionalities (e.g. enhanced security or additional bandwidth). Also, the *DHF* can be notified that an existing slice needs the addition of some

services (e.g. load balancing) in its chain of service. In all cases, the demand of the slice modification will be captured by the *Slice Service Orchestrator* which forwards it to the *VM Controller (VMC)*. The *VMC* will be inquired to implement the modification action into the VMs according to the new service functions chain.

If VMs and servers are congested or out of order, then a migration of those resources must be executed. In this case, the *DHF* notifies the controllers in order to migrate and re-provision indicated resources. After VNF migration, old resources must be released in order to be used for further implementations.

The decision of VNF removal may be the solution for the enhancement of the required QoS. In fact, while analyzing the *Information Base*, the *DHF* may notice the increase of the delay in a particular slice. In this case, some VNFs could be removed in order to decrease the number of forwarding devices to be traversed by the flow and hence decrease the delay. For that, the system will identify the list of VNFs that could be eliminated from the deployed slices. This list will be then communicated to the *VM Controller* in order to release the corresponding VMs. After these modifications, the service chain as well as the slice description will be updated.

3.5 Vehicular Network Use Case

In order to illustrate how our architecture works, we interest in this section on vehicular network slices use case. Vehicular Ad-hoc Networks (VANETs) are considered as a promising network design for intelligent transportation systems [73]. They enable a class of applications that requires time-critical responses, very high data rates and a consideration of the very high mobility of vehicles.

The architecture of VANETs is complex, inflexible and characterized by an absence of a centralized control which is a challenging case study in resource management and data scheduling. The SDN/NFV paradigm presents an opportunity for a more flexible and programmable network. In fact, the application of SDN in vehicular networks has recently been the focus of several researches [74], [75] and [76]. These works debate the use of the SDN approach in the context of vehicular networks. In fact, SDN and NFV enhance the scalability, reliability and flexibility while reducing latency and cost. For instance in [77], evaluation results demonstrate the benefits of SDN in managing a video

streaming application over several wireless interfaces relying on Vehicle to Vehicle (V2V) and Vehicle to Infrastructure (V2I) communications. In [78], authors propose a network architecture that enables the slicing function for vehicle network service and improves the bandwidth utilization. In [79], authors study the case of cooperative driving among autonomous vehicles.

In accordance to our architecture, we consider cars as the nodes of the virtual infrastructure. For the provision of vehicular network services, a dedicated slice will be deployed across the different network resources whenever a new service is requested. Those services are exposed to the drivers on a catalogue provided by the *Service Provider*. As example of services we cite:

- The smart parking use case is considered as one of the most popular *V2I* applications. Vehicles will retrieve data through sensors placed on the ground in order to find a vacant parking space.
- The video streaming sharing between vehicles is a second possible application where vehicles share videos about the road to better understand their environment and anticipate the risk of collision for example. For this service the network has to provide the type of video (HD streaming, Ultra-HD, etc.) as well as the area of coverage of this service (zones 2 and 3 at the city, for example).

For the establishment of vehicular service, the *Control Orchestrator* of the *Service Provider* domain must discover, identify and select the nodes of the virtual network . In the case of video streaming use case, it has to identify the nodes capable of transmitting information (existence of an on-board video camera in the vehicle) and/or nodes capable of transferring the video. Controllers and orchestrators proposed by the DANSO architecture for the vehicular slice will have a global view of all vehicles path and sensors location. Thus, the exchange of information and the communication between vehicles and between vehicles and fixed sensors will be more feasible in real time. The use of DANSO architecture makes the monitoring of road conditions in congestion and emergencies cases more flexible and scalable.

The main actions of the controllers and orchestrators for a new slice creation are presented in the next paragraphs. For each interaction we specify the number of the corresponding message in the diagram presented in Fig. 3.10.

1. When the user sends a demand (*message 1*), the *Control Orchestrator* is the first to receive its request. This request will be translated by the translation module of the *Control Orchestrator* (*message 2*).

2. After that, it will be sent to the *Network Store controller* in order to ask for the user's profile (*message 3*). The complete description of the user's demand is composed of the information about its policy and the translation of its request. This description is sent to the *slice controller* which searches in the *Information Base* for a slice that responds to this description (*messages 4 and 5*). In this example, we suppose that the *Slice Controller* does not find a suitable slice for the request (*message 7 and 8*). Thus, it informs the *Network Store Controller* that he did not find an already implemented slice (*message 9*).

3. When the *Network Store Controller* receives a negative ACK, it prepares a service chain composed of VNF from the *Network Store* according to the demand description (*messages 10, 11 and 12*). Then, it communicates this service chain to the *Infrastructure Controller* (*message 13*).

4. When the *Infrastructure Controller* receives the service chain, it studies the *Virtual Infrastructure* (VI) performances while considering the service chain requirements. This study is done due to the *Optimization Framework* that receives the required information and executes its algorithms in order to figure out if the slice can be implemented or not (*messages 14 and 15*). In our example, we suppose that the VI is able to support a new slice creation. In this case, the *infrastructure controller* asks the *VM controller* to implement the indicated VNF into the created VMs for this slice (*message 17*).

5. If the implementation is done successfully, then all the controllers will be informed and the *Information Base* will be updated (*messages [18,21]*).

6. When the *Control Orchestrator* is notified that the slice is ready to execute the user's application, it informs the user that its request is accepted and the communication through this slice will be set up (*messages 22 and 23*).

We note that the communication between all the modules of the *Control Layer* and also between the *Control Layer* and the *Network Store Layer* goes through the

Control Orchestrator. These interactions are not presented in this diagram for reason of simplicity.

Conclusion

This chapter presented a new architecture for the network slicing deployment and management based on several paradigms, such as SDN and NFV. This architecture considers the delegation approach for the orchestration, control and management of network slices. We described each level of the architecture which are the *Network Store Level*, *Control and Orchestration Level*, *Network Slices* and *Virtual Infrastructure Level*. We also detailed the slice creation and the slice management operations.

In the next chapters, we are going to detail the *Dynamic Handler Framework* and the *Optimization Framework*. We will present the developed algorithms for each framework for the creation and the management of network slices.

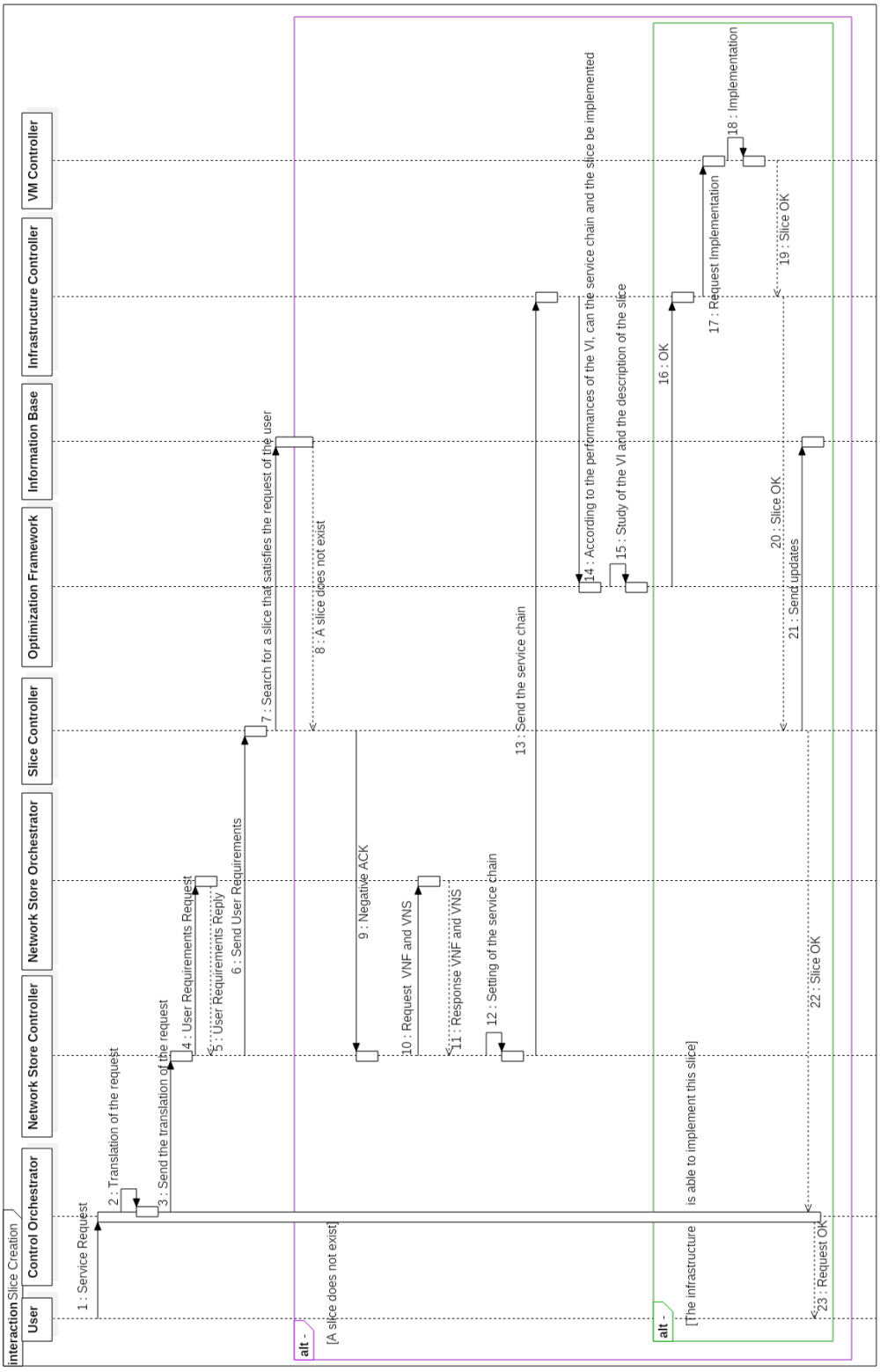


Figure 3.10: *Diagram of sequence of a new slice creation*

CHAPTER 4

Optimization of Network Slice Deployment

Introduction

THIS chapter addresses the problem of slice deployment optimization within the 5G context. In fact, the admission control and resources allocation issues are challenging aspects for the network slicing technology. Our work is related to this context. In fact, we interest on the deployment of E2E network slices for 5G use cases. For this purpose, we consider the requirements of each 5G use case class namely in terms of *reliability*, *availability* and *latency*. We present a mathematical formulation of the users' admission control problem and also the problem of slice deployment. Then, we present our heuristic algorithms which aim to implement the slice on the most convenient resources and map the user's requests to the most suitable slices.

4.1 Problem Statement

The use of the network slicing concept allows the network system to provide as many network slices as requested over the same network infrastructure. As shown in Fig. 4.1, a slice is composed of a set of VNFs. Each implemented network slice serves a specific type of application which is defined by a *Service-Level Agreement* (*SLA*) description. This composition of VNFs is adequately configured and chained according to the use

case requirements. It is then implemented on top of the virtual infrastructure.

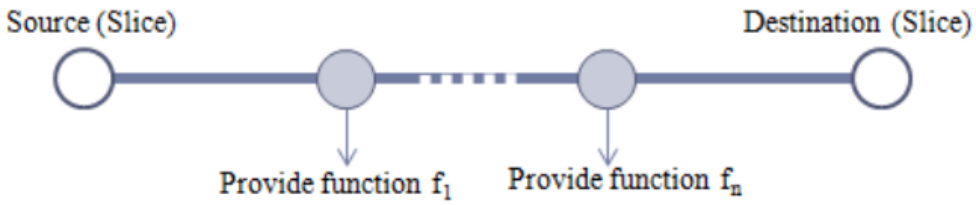


Figure 4.1: *An illustration of a Network Slice*

The E2E slice performance depends on the performance of all its constituent network resources and applications. Thus the E2E reliability and availability of the deployed slice depends on the reliability and availability of each constituent functional block. The combination of several components in the slice deployment is characterized by a serial dependency. In fact, all the constituent functional blocks need to be available and reliable at the same time in order to have the entire slice available and reliable. For example, the deployment of a VNF over the NFV Infrastructure (NFVI) is a serial dependency and its reliability and availability depends on the physical hardware, the hypervisor and the software of the VNF itself.

As presented in Chapter 2, we consider three classes of use cases which are i) massive machine type communication (mMTC), ii) critical communications, and iii) enhanced mobile broadband (eMBB). Each use case is characterized by special requirements in terms of reliability, availability and latency. Our work addresses the optimization of the mapping of users' demands within the deployed slices while considering their requirements. The reliability, availability and latency rate required by each 5G use case depends on its classification.

4.2 System model

According to our proposed architecture presented in Chapter 3, when the framework receives the user's demand, it analyses its requirements. Following this step, users' demands will be communicated to the *admission control mechanism* as shown in Fig. 4.2. the *admission control mechanism* searches for an existing slice which responds to the demand. Two cases are possible. First, the system may find a slice that suits perfectly the user's request. In this case, it will study the performances of this slice. If the slice is able to serve the user's demand then the system will map this demand into this slice.

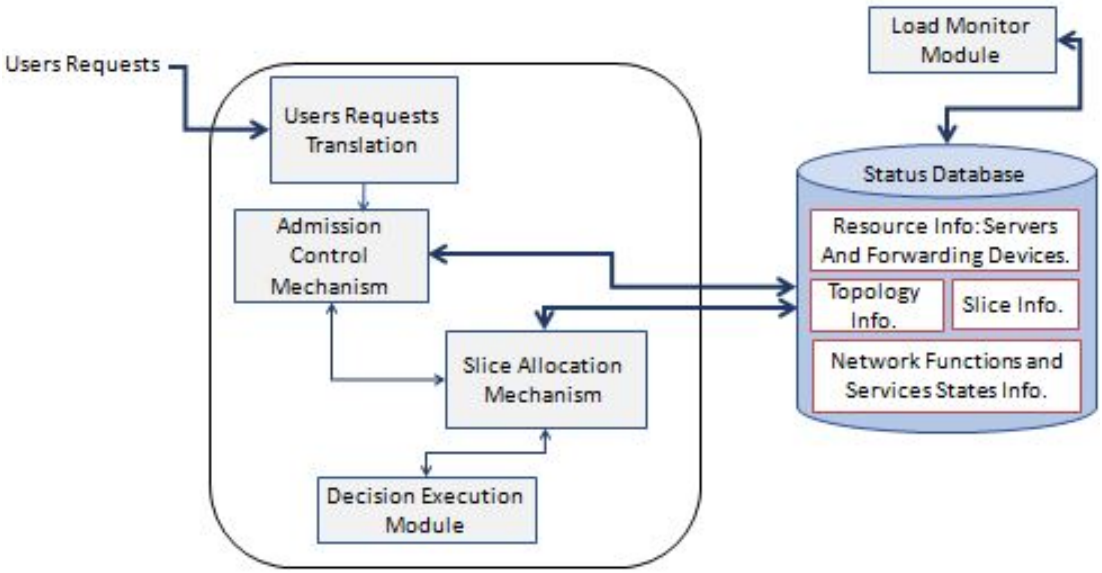


Figure 4.2: *Optimization framework*

The second case is when the system does not find a matching slice for this user's request. In this case, the system will create a new Network slice.

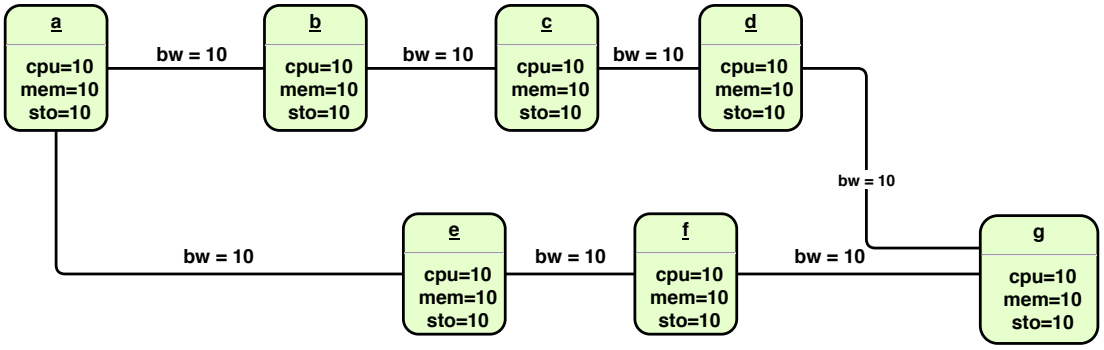
In this section, we present our system model. We represent both Virtual Network (VN) and Network Slice (NS) as weighted undirected graphs.

4.2.1 Virtual Networks Modeling

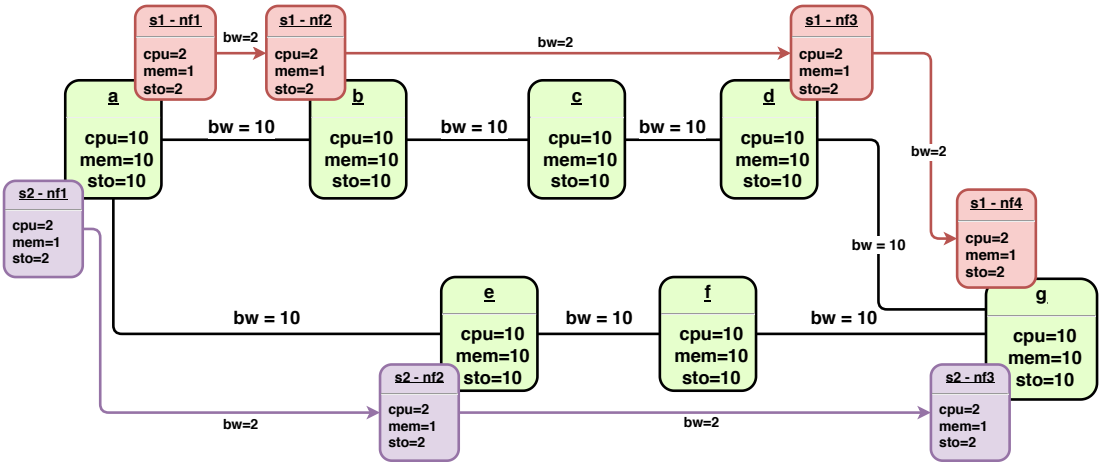
The *Virtual Infrastructure (VI)* provides an abstraction of the physical network resources layer. It contains N Forwarding Elements (FEs) interconnected according to the VI topology and a set of K servers which are connected to the FEs.

We model the virtual network infrastructure as a weighted undirected graph and we denote it by $G^S = (V^S, N^S, L^S)$ where V^S is the substrate virtual machines, N^S is the substrate forwarding nodes and L^S is the set of substrate links. On top of this VI, several slices are implemented according to users' requests.

In figure 4.3a, we depict an example of a virtual network modeled as a multi-dimension weighted undirected graph, where the virtual node set V is equal to $\{a, b, ..., g\}$; the virtual link set L is equal to $\{(a,b),(b,c),...,(d,g),(f,g)\}$; the weights associated with each virtual nodes indicate the CPU, memory and storage capacities of that node; and finally the weight associated to each virtual edge denote the bandwidth capacity of that link.



(a) *Virtual network topology*



(b) *Network Slice deployment on top of a Virtual Network*

Figure 4.3: *Network slice deployment topology*

As illustrated in the figure 4.3a, we consider a virtual network composed of virtual nodes connected via virtual links. To each virtual nodes and virtual links we associate a set of resources capacity such as CPU, memory and storage for virtual nodes and bandwidth for virtual links

4.2.2 Network Slice Modeling

A network slice can be viewed as a chain of Virtual Network Functions (VNFs) as well as physical network functions (PNFs) implemented on top of the virtual and/or physical infrastructure. This composition of VNFs/PNFs is adequately configured and chained according to the application requirements. In the following, we focus only on network slices composed of a set of VNFs.

In order to create a slice, an interconnection between virtual network resources is required. We indicate by $S = \{Slice_1, \ldots, Slice_n\}$ the set of implemented slices in the virtual network. Each slice $Slice_i$ is composed of an interconnection between virtual network resources. It is an association of a set of v Virtual Machines (VM) and m forwarding nodes interconnected together with a set of p weighted links and we denote it by $Slice_i = \{VM_v, N_m, Link_p\}$.

Figure 4.3b depicts an example of the deployment of two slices (s_1 and s_2) on a virtual network. The slice s_1 is formed of *4* virtual functions and the slice s_2 is formed of *3* virtual functions. As we can see, each slice starts from one virtual node considered as the source of the slice to reach another virtual node considered as the destination of the slice. In this example, both slices s_1 and s_2 starts at the node *"a"* and ends at the node *"g"*. We can also notice that the deployment of slices require the installation of some network functions on virtual nodes. For instance, the slice s_2 install three network functions $s_2(nf_1)$, $s_2(nf_2)$ and $s_2(nf_3)$ on virtual nodes *"a"*, *"l"* and *"g"*, respectively. This installation requires that the virtual node satisfies the resources constraint required by the network functions. In addition to the network function installation, we also need to guarantee the resources constraints related to the links chaining two network functions. For instance, for the slice s_2, we need to guarantee a bandwidth of $2Mb/s$ between network functions $s_2(nf_1)$ and $s_2(nf_2)$ as well as between network functions $s_2(nf_2)$ and $s_2(nf_3)$. As we can see, in the figure 4.3b, to guarantee the bandwidth between two NFs we need to reserve this bandwidth on all the virtual links used in

order to chain those two NFs. For example, to guarantee $2Mb/s$ between $s_2(nf_2)$ and $s_2(nf_3)$, we need to reserve $2Mb/s$ on both virtual links (e,f) and (f,g).

4.2.3 Resources Modeling

We denote by NR_j the network resource *"j"*. In order to propose a generic system model, we consider in this work different types of resources. As in most of the works proposed in the literature, we consider CPU (CPU_j) and memory (Mem_j), as virtual node resources. We consider also reliability (R_j), availability (*"*A_j*"*) and latency (L_j) as nodes characteristics. Basically, each virtual function asks for a certain amount of resources, when deployed on a given virtual node. In addition to the node resources, we consider the bandwidth required to chain two virtual functions.

4.2.4 User Requests Modeling

Each implemented network slice serves several user's requests. The set of user's requests is denoted by $UR = \{UR_1, \ldots, UR_k\}$.

The user's request UR_i is characterized by its required SLA. In our work, this SLA is described based on three parameters which are the *Reliability* R_i, *Availability* A_i and *Latency* L_i. When UR_i is received, the *Admission Control mechanism* will search from the information database the set of slices that corresponds to the required SLA.

4.3 Network slice Deployment Process

In this section, we consider the case of new slice implementation. We describe the mathematical model that we consider in order to formulate the slice placement problem. We present next our algorithms to solve this problem. Our objective is to maximize the slice deployment score by selecting the most convenient network resources.

4.3.1 Problem Formulation

The Slice creation process is characterized as a set of M tasks to be executed with a deadline d_{thr}. A task is defined as the allocation of network resources to the slice and the implementation of the required Virtual Network Function (VNF). Each task is characterized by its reliability, availability and latency requirements. We consider a binary variable $w_{(i,j)}$ to indicate if a the network resource NR_j is allocated to the slice

$Slice_i$ or not:

$$w_{(i,j)} = \begin{cases} 1 & \text{if the resource}{=}NR_j \text{ is allocated to } Slice_i \\ 0 & otherwise \end{cases} \tag{4.1}$$

In the following, we suppose that all the forwarding elements, links and also servers have unlimited computational capacities to implement all the requested slices. We interest only on the reliability, availability and latency characteristics of those network resources.

$$NR_j \begin{pmatrix} Reliability = & R_j \\ Availability = & A_j \\ Latency = & L_j \end{pmatrix} \tag{4.2}$$

When a new slice creation request (SCR) is received, the Network Orchestrator has to search for the target resources to allocate for this slice based on its service-level agreement (SLA). In fact, the required SLA of each slice is described based on the following parameters:

$$SCR_i \begin{pmatrix} Reliability = & R_i \\ Availability = & A_i \\ Latency = & L_i \end{pmatrix} \tag{4.3}$$

Our goal is the maximization of the availability and reliability for each created slice while considering the user's demand requirements. Therefore, the following constraints have to be considered:

- Availability constraint for SCR_i:

$$A_j > A_i \text{ for each request } i \text{ and } \forall \text{ the Network Resource } j \tag{4.4}$$

  Constraint 4.4 guarantees that the availability of the selected network resource is equal or higher than the required user's availability.

- Reliability constraint for SCR_i:

$$R_j > R_i \text{ for each request } i \text{ and } \forall \text{ the Network Resource } j \tag{4.5}$$

  Constraint 4.5 guarantees that the reliability of the selected network resource is

equal or higher than the required user's reliability.

- Delay constraint for SCR_i:

$$\sum_j D_j < D_i \text{ for each request } i \text{ and } \forall \text{ the Network Resource } j \tag{4.6}$$

Constraint 4.6 ensures that the E2E delay from the source node to the destination node is lower than the maximum tolerant delay by the requested service.

Our objective is the maximization of the E2E reliability (R) and E2E availability (A) of the deployed slice as shown in the following objective function:

$$\text{Maxmize } (A + R) \tag{4.7}$$

This problem is NP-Hard. Therefore, we introduce in the next section an heuristic algorithms which search for the optimal solution by selecting the most suitable resources for the requested slice in a reasonable amount of time.

4.3.2 Proposed Algorithm

In order to determine the target resources for the slice creation, we have opted for the utility theory. In fact, using the utility function, our algorithm computes the score of each candidate network resource while taking into consideration input parameters as well as their weights. Then, the selection will be based on the score of each resource. Our algorithm will select the resources that have the highest utility score. The utility function that has been retained is as follow [80]:

$$U(x) = 1 - e^{-\alpha x} \tag{4.8}$$

Where x is the decision vector and α is the corresponding weight (α>0). In this work, the decision vectors that we implement in the utility function are the network resources availability, reliability and latency. The formula used to compute the score of the network resource NR_j while taking into account the requirements of the slice

request i is the following:

$$SC(NR_j) = (1-e^{-\alpha_A \times A_j}) \times (A_i \otimes A_j) \times (1-e^{-\frac{\alpha_L}{L_j}}) \times (L_i \otimes L_j) \times (1-e^{-\alpha_R \times R_j}) \times (R_i \otimes R_j) \tag{4.9}$$

Where

$$(A_i \otimes A_j) = \begin{cases} 1 & if A_j \geqslant A_i \\ 0 & otherwise \end{cases}$$

$$(R_i \otimes R_j) = \begin{cases} 1 & if R_j \geqslant R_i \\ 0 & otherwise \end{cases}$$

$$(L_i \otimes L_j) = \begin{cases} 1 & if L_j \leqslant L_i \\ 0 & otherwise \end{cases}$$

α_A: Weight of the decision criterion A_i.

α_R: Weight of the decision criterion R_i.

α_L: Weight of the decision criterion L_i.

This utility function is an increasing function. Therefore, the variation of utility values is proportional to the changes in variable values. Also, it distinguishes between ascending and descending criteria. For example, the availability and reliability are ascending criteria that has to be maximized to better meet the user's satisfaction, while the latency is a descending criterion to minimize.

The chosen function returns values normalized to [0, 1] domain. For the decision criteria we set $\alpha = \frac{1}{3}$ which means that equal weight is given for the different performances criteria.

After computing different scores of network resources between the source node and the destination node, the algorithm will select the one with the best score value. This process enables us to determine the best network slice deployment scenario while considering network resources characteristics.

We present in the following the proposed algorithms that search for the slice implementation scenario that satisfies the user's request. The proposed Algorithm 1 determines at first the list of all possible paths from the source to the destination. In fact, the *Network Control System* has a complete view of all the network resources. For

each requested slice, it determines the different implementation scenarios. Which means that Algorithm 1 determines within a deadline execution d_thr the convenient available network slice implementation scenario between the source node and the destination node. This scenario is selected while executing the second proposed algorithm. In fact, Algorithm 2 computes the score of each selected path and the score of each resource belonging to this path. When the Algorithm 2 finds a path that has a non-null score, this scenario will be retained. According to the equation 4.9, a path has a non-null score only if its reliability, availability and latency characteristics correspond to the slice requirements.

Algorithm 1 Selection of the Slice Network Resources

src = Source node of the $Slice_i$
dst = Destination node of the $Slice_i$
Initialize S = Null; S is the retained network path assigned to the $Slice_i$
P = FController (src, dst); This function restitutes the list of all possible paths from src to dst computed by the controller.
while S = Null and $d < d_{Thr}$ **do**
    S = ScoredPath (src, dst, P, score); score is an input/output argument for ScoredPath function. It is used to compute the score of the retained path.
end while
if score <> 0 **then**
    confirm S as the target resources for the $Slice_i$
    return S
else
    return request timeout; that means d >= d_{Thr} or score=0
end if

4.3.3 Numerical Results for Network Slice Deployment

In this section, we present simulation results of our proposed *Utility-based Algorithm* (*UBA*) compared to the following mono-criterion algorithms:

- *Delay-based Algorithm* (*DBA*): the choice of network resources is based on the delay parameter. The aim of this algorithm is the minimizing of the E2E delay.
- *Availability-based Algorithm* (ABA): the slice is embedded on the most available resources.
- *Reliability-based Algorithm* (*RBA*): the selection of the network resources for the requested slice is based on the reliability parameter. The most reliable network resources are selected.

Algorithm 2 ScoredPath (input src, input dst, input P, input/output score)

L = Null; L is the retained network resources path for the $Slice_i$.
c1 = card1(P); The function card1 computes the cardinality or the number of possible paths from src to dst.
i = 1
PS = 0; the score of the $Path_i$ P (each path of the list P is referenced as $Path_i$)
score = 0; score of the retained path.
while i<=c1 and PS=0 **do**
    c2 = card2($Path_i$); The function card2 computes the cardinality or the number of network resources of the $Path_i$..
    v = 1
    for j = 1 .. c2 **do**
        Compute the score SC(NR_j) of the network resources NR_j of the $Path_i$ using the equation (9).
        ; compute the path score.
    end for
    PS = v;
    if PS <> 0 **then**
        L = $Path_i$; the retained path.
        score = PS; score of the retained path.
    end if
    i++
end while
return L

We have evaluated our algorithm using MATLAB for the three use cases families presented in Table 2.1.

Two scenarios of real network topologies were considered: i) the Abilene topology presented in Fig. 4.4 as a small network scenario and ii) the Geant topology presented in Fig. 4.5 as a large network scenario. Those two topologies are obtained from the Internet Topology Zoo project [81]. The reliability, availability and delay of network resources are generated randomly.

Fig. 4.6 presents the score of the slice deployment in the case of the small network scenario. We have evaluated the score for the enhanced mobile broadband, massive IoT and also critical communications use cases. This score has been evaluated by applying our proposed UBA algorithm as well as the ABA, RBA and DBA algorithms. Simulation results show that the total score of the slice deployment process for each use case family is enhanced when our algorithm is applied compared to the other considered algorithms. In fact, our algorithm takes into consideration the three network performance criteria (availability, reliability and latency) for the network resources

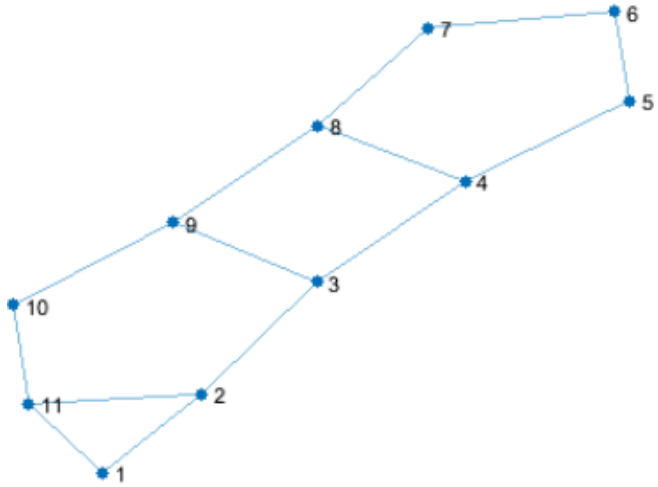


Figure 4.4: *The Abilene topology (11 nodes, 28 directed links)*

selection while the mono-criterion algorithms consider only one network performance indicator which reduces the score of the deployed slice.

Fig. 4.7 presents the E2E score generated by the different algorithms for the large network topology. Similarly to the small network scenario, simulation results show that the score of the slice deployment when our proposed algorithm is applied is higher than the score generated by the other mono-criterion algorithms. Moreover, for the other algorithms, the total slice creation score decreases with the increase of the number of network resources while we maintain a high score when we apply our proposed algorithm.

For the two network topologies, the eMBB and the massive IoT use cases families have the highest utility score for all the applied algorithms. In fact, those two families are characterized by a flexible requirements compared to the critical communications use case family which is characterized by stringent requirements. Simulation results demonstrate that the UBA increases the utility of the deployed slices for all the use case families as it selects the network resources based on the three network performance indicators.

As we can see, our algorithm is efficient even in a large scale network. The selection of the best slice creation scenario among available ones is insured when our proposal is applied. In fact, the three network performance indicators are considered in the UBA algorithm, while for the other algorithms only one indicator is considered and the others are ignored. Thus, our algorithm provides a better QoS for the deployed slices.

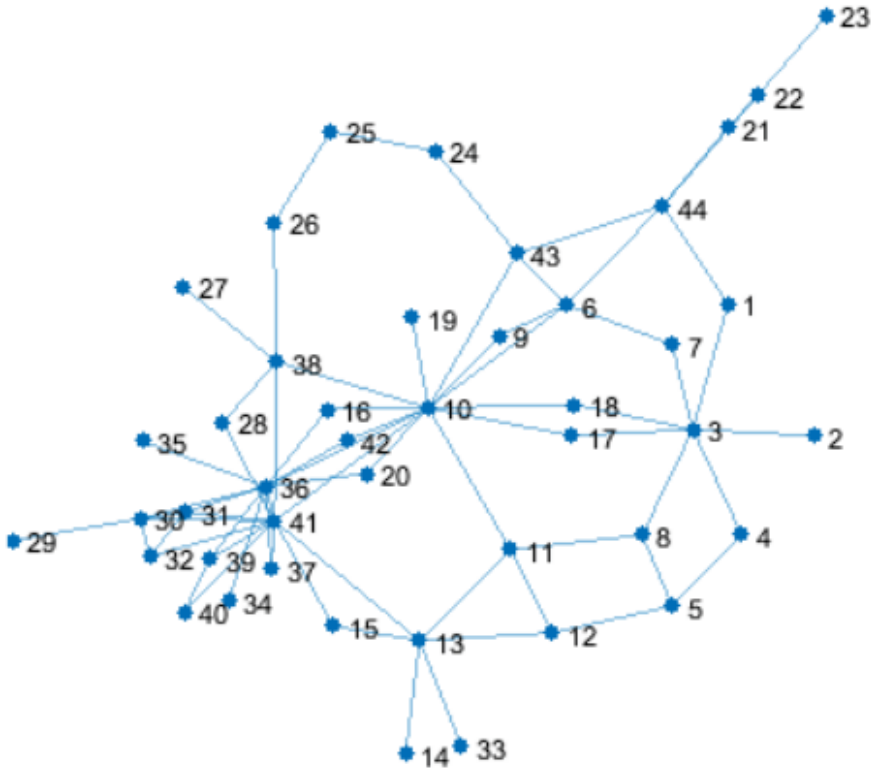


Figure 4.5: *The Geant topology (44 nodes, 142 directed links)*

4.4 Admission Control and Request Mapping Mechanism

In this section, we suppose that slices that suit the users' demands are already deployed in the network. Our goal is to select the best target slice among available slices for each user's request based on several network criteria such as the network resources latency, availability and reliability as well as their computational performances. For this purpose, we present, in this work, our proposed system model for the admission control. Then, we propose an optimization algorithm for the mapping of users' demands into the most convenient network slices according to their requirements description, the actual slices performances and also the offered E2E availability, reliability and latency. Our proposed algorithm aims to minimize the deployment cost of the requests while maintaining the required performances.

In the following subsections, a mathematical formulation of our proposed algorithm and a performance evaluation are presented. We demonstrate that by providing the ideal slice for users' requests, the quality of service afforded to the users will be enhanced.

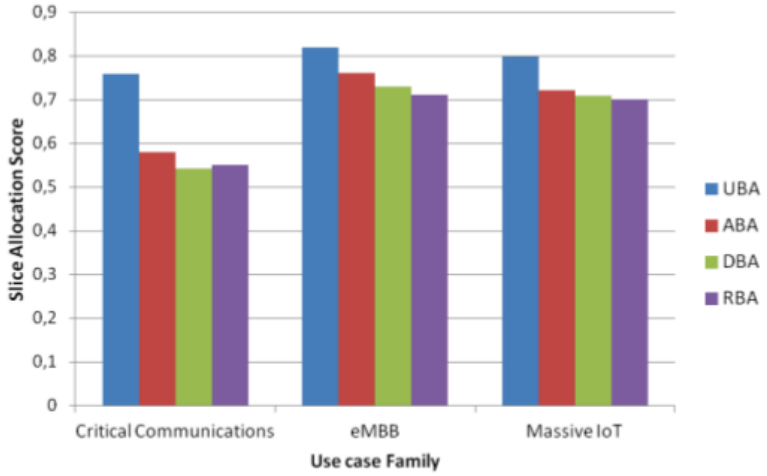


Figure 4.6: *Slice deployment score in Abilene topology*

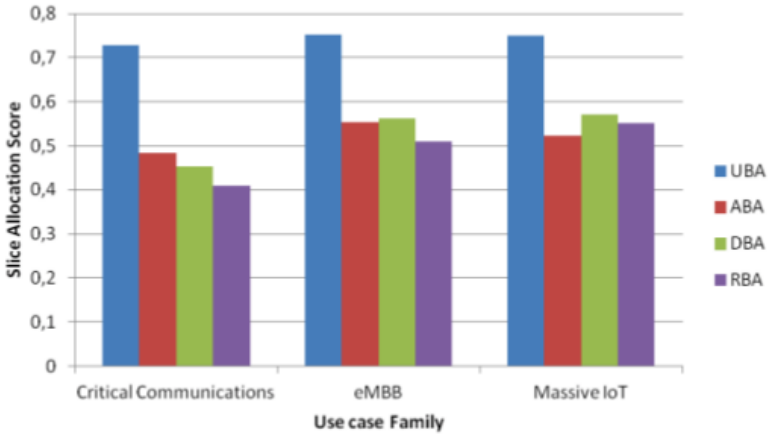


Figure 4.7: *Slice deployment score in Geant topology*

4.4.1 Problem Formulation

A $Slice_i$ is considered as a corresponding slice for the user's request if its availability A_j is equal or higher than the required user's availability, its reliability R_j is equal or higher than the required reliability and the afforded E2E latency L_j is lower than the maximum tolerant delay by the requested service. Basically:

$$\begin{cases} A_j \geqslant A_i \\ R_j \geqslant R_i \\ L_j \leqslant L_i \end{cases} \tag{4.10}$$

However, in order to map a user's request into an existing slice, several additional parameters need to be considered. In fact, each network resource of the infrastructure is characterized by a limited capacity. For instance, the restricted capacity of servers limits

the implementation of network functions on top of them. In fact, network functions will be implemented on virtual machines installed on those servers and they will be sharing the total resources in terms of memory, CPU, etc.

Each Virtual Machine $VM_i \in V^S$ is characterized by its capacity $C(VM_i)$ which depends on the CPU and Memory utilization rates. The capacity of each forwarding node $n_i \in N^S$ is defined by $C(n_i)$ and it depends on the occupation rate of its flow table. The capacity of each link $l_i \in L^S$ is characterized by its bandwidth capacity and denoted by $C(L_i)$.

4.4.2 Overload Cost and Requirements based Algorithm for Admission Control Mechanism

In order to determine the target slice for the user's request, we present in this part our proposed algorithm. Algorithm 3 describes the process of the selection of the best slice scenario in order to map the user's request to this slice within an execution deadline d_{thr}. It aims to minimize the total cost of requests mappings while considering the requested SLA.

As discussed in the previous sub-section, the algorithm starts by searching for a slice that corresponds to the request. After finding the set of slices that satisfy the user's request, the system has to verify the capacities of the VMs, switches and links that constitute this slice. To do so, we propose an overload cost function which computes the overloading rate of each network resource of the slice. Then, our proposed algorithm will select the slice that has the lowest overloading cost.

4.4.2.1 Overloading Cost of Virtual Machine

In order to compute the overloading cost of the virtual machine, we consider two parameters which are the CPU and the memory utilization rates. The formula used to compute the cost is the following:

$$OvC(VM_j) = (1-e^{-\alpha_{CPU} \times CPU_j}) \times (CPU_{Thr} \otimes CPU_j) \times (1-e^{-\alpha_{Mem} \times Mem_j}) \times (Mem_{Thr} \otimes Mem_j) \tag{4.11}$$

Where

$$(CPU_{Thr} \otimes CPU_j) = \begin{cases} 0 & \text{if} \quad CPU_{Thr} \leqslant CPU_j \\ 1 & otherwise \end{cases}$$

$$(Mem_{Thr} \otimes Mem_j) = \begin{cases} 0 & \text{if} \quad Mem_{Thr} \leqslant Mem_j \\ 1 & otherwise \end{cases}$$

α_{CPU}: the weight of the decision vector CPU_i.

α_{Mem}: the weight of the decision vector Mem_i.

$\alpha_{CPU} = \alpha_{Mem} = 0.5$.

CPU_{Thr}: a defined threshold for the CPU parameter.

Mem_{Thr}: a defined threshold for the Memory parameter.

4.4.2.2 Links Overloading Cost

In order to compute the overloading cost of virtual links, we consider the bandwidth capacity. The formula used to compute the cost is the following:

$$OvC(l_j) = (1 - e^{-\frac{1}{Bw_j}}) \times (Bw_{Thr} \otimes Bw_j) \tag{4.12}$$

Where

$$(Bw_{Thr} \otimes Bw_j) = \begin{cases} 1 & \text{if} \quad Bw_{Thr} \leqslant Bw_j \\ 0 & otherwise \end{cases}$$

Bw_{Thr}: a defined threshold for bandwidth parameter.

4.4.2.3 Switches Overloading Cost

For the computing of the overloading cost of forwarding nodes, we consider the occupation rate of its flow table. The formula used to compute the cost is the following:

$$OvC(n_j) = (1 - e^{-FToc_j}) \times (FToc_{Thr} \otimes FToc_j) \tag{4.13}$$

Where

$$(FToc_{Thr} \otimes FToc_j) = \begin{cases} 0 & \text{if} \quad FToc_{Thr} \leqslant FToc_j \\ 1 & otherwise \end{cases}$$

$FToc_{Thr}$: define the threshold for the occupation rate of the flow table parameter.

4.4.2.4 Slice Overloading Cost

After computing different scores of network resources of the slice, the following formula computes the overloading cost of the entire considered slice.

$$OvC(Slice_j) = \prod OvC(VM_j) \times \prod OvC(l_j) \times \prod OvC(n_j) \tag{4.14}$$

Our proposed Algorithm 3 will select the slice with the lowest overloading cost value. This process enables us to determine the best network slice deployment scenario while considering network resources characteristics.

Algorithm 3 Selection of the target slice for the user's request

1: UR_i = The request of $User_i$
2: Initialize L = Null; L is the retained network slice scenario assigned to the $User_i$
3: R = $Slice_1$, ..., $Slice_k$ = FController (R_i, A_i, L_i); This function restitutes the list of all possible slices for the user's request UR_i computed by the controller.
4: Initialize PS= OvC($Slice_1$)
5: L =$Slice_1$;
6: j=1;
7: **while** d < d_{thr} **do**
8: j++;
9: Compute the cost OvC($Slice_j$) of the $Slice_j$ using the equation (9).
10: **if** (OvC($Slice_j$) < PS and OvC($Slice_j$) >0 **then**
11: L = $Slice_j$; the temporary retained path.
12: PS = OvC($Slice_j$); cost of the temporary retained path.
13: **end if**
14: **end while**
15: **if** PS > 0 **then**
16: confirm L as the target slice for the UR_i
17: return L
18: **else**
19: **return** No corresponding slice was found; In this case, a new slice will be created in order to serve the user's request.
20: **end if**

4.4.3 Numerical Results for Admission Control Algorithm

In this section, we present simulation results of our proposed Overload Cost and Requirements based algorithm (OvC&R) compared to the following algorithms:

- *Overload Cost-based Algorithm (OvCA)*: the choice of the target slice is based only on the overloading cost of the slices. The slice that will be selected is the

less overloaded slice.

- *Requirements-based algorithm (RBA)*: the user's request is mapped to the slice that responds the best to the users' request in terms of latency, reliability and availability.

We have evaluated our proposed algorithm using MATLAB for the three use cases families presented in Table 2.1. The reliability, availability, latency, and utilization rates of each network resources of the different slices were generated randomly.

Fig. 4.8 presents the overloading cost of the selected slices by the different presented algorithms. We have evaluated this cost for the enhanced mobile broadband, massive IoT as well as critical communications use cases. This cost has been evaluated by applying our proposed OvC&R algorithm as well as the OvCA and RBA algorithms. Simulation results show that the total cost of the slice deployment process for each use case family is minimized when our proposed OvC&R algorithm and the OvCA were used for the three use cases families. The RBA algorithm does not consider the overloading rate when selecting the target slices.

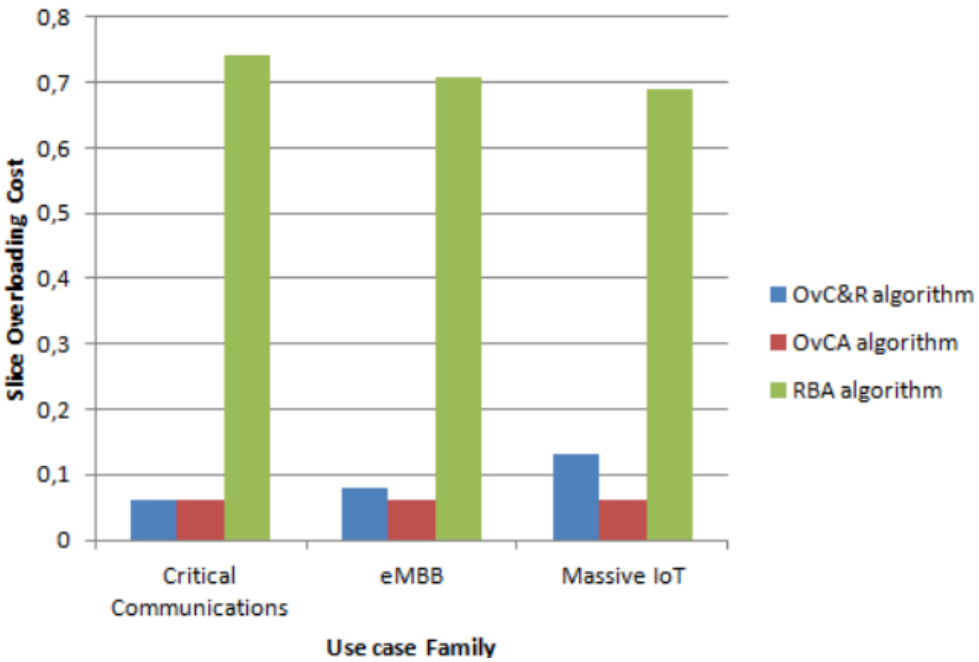


Figure 4.8: *Slice overloading cost*

Fig. 4.9 presents the percentage of choosing a slice with the right requested SLA description. Results show that the OvC&R and RBA algorithms select the most suitable slices in terms of reliability, availability and latency requirements compared to the slices

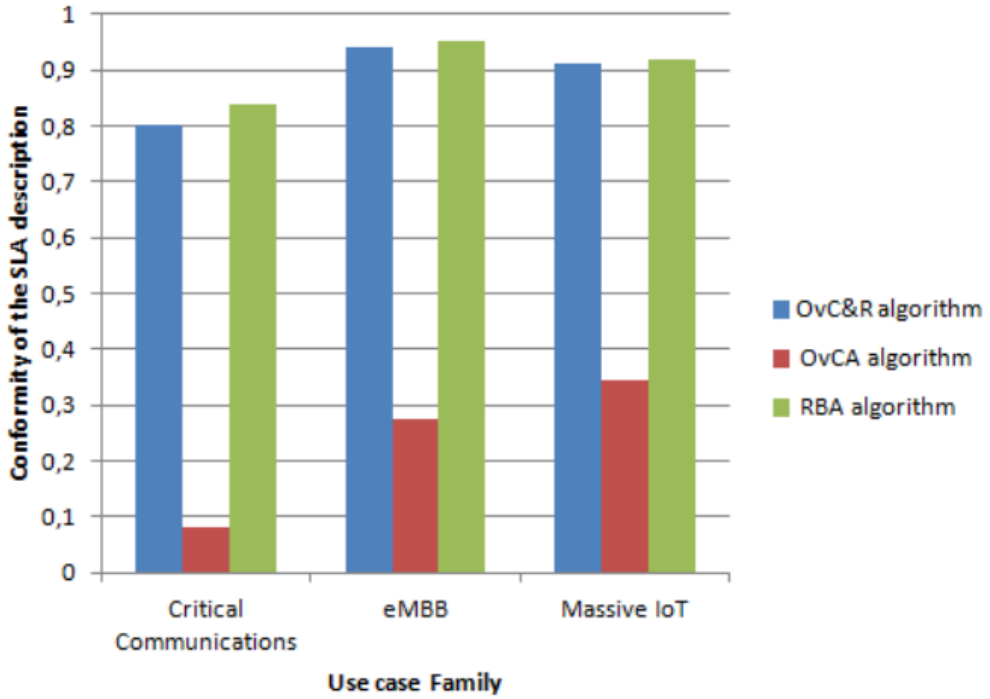


Figure 4.9: *Conformity of the SLA description*

selected by the OvCA. In fact, the OvCA does not guarantee any QoS when it selects the target slice.

For the obtained results, the eMBB and the massive IoT use cases are mapped to better compliant slices for the SLA description than the critical communications use cases. In fact, those two families are characterized by a flexible requirements compared to the critical communications use case family which is characterized by stringent requirements.

Therefore, our algorithm is efficient for the different use cases families. The selection of the best slice creation scenario among available ones is insured when our proposal is applied. In fact, our proposed algorithm takes into consideration both the SLA description and the performances of the available slices. For the other tested algorithms, only one performance indicator is considered and the others are ignored. Thus, our algorithm provides a better mapping scenario for the new users' requests.

Conclusion

This chapter proposes a *Utility-based* slice deployment algorithm (*UBA*) that aims to maximize the *QoS* for the deployed slices, in addition of the *Overload Cost and Requirements* based algorithm (*OvC&R*) that aims to maximize the *QoS* of the users' requests. Both UBA and *OvC&R* consider the use case requirements as well as the network resources characteristics in terms of *availability*, *reliability* and *latency*. In our scenario, we consider a network control and orchestration module that handles users' admission control and creates slices according to users' demands. Our objective is to maximize the slice deployment score and also to minimize the user's deployment cost by selecting the less overloaded slice. Simulation results show that the proposed algorithms are able to select the most convenient network slice deployment scenario while maximizing the total deployment score and also to select the most convenient network slice among available scenarios while minimizing the total overloading cost.

CHAPTER 5

Dynamic Handler Framework for Network Slices Management

Introduction

After the deployment of network slices, management challenges arise. In fact, network slice resources have to be managed in order to ensure the well operation and execution of the services. Therefore, the system has to react to the unexpected events that occur in the system. In this chapter, we study two management actions which are related to the slice congestion and users' mobility events. In the first section of this chapter, we detail our considered system model and the components of the *Dynamic Handler Framework*. The second section presents our proposal for the management of the slice congestion issue. We propose a fuzzy-based algorithm for the management of network resources. We interest mainly on the auto-scaling action as one of the most deployed strategy among management operation. The third section is related to the management of users' mobility event. We study the users' handover from one slice to another. We present our proposed algorithm for the management of the mobility between slices. Finally, we give some simulations and results of our proposed algorithms.

5.1 Presentation of the Dynamic Handler Framework

As presented in Chapter 3, we consider in our work a network architecture based on SDN, NFV and network slicing technologies. In this chapter, we interest particularly on the *Dynamic Handler Framework* (DHF).

The *DHF* deals with sudden events. It surveys the network state, analyzes it and informs the controllers in case of dysfunction. The *DHF* may detect several types of events and for each event a different action will be executed by the network. For instance, if the slice is congested, the *DHF* asks the system to either increase the performances of this slice through the migration of some VNF or the scaling-in of its resources.

In the first aspect of this work, we interest on the management of network slices based on their load state and the predicted future state. We propose a proactive and dynamic management algorithm which will predict the state of slice resources and then apply the convenient management decision based on fuzzy logic system. Fuzzy logic system is used to handle multiple decision issues. In fact, fuzzy controllers are characterized by their ability to analyze unclear data behavior such as the variation of network load [82]. In a second aspect, we focus on the support of vehicular network slices. We propose a management algorithm for the mobility of vehicles from one slice to another based on the load situation of available slices.

The *DHF* is composed of many sub-modules as depicted in Fig. 5.1. In the following we define each sub-module and we present its functionalities.

- *Load Monitor Module*: this module collects the CPU, disk, and memory utilization rates of the deployed slices and network resources. Thereafter, It stores these measures in the *Status Database*.

- *Status Database*: this database contains information about the state of network slices (e.g. Received Signal Strength (RSS)) and virtual machines. The state and computational capacities of each network slice is then updated periodically by the *Load Monitor Module*.

- *Workload Analyzer*: this module analyzes the *Status database* in order to compute the load state of each network slice future slice state. The results of this module

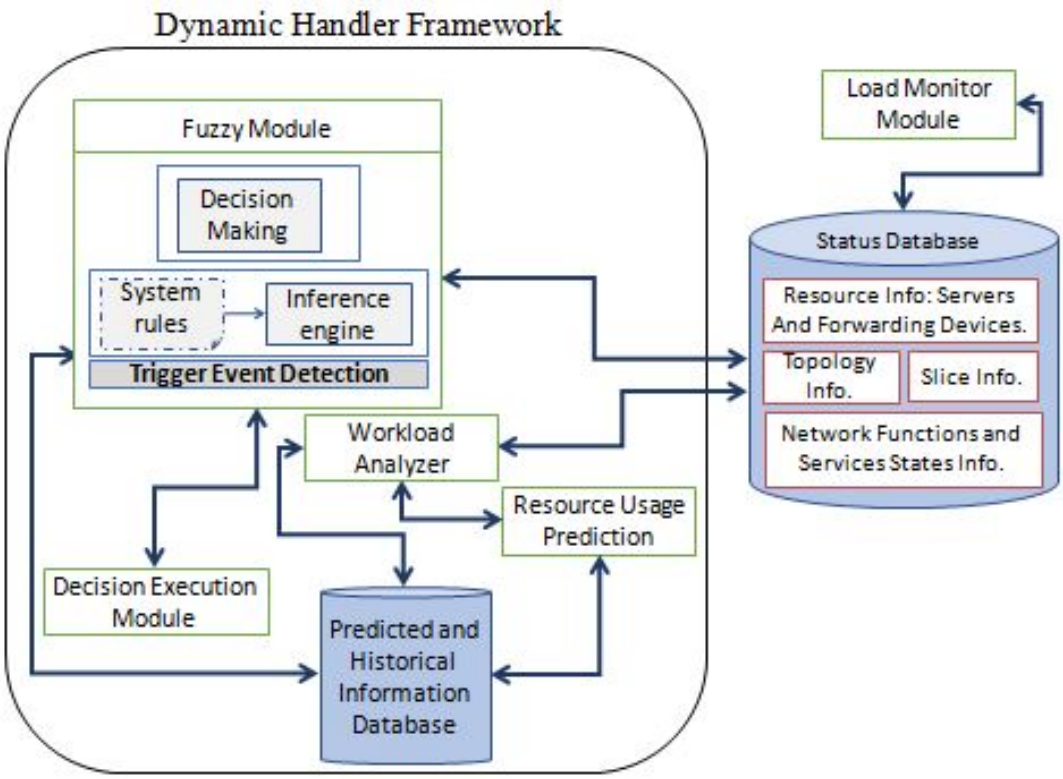


Figure 5.1: *Modules of the Dynamic Handler Framework (DHF)*

are stocked in the *Predicted and Historical Information Database* and will be used by the *Resource Usage Prediction* module in order to predict the future resource utilization rate of the corresponding slice.

- *Resource Usage Prediction*: this module will forecast for each slice the resource utilization rate of VNFs which are running on top of virtual machines. It will predict the future load state based on historical information about the load state. In this work, the prediction module is designed using the *Support Vector Regression (SVR)* model. By deploying a proactive solution, we can anticipate some problems that may occur in the slice. For instance, we will overcome the increase of the *processing delay* and avoid unnecessary resources migration.

- *Predicted and Historical Information Database*: this database contains historical and predicted information about the total load state of each network slice. These information will be used by the *Optimization module* and the *DHF* in order to better optimize and manage network slices.

- *Fuzzy module*: it deals with unexpected events like the congestion of the slice, the

failure of VMs, energy waste, etc. For this purpose, it examines periodically the *Predicted and Historical Information Database* and detects the trigger events for slices management (e.g. addition, deletion or modification of network resources). Thus, this module studies the slices performances and decides what management action is needed to be exercised on each slice.

- The *Decision Execution Module*: this module is responsible for the execution of the taken decision on the considered slices.

5.2 Management of Network Slices Resources

After the slice creation, the requested service is started. Performance information about slice resources are collected and stored in the *Status Database*. The *Load Monitor Module* studies this information, calculates load state which is characterized by the *load ratio* and the *predicted load-time fairness index* and then stores these measures in the *Status Database*. The *DHF* deals with sudden events like the congestion of the slice, the failure of the VM implementation, the user mobility, etc. It analysis regularly the *Status Database* and detects the trigger events for the addition, deletion and modification of slice resources. In order to decide if the slice has to be modified or not, the DHF executes the *Resource Management Decision (RMD)* algorithm which studies the slices performances and decides what management action is needed to be performed on each slice. The *Decision Execution Module* receives trigger events from the *DHF* and executes the action on the considered slices.

Our proposed solution is a multi-criteria and context-aware solution based on the fuzzy logic system for sudden events management. As shown in Fig. 5.2, it is composed of mainly three phases which are the *information gathering phase*, the *management decision phase* and the *management decision execution phase*.

5.2.1 Information Gathering Phase

We introduce in this part the considered metrics for the monitoring of network slices which are: *load ratio* and *predicted load-time fairness index*.

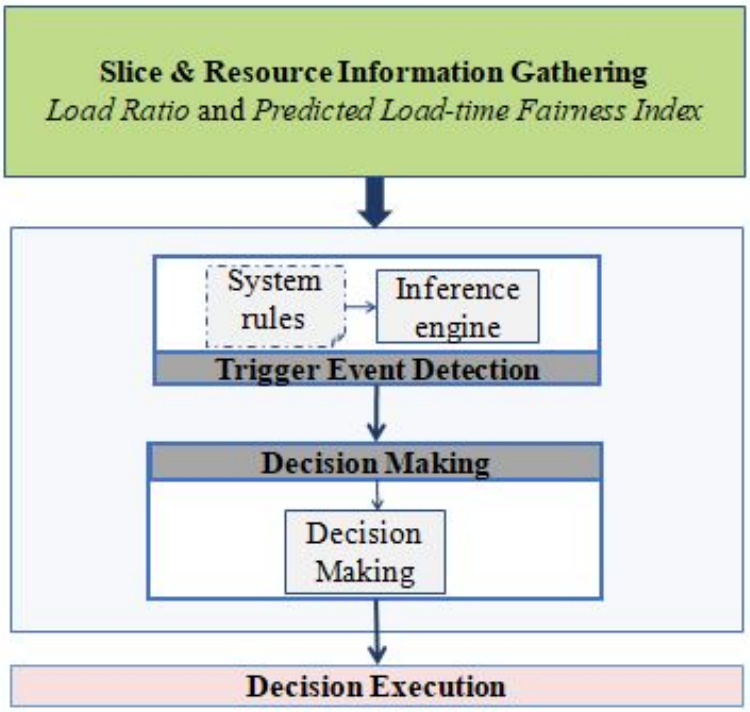


Figure 5.2: *Network slice management process*

5.2.1.1 Load Ratio

Considering the varying law of VMs and network elements load, the time T can be divided into n time periods. Thus, we define $T = [(t_1 - t_0), (t_2 - t_1), ..., (t_n - t_{n-1})]$ as the batch mode time interval. The load on the virtual element i is computed as follow:

$$L_i = \frac{1}{n} \sum_{k=1}^{n} L_{i,k} \tag{5.1}$$

where $L_{i,k}$ is the load on the virtual element i at time t_k.

For Load computation $L_{i,k}$, we take into consideration three resources which are *CPU*, *memory*, and *disk*. We define $L_{i,k}$ as the sum of *CPU load* $L_{i,k}^{CPU}$, the *memory load* $L_{i,k}^{Mem}$ and the *disk load* $L_{i,k}^{disk}$.

$$L_{i,k}^{CPU} = \sum_{j}^{F_j} r_{i,j} \tag{5.2}$$

$$L_{i,k}^{Mem} = \sum_{j}^{F_j} PF_{i,j} \tag{5.3}$$

$$L_{i,k}^{disk} = \sum_{j}^{F_j} r_{i,j} \times R_{i,j} \tag{5.4}$$

Where F_j is the total number of VNFs assigned to the VM_i. $r_{i,j}, PF_{i,j}$, and $R_{i,j}$ are respectively the *CPU life time*, the *total amount of virtual memory allocated* and *allocated disk for the* VNF_j *on* VM_i.

The total load of a slice is defined as below:

$$L = \sum_{i=1}^{N} L_i \tag{5.5}$$

where N is the total number of VMs implemented on this slice

We define the load ratio L_{ocp} as the load from the whole allocated virtual resources available on the slice.

$$L_{ocp} = \frac{L}{R} \tag{5.6}$$

where R is the available resources in the slice.

Based on L_{ocp} we can classify the load into several states (high, moderate, low and too low).

5.2.1.2 Resource usage Prediction

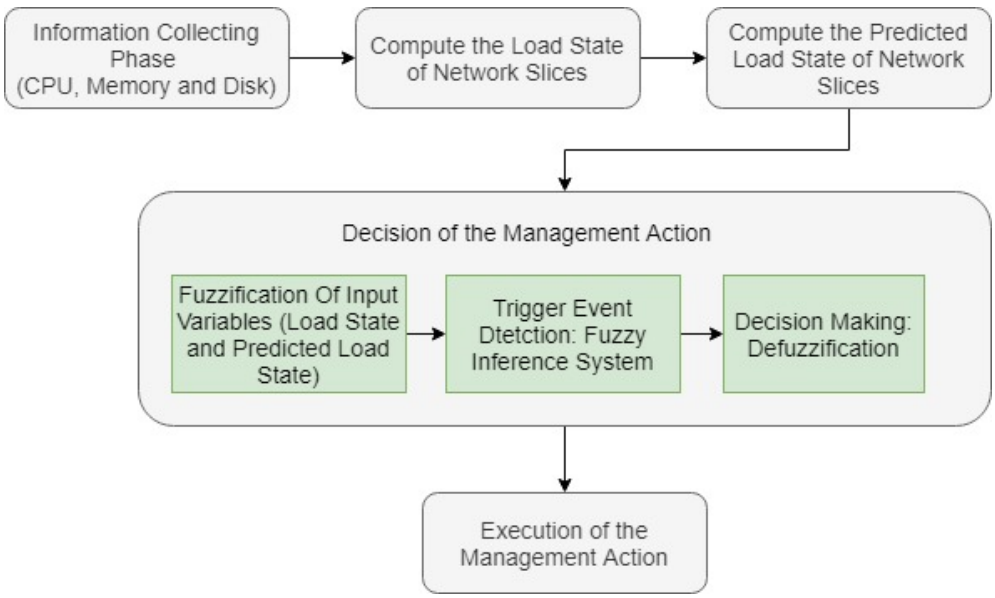


Figure 5.3: *Network slice management algorithm*

Load utilization rates of virtual network resources are characterized by their high

variability. Thus, the computational performances of the network slice are not stationary all the time and have a high variance. This makes hard to satisfy the slice requirements during its execution time. Therefore, the load prediction of network slices resources is fundamental.

Several methods exist for the prediction of future application state such as Queuing Network (QN) models, Linear Regression (LR), ARIMA, etc. In our work, we opt for the *Support Vector Regression (SVR)* prediction model to forecast the stationarity of the slice state [83].

SVR algorithm has presented better *Root Mean Squared Error* (*RMSE*) measure compared to *LR* and *ARIMA* algorithms. In fact, prediction algorithms are evaluated through *Mean Squared Error* (*MSE*) and *RMSE* which are equals to:

$$MSE = \frac{1}{N} * \sum_{1}^{N} (A - P)^2 \tag{5.7}$$

$$RMSE = \sqrt{MSE} \tag{5.8}$$

Where A is the actual value; P is the predicted value and N is the size of the prediction set.

In this work, *Radial Basis Function (RBF)* is used as the kernel function for the SVR model. This model is shown in Eq. 5.9.

$$K(t_i, t_j) = exp(-0.5 \|t_i - t_j\|^2 / \sigma^2) \tag{5.9}$$

$K(t_i, t_j)$ is the kernel function, t_i and t_j are the input vectors and σ is the value of sigma in the Gaussian kernel.

The SVR model is also defined by C parameter, which is the trade-off between the empirical risk and the model flatness and also by ϵ which is the value of epsilon in the insensitive loss function. For our simulation in section 5.2.5, we fixed the value of C at *1e3* and the value of σ to *1e2*.

After the prediction step, we test the stationarity of the future state of the network slice in order to anticipate management actions.

5.2.1.3 Predicted Load-Time Fairness Index

In our work, we will use the *predicted load-Time fairness index* to forecast how dynamic the real network environment is in order to be proactive and anticipate action of migration as it is a time consuming operation that impacts latency.

The predicted vector $\Omega = (\widehat{\omega_1}, \widehat{\omega_2}, ..., \widehat{\omega_m})$ contains the predicted values $\widehat{\omega_l}$ of each load sate ω_l. $\widehat{\omega_l}$ is determined by the following equation:

$$\widehat{\omega_l} = E(\omega_i|X_j) = \sum_{i=1}^{m} \omega_i P(\omega_i|X_j) \tag{5.10}$$

m is the number of load states of the VM. This number has to be not too big in order to not consider distant past and waste memory space to record m values. The best value of m can be the object of future work. $X = (X_1, X_2, \ldots, X_m)$ is the evidence vector containing independent features of VM load fluctuation at different times.

5.2.2 Management Decision Phase

The management decision is based on the load information. It aims to determine the best moment to trigger the management action. For instance, a good initiation of the auto-scaling action reduces the unnecessary auto-scaling actions and also avoids energy waste.

5.2.3 Management Execution Phase

The last phase of the management process is the execution of the decision. Based on the decision received from the *Management Decision Module*, the *controller* will increase or decrease the allocated resources for a slice.

5.2.4 Proposed Algorithm

In our work, we propose the *RMD* algorithm which is based on the fuzzy logic system to perform the management process. It aims to select the adequate strategy for slices management based on input parameters. It takes into account multiple levels of input parameters while deterministic approaches make binary decision. Thus, fuzzy logic system provides intelligence to the system in order to improve problem solving tasks by developing computational methods. Those computational methods define the fuzzy

inference system which is based on the concepts of fuzzy set theory, fuzzy if-then rules, and fuzzy reasoning. This work was carried out using the MATLAB fuzzy logic toolbox. We used the Mamdani fuzzy inference system which includes four functional blocks: the fuzzifier, the set of fuzzy rules, the fuzzy inference engine and the defuzzifier. Figure 5.4 shows the structure of the fuzzy logic controller. The next subsections detail those blocks.

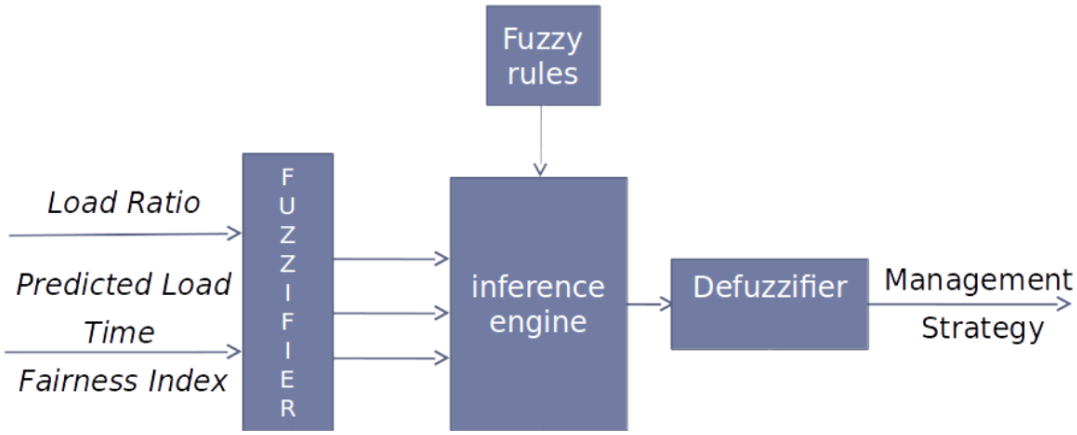


Figure 5.4: *Structure of the fuzzy logic controller*

5.2.4.1 The Fuzzification

The fuzzification step is performed after measuring network criteria values. In fact, the input parameters for the fuzzy logic controller are the information collected during the *Information Gathering phase* (*load ratio* and *predicted load-time fairness index*). It converts the numerical collected information into linguistic variables using the membership functions. In our fuzzy model, we define for each input variable a membership function which has several membership degrees which are:

- Load index (Refer to Fig 5.5): denoted as L_{ocp} and it provides a measure of the load ratio on all VMs of the network slice. We define the fuzzy set of L_{ocp} as the following: Too low, Low, Moderate and High.
- Predicted load-time fairness index (Refer to Fig 5.6): denoted as V_t. We define the fuzzy set of V_t as the following: stationary, variant.

5.2.4.2 The Fuzzy Inference System (FIS)

After the fuzzification of the input values, results are used and evaluated by the FIS. In fact, FIS analyses fuzzy sets according to the inputs variables and fuzzy rules which are

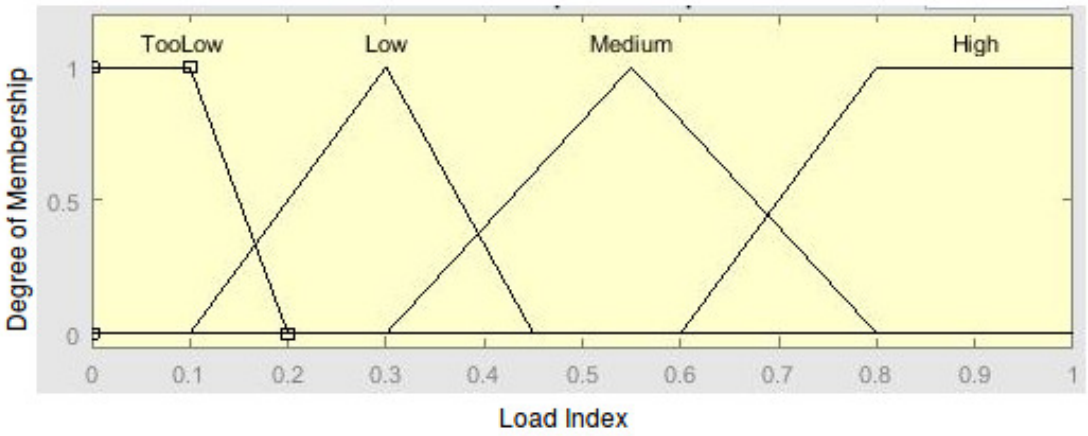


Figure 5.5: *Membership graph for load index*

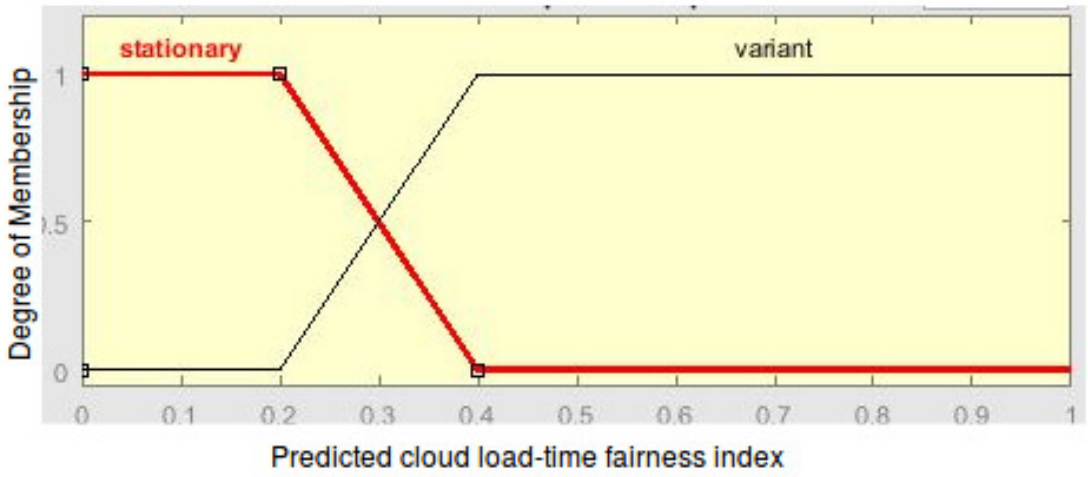


Figure 5.6: *Membership graph for predicted load-time fairness index*

based on a set of *If(condition)-THEN(action)* rules. By applying the fuzzy inference rules, a linguistic decision about the management strategy can be generated based on fuzzy input.

In our system model, there are two input variables which are the *load index* and the *predicted load time fairness index*. The input variable *load index* is composed of four fuzzy sets while *predicted load time fairness index* is composed of two fuzzy sets. Therefore the maximum possible number of rules used to build the rule base is: $N_r = 2 * 4 = 8$. Table 5.1 presents our considered rules.

The decision algorithm implements fuzzy logic in order to select the best strategy of management while considering the characteristics of load. Depending on these characteristics, the algorithm will decide about the strategy of resource management that should be adopted (refer to figure 5.7). We define the fuzzy set strategies as the

following: *resource migration, scaling-down, scaling-up, scaling-out, load balancing* or *no slice modification.*

The resource migration strategy has to determine which VM should be selected for migration and where it should be deployed. In this work, we consider the migration of slice resources from one VM to another either on the same slice or not. The load balancing strategy aims to eliminate hot-spots and improve resource utilization efficiency. In this work, we consider the Round Robin strategy [84]. Energy saving strategy aims to decrease the energy consumption. For the auto-scaling actions we consider the addition (scaling-out), increase (scaling-up) or decrease (scaling-down) of network resources [65].

- When the load ratio is too low and the *predicted load-time fairness index* is stationary, we can migrate slice resources without any impact on QoS in order to minimize energy consumption. In fact, if the energy is not used in an efficient way, it is considered as an energy waste. When the load ratio is low and the *predicted load-time fairness index* is stationary, we will scale down slice resources in order to decrease the computing performances of those resources. We opt for the scaling-down strategy instead of resource migration in order to regain the required QoS and avoid the additional cost of resource migration.

- If the *predicted load-time fairness index* is variable and the load ratio is either low or too low, no management decision will be selected in order to avoid the unnecessary actions of resource migration and scaling down resources.

- If the load ratio is moderate and the predicted load-time fairness index is stationary, the algorithm will not request an execution of a management algorithm on the resources. The slice will keep its actual situation. If the load ratio is moderate and the predicted load-time fairness index is variable, the strategy will first detect the hot-spot nodes of the slice, and then scale up their resources.

- When the load ratio is high, we have to increase slice capacities in order to guarantee the required level of QoS. If the predicted load-time fairness index is stationary, the strategy will be to scale up existing resources on the slice and so to increase the computing resources assigned to VMs. If the predicted load-time fairness index is variable, the strategy will be achieving efficiency by a scaling-out action (adding resources to the slice) and balancing load over machines in order

to decrease energy consumption and minimize execution delay. The scaling out of resources is required in order to assign more resources to VMs which are expected to be hot-spots. Balancing load will ensure a lower *processing delay* on VMs, and lower energy consumption for the physical machines. The *processing delay* is defined as the time it takes VMs to process incoming tasks.

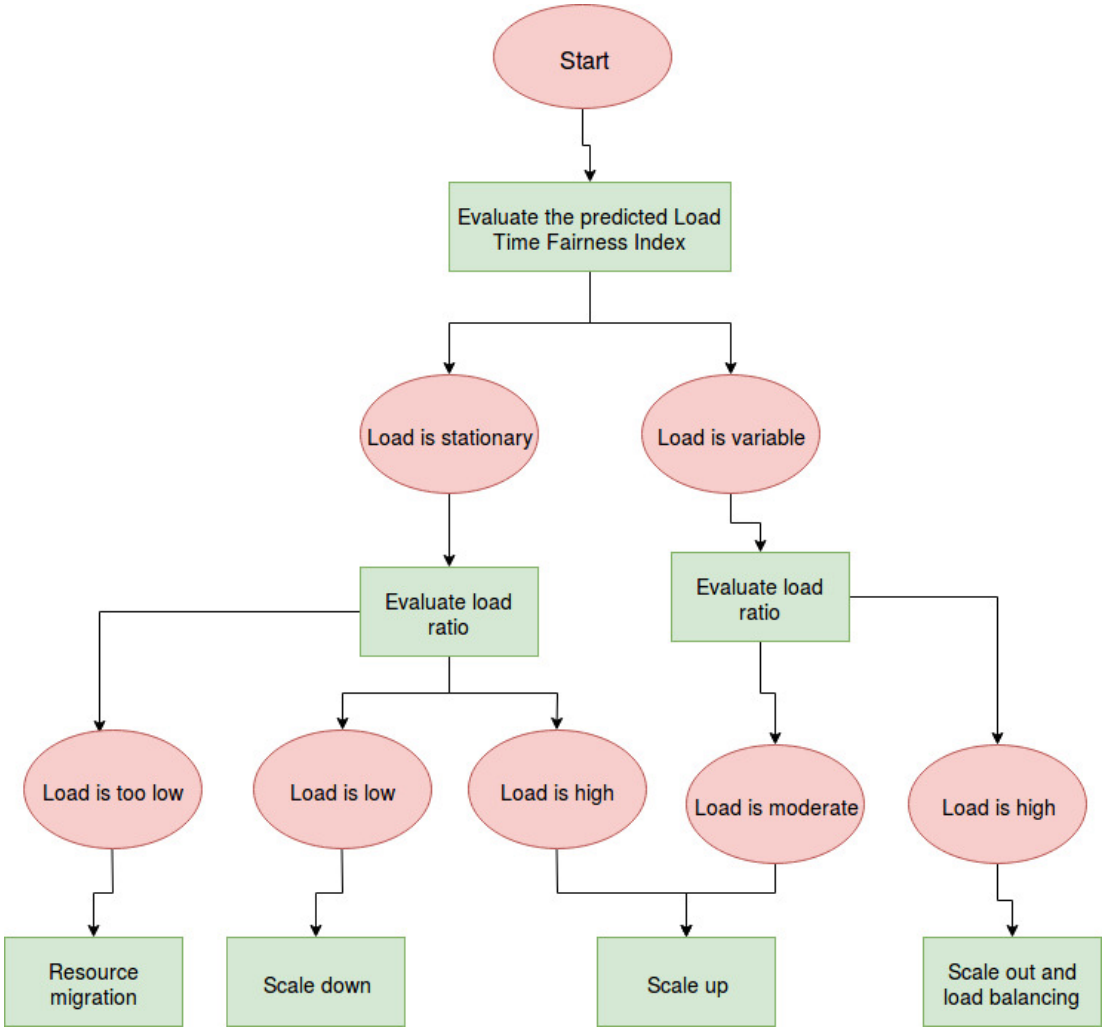


Figure 5.7: *RMD Algorithm*

5.2.4.3 The Defuzzification

The last step is the defuzzification. It makes the final decision of the initiation of the management action. In fact, the defuzzifier converts the decision sets into precise value to determine the management initiation decision. There are many types of defuzzifier. In our system, we used the maximum defuzzifier technique. The output of our algorithm follows a Gaussian shape as in Fig. 5.8.

Table 5.1: *Example of fuzzy decision rules*

Num	Input variables		Management decision
	load ratio	predicted load-time fairness index	
1	Too Low	Stationary	Resource migration
2	Too Low	Variant	No management decision
3	Low	Stationary	Scale down
4	Low	Variant	No management decision
5	Medium	Stationary	No management decision
6	Medium	Variant	Scale up
7	High	Stationary	Scale up
8	High	Variant	scale out and load balancing

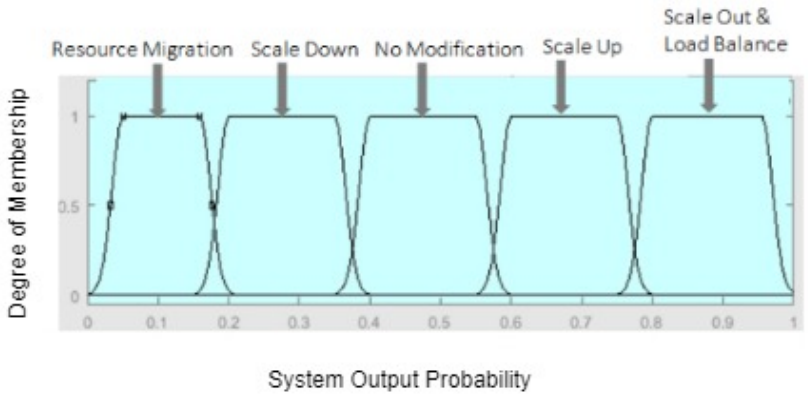


Figure 5.8: *Membership graph of management strategy*

5.2.5 Validation of the Resource Usage Prediction Module

In this section, our objective is to study the performances of the *Resource Usage Prediction* module and the *Dynamic Handler Framework*.

We consider real data-set traces in order to validate the prediction module. In fact, prediction tests have been performed on a trace dataset provided by Alibaba which was released on September 2017 [85]. This data set details the performances of 11089 online service jobs and 12951 batch jobs running on top of 1300 machines during 12 hours. The analysis of this public trace is performed in [86].

Fig. 5.9 presents the results of the prediction module based on the Alibaba dataset for CPU utilization rate [85]. It shows that this prediction allows to forecast the variability of the CPU utilization rate. Despite the fact that the predicted data is not equal to real data, we can predict the variability of the utilization rate.

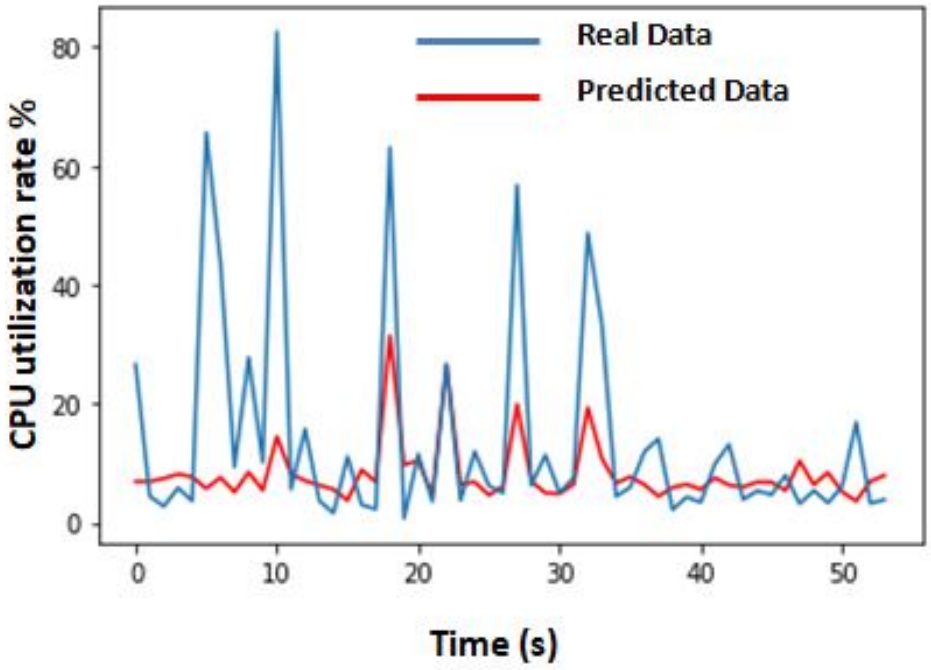


Figure 5.9: *Prediction of the CPU utilization rate of machines during execution time*

After the validation of our prediction module, we consider a network slice scenario characterized by a variable load state as shown in Fig. 5.10. We suppose that the considered slice has the capacity to serve at most 1000 users. We suppose also that, at time t=0, all the VMs of this slice have the same computational capacities.

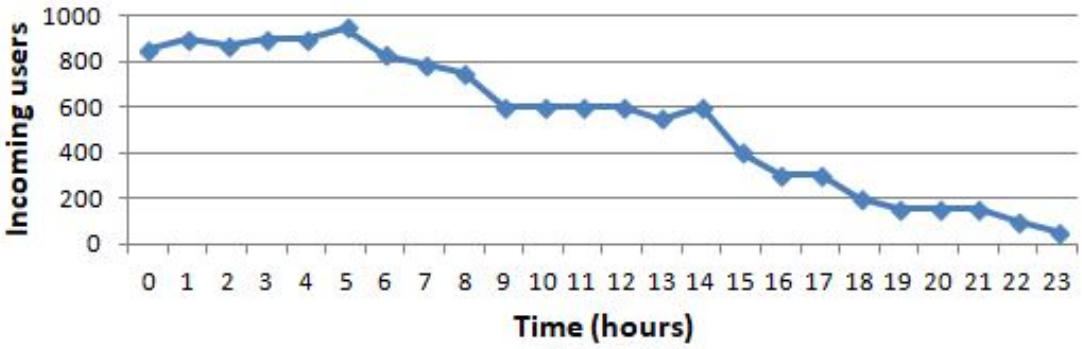


Figure 5.10: *Load state of the considered network slice*

Given resources utilization rate and predicted slice state, the fuzzy decision maker gives the management decision to execute. According to our DHF, several actions can be executed such as scaling-in, scaling-out and scaling down. Fig. 5.11 illustrates the progress in time of the *processing delay* of each slice under different load state. The *processing delay* is defined as the time it takes VMs to process incoming tasks. For the considered slice, at time t=1 the DHF decides to increase the number of allocated

resources in order to satisfy the pic of users at this time. Then, as the load at this slice decreases over the time, it decides to decrease the number of allocated resources and perform a scale-down action. The *processing delay* when our algorithm was used is lower than the generated *delay* when no management decision was implemented. In fact, for each slice, our proposed algorithm will consider the predicted values of resources utilization rates and decides according to the actual and predicted state the best management decision to execute.

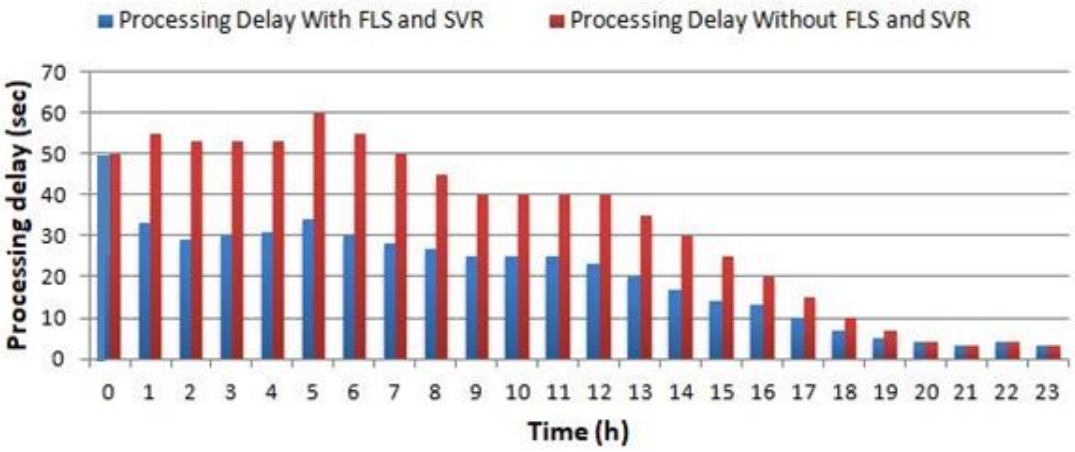


Figure 5.11: *Progress in time of the processing delay*

5.2.6 Validation of the RMD Algorithm

In our simulations, we consider several slices deployed on the corresponding virtual infrastructure. Our objective is to study the performances of the proposed *Resource Management Decision* algorithm (RMD) for each deployed slice. We consider different scenarios under different load state as shown in Fig. 5.12. We suppose that all the considered slices have the same capacities and they can serve at most 1000 users. We suppose also that, at time t=0, all the VMs of each slice have the same computational capacities.

Fig. 5.13(a) and Fig. 5.13(b) show a correlation between the *processing delay* and resource usage of VMs. They show respectively the relationship between the processing delay and the CPU utilization rate as well as between the *processing delay* and the *memory utilization rate*. These templates are derived from historical data for two different VM categories which are Bare metal [87] for VM1 and thin hypervisor (Xen) [88] for VM2. The characteristics of each VM is presented in Table 5.2. The aim of

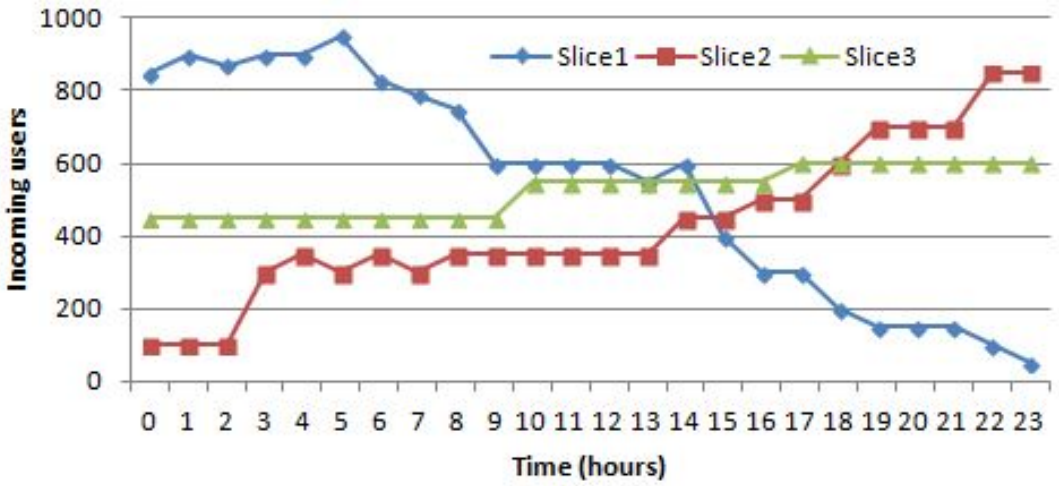


Figure 5.12: *Load state of each network slice*

these templates is to compute the needed resource amount that satisfies the required QoS. Fig. 5.13 demonstrates that the technology of the VM does not really affect the *processing delay*. Thus the *processing delay* depends only on the CPU and memory utilization rate. Adaptiveness and QoS of our DHF module are ensured by considering the templates of the *processing delay* and *resources utilization rate* for each VM of the slice.

Table 5.2: *Virtual machines characteristics*

VM ID	MIPS	Image SIZE (MByte)	RAM (MByte)	BW (kbit/s)
VM1	150	16000	2048	2000
VM2	350	24000	2048	2000

When an auto-scaling action is needed, the algorithm will perform a mapping between the required delay and the minimal required resource utilization rate and then perform an auto-scaling according to those requirements.

Given load state, the fuzzy decision maker gives the approach to adopt. Fig. 5.14, presents the decision that was selected for each slice over the time. Fig. 5.15 illustrates the progress in time of the *processing delay* of each slice under different load state. Slice 3 is characterized by a relatively static traffic load. At times 10 and 17 the DHF detects that the traffic at this slice is moderate and variable so a scaling-up of its resources is performed in order to guarantee the required QoS. For Slice 2 at time 0 the DHF detects that the traffic load of this slice is too low and it is stable so it performs a

scaling-down of this slice resources in order to accomplish an energy saving. When the load increases, the DHF decides a scaling-up action in order to increase the resources capacities. When the traffic is so high the algorithm decides to add some resources to the slice and to perform a load balancing between VMs to be able to increase the slice resources capacity. For Slice 1, at time 5 the RMD decides to increase the number of allocated resources in order to satisfy the pic of users at this time. Then, as the load at this slice decreases over the time, the RMD decides to decrease the number of allocated resources and perform a scale-down action. For all the considered slices, the *processing delay* when our algorithm was used is lower than the generated *delay* when no management decision was implemented. In fact, for each slice, our proposed algorithm will compute the required resource utilization as recommended by the templates.

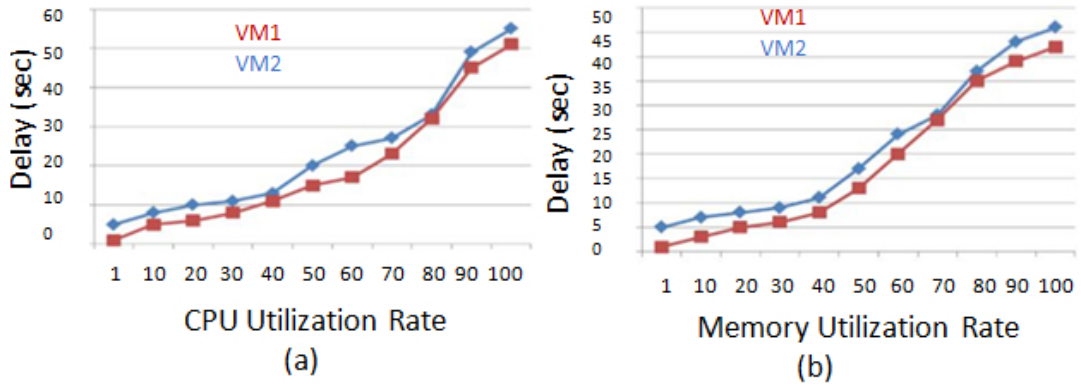


Figure 5.13: *(a) The template of the processing delay-CPU Utilization Rate (b)The template of the processing delay-Memory Utilization Rate*

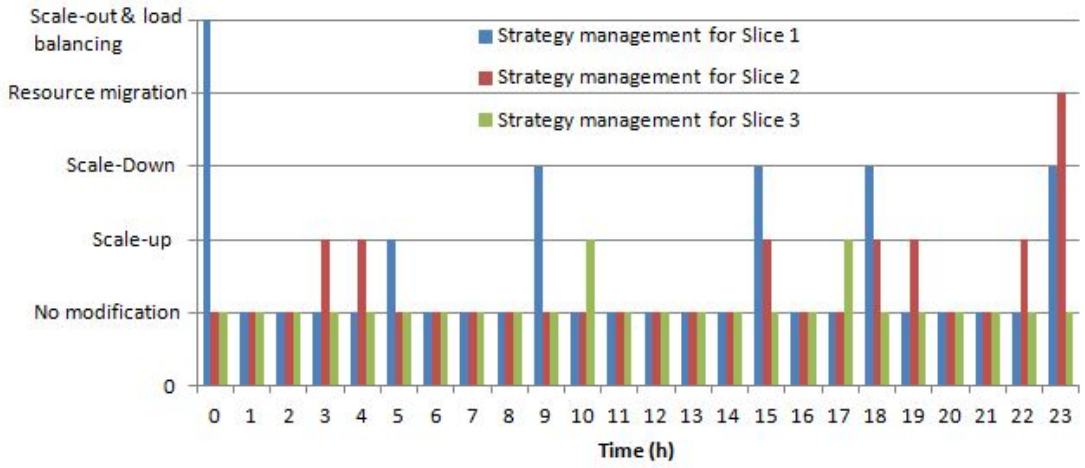


Figure 5.14: *The management decision selected for each slice*

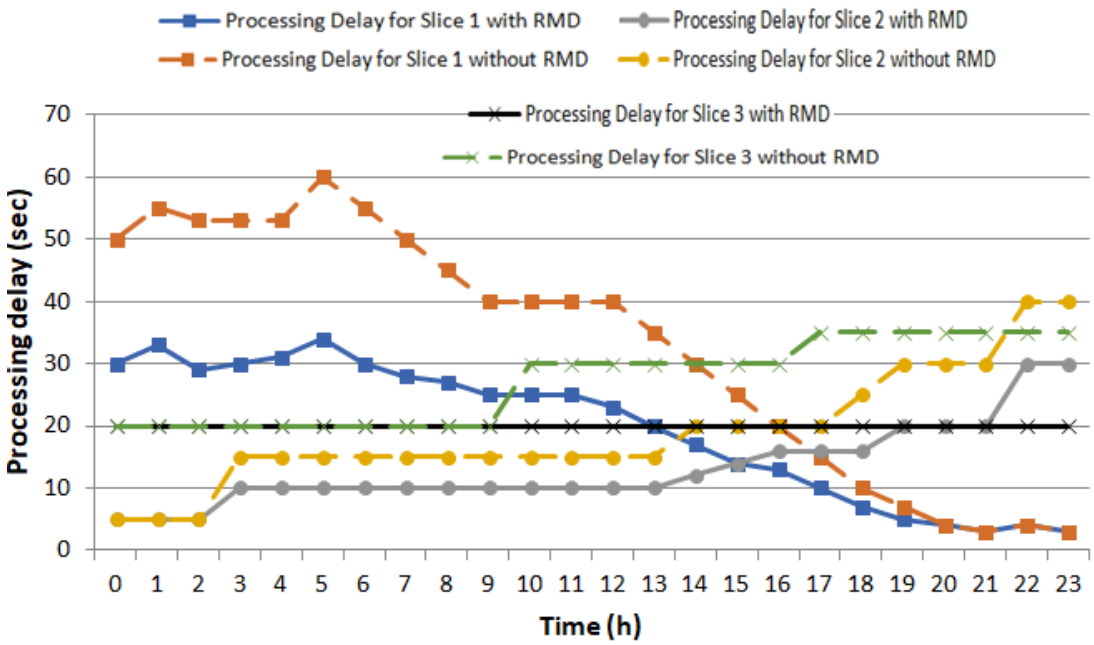


Figure 5.15: *Progress in time of the processing delay for each slice*

5.3 Inter Slice Mobility Management

In this part, we interest on the management of the use case of vehicular network slices. Actually, thanks to *Global Positioning System (GPS)* and *Radio Detection and Ranging (RADAR)*, the cooperative driving and the management of connected vehicles become more feasible. Moreover, connected cameras of vehicles enhance vehicles safety by providing information about road obstacles, conditions, and future moves.

Vehicular Ad-hoc Networks (VANETs) are considered as a promising network design for intelligent transportation systems [73]. They enable a class of applications that requires time-critical responses, very high data rates and a consideration of the very high mobility of vehicles. The architecture of VANETs, as shown in Fig 5.16, is complex, inflexible and characterized by an absence of a centralized control which give rise to challenges in resource management and data scheduling. In this context, the SDN/NFV paradigm presents an opportunity for a more flexible and programmable network [75].

The application of SDN in vehicular networks has recently been the focus of several researches. In [77], evaluation results demonstrate the benefits of SDN in managing a video streaming application over several wireless interfaces relying on V2V and V2I communications. In [78], authors propose a network architecture that enables the slicing function for vehicle network service and improves the bandwidth utilization. In [79],

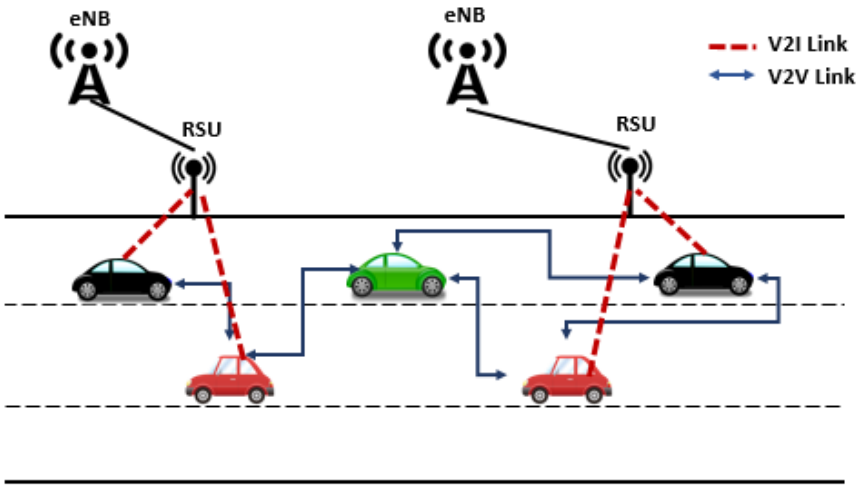


Figure 5.16: *VANET architecture*

authors study the case of cooperative driving among autonomous vehicles. They propose a centralized resource management framework for the computing and storage resources on cloud-computing based on SDN and NFV control modules.

In the literature, some works interest on the mobility management on the context of SDN networks. In [89] authors interest on the application of SDN to aeronautical communications for traffic management purposes. In this context, they study the handover mechanism for video streaming application. In [90], authors propose a vertical handover decision algorithm in SDN context based on *Received Signal Strength (RSS)*, *bandwidth* and *delay*. Their aim is to enhance the delivered QoS to network users. In [91], authors propose an architecture to support Vehicle-to-Everything (V2X) network slices. They study the handover decision with and without slice federation while considering the load state of target slices.

5.3.1 Mobility Management Process

In the context of vehicular networks, mobility management and users' roaming are complicated issues that require inter-slices interactions. The goal is to maintain the required QoS for the user when he moves from its home slice to the visited slice. Our proposed algorithm, presented in Algorithm 1, will decide the need to perform the handover and when it should be initiated whereas vehicles are moving between network slices. The main interactions during handover action are presented in Fig. 5.17.

To perform the handover three phases are required. The first phase is the system

discovery. In this phase, the controllers collect the contextual information about all the available slices. The second phase is the handover decision. This phase determines the best moment to trigger the handover and chooses the most suitable target slice. The last phase is the handover execution. It consists on transferring the current session to the selected slice.

The *Dynamic Handler Framework* continuously monitors the *Status database*. It examines the status of the users' home slice, the planned trajectory and the actual position of the vehicles. This information will be used to determine if a vehicle needs a handover action or not and to which slice the handover will be performed. The *Status database* is updated by the *Infrastructure Controller*, *Slice Controller* and *SDN controller*. In our algorithm we consider the RSS value and load state of the network slices as well as the delivered QoS to the users in order to decide about the initiation of the handover as shown in Algorithm 4.

Whenever the *Dynamic Handler Framework* detects that a handover action should be performed, it notifies the *Slice Controller* which in its turn interacts with the corresponding *Slice Orchestrator*. The *Slice Orchestrator* of the home slice communicates with the *Slice Orchestrator* of the target slice via their slice controllers to exchange the information about the service requirements of the vehicle.

The *Slice Controller* of the target slice will select a slice that corresponds to the user's requirements. If there is a slice that fits the user requirements, the user will be mapped to this slice. Otherwise, the slice that better suits the request is selected to host the user. Finally, the Slice Controller ensures the migration of the user's flows and updates the data base.

5.3.2 Simulations and Results

For our simulations, we use the Mininet-WiFi network emulator [92]. Mininet-WiFi is a fork of Mininet emulator with WiFi features. It allows the evaluation of wireless networks with OpenFlow switches and wireless access points. In our simulation, WiFi slices are considered. Emulated vehicles are multi-interfaces hosts that connect to both radio networks and access points. Ryu Controller was implemented in the system. It is a Python controller that supports OpenFlow 1.3.

As shown in Fig. 5.18, we consider in our simulation three overlapped slices. Slice 1

Algorithm 4 Mobility Management Process

```
1: Controllers send measurements to the Status database
2: The Dynamic Handler Framework analyzes the Status database
3: if The RSS of the home slice < RSS threshold then
4:     Search a target slice
5:     if The system finds a slice that corresponds to the user's requirements then
6:         Migrate the user's session to this slice
7:     else
8:         Search a similar slice from another slice provider domain
9:         if A slice that belongs to another domain is found then
10:            Communicate with the controller of the other domain and migrate the user's
               session to the founded slice
11:        else
12:            Trigger the handover to the slice or the network that offers the closest
               requestet QoS
13:        end if
14:    end if
15:    if Load of the home slice > 80% and QoS delivered to the user decreases then
16:        if RSS target slice > RSS home slice then
17:            Trigger Handover
18:        else
19:            No handover
20:        end if
21:    end if
22: end if
```

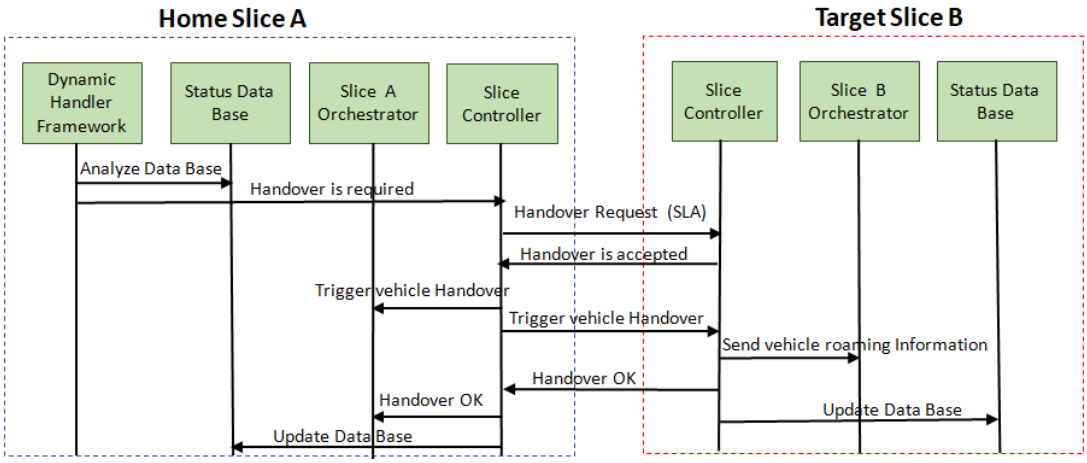


Figure 5.17: *Main interactions of the handover process*

and slice 2 belong to the same slice provider. They are controlled by the same *Slice Controller*. However, the slice 3 belongs to another domain and it is controlled by another *Slice Controller*. At the beginning, the target vehicle is served by the slice 1. When it moves from the zone served by the first slice towards the zone served by the second slice, a handover to slice 2 should be performed. Vehicles are running video streaming application during there mobility between slices. The video streaming traffic was simulated using the Iperf traffic generator. Parameters of our simulation are summarized in Table 5.3. When the user will be served by the second slice, a migration of the routing information should be done by the slice controller [93].

Table 5.3: *Simulation parameters*

Simulator	Mininet-WiFi
Controller	Ryu (OpenFlow 1.3)
Simulation duration	200 s
Number of vehicles at handover time	1, 5, 10
Vehicle Velocity	10 m/s
Node starts moving at	0 s
Traffic starts at	1 s
Traffic Type	Video streaming
Number of slices	3

For our simulations, we evaluate the offered QoS to vehicles when they move from one slice to another for three use cases. In the first use case, we evaluate the performance of the system when only one vehicle is moving between slices at the handover time. For

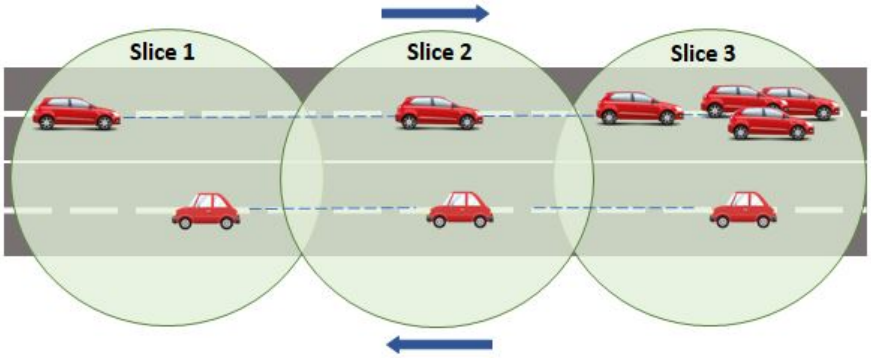
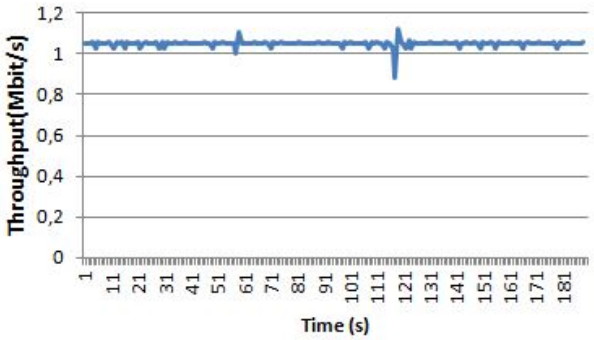


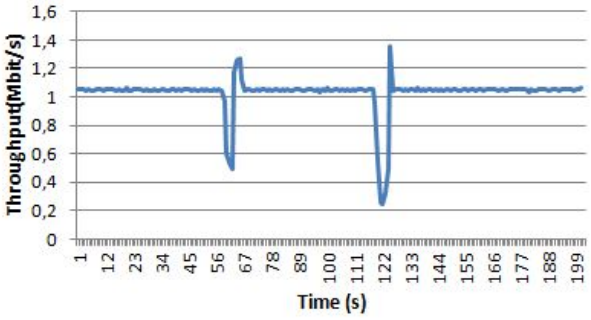
Figure 5.18: *Users' mobility from one slice to another*

the second case, we evaluate these performances for 5 cars and for the last slice we simulate 10 moving cars.

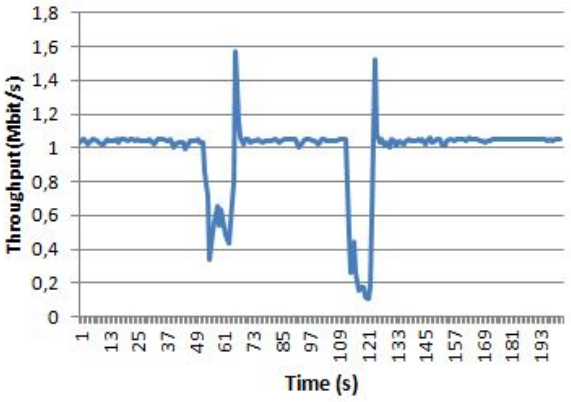
Figures 5.19 and 5.20 present the obtained results in terms of *throughput* and *E2E delay*. We notice that the *throughput* decreases at the handover time when the vehicles move towards the target slice. The degradation of the perceived QoS is more significant when the number of vehicles at the handover time increases. The first handover from slice 1 to slice 2 occurs at time t=60, while the second handover from slice 2 to slice 3 occurs at time t=120. We notice that the performances at the first handover time are better than performances in the second handover. In fact, as the target slice and the home slice are not controlled by the same controller, an additional latency will occur in the system for the handover of vehicles.



(a) *Throughput results for 1 car*

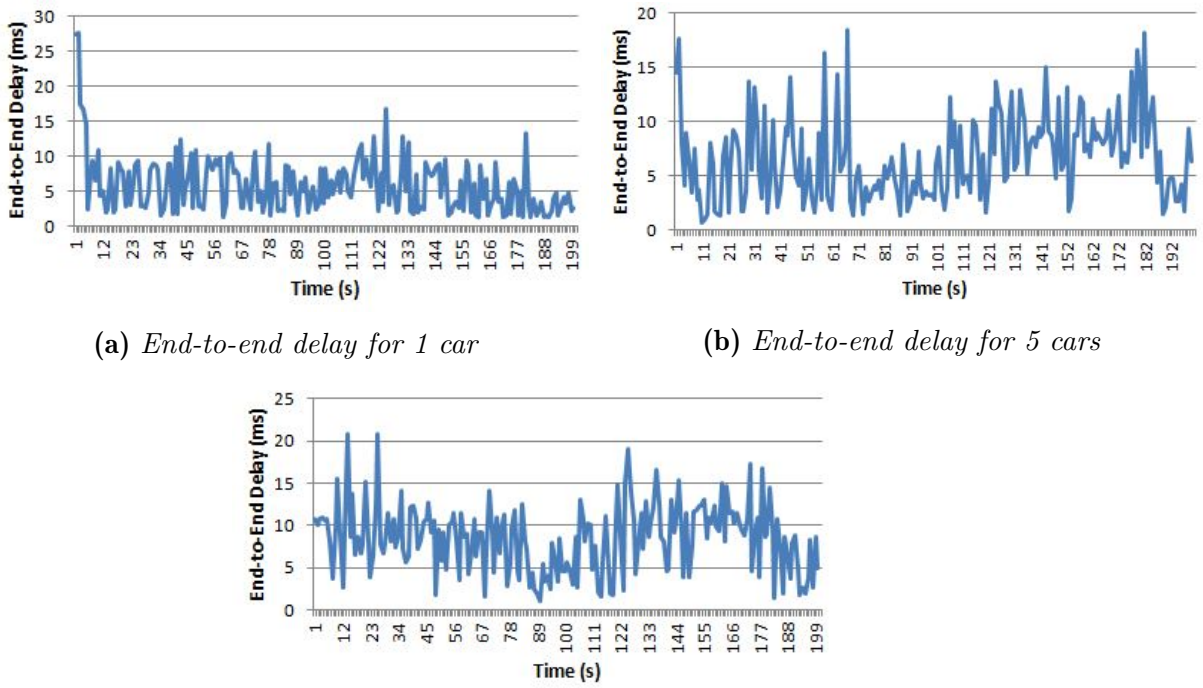


(b) *Throughput results for 5 car*



(c) *Throughput results for 10 cars*

Figure 5.19: *Slice performance when cars are moving from one slice to another in terms of throughput*



(a) *End-to-end delay for 1 car*

(b) *End-to-end delay for 5 cars*

(c) *End-to-end delay for 10 cars*

Figure 5.20: *Slice performance when cars are moving from one slice to another in terms of end-to-end delay*

Conclusion

Slice management is a critical aspect in network slicing and it affects the QoS delivered by the service providers. In this work, we studied two management aspects for network slicing. The first aspect is the management of network slice resources. Our aim of the slice resource management is to maintain the required QoS, increase resource efficiency and reduce unnecessary migration. To solve this problem, we opt for a fuzzy logic system that chooses the management decision depending on the actual and predicted slice conditions. We set several management strategies which are the resource migration, the load balancing and the auto scaling. According to our results, the *Resource Management Decision* algorithm (RMD) adapts the number of CPU allocated to each slice according to its utilization rate and the required QoS. As a second aspect, we studied the mobility management between network slices. We proposed a handover algorithm for the decision of the handover execution and the selection of the target slice. We studied the performances of running services during the handover time when vehicles roam between slices under different load scenarios.

CHAPTER 6

General Conclusion and Perspectives

The emergence of new services with stringent requirements requires a new network architecture which is more flexible and extensible. For this purpose, in this thesis we aimed first to present a new network architecture for services provisioning based on SDN and NFV paradigms. We considered the network slicing technology in order to provide an adequate slice to the requested services. The second goal was to optimize the deployment of network slices over a shared infrastructure while considering the requirements of services in terms of reliability, availability and latency. Finally, we interested on the management of network slices in order to avoid their congestion and react to sudden events. This conclusion provides an overview of our proposed contributions and presents future research that will improve the proposals.

Main Contributions

Our first contribution was the *DANSO* architecture which defines four different levels. The first level is the *Network Store* which maintains a catalog of pre-defined services ready to be deployed for network slices. The second level is the *Control System level* which defines the required modules to deploy and configures slice resources. The third level is the *Slice Service Level* which contains the set of deployed slices and manages the slices services and their life-cycle. Finally, the *Virtual Infrastructure level* which constitutes the *DANSO* capacity for deployments (i.e. an abstracted view of resources).

DANSO proposes also a set of managers, controllers and orchestrators that are associated to different levels of the architecture.

As a second contribution, we designed a new algorithm which optimizes the network slice deployment. In fact, when the system receives the users' demands, it has to either map this request to an existing slice or to create a new slice. In this regard, we proposed at first a heuristic algorithm for the admission control of users' requests. This algorithm will search a corresponding slice that can serve these demands. If the demand cannot be served by a deployed slice, we propose a second algorithm that will create a new network slice while considering the required QoS and also the offered quality by the underlying infrastructure. In this contribution, we focused essentially on the reliability, availability and latency requirements.

The third contribution was related to the management aspect. In fact, when the slice is running, several sudden events may occur. In our work we studied the congestion of the slice and also the users' mobility events. We proposed a fuzzy logic based algorithm in order to detect the need to perform a management action when the slice will be overloaded. This decision is based on the actual load state of the slice as well as the predicted value of the load. We considered the SVR technique in order to perform the prediction of the future load values. In this case, we considered several auto-scaling action in order to overcome the congestion issue. Finally, we studied the problem of vertical handover management in the context of network slices. We proposed an algorithm that decides when to perform the handover and which slice will be selected as the target slice.

Future Work

Numerous perspectives could be considered in order to extend our presented work. In fact, the network slices creation and management problems will continue to face research improvement. Network slicing has been a very big opportunity for several vendors and service providers. In this regard, we aim to prototype our proposed architecture in order to test our proposed algorithms in real environment. We aim also to consider a specific 5G use case, such as the IoT, in order to highlight quantitatively the efficiency of our proposal compared to legacy solutions.

For our optimization algorithm, we will try to better investigate the network perfor-

mances by handling the interference between the several deployed slices and consider inter and intra-slices priorities.

For our management algorithm, we will study the trade-off between introducing other input variables to our fuzzy system and the cost that will be induced in terms of memory capacity and processing delay. Moreover, we aim to study the utilization rate of network resources when the RMD algorithm is applied compared to other algorithms. We manage also to improve the considered SVR parameters in order to have better predictions. For our handover algorithm, we plan to study the scalability issue and evaluate the handover performances with ONOS and OpenDayLight controllers. Furthermore, we aim to consider other handover criteria like the energetic cost and the vehicle velocity for a better handover decision.

List of Publications

1. A. Kammoun, N. Tabbane, G. Diaz, N. Achir and A. Dandoush, " Inter-Slice Mobility Management in the Context of SDN/NFV Networkst," 2019 Distributed Computing for Emerging Smart Networks (DiCES-N), Hammamet, 2019.

2. A. Kammoun, N. Tabbane, G. Diaz, N. Achir and A. Dandoush, "Dynamic Handler Framework for Network Slices Management," 2019 27th International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split, 2019.

3. A. Kammoun, N. Tabbane, G. Diaz, N. Achir and A. Dandoush, " Proactive Network Slices Management Algorithm Based on Fuzzy Logic System and Support Vector Regression Model," 2019 14th International Conference on Broad-Band Wireless Computing, Communication and Applications (BWCCA), Antwerp, 2019.

4. A. Kammoun, N. Tabbane, G. Diaz, A. Dandoush and N. Achir, "End-to-End Efficient Heuristic Algorithm for 5G Network Slicing," 2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA), Krakow, 2018, pp. 386-392.

5. A. Kammoun and N. Tabbane, "Fuzzy utility decisional vertical handover algorithm for enhancing network performances," 2016 5th International Conference on Multimedia Computing and Systems (ICMCS), Marrakech, 2016, pp. 337-343.

6. A. Kammoun, N. Tabbane, G. Diaz and N. Achir, "Admission Control Algorithm for Network Slicing Management in SDN-NFV Environment," 2018 6th International Conference on Multimedia Computing and Systems (ICMCS), Rabat, 2018, pp. 1-6.

Bibliography

[1] M. R. Sama et al. “Reshaping the mobile core network via function decomposition and network slicing for the 5G Era”. In: *2016 IEEE Wireless Communications and Networking Conference*. Apr. 2016, pp. 1–7. DOI: 10.1109/WCNC.2016.7564652 *Cited on page 1.*

[2] M. Jiang, M. Condoluci, and T. Mahmoodi. “Network slicing management prioritization in 5G mobile systems”. In: *European Wireless 2016; 22th European Wireless Conference*. May 2016, pp. 1–6 *Cited on pages 4, 25.*

[3] *Network Functions Virtualisation (NFV); Ecosystem; Report on SDN Usage in NFV Architectural Framework*. v. 1.1.1. ETSI GS NFV-EVE 005. Dec. 2015 *Cited on pages 5, 14, 15, 42.*

[4] Ericsson. *ERICSSON WHITE PAPER, 5G SYSTEMS*. https://www.ericsson.com/assets/local/publications/white-papers/wp-5g-systems.pdf. Accessed: 03-04-2018. 2017 *Cited on pages 10, 38.*

[5] *NGMN 5G White Paper*. v. 1.0. NGMN Alliance. 2015 *Cited on pages 10, 17, 38.*

[6] *feasibility study on new services and markets technology enablers, stage 1, rel-14, V14.2.0*. 3GPP TR 22.891. 2016 *Cited on pages 10, 17, 22, 38.*

[7] *Network Functions Virtualisation (NFV); Resiliency Requirements*. V1.1.1. ETSI GS NFV-REL 001. 2015 *Cited on pages 10, 11, 26, 27.*

[8] R. Mijumbi et al. “Network Function Virtualization: State-of-the-Art and Research Challenges”. In: *IEEE Communications Surveys Tutorials* 18.1 (Firstquarter 2016), pp. 236–262. ISSN: 1553-877X. DOI: 10.1109/COMST.2015.2477041 *Cited on pages 11, 12.*

[9] *network functions virtualisation (nfv); virtual network functions architecture*. V1.1.1. ETSI Industry Specification Group (ISG) NFV. Etsi gs nfv-swa 001. Dec. 2014 *Cited on page 12.*

[10] *Network Functions Virtualisation (NFV); Architectural Framework*. v. 1.1.1. ETSI GS NFV-EVE 002. Oct. 2013 *Cited on page 12.*

[11] D. Kreutz et al. “Software-Defined Networking: A Comprehensive Survey”. In: *Proceedings of the IEEE* 103.1 (Jan. 2015), pp. 14–76. ISSN: 0018-9219. DOI: 10.1109/JPROC.2014.2371999 *Cited on pages 12, 13.*

[12] *Open Network Foundation: SDN architecture*. Tech. rep. June 2014 *Cited on page 12.*

[13] *Network Functions Virtualisation (NFV); Management and Orchestration*. v. 1.1.1. GS NFV-MAN 001. Dec. 2014 *Cited on pages 14, 29.*

[14] *Open Source MANO*. https://osm.etsi.org/. Accessed: 2019-03-06 *Cited on page 15.*

[15] *Open Baton*. http://openbaton.github.io/. Accessed: 2019-03-06 *Cited on pages 16, 30.*

[16] *ONAP*. https://www.onap.org/. Accessed: 2018-03-28 *Cited on page 16.*

[17] Sahel Sahhaf et al. “Network service chaining with optimized network function embedding supporting service decompositions”. In: *Computer Networks* 93 (2015). Cloud Networking and Communications II, pp. 492–505. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2015.09.035 *Cited on pages 16, 25.*

[18] J. Gil Herrera and J. F. Botero. “Resource Allocation in NFV: A Comprehensive Survey”. In: *IEEE Transactions on Network and Service Management* 13.3 (Sept. 2016), pp. 518–532. ISSN: 1932-4537. DOI: 10.1109/TNSM.2016.2598420 *Cited on page 16.*

[19] *P. Mell and T. Grance. Nist special publication 800-145 : The nist definition of cloud computing*. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf , 2011. Accessed: 2019-06-20 *Cited on pages 16, 29.*

[20] Z. Chang et al. “Towards Service-Oriented 5G: Virtualizing the Networks for Everything-as-a-Service”. In: *IEEE Access* 6 (2018), pp. 1480–1489. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2779170 *Cited on page 16.*

[21] T. Cooklev et al. “Enabling RF data analytics services and applications via cloudification”. In: *IEEE Aerospace and Electronic Systems Magazine* 33.5-6 (May 2018), pp. 44–55. ISSN: 0885-8985. DOI: 10.1109/MAES.2018.170108 *Cited on page 16.*

[22] ONF. https://www.opennetworking.org/ *Cited on page 17.*

[23] *Next Generation Protocols(NGP);E2E Network Slicing Reference Framework and Information Model*. v. 1.1.1. ETSI GR NGP 011. Sept. 2018 *Cited on page 17.*

[24] Navid Nikaein et al. “Network Store: Exploring Slicing in Future 5G Networks”. In: *Proceedings of the 10th International Workshop on Mobility in the Evolving Internet Architecture*. MobiArch ’15. Paris, France: ACM, 2015, pp. 8–13. ISBN: 978-1-4503-3695-6. DOI: 10.1145/2795381.2795390 *Cited on page 18.*

[25] A. Devlic et al. “NESMO: Network slicing management and orchestration framework”. In: *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*. May 2017, pp. 1202–1208. DOI: 10.1109/ICCW.2017.7962822 *Cited on page 19.*

[26] Y. Li et al. “A unified control and optimization framework for dynamical service chaining in software-defined NFV system”. In: *IEEE Wireless Communications* 22.6 (Dec. 2015), pp. 15–23. ISSN: 1536-1284. DOI: 10.1109/MWC.2015.7368820 *Cited on page 19.*

[27] 5gppp. https://5g-ppp.eu/ *Cited on page 19.*

[28] S. Dräxler et al. “SONATA: Service programming and orchestration for virtualized software networks”. In: *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*. May 2017, pp. 973–978. DOI: 10.1109/ICCW.2017.7962785 *Cited on page 19.*

[29] Carlos J. Bernardos et al. “5GEx: realising a Europe-wide multi-domain framework for software-defined infrastructures”. In: *Transactions on Emerging Telecommunications Technologies* 27.9 (2016), pp. 1271–1280. DOI: 10.1002/ett.3085. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/ett.3085 *Cited on page 19.*

[30] F. S. D. Silva et al. “NECOS Project: Towards Lightweight Slicing of Cloud Federated Infrastructures”. In: *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*. June 2018, pp. 406–414. DOI: 10.1109/NETSOFT.2018.8460008 *Cited on page 19.*

[31] A. de la Oliva et al. “5G-TRANSFORMER: Slicing and Orchestrating Transport Networks for Industry Verticals”. In: *IEEE Communications Magazine* 56.8 (Aug. 2018), pp. 78–84. ISSN: 0163-6804. DOI: 10.1109/MCOM.2018.1700990 *Cited on page 20.*

[32] P. Rost et al. “Network Slicing to Enable Scalability and Flexibility in 5G Mobile Networks”. In: *IEEE Communications Magazine* 55.5 (May 2017), pp. 72–79. ISSN: 0163-6804. DOI: 10.1109/MCOM.2017.1600920 *Cited on page 20.*

[33] Q. Wang, J. Alcaraz-Calero, and al. “SliceNet: End-to-End Cognitive Network Slicing and Slice Management Framework in Virtualised Multi-Domain, Multi-Tenant 5G Networks”. In: *2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. June 2018, pp. 1–5. DOI: 10.1109/BMSB.2018.8436800 *Cited on page 20.*

[34] Christian Jacquenet. “Service Parameter Negotiation in the 5G Era”. In: *Keynote in ICIN 2018,* *Cited on page 21.*

[35] J. Ordonez-Lucena et al. “Network Slicing for 5G with SDN/NFV: Concepts, Architectures, and Challenges”. In: *IEEE Communications Magazine* 55.5 (May 2017), pp. 80–87. ISSN: 0163-6804. DOI: 10.1109/MCOM.2017.1600935 *Cited on page 21.*

[36] J. Li et al. “Online Joint VNF Chain Composition and Embedding for 5G Networks”. In: *2018 IEEE Global Communications Conference (GLOBECOM)*. Dec. 2018, pp. 1–6. DOI: 10.1109/GLOCOM.2018.8647700 *Cited on page 23.*

[37] P. T. A. Quang et al. “Single and Multi-Domain Adaptive Allocation Algorithms for VNF Forwarding Graph Embedding”. In: *IEEE Transactions on Network and Service Management* 16.1 (Mar. 2019), pp. 98–112. ISSN: 1932-4537. DOI: 10.1109/TNSM.2018.2876623 *Cited on page 23.*

[38] H. Cao, H. Zhu, and L. Yang. “Dynamic Embedding and Scheduling of Service Function Chains for Future SDN/NFV-Enabled Networks”. In: *IEEE Access* 7 (2019), pp. 39721–39730. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2906874 *Cited on page 23.*

[39] Edoardo Amaldi et al. “On the computational complexity of the virtual network embedding problem”. In: *Electronic Notes in Discrete Mathematics* 52 (2016). INOC 2015 – 7th International Network Optimization Conference, pp. 213–220. ISSN: 1571-0653. DOI: https://doi.org/10.1016/j.endm.2016.03.028 *Cited on page 24.*

[40] S. Agarwal et al. “Joint VNF Placement and CPU Allocation in 5G”. In: *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*. Apr. 2018, pp. 1943–1951. DOI: 10.1109/INFOCOM.2018.8485943 *Cited on page 24.*

[41] A. H. M. Jakaria and M. A. Rahman. “A Formal Framework of Resource Management for VNFaaS in Cloud”. In: *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*. June 2017, pp. 254–261. DOI: 10.1109/CLOUD.2017.40 *Cited on page 24.*

[42] J. Chase et al. “Joint virtual machine and bandwidth allocation in software defined network (SDN) and cloud computing environments”. In: *2014 IEEE International Conference on Communications (ICC)*. June 2014, pp. 2969–2974. DOI: 10.1109/ICC.2014.6883776 *Cited on page 24.*

[43] S. Parsaeefard et al. “Joint resource provisioning and admission control in wireless virtualized networks”. In: *2015 IEEE Wireless Communications and Networking Conference (WCNC)*. Mar. 2015, pp. 2020–2025. DOI: 10.1109/WCNC.2015.7127778 *Cited on page 25.*

[44] N. Zhang et al. “Network Slicing for Service-Oriented Networks Under Resource Constraints”. In: *IEEE Journal on Selected Areas in Communications* 35.11 (Nov. 2017), pp. 2512–2521. ISSN: 0733-8716. DOI: 10.1109/JSAC.2017.2760147 *Cited on page 25.*

[45] Xuemin Wen et al. “Towards reliable virtual data center embedding in software defined networking”. In: *MILCOM 2016 - 2016 IEEE Military Communications Conference*. Nov. 2016, pp. 1059–1064. DOI: 10.1109/MILCOM.2016.7795470 *Cited on page 25.*

[46] Oussama Soualah et al. “Online and batch algorithms for VNFs placement and chaining”. In: *Computer Networks* 158 (2019), pp. 98–113. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2019.01.041 *Cited on page 25.*

[47] M.M. Tajiki et al. “Software defined service function chaining with failure consideration for fog computing”. In: *Concurrency and Computation: Practice and Experience* 31.8 (2019). e4953 cpe.4953, e4953. DOI: 10.1002/cpe.4953. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.4953 *Cited on page 27.*

[48] Mohammad Mahdi Tajiki et al. “Joint Failure Recovery, Fault Prevention, and Energy-efficient Resource Management for Real-time SFC in Fog-supported SDN”. In: *CoRR* abs/1807.00324 (2018). arXiv: 1807.00324 *Cited on page 27.*

[49] T. Ahammad, U. Kumar Acharjee, and M. M. Hasan. “Energy-Effective Service-Oriented Cloud Resource Allocation Model Based on Workload Prediction”. In: *2018 21st International Conference of Computer and Information Technology (ICCIT)*. Dec. 2018, pp. 1–6. DOI: 10.1109/ICCITECHN.2018.8631953 *Cited on page 27.*

[50] Wei Zhong et al. “The Cloud Computing Load Forecasting Algorithm Based on Wavelet Support Vector Machine”. In: *Proceedings of the Australasian Computer Science Week Multiconference*. ACSW ’17. Geelong, Australia: ACM, 2017, 38:1–38:5. ISBN: 978-1-4503-4768-6. DOI: 10.1145/3014812.3014852 *Cited on page 27.*

[51] Deepak Nadig et al. “Large Data Transfer Predictability and Forecasting using Application-Aware SDN”. In: Dec. 2018, pp. 1–6. DOI: 10.1109/ANTS.2018.8710165 *Cited on page 27.*

[52] Manuel Pérez et al. “Self-Organizing Capabilities in 5G Networks: NFV SDN Coordination in a Complex Use Case”. In: June 2018 *Cited on page 27.*

[53] B. Tola et al. “Modeling and Evaluating NFV-Enabled Network Services under Different Availability Modes”. In: *2019 15th International Conference on the Design of Reliable Communication Networks (DRCN)*. Mar. 2019, pp. 1–5. DOI: 10.1109/DRCN.2019.8713765 *Cited on page 27.*

[54] Syed Riffat Ali. “Network Function Virtualization”. In: *Next Generation and Advanced Network Reliability Analysis: Using Markov Models and Software Reliability Engineering*. Cham: Springer International Publishing, 2019, pp. 131–156. ISBN: 978-3-030-01647-0. DOI: 10.1007/978-3-030-01647-0_5 *Cited on page 27.*

[55] M. Miyazawa, M. Hayashi, and R. Stadler. "vNMF: Distributed fault detection using clustering approach for network function virtualization". In: *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. May 2015, pp. 640–645. DOI: 10.1109/INM.2015.7140349 *Cited on page 28.*

[56] T. Niwa et al. "Universal fault detection for NFV using SOM-based clustering". In: *2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. Aug. 2015, pp. 315–320. DOI: 10.1109/APNOMS.2015.7275446 *Cited on page 28.*

[57] U. S. Hashmi, A. Darbandi, and A. Imran. "Enabling proactive self-healing by data mining network failure logs". In: *2017 International Conference on Computing, Networking and Communications (ICNC)*. Jan. 2017, pp. 511–517. DOI: 10.1109/ICCNC.2017.7876181 *Cited on page 28.*

[58] R. C. Turchetti and E. P. Duarte. "Implementation of Failure Detector Based on Network Function Virtualization". In: *2015 IEEE International Conference on Dependable Systems and Networks Workshops*. June 2015, pp. 19–25. DOI: 10.1109/DSN-W.2015.30 *Cited on page 28.*

[59] H. Farooq, M. S. Parwez, and A. Imran. "Continuous Time Markov Chain Based Reliability Analysis for Future Cellular Networks". In: *2015 IEEE Global Communications Conference (GLOBECOM)*. Dec. 2015, pp. 1–6. DOI: 10.1109/GLOCOM.2015.7417594 *Cited on page 28.*

[60] Q. Liao and S. Stanczak. "Network State Awareness and Proactive Anomaly Detection in Self-Organizing Networks". In: *2015 IEEE Globecom Workshops (GC Wkshps)*. Dec. 2015, pp. 1–6. DOI: 10.1109/GLOCOMW.2015.7414141 *Cited on page 28.*

[61] R. Cziva et al. "SDN-Based Virtual Machine Management for Cloud Data Centers". In: *IEEE Transactions on Network and Service Management* 13.2 (June 2016), pp. 212–225. ISSN: 1932-4537. DOI: 10.1109/TNSM.2016.2528220 *Cited on page 29.*

[62] L. Cui et al. "PLAN: Joint Policy- and Network-Aware VM Management for Cloud Data Centers". In: *IEEE Transactions on Parallel and Distributed Systems* 28.4 (Apr. 2017), pp. 1163–1175. ISSN: 1045-9219. DOI: 10.1109/TPDS.2016.2604811 *Cited on page 29.*

[63] S. Ejaz et al. "Traffic Load Balancing Using Software Defined Networking (SDN) Controller as Virtualized Network Function". In: *IEEE Access* 7 (2019), pp. 46646–46658. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2909356 *Cited on page 29.*

[64] Jiefei Ma et al. "A Comprehensive Study on Load Balancers for VNF chains Horizontal Scaling". In: *CoRR* abs/1810.03238 (2018). arXiv: 1810.03238 *Cited on page 29.*

[65] G. A. Carella et al. “An extensible Autoscaling Engine (AE) for Software-based Network Functions”. In: *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. Nov. 2016, pp. 219–225. DOI: 10.1109/NFV-SDN.2016.7919501 *Cited on pages 29, 83.*

[66] F. Tseng et al. “A Lightweight Autoscaling Mechanism for Fog Computing in Industrial Applications”. In: *IEEE Transactions on Industrial Informatics* 14.10 (Oct. 2018), pp. 4529–4537. ISSN: 1551-3203. DOI: 10.1109/TII.2018.2799230 *Cited on page 29.*

[67] Hamid Arabnejad et al. “A Comparison of Reinforcement Learning Techniques for Fuzzy Cloud Auto-Scaling”. In: *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. CCGrid ’17. Madrid, Spain: IEEE Press, 2017, pp. 64–73. ISBN: 978-1-5090-6610-0. DOI: 10.1109/CCGRID.2017.15 *Cited on page 30.*

[68] *Auto-scaling actions*. https://medium.com/faun/scaling-applications-in-the-cloud-52bb6dfbac4e. Accessed: 2019-09-4 *Cited on page 30.*

[69] G. Xilouris et al. “T-NOVA: A marketplace for virtualized network functions”. In: *2014 European Conference on Networks and Communications (EuCNC)*. June 2014, pp. 1–5. DOI: 10.1109/EuCNC.2014.6882687 *Cited on page 37.*

[70] Ines Ayadi, Gladys Diaz, and Noëmie Simoni. “QoS-based network virtualization to future networks: An approach based on network constraints”. In: *IEEE Network of the Future (NOF), 2013 Fourth International Conference on the* 9.4 (Oct. 2013), pp. 1–5 *Cited on page 43.*

[71] A. Kammoun et al. “End-to-End Efficient Heuristic Algorithm for 5G Network Slicing”. In: *2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)*. May 2018, pp. 386–392. DOI: 10.1109/AINA.2018.00065 *Cited on page 43.*

[72] M.R.M. Assis and L.F. Bittencourt. “A survey on cloud federation architectures: Identifying functional and non-functional properties”. In: *Journal of Network and Computer Applications* 72 (2016), pp. 51–71. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2016.06.014 *Cited on page 45.*

[73] Felipe Cunha et al. “Data communication in VANETs: Protocols, applications and challenges”. In: *Ad Hoc Networks* 44 (2016), pp. 90–103. ISSN: 1570-8705. DOI: https://doi.org/10.1016/j.adhoc.2016.02.017 *Cited on pages 48, 90.*

[74] Manisha Chahal et al. “A Survey on software-defined networking in vehicular ad hoc networks: Challenges, applications and use cases”. In: *Sustainable Cities and Society* 35 (2017), pp. 830–840. ISSN: 2210-6707. DOI: https://doi.org/10.1016/j.scs.2017.07.007 *Cited on page 48.*

[75] S. A. A. Shah et al. “5G for Vehicular Communications”. In: *IEEE Communications Magazine* 56.1 (Jan. 2018), pp. 111–117. ISSN: 0163-6804. DOI: 10.1109/MCOM.2018.1700467 *Cited on pages 48, 90.*

[76] I. Yaqoob et al. “Overcoming the Key Challenges to Establishing Vehicular Communication: Is SDN the Answer?” In: *IEEE Communications Magazine* 55.7 (July 2017), pp. 128–134. ISSN: 0163-6804. DOI: 10.1109/MCOM.2017.1601183 *Cited on page 48.*

[77] R. Dos Reis Fontes et al. “From Theory to Experimental Evaluation: Resource Management in Software-Defined Vehicular Networks”. In: *IEEE Access* 5 (2017), pp. 3069–3076. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2671030 *Cited on pages 48, 90.*

[78] Rentao Gu et al. “Network slicing and efficient ONU migration for reliable communications in converged vehicular and fixed access network”. In: *Vehicular Communications* 11 (2018), pp. 57–67. ISSN: 2214-2096. DOI: https://doi.org/10.1016/j.vehcom.2018.01.003 *Cited on pages 49, 90.*

[79] Hai-xia Peng, Qiang Ye, and Xuemin Shen. “SDN-Based Resource Management for Autonomous Vehicular Networks: A Multi-Access Edge Computing Approach”. In: *CoRR* abs/1809.08966 (2018). arXiv: 1809.08966 *Cited on pages 49, 90.*

[80] L. Suciu and K. Guillouard. “A Hierarchical and Distributed Handover Management Approach for Heterogeneous Networking Environments”. In: *International Conference on Networking and Services (ICNS ’07)*. June 2007, pp. 77–77. DOI: 10.1109/ICNS.2007.9 *Cited on page 60.*

[81] *Zoo Project*. http://www.topology-zoo.org/. Accessed: 2019-09-05 *Cited on page 63.*

[82] S. Dotcenko, A. Vladyko, and I. Letenko. “A fuzzy logic-based information security management for software-defined networks”. In: *16th International Conference on Advanced Communication Technology*. Feb. 2014, pp. 167–171. DOI: 10.1109/ICACT.2014.6778942 *Cited on page 74.*

[83] Corinna Cortes and Vladimir Vapnik. “Support-vector networks”. In: *Machine Learning* 20.3 (Sept. 1995), pp. 273–297. ISSN: 1573-0565. DOI: 10.1007/BF00994018 *Cited on page 79.*

[84] Rasmus V. Rasmussen and Michael A. Trick. “Round robin scheduling – a survey”. In: *European Journal of Operational Research* 188.3 (2008), pp. 617–636. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2007.05.046 *Cited on page 83.*

[85] *Alibaba Cluster Trace Program*. https://github.com/alibaba/clusterdata. Accessed: 2019-07-31 *Cited on page 85.*

[86] C. Lu et al. “Imbalance in the cloud: An analysis on Alibaba cluster trace”. In: *2017 IEEE International Conference on Big Data (Big Data)*. Dec. 2017, pp. 2884–2892. DOI: 10.1109/BigData.2017.8258257 *Cited on page 85.*

[87] Abel Gordon et al. “ELI: Bare-Metal Performance for I/O Virtualization”. In: *Proc. 17th Intl. Conference on Architectural Support for Programming Languages Operating Systems (ASPLOS’12)*. London, UK, 2012 *Cited on page 87.*

[88] yoji yamato. “OpenStack hypervisor, container and baremetal servers performance comparison”. In: *IEICE Commun. Express 4(7)*. 2015, pp. 228–232 *Cited on page 87.*

[89] Doanh Kim Luong et al. “Seamless Handover for Video Streaming over an SDN-based Aeronautical Communications Network”. In: *SIGMETRICS Perform. Eval. Rev.* 46.3 (Jan. 2019), pp. 98–99. ISSN: 0163-5999. DOI: 10.1145/3308897.3308943 *Cited on page 91.*

[90] J. Rizkallah and N. Akkari. “SDN-based vertical handover decision scheme for 5G networks”. In: *2018 IEEE Middle East and North Africa Communications Conference (MENACOMM)*. Apr. 2018, pp. 1–6. DOI: 10.1109/MENACOMM.2018.8371040 *Cited on page 91.*

[91] Claudia Campolo et al. “Slicing on the Road: Enabling the Automotive Vertical through 5G Network Softwarization”. In: *Sensors* 18.12 (2018). ISSN: 1424-8220. DOI: 10.3390/s18124435 *Cited on page 91.*

[92] *Mininet-WiFi*. https://github.com/intrig-unicamp/mininet-wifi. Accessed: 2019-08-25 *Cited on page 92.*

[93] Yimeng Zhao et al. “Virtual Network Migration on the GENI Wide-Area SDN-Enabled Infrastructure”. In: *arXiv e-prints*, arXiv:1701.01702 (Jan. 2017), arXiv:1701.01702. arXiv: 1701.01702 [cs.NI] *Cited on page 94.*

Titre : **Gestion des réseaux virtuels dans un contexte SDN/NFV**

Mots clefs : SDN, NFV, Optimisation, Gestion, Découpage du Réseau, Fiabilité, Disponibilité, Prévision, Handover, Mobilité, Logique floue.

Résumé : Les dernières directions lancées dans le contexte de l'Internet du futur reposent sur la mise en place d'une couche logicielle programmable permettant une vue globale du réseau et une gestion dynamique des ressources. Trois nouveaux paradigmes sont aujourd'hui au cœur de cette tendance: le SDN (Software Defined Networking), le NFV (Network Function Virtualization) et le Network Slicing. Ces approches visent à introduire une certaine agilité pour le déploiement et l'évolution des services réseau.
Afin de répondre aux interrogations présentées, nous nous intéressons dans la première partie de la thèse au processus de déploiement des services réseaux dans un contexte évolutif en termes de technologies et d'usages. En effet, nous nous intéressons au processus d'orchestration permettant l'automatisation du processus de création, de monitoring et de gestion continue des services réseaux au niveau de l'architecture proposée. Dans la deuxième partie de nos travaux de recherche, nous proposons un nouveau algorithme pour l'automatisation et l'optimisation des opérations de création des slices réseaux. La troisième partie de nos travaux est consacrée pour l'étude de la gestion des événements qui occurrent dans les slices tels que la congestion des slices, la mobilité des utilisateurs, etc. Pour Pallier ces problèmes, nous proposons un algorithme basé sur la logique floue. Cet algorithme analyse la valeur actuelle de la charge ainsi que sa future valeur prédite. Nous étudions aussi la mobilité des utilisateurs entre les slices. Nous proposons un algorithme pour la gestion du handover des utilisateurs d'un slice à un autre. L'objectif est de garantir la continuité du service avec la meilleure qualité de service possible.

Title : **SDN/NFV-based Network Slicing Management**

Keywords : SDN, NFV, Optimization, Management, Network Slicing, Reliability, Availability, Prediction, Handover, Mobility, Fuzzy Logic.

Abstract : 5G networks try to cope with the limitations of current network implementations by proposing a new system aiming to meet new challenges. Indeed, unlike previous technologies, 5G will not only enhance the network system but will also provide an end-to-end infrastructure that will support emergent services and respond to stringent user requirements. The key concepts for the 5G vision are the SDN, NFV and Network Slicing technologies. Those paradigms allow the network to provide services for various scenarios under different requirements. They permit to achieve higher performance and flexibility for the network.
In this context, this thesis aims first to propose a new architecture for network slices provisioning and management. We propose the DANSO architecture which considers the SDN, NFV and network slicing technologies in order to present a programmable and flexible framework for services provisioning. The second aim is the optimization of the process of the network slices creation and the admission control of users' request. For this purpose, we propose heuristic algorithms in order to either map the users' demands to existing slices or to create a new network slices. Our algorithms consider the *reliability*, *availability* and *latency* requirements as well as the offered quality by the underlying infrastructure. The third aim is related to the management aspect. In fact, we interest on the management of sudden events that occur in the slice during its running time. In this regard, we study the congestion of the slices and users' mobility events. We propose a fuzzy logic-based algorithm that considers the actual and the predicted load state of the slice in order to perform auto-scaling actions. The future load values are determined using the SVR technique. Finally we interest on the problem of vertical handover management in the context of network slices. We propose an algorithm that decides when to perform the handover and selects the target slice.

École doctorale Galilée, Université Paris 13,
Laboratoire de Traitement et Transport de l'Information (L2TI).
Villetaneusse, France