NFV Management and Orchestration in Large-Scale Distributed Systems

Mohammad Abu-Lebdeh

A thesis
In
The Concordia Institute
for
Information Systems Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of
Doctor of Philosophy (Information and Systems Engineering) at
Concordia University
Montréal, Québec, Canada

April 2018

Concordia University

School of Graduate Studies

This is to certify that the thesis prepared

By: **Mohammad Abu-Lebdeh**

Entitled: **NFV Management and Orchestration in Large-Scale Distributed Systems**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Information and Systems Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Dr. Ion Stiharu	Chair
Dr. Jean-Charles Grégoire	External Examiner
Dr. Dongyu Qiu	External to Program
Dr. Chadi Assi	Examiner
Dr. Jamal Bentahar	Examiner
Dr. Roch Glitho	Supervisor

Approved by _____

Dr. Chadi Assi, Graduate Program Director

12 June 2018 _____

Dr. Amir Asif, Dean

Faculty of Engineering and Computer Science

Abstract

NFV Management and Orchestration in Large-Scale Distributed Systems

Mohammad Abu-Lebdeh, Ph.D.
Concordia University, 2018

Network Functions Virtualization (NFV) radically transforms the way network operators design and manage network services, promising a lot of potential benefits such as agility, flexibility, reduction of CAPEX and OPEX. It eliminates the dependency between the network function software and hardware enabling pure-software based network function that runs on commodity hardware, called Virtualized Network Function (VNF). NFV, along with other emerging technologies such as Software-Defined Networking (SDN), enables network operators to create dynamic and programmable network services, wherein VNFs are deployed on-demand, dynamically chained and optimized over time to cope with emerging business needs. The European Telecommunications Standards Institute (ETSI) developed the NFV Management and Orchestration (MANO) framework, which consists of Virtualized Infrastructure Manager (VIM), VNF Manager (VNFM) and NFV Orchestrator (NFVO), in order to provide network operators with the sophisticated capabilities needed to manage the dynamic aspects of infrastructure, VNFs and network services.

This thesis elaborates and addresses key architectural and algorithmic research challenges related to the NFV management and orchestration in distributed and large-scale systems. We look at orchestration scalability from an architectural perspective and propose to leverage two-layer hierarchical service orchestration to manage network services over distributed infrastructure. We also propose an architecture of Virtual Network Platform-as-a-Service (VNPaaS) that utilizes the hierarchical orchestration to offer next-generation mobile networks as-a-service. The architecture is illustrated by offering the 3GPP Home Subscriber Server (HSS) as-a-Service (HSSaaS), in which the HSS is decomposed into VNFs with a granularity finer than what is known today. On the algorithmic side, a key challenge is to identify the number and location of the NFVO and VNFM functional blocks since they have a significant impact on the overall system cost and performance, among others. In particular, we tackle the online placement of VNFM to enable network operators to adjust the number and location of VNFMs in response to variation in workload. There, we assume a fixed location of NFVO and aim at minimizing the operational cost. Owing to its complexity, we propose a tabu search heuristic and numerically show that it is

faster than the mathematical formulation by many orders of magnitude. We further study the joint placement of NFVO and VNFM. We first address the problem in the context of the multi-orchestrator system and seek to minimize the number of NFVOs and VNFMs. We mathematically formulate the problem and propose a two-step placement heuristic to solve the problem efficiently. Finally, we investigate the same problem in the context of single- and multi-orchestrator systems providing a comparative study of the worst-case delay in both scenarios. We also propose a late acceptance hill-climbing heuristic to solve the problem in a reasonable time frame.

Acknowledgments

All praise is due to Allah who gave me the strength, determination, patience and guided me throughout this work and beyond.

First, and foremost, I am grateful to my Ph.D. supervisor Prof. Roch Glitho for his guidance, support, patience, and encouragement. He is the dedicated, professional and caring advisor. Thank you.

I gratefully acknowledge my Ph.D. committee members, Prof. Chadi Assi, Prof. Jamal Bentahar and Prof. Dongyu Qiu for their time, effort and constructive comments. I would also like to extend my appreciation to the external examiner Prof. Jean-Charles Grégoire for accepting to serve in my Ph.D. thesis committee.

I am also thankful to Dr. Fatna Belqasmi, Dr. Jagruti Sahoo, Dr. Sami Yangui, Dr. Diala Naboulsi and Mr. Constant Wette Tchouati for all the enlightening discussions, comments and collaboration. It was a great pleasure working with you.

Furthermore, I am thankful to all of my colleagues in the telecommunication service engineering lab at Concordia University. In particular, I would like to thank Abbas Soltanian, Carla Mouradian, and Narjes-Elaheh Tahghigh for their companionship, support, ideas, and fruitful discussions.

Last but not least, I am forever indebted to my family for their encouragement, continuous support, love and prayers. My sincere thanks go to my mother, Aisha. Without you, this thesis would not have been possible. There are no words that can express my gratitude and love for you.

Contents

List of Figures

List of Tables

List of Acronyms

3GPP	3rd Generation Partnership Project
AIR	Authentication Information Request
API	Application Programming Interface
CDF	Cumulative Distribution Function
CDN	Content Delivery Network
CSCF	Call State Control Function
EM	Element Management
EPC	Evolved Packet Core
ePDG	evolved Packet Data Gateway
EPS	Evolved Packet System
ETSI	European Telecommunications Standards Institute
FCAPS	Fault, Configuration, Accounting, Performance, and Security
FE	Front End
FLP	Facility Location Problem
GSO	Global Service Orchestrator
HSS	Home Subscriber Server
HSS-FE	Home Subscriber Server Front End
HSSaaS	Home Subscriber Server as-a-Service
I-CSCF	Interrogating Call State Control Function
ILP	Integer Linear Programming

IMS	IP Multimedia Subsystem
ISG	Industry Specification Group
KPI	Key Performance Indicator
LAHC	Late Acceptance Hill-Climbing
LIR	Location Info Request
LTE	Long Term Evolution
MANO	Management and Orchestration
MAR	Multimedia Authentication Request
MME	Mobility Management Entity
MPP	VNFM Placement Problem
MVNO	Mobile Virtual Network Operator
NFaaS	Network Functions-as-a-Service
NFV	Network Function Virtualization
NFVI	Network Function Virtualization Infrastructure
NFVIaaS	NFVI-as-a-Service
NFVO	Network Function Virtualization Orchestrator
OMP-MO	NFVO and VNFM Placement in Multi-Domain
OMP-SO	NFVO and VNFM Placement in Single-Domain
ONAP	Open Network Automation Platform
P-CSCF	Proxy Call State Control Function
PaaS	Platform-as-a-Service
PCC	Policy and Charging Control
PCRF	Policy and Charging Rule Function
PDN-GW	Packet Data Network Gateway
PoP	Points of Presence
PUR	Purge UE Request

S-CSCF	Serving Call State Control Function
S-GW	Serving Gateway
SAR	Server Assignment Request
SDN	Software-Defined Networking
SIP	Session Initiation Protocol
SIP-AS	Session Initiation Protocol-Application Server
SPR	Subscription Profile Repository
TOSCA	Topology and Orchestration Specification for Cloud Applications
TSP	Two-Step Placement
UAR	User Authorization Request
UDC	User Data Convergence
UDR	User Data Repository
UE	User Equipment
ULR	Update Location Request
vCPE	virtual Customer Premises Equipment
VM	Virtual Machine
VNaaS	Virtual Network-as-a-Service
VNF	Virtualized Network Function
VNFaaS	VNF-as-a-Service
VNFM	Virtualized Network Function Manager
VNPaaS	Virtual Network Platform-as-a-Service
WAN	Wide Area Network

Chapter 1

Introduction

1.1 Overview

During the past decade, cloud computing has gained significant momentum for delivering computing resources (e.g., networks, servers and storage) as utility [1]. These resources are pooled, generally using virtualization technologies, and offered to different users (individuals and enterprises), who can dynamically provision and release these resources to accommodate their demand following the pay-as-you-go financial model. Cloud computing replaces up-front resource provisioning with elastic resource allocation, which stimulates the emergence of sophisticated cloud automation tools that accelerate innovation through automated provisioning, governance, and management of the cloud services, enabling enterprises to provide agile, scalable and cost-efficient cloud services.

The success of cloud computing, as an approach for providing scalable and cost-efficient services, motived leading telecommunication network operators to initiate an Industry Specification Group (ISG), within the European Telecommunications Standards Institute (ETSI) to overcome current network deficiencies by leveraging cloud technologies (e.g., virtualization and automation) [2]. Today, introducing new network services often requires the deployment of additional proprietary hardware appliances at fixed locations in the infrastructure. These appliances are designed to perform particular network functions and cannot be easily modified to support new operations. This static approach in service management

limits the capability of innovation and support for new services. It also leads to low resource utilization, high capital and operational expenditures [3].

ETSI proposed the notion of Network Functions Virtualization (NFV) to enable network functions to run as cloud applications and allow network services to be provisioned as cloud services [2]. NFV [4] eliminates the dependency between the network function software and underlying hardware and implements the network function in software module called Virtualized Network Function (VNF), which enables consolidation of many network equipment onto standard high volume servers, switches and storage. By that, NFV, along with other emerging technologies such as Software-Defined Networking (SDN), enables network operators to create dynamic and programmable network services, wherein VNFs are deployed on-demand at any point in the infrastructure, dynamically linked and optimized over time to cope with emerging business cases and needs [5].

Currently, ETSI is leading the way in the standardization of NFV. So far, it has developed a set of requirements, specifications and architectures that cover various aspects of NFV technology. Among them is the NFV architectural framework [6] depicted in Figure 1.1. It describes the building blocks of an NFV system in an administrative domain. It encompasses VNFs, NFV Infrastructure (NFVI) and NFV Management and Orchestration (MANO) framework. VNFs are the software implementation of the network functions. NFVI is the environment in which VNFs are deployed. It is a combination of hardware and software resources that may span several geographically distributed locations. A single location, where a network function could be deployed as VNF, is called a Point of Presence (PoP). Moreover, the MANO framework [7] is responsible for the orchestration and lifecycle management of network services including all relevant functions, such as deploying VNFs, optimizing their performance and managing their associated resources. Currently, it is the most prominent NFV management framework and has been adopted by a majority of open-source and commercial NFV platforms [8, 9]. The next section sheds more light on this framework.

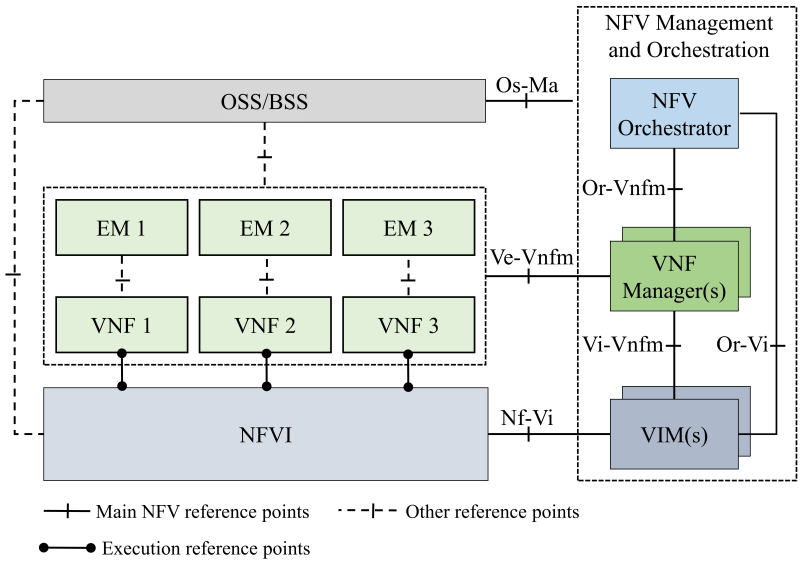


Figure 1.1: ETSI NFV architectural framework [6]

1.2 NFV Management and Orchestration Framework

The ETSI NFV MANO framework consists of three functional blocks: the Virtualized Infrastructure Manager (VIM), the VNF Manager (VNFM) and the NFV Orchestrator (NFVO). These functional blocks form three distinct management layers with different functional roles as explained next:

1. VIM: The VIM manages and controls the NFVI compute, storage and network resources. For example, it performs resource allocation and de-allocation on behalf of the NFVO and VNFM. It also collects and reports resource fault and performance information. The NFVI resources can be managed by one or more of VIMs. Each VIM can manage a subset of resources within a PoP, all resources within a PoP, or the resources across multiple PoPs [7].

2. VNFM: The VNFM is responsible for the lifecycle management of one or more VNF instances. As such, each VNF instance is associated with a VNFM. Lifecycle management refers to the set of functions required to manage the instantiation, maintenance and termination of a VNF or network service [10]. In this context, for instance, the VNFM can collect the virtualized resource performance information from the VIM, and the VNF indicators

from the Element Management (EM) or VNF instance. An indicator is application-level information that provides insight into the VNF behavior [11]. The VNFM uses the collected information for decision making, such as VNF scaling and healing. Furthermore, the VNFM can be either generic or VNF-specific [12]. A generic VNFM can manage VNF instances of different types that might be provided by different VNF providers. A VNF-specific VNFM has a dependency on the VNFs and can manage VNF instances of defined type(s), usually provided by the VNF provider [12]. In an administrative domain, one or more VNFMs can be used to manage the VNF instances. Further, as shown in Figure 1.1, the VNFM may communicate with Element Management (EM) to manage the VNF instances. EM is responsible for FCAPS (Fault, Configuration, Accounting, Performance, and Security) management functionality for one or more of VNF instances. It has overlapping functions with the VNFM. However, the key difference between them is that the EM manages a VNF instance through a proprietary reference point, whereas the VNFM uses a standard reference point [13]. By that, the EM can play the role of proxy by exposing the VNF management functions to the VNFM through a standard reference point [13].

3. NFVO: Two main tasks are delegated to the NFVO. First, it performs resource orchestration across multiple VIMs. The role of NFVO involves, but not limited to, resource request authorization and capacity management. Second, it is in charge of the lifecycle management of network services which involves coordination with VNFMs in managing the lifecycle of VNF instances. For example, the NFVO and VNFMs work jointly to ensure that the VNF instances meet the desired requirements (e.g., performance). The NFVO can collect the VNF indicators and virtualized resource performance metrics from VNFMs. It analyzes this information to assure that the network services satisfy their requirements. Finally, it is worth noting that a single NFVO exists in an administrative domain.

Moreover, the NFV MANO framework includes a set of reference points, as shown in Figure 1.1, to enable communications among MANO functional blocks as well as the communication with other NFV functional blocks such as EM and VNF. ETSI specifications [11, 14–17] define the interfaces, operations and information model supported by the reference

points. However, the details of the operations and the communication protocols are not discussed. We highlight in particular, the following reference points that are relevant to our work: *Or-Vnfm*, a reference point between the NFVO and VNFM; *Or-Vi*, a reference point between the NFVO and VIM; *Vi-Vnfm*, a reference point between VNFM and VIM; *Ve-Vnfm*, a reference point between VNFM and EM/VNF instance.

1.3 Challenges and Thesis Contributions

The responsibilities associated with MANO functional blocks leave no doubt that their performance is crucial to network operators. Today, there are many emerging NFV scenarios where the network services will span large geographical area (e.g., country or continent) and the number of VNF instances will grow tremendously. For instance, the forthcoming 5G cellular system, for which NFV is considered an essential enabling technology [18, 19], is likely to run on a highly distributed NFVI to satisfy the requirement of 1 ms round-trip latency. 5G also requires scalable NFV architecture to deliver the required massive capacity and connectivity [20]. Another example is the Content Delivery Network (CDN) which is typically deployed over a large geographical area to deliver contents (e.g., video) to end-users with low delay. In these scenarios, the distributed nature and scale of NFV deployments lead to scalability and performance issues:

- Communication delay: The decision making in NFV MANO framework is distributed among the three functional blocks (i.e., VIM, VNFM and NFVO). These functional blocks communicate with each other and with other NFV functional blocks (e.g., VNFs) in order to fulfill their functionalities. The physical distance between the functional blocks in NFV system is one of the factors that introduce communication delay. When the communication delay is high, the execution time of the MANO operations becomes longer which decreases the scalability and degrades the performance. According to [8], the communication overhead and delay may prevent the frequent collection and analysis of monitoring data (e.g., performance information) from the environment (e.g., VNF instances and VIMs). Another example is the VNF fault

management [21]. Fast failure notification and recovery are necessary to minimize the impact of the failure and maintain the reliability of the network services.

- Centralized MANO: Relying on single NFVO in a distributed and large-scale deployment will hinder the scalability of the orchestration process [22]. As the NFV deployment grows with respect to the infrastructure size and the number of VNFs, the NFVO will have to cope with more requests and events. Since the system is bounded by the processing capacity of the NFVO, it can become a bottleneck as the load grows with the size of the deployment. Besides, if the NFVI has a large diameter, no matter where the NFVO is placed, the communication with other functional blocks, especially VIM, would encounter a high delay.

- Number of VNFMs: The VNFM functional block is responsible for the lifecycle management of VNF instances. However, in large-scale deployments, the VNFMs should manage thousands of VNF instances without compromising the performance of its management functions. Thus, the number of VNFMs must be adequate to cope with the number of VNF instances.

These obstacles must be addressed before NFV can advance to reality, especially when the network services in production deployments are associated with carrier-grade requirements. This thesis aims to supplement the undergoing research efforts towards design and operate NFV MANO platforms. The thesis addresses fundamental architectural and resource allocation challenges related to the NFV MANO. These problems are summarized next.

1.3.1 VNF Manager Placement Problem

As defined by the NFV MANO framework, the VNFM functional block is responsible for the lifecycle management of the VNFs. However, these VNFs can be instantiated on-demand when and where needed and elastically scaled to meet the demand variability while maintaining cost efficiency. Hence, the number of VNF instances, their types (e.g., firewall) and locations can vary over time. The VNFMs should manage the lifecycle of thousands

of VNF instances without compromising the performance of its management functions. At any time, the number of VNFMs should be adapted to the VNF instances deployed in the system, for optimal performance of the system. Besides, the placement of VNFM can significantly affect the overall system performance and operational cost. Adapting the number and placement of VNFMs accordingly can thus result in important savings for operators.

In this thesis, we introduce and investigate the VNFM Placement Problem (MPP). We present two versions of the problem: static MPP with permanent placement decisions and dynamic MPP with placement decisions that change over time. Assuming that we are given the placement of the NFVO, the dynamic placement configuration of a set of VNF instances over geographically distributed NFVI, we aim at finding the optimal number and placement of the VNFMs, at each moment, that minimizes the operational cost. We mathematically formulate the VNFM placement problem and propose a tabu search metaheuristic to solve large instances of the problem. We compare our tabu approach against the mathematical model over various NFVI topologies. Our numerical results show that our tabu search heuristic yields high-quality solutions in considerably fast runtime. Moreover, we study the impact of crucial aspects, i.e., NFVO location, VNFM architectural options (generic and VNF-specific) on the outcome of the problem. We show that they can have a notable impact on the placement decisions and require adequate tuning according to the operators' requirements.

1.3.2 A Scalable Architecture for NFV Management and Orchestration

Centralized NFV MANO solutions, which were the common approach in literature until recently, are prone to scalability and performance issues in large-scale and distributed deployments [8, 22]. These solutions rely on a single NFVO in the system which would hinder the scalability of the orchestration process [22]. On the one hand, the number of network services and their constitute VNFs would grow far beyond the processing capacity of an NFVO. On the other hand, the delay between the NFVO and other functional blocks would increase with the increase of the size of the covered geographical area. A high network delay

increases the communication overhead and can negatively impact critical functions.

In this thesis, we propose to employ two-layer hierarchical service orchestration in order to address these challenges, wherein the NFVI is decomposed into domains (or orchestration zones) and an NFVO is assigned the resource and service orchestration within a domain. By that, NFVO can be placed close to hosting infrastructure and the communication overhead is minimized. Besides, a global service orchestrator performs the end-to-end service orchestrations across different domains. We propose an architecture of Platform-as-a-Service, which utilizes the two-layer hierarchical service orchestration approach, for provisioning 3GPP 4G and beyond core networks as-a-service. We also present a proof-of-concept prototype to validate the feasibility of the approach. We use the Home Subscriber Server (HSS) as-a-Service (HSSaaS) as an illustrative use case. It relies on a novel NFV-based architecture of HSS, in which the HSS is decomposed into VNFs with a granularity finer than what is known today. The new architecture allows the different diameter interfaces of HSS to be deployed and scaled independently. It also enables performance isolation between these interfaces, which is further demonstrated by experimentation.

1.3.3 Joint Placement of NFV Orchestrator and VNF Manager Problem: The Multi-Orchestrator Case

As mentioned earlier, we propose multi-orchestrator and hierarchical orchestration architecture to address the NFV MANO scalability and performance challenges for large-scale and distributed NFV systems, wherein multiple instances of NFVO and VNFM are used to manage the lifecycle of network services and VNFs. However, there is still the challenge of finding the optimal number and placement for these functions blocks that provide the required capacity and performance. Hence, we introduce and study the joint placement of NFVO and VNFM in the context of the multi-orchestrator system. In particular, given the NFVI topology, a set of VNF instances, and the location of the global service orchestrator, we aim at finding the number and placement of NFVOs and VNFMs needed in the system that minimizes their number, as it is a measure of the cost. We mathematically formulate the problem, propose a two-step placement heuristic and evaluate it.

1.3.4 Joint Placement of NFV Orchestrator and VNF Manager Problem: The Single and Multi-Orchestrator Cases

Since the locations of NFVO and VNFM functional blocks have a significant impact on the delay experienced in the communication, we revisit the joint placement of NFVO and VNFM problem aiming at minimizing the total worst-case delay between the various functional blocks for both single- and multi-orchestrator systems. We also investigate the impact of the number of NFVOs and VNFMs on the worst-case delay providing a comparative study of the delay in both scenarios. Moreover, we present mathematical formulations of both problems and propose a late acceptance hill-climbing heuristic to solve them in a reasonable time frame.

1.4 Thesis Outline

The rest of the thesis is organized as follows. Chapter 2 presents the requirements of the NFV MANO related to our work, followed by a thorough review of the state-of-the-art. Chapter 3 discusses the VNFM placement problem. In chapter 4, we propose a scalable architecture for NFV management and orchestration. Chapter 5 discusses the joint placement of NFVO and VNFM functional blocks for the multi-orchestrator system. We revisit the placement of NFVO and VNFM problem in chapter 6 and investigate the problem for both single- and multi-orchestrator systems. We conclude this manuscript in chapter 7 and provide future directions for this research.

Chapter 2

Related Work

In this chapter, we first present the requirements of the NFV MANO related to our work. After that, in the light of these requirements, we survey the state-of-the-art and review it.

2.1 Requirements

We consider the following requirements to be the most pertinent to NFV MANO:

1. Scalability: NFV MANO functions are important for realizing network services associated with the carrier-grade characteristics. The NFV MANO platforms should be scalable to exploit the benefits of NFV, given that the design aspects of NFV MANO such as workload and propagation delay among functional blocks can lead to long response time, affecting the ability to respond rapidly to events and reducing the reliability of the system. Addressing such a problem requires a solution with a scalable architecture. Besides, both the number and location of NFVO and VNFM functional blocks must be planned to provide the needed scalability and performance.

2. Elasticity: VNFs can be deployed and scaled dynamically to meet demand. This will lead to workload fluctuation on the NFVO and VNFM functional blocks. Depending on the level of variation, adding new instances of these functional blocks or removing existing ones might be necessary to cope with the workload and ensure cost-efficiency.

This is, in particular, more relevant to the VNFM as it is responsible for VNF lifecycle management.

2.2 Related Work

In this section, we will discuss the works from the literature that are closely related to each of the thesis contributions. We first discuss and analyze the works related to the NFV MANO architecture. After that, we review the works related to the resource allocation for NFV MANO functional blocks.

2.2.1 NFV Management and Orchestration Architecture

The NFV community has paid particular attention to NFV MANO challenges and designed platforms that can provision network services over distributed infrastructure. We classify these solutions into two categories based on whether they follow the centralized or distributed management approach. We consider the management is centralized when a single NFVO is in charge of resource and service orchestration in the system. On the other hand, the management is distributed when the architecture employs multiple NFVOs to perform the MANO operations (i.e., multi-orchestrator system).

2.2.1.1 Centralized Approach

T-NOVA [23, 24] is a European funded project that designs and implements an NFV management and orchestration platform for provisioning network functions-as-a-service over a distributed infrastructure. It provides a VNF marketplace for third-party which allows VNF developers to describe and publish service offerings. It also enables customers to browse, select and deploy these services. The proposed solution covers the entire ETSI NFV MANO framework stack (i.e., VIM, VNFM and NFVO). It encompasses an orchestrator called TeNOR that provides ETSI NFVO and VNFM functionalities and can automate four phases of the network service lifecycle management, namely: resource discovery, service mapping, service deployment and monitoring. SONATA [25] is another European

project that provides network services development and orchestration functionalities. The proposed architecture consists of two main components: a software development kit and a service orchestration platform. The software development kit allows developers to define complex services consisting of multiple VNFs that can be deployed and managed on SONATA service platform. SONATA service platform provides a customizable management and orchestration framework.

Moreover, Sciancalepore et al. [26] study the impact of co-existence of multi-access edge computing applications and VNFs on the same infrastructure in the context of 5G systems. The authors propose an extension to the ETSI NFV MANO framework in order to enable the join orchestration of VNFs and multi-access edge computing applications and consequently reduce the infrastructure and operational cost. Garcia et al. [27] propose a platform-as-a-service architecture to enable deployment and provisioning of real-time multimedia communications and media processing services in an NFV environment. The proposed architecture encompasses NFVO and VNFM functional blocks in order to support lifecycle management of media servers and cloud repository. Vilalta et al. [28] propose an architecture to manage network services over for multi-domain transport networks and distributed NFVI. The architecture is illustrated through two use cases: virtual path computation element and virtual SDN controllers.

Furthermore, there are several open-source projects that aim to provide reference implementations of the ETSI NFV MANO framework. For instance, Tacker [29] emerged as an OpenStack project to provide the functionalities of NFVO and generic VNFM to deploy and operate VNFs. OpenMANO [30] is an ETSI hosted project that aims to build management and orchestration stack aligned with ETSI NFV specifications. OpenBaton [31] is another example which provides an NFV orchestration solution that supports NFVO, generic VNFM and generic EM capabilities to enable VNF deployment on the top of multiple cloud infrastructure.

All presented MANO solutions follow the same approach and try to build an NFV platform with centralized management. As discussed earlier, this approach leads to scalability and performance issues as the NFVO would become a potential bottleneck in the system. In

addition, the design and operational aspects of NFV MANO, such as number and placement of MANO functional blocks, are not discussed.

2.2.1.2 Distributed Approach

Recently, several studies have embraced distributed MANO approach in order to address various orchestration challenges such as multi-technology and multi-administrative domains orchestration. In these architectures, multiple orchestrators collaborate in performing the resource and network service orchestration in the system. Each orchestrator performs the orchestration functions over part of the infrastructure, often referred to as a domain. The orchestrators coordinate with each other to ensure the delivery of end-to-end network service. However, since there is no standard architecture to define the interactions among these orchestrators, several design choices have emerged in the literature. Broadly speaking, these approaches can be classified into three models, namely: hierarchical, flat (or peer-to-peer) and hybrid models.

The hierarchical model organizes the orchestrators into two or more layers. The orchestrators in the bottom layer perform resource and service orchestration within their domains. Meanwhile, the orchestrators in the second layer and upwards are responsible for the service orchestration across multiple orchestrators in the lower adjacent layer. The top layer often encompasses a global service orchestrator that maintains a global view of the entire system and is in charge of end-to-end service orchestration. This model relies on vertical communications between the orchestrators in adjacent layers. In contrast, the flat model does not exercise hierarchical control among orchestrators. It is, in fact, adopts horizontal communications and allows individual orchestrators to communicate directly with other orchestrators. Moreover, the hybrid model is a composition of the hierarchical and flat models. It organizes the orchestrators into layers with vertical communications between adjacent layers. Nevertheless, it allows horizontal communications between the orchestrators in the same layer.

The majority of the works available in the state-of-the-art adopted hierarchical orchestration. For instance, Garay et al. [22] propose a novel service graph model that can be

split into subgraphs according to the orchestrator responsible for the assigned resources and consequently it enables hierarchical orchestration of network services. Further, an ETSI report [15] presents two-layer hierarchical service orchestration as an architectural option emerged due to the design flexibility of ETSI NFV MANO framework. The architecture aims to address the challenge of providing end-to-end network services across two administrative domains. There, the architecture encompasses two layers of orchestrators. In the bottom layer, there is an NFVO in each domain that performs resource and network service orchestration within its domain. The top layer includes an umbrella NFVO that is responsible for the orchestration of network services across the two domains.

Two-layer hierarchical service orchestration has also been applied by [32, 33]. Katsalis et al. [32] focus on multi-domain orchestration over multi-technology domains. Open Network Automation Platform (ONAP) [33] is an open source platform that is derived from AT&T's OpenECOMP and Open-O projects. It provides policy-driven orchestration and automation of physical and virtual network functions. The ONAP Amsterdam release supports hierarchical service orchestration wherein the service orchestrator component is in charge of end-to-end service orchestration and virtual function controller component, which provides ETSI NFV compliant NFVO functions, is responsible for lifecycle management of network services [34].

Further, Li et al. [35] propose a 5G cross-haul architecture that encompasses three network segments: access, cross-haul and core networks. Each of the segments has its dedicated MANO functional blocks, i.e., NFVO, VNFM and VIM. The authors claim that both hierarchical and flat models are applicable to provide end-to-end network services across all segments. Besides, the European 5G Exchange (5GEx) project [36] proposes a platform to enable service orchestration over multiple domains for the same or different administrations in the context of 5G. The platform supports hybrid orchestration model.

In all above-discussed studies, the NFV MANO scalability challenge has been discussed only in [22]. However, the study does not include an architecture realizing the idea; rather, it stays at the conceptual level. The design aspects of NFV MANO are out of the scope of the remaining works, which aim at addressing other challenges such as orchestration of

multi-technology and multi-administrative domains.

2.2.2 Resource Allocation

To the best of our knowledge, there are no previous studies that discuss and target the placement of NFVO MANO functional blocks. We thus review the works that have been done on similar problems in the area of NFV and SDN, in particular, the VNF placement and SDN controller placement problems.

2.2.2.1 VNF Placement

The problem of placing chains of VNF instances has been extensively studied [37]. There, the idea is to optimize the placement of chains of VNF instances, over commodity servers in the system, by reserving resources as needed and according to a predefined objective. A chain of VNF instances is a sequence of VNF instances that together offer a network service. A variety of objectives has been covered in the literature. Kim et al. [38] propose VNF placement strategy that seeks to minimize the overall energy consumption while guaranteeing the service latency requested by end-users. Bhamare et al. [39] aim at minimizing inter-cloud traffic and response time over geographically distributed clouds. Cao et al. [40] investigate the VNF placement for 5G mobile networks in order to achieve lower bandwidth consumption and lower maximum link utilization. Pham et al. [41] propose a placement strategy that minimizes energy and traffic cost. Moens et al. [42] focus on minimizing the number of used nodes. Mechtri et al. [43] aim at enabling efficient resource utilization in the system. The objective targeted by Hirwe et al. [44] is instead to minimize the length of paths traversed by flows. Qu et al. [45] and Xia et al. [46] target the minimization of communication cost. Minimizing the global operational cost, covering setup and network traffic costs, has been the focus of several works including Ghaznavi et al. [47], Cohen et al. [48] and Bouet et al. [49]. Mehraghdam et al. [50] push the analysis even further and study trade-offs among different optimization objectives including maximizing the data rate, minimizing the number of used nodes and minimizing the delay.

While significant effort has been put to study VNF placement problem in the NFV

community, the resource allocation for NFV MANO has not received any attention so far. The resource allocation for NFV MANO differs from the VNF placement problem and needs to be studied per se. In fact, different functional blocks are implied in each of them. As per definition, the VNF placement problem implies solely the VNF instances [37], aiming at serving requests, and disregards NFV MANO functional blocks. Instead, in the resource allocation for NFV MANO, various functional blocks are involved. These include the placed VNF instances, EMs, VIMs, VNFMs and the NFVO.

2.2.2.2 SDN Controller Placement

Similar to our problems in the area of SDN is the controller placement problem. The problem can be stated as selecting the location of one or more SDN controllers and allocating switches to these controllers to optimize a certain objective [51]. The problem has been introduced by Heller et al. [52] who showed that random placement of a controller might yield a solution with five times worse latency than an optimal one. The problem has gained significant attention over the past few years in the networking community as it impacts different aspects of SDN networks like performance, cost, and resiliency. The proposed placement strategies in the literature can be classified into a static approach and a dynamic approach. The static problem corresponds to a static network design problem which uses static mapping of switches to controllers. The dynamic placement dynamically adapts the number of controllers and their locations with changing network conditions. Different metrics have been considered to find the number and location of the controller(s). According to Huque et al. [53], the placement strategies can be classified into two categories depending on the used metrics. The first category uses only the latency between switches and controllers to identify the number and locations of controllers in the network. The second category considers the latency and traffic load of switches to find the number and location of controllers.

A variety of objectives has been targeted in literature. For example, Killi et al. [54] propose a placement strategy that plans for controller failures and assigns every switch to more than one controller to ensure reliability; one controller serves as a primary controller

while the others are backup. The proposed strategy aims to minimize the maximum sum of the latency from the switch to the closest controller with enough capacity (primary controller) and the latency from the primary controller to its closest controller with enough capacity (backup controller). Bari et al. [55] consider the dynamic controller placement problem where the number and location of controllers are adjusted according to network dynamics. The authors propose a management framework for dynamically deploying multiple controllers within a Wide Area Network (WAN) in order to minimize the operational cost while satisfying the demand. Similarly, Huque et al. [53] target the dynamic controller placement, but in the context of large-scale SDN deployments. The proposed solution seeks to minimize the latency between switches and controllers.

Moreover, Sallahi and St-Hilaire [56] discuss the challenge of adding new switches to an existing SDN network. The authors propose an expansion model that allows network operators to expand their network at minimum costs. Muller et al. [57] propose a controller placement strategy that maximizes the connectivity between switches and controllers. The authors present the average number of node-disjoint paths between switches and controllers as a metric to characterize the connectivity. Lange et al. [58] study the trade-offs that exist among a variety of metrics including delay, resilience and load balancing.

The controller placement problem has similarity to our problems in the sense that it selects the best locations for functional blocks (or nodes) in a given network topology. The controller placement, when network delay is considered, resembles the facility location problem [51, 52]. However, the joint placement of NFVO and VNFM problem corresponds to the hierarchical facility location problem since it finds the locations for different facility types (i.e., NFVO and VNFM) in a multi-level system. On the other hand, the VNFM placement problem corresponds to the facility location problem. Nevertheless, each of controller placement and VNFM placement problems has distinct particularities since the SDN controller and VNFM perform distinct functions in different systems.

Chapter 3

VNF Manager Placement Problem

3.1 Introduction

Given the roles associated with NFV MANO functional blocks, their placement has a critical impact on the system scalability and performance, especially in large-scale and distributed NFV environments. There, communication among these functional blocks takes place over WAN links and thus it can suffer from high delay that is not tolerable by the management functions. Moreover, the placement of MANO functional blocks can significantly affect the overall system operational cost. This is particularly true for VNFMs that can be numerous in the system. In fact, resources cost, including both compute and bandwidth resources cost, differ depending on the location and time, due to differences and changes in energy cost [59] and pricing policies of multiple NFVI as-a-Service [60] providers. Also, network traffic changes dynamically in the system. Adapting the number and location of VNFMs accordingly can thus result in important savings for operators.

To this extent, the placement of the MANO functional blocks is indeed an important problem to address. In this chapter, we tackle in particular the problem of placing VNFMs dynamically in the context of large-scale and distributed NFV systems. We exclude from our problem decisions on the placement of VIMs, as they are part of NFVI design options. We further exclude decisions on the placement of the NFVO to narrow the scope of the problem. Therefore, assuming that we are given the placement of the NFVO, the dynamic placement

configuration of a set of VNF instances over geographically distributed PoPs, together with the corresponding VIMs, we aim at finding the optimal number and placement of the VNFMs, at each moment, that minimizes the operational cost under delay and capacity constraints. More precisely, delay limitations over reference points, enabling communication between a VNFM and other functional blocks in the system, are considered. Moreover, VNFMs capacity limitation in terms of the number of assigned VNF instances is covered. Bandwidth capacity limitation over communication links is also considered. We refer to this problem as the VNFM Placement Problem (MPP). To the best of our knowledge, we are the first to address this problem.

The MPP has two versions, static MPP, with permanent placement decisions and dynamic MPP with placement decisions that change over time. We propose a general Integer Linear Program (ILP) formulation of the problem. It allows determining the number and placement of VNFMs at minimum overall management cost for operators, at each moment. We also propose to employ a tabu search metaheuristic to solve the MPP problem in both static and dynamic schemes. Tabu search is an efficient neighborhood search method that uses adaptive memory. We carefully design its steps, in the light of the peculiarities of our problem. Moreover, we assess the performance of the tabu search metaheuristic over a realistic dataset. We compare its solution to the optimal one derived based on the ILP model. We also compare its results to those obtained based on a first-fit greedy approach. Our small- and large-scale evaluations confirm that the tabu search metaheuristic allows deriving high-quality solutions in a very short time. We also study the impact of key aspects, e.g., NFVO location, VNFM architectural options (generic and VNF-specific) and objective function weight, on the outcome of the problem. We show that they can have a notable impact on the placement decisions and require adequate tuning according to the operators requirements. Finally, we show that dynamic placement decisions, derived according to dynamic MPP, lead to significant reductions in cost with respect to static MPP.

3.2 The VNFM Placement Problem

This section is dedicated to the presentation of the problem. We start first by motivating the VNFM placement problem and stating it in section 3.2.1. We then present the system model in section 3.2.2 that allows us to formulate the problem in section 3.2.3.

3.2.1 Motivation and Problem Statement

The next generation mobile system (5G) is an anticipated large-scale and distributed NFV deployment. NFV is foreseen as a key enabling technology for 5G to reduce the overall cost and to connect a massive number of users and devices [19]. Mobile systems are highly distributed [18], and 5G is not an exception. In fact, 5G is likely to have a more distributed architecture to cope with the envisioned ultra-low latency requirement. ITU-R recommendation sets the 5G goal to support a round-trip latency of about 1 ms [20]. Further, the usage patterns of mobile traffic and services encounter significant variations over time and space [61]. On a typical working day, traffic is concentrated during work hours in business areas and gets shifted to residential areas later during the evening. Moreover, special events, such as an occasional football match in a stadium, can cause occasional spikes in traffic over a limited duration.

In that context, the NFV MANO will manage network services associated with service level agreement (on performance, availability, etc.) that should be enforced [62]. In NFV, the network services and VNFs can be instantiated on-demand when and where needed and elastically scaled to meet the demand variability while maintaining cost efficiency. Hence, the number of VNF instances, their types (e.g., firewall) and locations can vary over time. The VNFMs should manage the lifecycle of thousands of VNF instances without compromising the performance and reliability of its management functions. At any time, the number of VNFMs should be adapted to the number of VNF instances deployed in the system so that the system maintains adequate capacity and performance.

As VNF instances would span geographically distributed PoPs, inter-PoP WAN becomes an important pillar in the performance and operational cost of the system. A VNFM

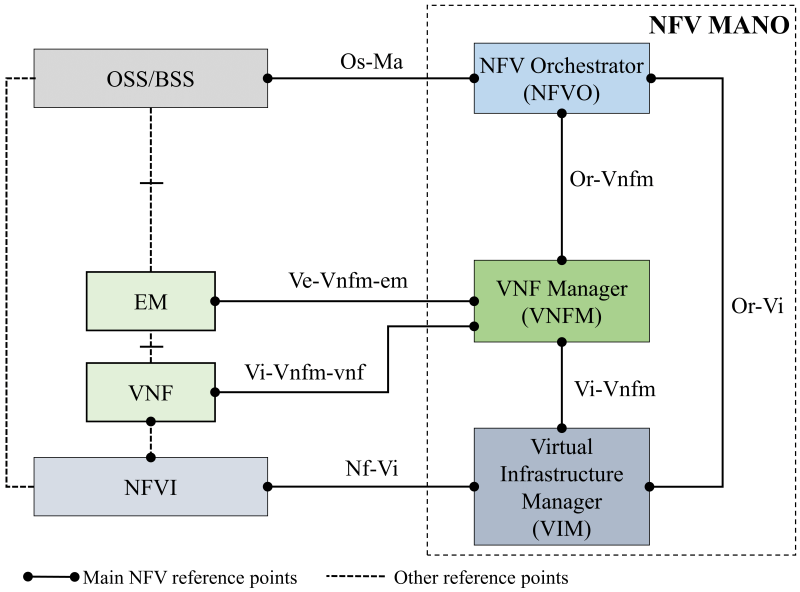
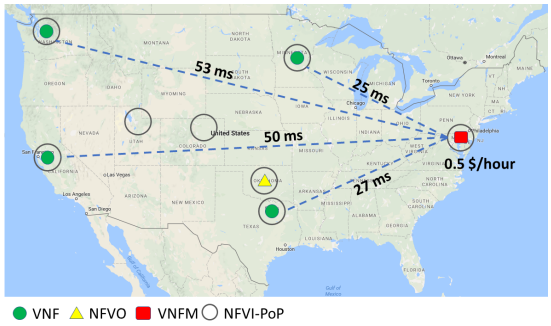


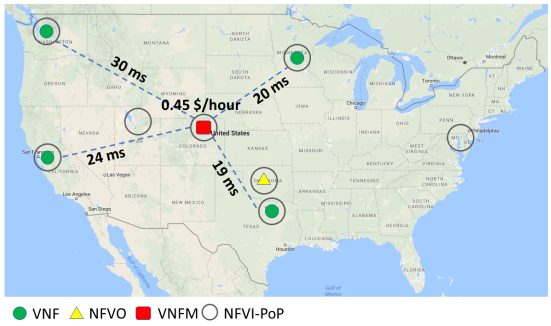
Figure 3.1: ETSI NFV MANO framework [7]

interacts with the VNF instances, VIMs and NFVO over the set of reference points referred to respectively as *Ve-Vnfm*, *Vi-Vnfm* and *Or-Vnfm*, as shown in Figure 3.1. The communication between the VNFM on the one hand, and these functional blocks, on the other hand, may take place over WAN links, depending on their locations in the NFVI. The VNFM location plays a vital role in determining the delay over the VNFMs reference points. Consequently, an unplanned placement of VNFMs can lead to intolerable delay, which negatively affects the performance and reliability of the system. To guarantee the performance, each reference point can be bound by a delay limit. The latter depends on the VNF instance, the reference point or other factors. As an example, for one specific VNF instance, the reference point *Or-Vnfm*, between the VNFM and the NFVO, can be bound by a delay limit that differs from that of the reference point *Ve-Vnfm*.

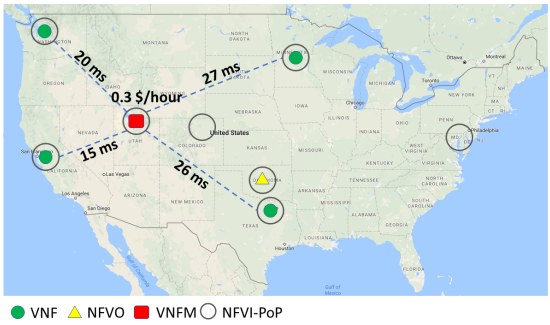
We illustrate in Figures 3.2(a), (b) and (c) the need for adequate planning of VNFM placement in the system according to its state. There, we explore different placement options over an NFVI that consists of 8 PoPs distributed across the USA. In this illustrative example, we consider a set of 4 VNF instances are deployed over the NFVI and need to be managed. We consider that the location of the NFVO is already given. We assume a VNFM can manage 10 VNF instances at a time. We also assume the reference point *Ve-Vnfm* is



(a) Violate delay bound over *Ve-Vnfm*



(b) Satisfy constraints with a high resource cost



(c) Satisfy constraints with a low resource cost

Figure 3.2: Illustration of the motivation for the VNFM placement problem

bound by a round-trip delay limit of 35 ms. In each figure, we highlight the round-trip delay over the communication links, as well as the cost of resources where a VNFM is placed.

In Figure 3.2(a), we cover the case of placing a VNFM over a PoP over the east coast of the USA. As shown, although this placement satisfies the capacity limit of a VNFM, it leads to round-trip delay that violates the bound over the reference point *Ve-Vnfm* (this is the case for the VNF instances located on the west coast of the USA). This example shows that VNFM placement is critical from the system performance perspective and needs to be planned cautiously.

In Figures 3.2(b) and (c), we show instead two scenarios where the VNFM is placed over a PoP that allows satisfying the delay bound over the *Ve-Vnfm* reference point. However, the placement in Figure 3.2(c) implies a lower cost with respect to that in Figure 3.2(b), due to the differences in resource cost over the corresponding PoPs. Thus, an adequate placement of VNFMs can further reduce the overall cost in the system.

The VNFM placement is therefore critical to the performance of the entire NFV system as well as the cost. We refer to the corresponding optimization problem as the VNFM Placement Problem (MPP). Formally, we define the problem as follows: given the location of the VNF instances and a fixed location of the NFVO, the goal is to find (1) the optimal number of VNFMs required to manage the VNF instances, (2) their types (e.g., generic VNFM, VNFM for managing VNFs from a specific provider, etc.), (3) the placement of VNFMs over distributed PoPs, and (4) the associations they hold with VNF instances. We aim at doing so at a minimum operational cost while satisfying delay (e.g., over communication links) and capacity constraints (e.g., of VNFMs) in the system. From an operations research perspective, our problem can be mapped to the Facility Location Problem (FLP). The FLP has received significant attention in the operations research community [63].

The MPP includes both static and dynamic versions. In the static MPP, the VNFM placement and association with VNF instances are permanent and do not change with time. This is applicable in scenarios where the changes in the system (e.g., number of VNF instances) are insignificant to readjust the VNFM placement. The static MPP can still be applied even if there are changes in the system. However, this requires an estimation of the maximum number of VNF instances that can exist in the system and their location, which can be used to derive the VNFM placement decisions. However, this scenario can lead to over-provisioning in the number of VNFMs. In contrast, the dynamic MPP seeks to adapt the VNFM placement to the changes in the system. The number of VNF instances, their locations and network conditions can vary over time and the dynamic MPP aims to attain potential gain by readjusting the VNFM placement.

3.2.2 System Model

The design of ETSI NFV framework allows a plurality of implementation and deployment models to emerge. Therefore, for the purpose of simplification, the following three assumptions are made: (1) the NFVO and VNFM are implemented as distinct components, (2) a VNF instance and its EM are deployed at the same PoP, and (3) a VIM manages the resources within one PoP.

Our model operates over a set of snapshots. We define a snapshot t as a representation of the system state over a fixed time interval. The system state (e.g., number of VNF instances) may vary from one snapshot to another. Therefore, when the system transits from one snapshot to the next, our model considers four mechanisms to adapt the system in response to changes: (1) add new VNFM(s) to cope with the increment of the VNF instances, (2) remove existing VNFM(s) to reduce the cost, (3) migrate existing VNFM(s) to new location(s), and (4) reassign VNF instance(s) to another VNFM(s).

In the following, we present our system model covering different entities in the system, as well as the network traffic. Table 3.1 presents a description of the inputs and decision variables used in our problem.

1. NFVI: We represent the NFVI using a graph structure $G = (P, E)$. There, P is a set of nodes, with each node p representing a PoP and E is a set of edges linking them. An edge $(p, q) \in E$ linking a couple of PoPs p and q represents a logical communication link between them. We employ $\gamma_{p,q}(t)$ and $\delta_{p,q}(t)$ to represent capacity and delay of edge $(p, q) \in E$, in snapshot t, respectively. We use $c_p^{com}(t)$ and $c_{p,q}^{net}(t)$ to denote the cost of one unit of compute resource at $p \in P$ and one unit of network bandwidth over the edge $(p, q) \in E$, in snapshot t, respectively.

2. NFVO: We assume that the NFVO is deployed at a given PoP. We use $h_p \in \{0, 1\}$ to refer to its location, such that h_p is equal to 1 if the NFVO is placed at $p \in P$, and 0 otherwise.

3. VNFM: We define $M(t)$ as the set of VNFMs m that can be used in snapshot t. We consider that different VNFM types (e.g., generic VNFM, specific VNFM for managing firewall VNF, etc.) exist and define K as the set of VNFM types. n_k denotes the capacity of a VNFM m of type $k \in K$. It represents the maximum number of VNF instances that can be managed by a single VNFM. We define $M_k(t) \in M(t)$ as the set of VNFMs of type $k \in K$ that can be used over snapshot t. $\widehat{M}_k(t)$ represents the set of VNFMs of type $k \in K$, selected from $M_k(t)$ to be used over snapshot t, e.g., active VNFMs. Further, we employ $g_{m,k}$ to refer to the bandwidth consumed in migrating the VNFM m of type k, over the

edge between the previous PoP and the new one. We also denote the cost of migrating the VNFM m of type $k \in K$ by $c^{mig}_{m,k}(t)$; it represents the penalty for the service disruption caused by the migration.

4. VNF: $V(t)$ is the set of VNF instances v to manage over snapshot t. $V_k(t)$ is a subset of $V(t)$ that includes all VNF instances that require VNFM of type $k \in K$ to manage them. We use $l_{v,p}(t) \in \{0, 1\}$ to identify the location of a VNF instance, such that $l_{v,p}(t)$ is equal to 1 if the VNF instance v is placed at $p \in P$ over snapshot t, otherwise 0. Further, the communication overhead, introduced in managing the lifecycle of the VNF instance v, is controlled through two delay limits. The first limit is $\varphi_v(t)$ which represents the maximum permissible delay between the VNF instance v and the VNFM managing it over snapshot t. Due to the assumptions (2) and (3), $\varphi_v(t)$ is also considered the maximum delay between the VNFM on the one hand, and the VIM and EM on the other, for the VNF instance v over snapshot t. The second limit, $\omega_v(t)$, refers to the maximum permissible delay between the NFVO and the VNFM managing the VNF instance v over snapshot t. We use s_v and $\widehat{s}_v$ to denote the bandwidth consumed in the reassignment of the VNF instance v between the NFVO and old VNFM as well as the NFVO and new VNFM, respectively. We also assume that $c^{rea}_v(t)$ refers to the reassignment cost of the VNF instance v; it represents the penalty paid for reconfiguring the system to ensure its stability.

5. Network Traffic: NFV MANO functional blocks interact with each other and with other non-MANO functional blocks (e.g., EM) to manage the lifecycle of the VNF instances. Herein, for the VNF instance v, we assume that $u^{O,M}_v(t)$, $u^{O,I}_v(t)$, $u^{M,I}_v(t)$, $u^{M,V}_v(t)$ represent the units of bandwidth consumed during communications between the NFVO and VNFM, NFVO and VIM, VNFM and VIM, VNFM and VNF instance v over reference points *Or-Vnfm*, *Or-Vi*, *Vi-Vnfm* and *Ve-Vnfm*, respectively, over snapshot t.

3.2.3 Problem Formulation

We formulate the MPP as an ILP problem, where we aim at deriving decisions over individual snapshots. For the static MPP, we operate only over a single snapshot t, representing

Table 3.1: Notations Description

Inputs	
$G(P, E)$	NFVI G with PoPs P and edges linking them E
E	The set of edges (i.e., logical communication links) in the network, $E = \{(p, q) \mid p \in P, q \in P, p \neq q\}$
$\gamma_{p,q}(t)$	Capacity of edge $(p, q) \in E$ in snapshot t
$\delta_{p,q}(t)$	Delay of edge $(p, q) \in E$ in snapshot t
$c_p^{com}(t)$	Cost of compute resource at $p \in P$ in snapshot t
$c_{p,q}^{net}(t)$	Cost of network bandwidth over the edge $(p, q) \in E$ in snapshot t
$h_p \in \{0, 1\}$	$h_p = 1$ if NVFO is placed at $p \in P$
K	Set of VNFM types
$M(t)$	Set of VNFMs in snapshot t
$M_k(t)$	Set of VNFMs of type $k \in K$ in snapshot t
$\widehat{M}_k(t)$	Set of active VNFMs of type $k \in K$ in snapshot t
$g_{m,k}$	Bandwidth consumed in migrating the VNFM m of type $k \in K$
$c_{m,k}^{mig}(t)$	Penalty for the migration of VNFM m of type $k \in K$ in snapshot t
n_k	Capacity of a VNFM of type k
$V(t)$	Set of VNF instances in snapshot t
$V_k(t)$	Set of VNF instances that require VNFM of type $k \in K$ in snapshot t
$l_{v,p}(t) \in \{0, 1\}$	$l_{v,p}(t) = 1$ if VNF instance v is placed at $p \in P$ in snapshot t
$\varphi_v(t)$	Maximum delay between VNF instance v and VNFM over snapshot t
$\omega_v(t)$	Maximum delay between NFVO and VNFM managing VNF instance v over snapshot t
$s_v, \widehat{s}_v$	Bandwidth used in the reassignment of VNF instance v between NFVO and old VNFM, NFVO and new VNFM
$c_v^{rea}(t)$	Reassignment cost for VNF instance v in snapshot t
$u_v^{O,M}(t)$	Bandwidth used between NFVO and VNFM regarding VNF v in snapshot t
$u_v^{O,I}(t)$	Bandwidth used between NFVO and VIM regarding VNF v in snapshot t
$u_v^{M,I}(t)$	Bandwidth used between VNFM and VIM regarding VNF v in snapshot t
$u_v^{M,V}(t)$	Bandwidth used between VNFM and VNF v in snapshot t
Decision Variables	
$x_{m,k,p}(t) \in \{0, 1\}$	$x_{m,k,p}(t) = 1$ f $m \in M_k(t)$ is placed at $p \in P$ in snapshot t
$y_{v,m,k,p}(t) \in \{0, 1\}$	$y_{v,m,k,p}(t) = 1$ if $v \in V_k(t)$ is assigned to $m \in M_k(t)$ that is placed at $p \in P$

the permanent state of the system. We derive over it the permanent configurations of the system. For the dynamic MPP, we operate over each couple of consecutive snapshots in the system $(t-1)$ and t. More precisely, given snapshots $(t-1)$ and t, at the end of the snapshot $(t-1)$, we solve the problem to decide the VNFM placement along with the associations they hold with VNF instances over snapshot t. The placement decisions made for all VNFMs in $M(t)$ allow determining whether to add new VNFMs, as well as to keep, remove, or migrate existing VNFMs. We define $M(t)$ as follows.

$$M_k(t) = F_k(t) \cup \widehat{M}_k(t-1) \tag{3.1}$$

where $F_k(t)$ is the set of new VNFMs m of type k that can be added to the system to manage the VNF instances over snapshot t, such that:

$$|F_k(t)| = \sum_{p \in P} \left\lceil \frac{\sum_{v \in V_k(t)} l_{v,p}(t)}{n_k} \right\rceil \tag{3.2}$$

By that, $|F_k(t)|$ represents an upper bound on the number of VNFMs m of type k that can be added to the system over snapshot t. $\widehat{M}_k(t-1)$ is the set of active VNFMs of type k used to manage the VNF instances over snapshot $(t-1)$.

Our decision variables are the following.

$$x_{m,k,p}(t) = \begin{cases} 1, & \text{if } m \in M_k(t) \text{ is placed at } p \in P, \\ 0, & \text{otherwise.} \end{cases}$$

$$y_{v,m,k,p}(t) = \begin{cases} 1, & \text{if } v \in V_k(t) \text{ is assigned to } m \in M_k(t) \text{ that is placed at } p \in P, \\ 0, & \text{otherwise.} \end{cases}$$

Operational Cost: Four different cost components contribute to our definition of the operational cost, defined as follows.

1. Lifecycle Management Cost $(C^{lif}(t))$: The lifecycle management cost represents the cost of the network bandwidth consumed in the communication performed through the

lifecycle management of all VNF instances in the system over snapshot t.

$$C^{lif}(t) = \sum_{(p,q)\in E} B^{lif}(p,q)\, c^{net}_{p,q}(t) \tag{3.3}$$

where

$$B^{lif}(p,q) = \sum_{k\in K} \sum_{v\in V_k(t)} \sum_{m\in M_k(t)} \Big\{ y_{v,m,k,p}(t)\, h_q\, u^{O,M}_v(t) + y_{v,m,k,p}(t)\, l_{v,q}(t) \Big(u^{M,V}_v(t) + u^{M,I}_v(t) \Big) + l_{v,p}(t)\, h_q\, u^{O,I}_v(t) \Big\}$$

2. Compute Resources Cost ($C^{com}(t)$): The compute resources cost represents the cost of compute resources assigned to VNFMs over snapshot t.

$$C^{com}(t) = \sum_{k\in K} \sum_{m\in M_k(t)} \sum_{p\in P} x_{m,k,p}(t)\, c^{com}_p(t) \tag{3.4}$$

With that, we assume that a VNFM requires a single unit of compute resource. This assumption is due to the lack of available data on resource allocation for NFV MANO functional blocks.

3. Migration Cost ($C^{mig}(t)$): It represents the cost implied by migrating a VNFM from one PoP to another while switching from snapshot $(t-1)$ to snapshot t. It concerns only the VNFMs that were placed over snapshot $(t-1)$, e.g., in $\widehat{M}_k(t-1)$.

$$C^{mig}(t) = \sum_{k\in K} \sum_{m\in \widehat{M}_k(t-1)} \sum_{(p,q)\in E} x_{m,k,p}(t)\, x_{m,k,q}(t-1)\, c^{mig}_{m,k}(t) \tag{3.5}$$

4. Reassignment Cost ($C^{rea}(t)$): While switching from snapshot $(t-1)$ to snapshot t, VNF instances that remain in the system may be reassigned to new VNFMs. We compute the cost of reassigning these VNF instances as follows.

$$C^{rea}(t) = \sum_{k\in K} \sum_{v\in V_k(t)\cap V_k(t-1)} \sum_{m\in M_k(t)} \sum_{p\in P} y_{v,m,k,p}(t) \Big(1 - \sum_{q\in P} z_{v,m,k,q}(t) \Big) c^{rea}_v(t) \tag{3.6}$$

where

$$z_{v,m,k,p}(t) = \begin{cases} y_{v,m,k,p}(t-1), & \text{if } v \in V_k(t) \cap V_k(t-1) \text{ and } m \in \widehat{M}_k(t-1), \\ 0, & \text{otherwise.} \end{cases}$$

The objective of our optimization problem is to minimize the weighted sum of the aforementioned four costs and can be expressed as follows.

$$\textit{Minimize} \quad \varepsilon\, C^{lif}(t) + \theta\, C^{com}(t) + \mu\, C^{rea}(t) + \rho\, C^{mig}(t) \tag{3.7}$$

We note that in the case of static MPP, as we operate only over a single snapshot t, the objective function in equation (3.7) does not include the terms $C^{rea}(t)$ and $C^{mig}(t)$.

Constraints: Each VNF instance should be assigned to one VNFM, as indicated in constraint (3.8).

$$\sum_{m \in M_k(t)} \sum_{p \in P} y_{v,m,k,p}(t) = 1 \qquad \forall k \in K, v \in V_k(t) \tag{3.8}$$

Equation (3.9) stipulates that a VNF instance cannot be assigned to a VNFM at PoP p unless that VNFM is placed at that location.

$$y_{v,m,k,p}(t) \leq x_{m,k,p}(t) \qquad \forall k \in K, v \in V_k(t), m \in M_k(t), p \in P \tag{3.9}$$

In constraint (3.10), we ensure that the number of VNF instances assigned to each VNFM does not exceed its capacity.

$$\sum_{v \in V_k(t)} y_{v,m,k,p}(t) \leq n_k \qquad \forall k \in K, m \in M_k(t), p \in P \tag{3.10}$$

A VNFM can be located only at one PoP. This constraint is defined by (3.11).

$$\sum_{p \in P} x_{m,k,p}(t) \leq 1 \qquad \forall k \in K, m \in M_k(t) \tag{3.11}$$

Equation (3.12) ensures that a VNFM is active only when it manages at least one VNF instance.

$$x_{m,k,p}(t) \leq \sum_{v \in V_k(t)} y_{v,m,k,p}(t) \qquad \forall k \in K, m \in M_k(t), p \in P \tag{3.12}$$

Each VNF instance has two delay limits to control the delay over the reference points of its assigned VNFM. We enforce these constraints by (3.13) and (3.14).

$$y_{v,m,k,p}(t)\, l_{v,q}\, \delta_{p,q}(t) \leq \varphi_v(t) \qquad \forall k \in K, v \in V_k(t), m \in M_k(t), (p,q) \in E \tag{3.13}$$

$$y_{v,m,k,p}(t)\, h_q\, \delta_{p,q}(t) \leq \omega_v(t) \qquad \forall k \in K, v \in V_k(t), m \in M_k(t), (p,q) \in E \tag{3.14}$$

Constraint (3.15) guarantees that the utilized bandwidth on each edge does not exceed its capacity.

$$B^{lif}(p,q) + B^{mig}(p,q) + B^{rea}(p,q) \leq \gamma_{p,q}(t) \qquad \forall (p,q) \in E \tag{3.15}$$

where

$$B^{mig}(p,q) = \sum_{k \in K} \sum_{m \in \widehat{M}_k(t-1)} x_{m,k,p}(t)\, x_{m,k,q}(t-1)\, g_{m,k}$$

$$B^{rea}(p,q) = \sum_{k \in K} \sum_{v \in V_k(t) \cap V_k(t-1)} \sum_{m \in M_k(t)} \Big\{ z_{v,m,k,p}(t)\, h_q \Big(1 - \sum_{\acute{p} \in P} y_{v,m,k,\acute{p}}(t)\Big) \widehat{s}_v + y_{v,m,k,p}(t)\, h_q \Big(1 - \sum_{\acute{p} \in P} z_{v,m,k,\acute{p}}(t)\Big) s_v \Big\}$$

We note that in the case of static MPP, as we operate only over a single snapshot t, constraint (3.15) does not include the terms B^{mig} and B^{rea}.

3.3 Resolution Approach

This section presents the approach that we propose to solve the MPP over each snapshot. We recall that MPP can be mapped to FLP [63]. Exact, approximation and heuristic algorithms have been employed to solve FLP. However, FLP is known to be NP-hard. As a result, in large-scale scenarios, deriving optimal solutions becomes unfeasible and mainly

Algorithm 3.1: Tabu Search Heuristic

```
1  S_0(t) ⟵ GreedyInitialSolution
2  S_current(t) ⟵ S_0(t), S_best(t) ⟵ S_0(t), j ⟵ 0
3  repeat
4      moves-list ⟵ create candidate moves list
5      best-move  ⟵ choose from moves-list the move that generates the best solution
6      apply best-move on S_current(t) solution
7      add best-move to tabu list for i_tabu iterations
8      j ⟵ j + 1
9      if f(S_current(t)) < f(S_best(t)) then
10         S_best(t) ⟵ S_current(t)
11         j ⟵ 0
12     end
13 until j = i_stop
14 return S_best(t)
```

heuristic and metaheuristic algorithms have been designed. While heuristics are typically tightly linked to the specific problem context, metaheuristics rely on general techniques that can be employed in different scenarios. There, a variety of techniques has been covered [64, 65], including simulated annealing, genetic algorithm and tabu search algorithm. In [64], the performance of the various metaheuristic techniques has been assessed for several instances of the FLP problem. The results underline the fact that genetic algorithms and tabu search techniques provide superior results with respect to other techniques. Therefore, we rely on tabu search [66] to solve our MPP problem.

Tabu search [66] is a metaheuristic designed for guiding a local search procedure to find a near optimal solution of combinatorial optimization problems. It starts exploring the search space from an initial solution and iteratively performs moves to transit from the current solution to another one in its neighborhood until the termination criterion is satisfied. Next, we present the different components of the proposed tabu search heuristic, outlined in Algorithm 3.1.

3.3.1 Initial Solution

We employ a simple greedy heuristic to obtain the initial solution $S_0(t)$ for snapshot t. The heuristic has two main steps, outlined in Algorithm 3.2. The first step is executed from the second snapshot in the system onwards, e.g., when there is a snapshot $(t-1)$. In this step, the heuristic implements the decisions made in the snapshot $(t-1)$. Hence,

Algorithm 3.2: Greedy Initial Solution Heuristic

```
1  ∀k ∈ K: M̂_k(t) ⟵ ∅
   /* Step One                                                          */
2  if M̂(t−1) ≠ ∅ then
3      foreach k ∈ K, m ∈ M̂_k(t−1), p ∈ P do
4          x_{m,k,p}(t) ⟵ x_{m,k,p}(t−1)
5          if x_{m,k,p}(t) = 1 then
6              M̂_k(t) = M̂_k(t) ∪ {m}
7          end
8      end
9      foreach k ∈ K, v ∈ V_k(t) ∩ V_k(t−1), m ∈ M̂_k(t−1), p ∈ P do
10         y_{v,m,k,p}(t) ⟵ y_{v,m,k,p}(t−1)
11     end
12 end
   /* Step Two                                                          */
13 foreach k ∈ K, v ∈ V_k(t) \ V_k(t−1) do
14     q ⟵ p ∈ P | l_{v,p} = 1
15     if ∃ m ∈ M̂_k(t) | x_{m,k,q}(t) = 1 and m has capacity then
16         y_{v,m,k,q}(t) ⟵ 1
17     end
18     else
19         choose first VNFM m from M_k(t) \ M̂_k(t)
20         M̂_k(t) ⟵ M̂_k(t) ∪ {m}
21         x_{m,k,q}(t) ⟵ 1, y_{v,m,k,q}(t) ⟵ 1
22     end
23 end
24 return {x_{m,k,p}(t), y_{v,m,k,p}(t)}
```

it places each active VNFM $m \in \widehat{M}_k(t-1), \forall k \in K$, at the same PoP and assigns each $v \in V_k(t) \cap V_k(t-1), \forall k \in K$, to the same VNFM. It is very challenging to determine the best mechanism (e.g., VNFM migration and VNF reassignment) to readjust the current system configurations. Thus, the goal of this step is to maintain current configurations in the initial solution and delegate the decision-making responsibility to tabu search heuristic, which can evaluate all possible alternatives and make the decisions.

In the second step, the heuristic starts assigning every new VNF instance $v \in V_k(t) \setminus V_k(t-1), \forall k \in K$, to an active VNFM that has enough capacity and is located at the same PoP as the VNF instance. We use the total delay between the VNFMs and their corresponding VNF instances in our objective function, as will be discussed in section 3.3.4. Therefore, the condition of having the VNFM and its assigned VNF instance at the same location is imposed to minimize the total delay in the initial solution and subsequently minimize the tabu search iterations. If none of the active VNFMs meets the aforementioned

conditions, a new VNFM $m \in M_k(t) \setminus \widehat{M}_k(t)$ is activated and placed at the PoP where the VNF instance is located. Then, the VNF instance is assigned to the that VNFM.

Further, every VNFM $m \in M_k(t) \setminus \widehat{M}_k(t), \forall k \in K$, does not have any assigned VNF instance in the initial solution, is considered as an inactive VNFM that can be activated later by the tabu search heuristic. The generated initial solution ensures that all constraints are satisfied, except (3.14) and (3.15), which may be violated. The proposed tabu search heuristic would substantially improve the solution quality and ensure satisfying any of the violated constraints.

3.3.2 Neighborhood Structure

Our tabu search heuristic employs four move types to generate a neighborhood solution, defined as follows.

1. VNF Reassignment: A VNF instance is selected randomly and reassigned to another VNFM. The new VNFM may be active or inactive. If it is the latter, the VNFM is activated. The old VNFM is deactivated if it has no more VNF instances associated with it.

2. VNFM Relocation: An active VNFM is selected randomly and moved to another PoP. The new location is chosen such that the delay constraints for all assigned VNF instances are not violated.

3. Bulk VNF Reassignment: It is a composite move that consists of a finite number of "VNF reassignment" moves. To that extent, an active VNFM is drawn randomly. Then, if the remaining active VNFMs have enough capacity to manage its assigned VNF instances while satisfying their delay constraints, the VNF instances are reassigned, and the chosen VNFM is deactivated.

4. VNFM Deactivation: This is another composite move that includes a finite number of "VNF reassignment" and "VNFM relocation" moves. Let $P_m(t) \subseteq P$ encompass all PoPs, where an active VNFM $m \in \widehat{M}(t)$ can be placed while satisfying the delay constraints of its corresponding VNF instances. Then, two VNFMs m and $\widehat{m}$ have overlapping coverage

if $\{P_{m,\widehat{m}}(t) = P_m(t) \cap P_{\widehat{m}}(t)\} \neq \emptyset$. To that extent, two VNFMs m and $\widehat{m}$ are selected randomly and overlapping coverage is assessed. If the condition holds and the VNFM m has the enough capacity to serve the VNF instances assigned to $\widehat{m}$, then these VNF instances are reassigned to m and the VNFM $\widehat{m}$ is deactivated. If the VNFM m is not already placed at $p \in P_{m,\widehat{m}}(t)$, then it is relocated to a PoP that belongs to that set, e.g., $P_{m,\widehat{m}}(t)$.

3.3.3 Tabu List and Aspiration Criterion

Tabu search uses memory structure, called tabu list, to record information about the recent history of the search. The search uses this information to avoid the local optimums and prevent the cycling to previously visited solutions. In our heuristic, tabu list records the moves that have been made in the recent past and forbids them as long as they are on the list. These moves are known as tabu moves. They stay on the tabu list for a certain number of iterations (i_{tabu}). We set this number to a constant value that is equal to 300. The number is small compared to the number of VNF instances (e.g., $|V(t)|$) in large-scale deployments. However, it is big enough to prevent the cycling problem. Further, a tabu move can be selected and implemented if it meets a condition known as the aspiration criterion. We define it as releasing a move from its tabu status and accepting it if that move produces a solution better than the best solution found so far.

3.3.4 Acceptance Criterion

In each iteration, the tabu search heuristic evaluates a set of candidate moves and selects the best move, e.g., the move that generates the best neighbor solution. The heuristic uses a hierarchical objective function (f) to evaluate the neighbor solutions, where the primary objective is minimized first and then, for the same primary objective value, the secondary objective is minimized. The primary objective is defined as the sum of the model objective function defined by (3.7) and the total penalty associated with the solution. The proposed heuristic penalizes the violation of the constraints (3.10), (3.13), (3.14) and (3.15). The solution is assigned a penalty proportional to the level of the violation. The secondary objective is the sum of the delay between the active VNFMs and their associated VNF

instances. The rationale for that is to reward the move that reassigns a VNF instance to a closer VNFM. The ultimate goal is to maximize $P_m(t)$ for all active VNFMs, then detect and eliminate any overlapping coverage (if possible).

3.3.5 Termination Criterion

The heuristic stops when the best solution found does not improve for a certain number of consecutive iterations (i_{stop}). This number is defined as $25\sqrt{|V(t)|} + |M(t)|$. This formula is designed to allow the number to grow with respect to $V(t)$ and $M(t)$, but to a lesser degree than a linear function of $V(t)$ in the large-scale deployments. The multiplier 25 is adjusted experimentally to make the trade-off between the execution time and the final solution quality.

3.4 Evaluation Scenarios

We perform several experiments considering both static and dynamic MPP. The experiments of static MPP are designed to (1) evaluate the proposed tabu search heuristic in terms of the solution quality and execution time, (2) study the impact of the optimization objective weight, and (3) investigate the effect of NFVO location and the architectural options related to VNFM (generic vs. VNF-specific). On the other hand, the experiments related to the dynamic MPP aim to study the gain that could potentially be achieved by reconfiguring the system. Next, we first present the simulation setup, followed by the description of the experiments for the static and dynamic MPP.

3.4.1 Simulation Setup

1. NFVI: We use the network of WonderNetwork [67] to build the NFVI used in our experiments. WonderNetwork is a networking solution provider that operates a network of servers distributed over the globe. It provides real-time hourly delay information between each pair of locations. We consider each location in this network as a potential PoP. The hourly cost of the compute resource at each PoP ($c_p^{com}(t)$) is set to the electricity price for

its location. The price ranges within (0.0833 to 0.1776) \$. In addition, the average delay is used as edge delay ($\delta_{p,q}(t)$) between each pair of PoPs. The bandwidth cost associated with all edges ($c^{net}_{p,q}(t)$) is linear with the traffic volume and equals to \$0.155/GB. The capacity of all edges ($\gamma_{p,q}(t)$) is set to 10 Gb/s.

2. VNFM: The live migration of a Virtual Machine (VM) involves transferring the VM running state (e.g., memory) and the virtual disk [68]. We assume that the VM hosting a VNFM is a medium size with 2 CPUs, 40 GB disk and 4 GB memory. Then, for $m \in M_k(t), \forall k \in K$, the bandwidth consumed in migrating the VNFM ($g_{m,k}$) is set to the sum of the disk size and memory size, e.g., 44 GB. The migration cost ($c^{mig}_{m,k}(t)$) is computed as the cost of the bandwidth consumed in the migration which is equal to $44 \times \$0.155$. The VNFM capacity ($n_k$) is set to 80.

3. VNF: We assume that the VNF instances managed by the MANO fall into two classes: class one (C1) and class two (C2). Class one contains complex and high-throughput transactional VNFs such as the Serving Call Session Control Function (S-CSCF) and the Policy and Charging Rules Function (PCRF). These VNFs have stringent reliability and performance requirements since the performance anomaly (e.g., VNF failure and performance degradation) has a significant impact on thousands of users. On the other hand, class two includes simple VNFs such as the firewall that may be used in a residential virtual Customer Premises Equipment (vCPE). In this case, VNF performance anomaly affects small group of users. Hence, these VNFs have relaxed requirements compared to C1. In accordance with that, we set the delay limits on the links, e.g., $\varphi_v(t)$ and $\omega_v(t)$, to be smaller in the case of C1 compared to C2. However, for both C1 and C2, $\varphi_v(t)$ is set smaller than $\omega_v(t)$ to place the VNFM close to the VNF instance. Table 3.2 shows the selected delay limits. Moreover, and for simplicity, s_v and $\widehat{s}_v$ (bandwidth consumed in VNF instance reassignment) are assumed equal and are set to 2 MB. The VNF reassignment cost $c^{rea}_v(t)$ is considered the cost of the bandwidth consumed in the reassignment and computed as $\frac{(2+2)\times\$0.155}{1024}$.

Table 3.2: Experiment Parameters

Parameter	VNF C1	VNF C2
Number of VNF indicators	30	15
Number of resource performance metrics	60	20
Maximum delay between VNFM and VNF instance, EM, VIM ($\varphi_v(t)$)	35 ms	60 ms
Maximum delay between NFVO and VNFM ($\omega_v(t)$)	70 ms	100 ms
VNFM data collection period	30 s	60 s
NFVO data collection period	180 s	360 s
Bandwidth used between NFVO and VNFM ($u_v^{O,M}(t)$)	0.51 MB/h	0.11 MB/h
Bandwidth used between NFVO and VIM ($u_v^{O,I}(t)$)	0 MB/h	0 MB/h
Bandwidth used between VNFM and VIM ($u_v^{M,I}(t)$)	1.26 MB/h	0.33 MB/h
Bandwidth used between VNFM and VNF/EM ($u_v^{M,V}(t)$)	1.77 MB/h	0.31 MB/h

4. Network Traffic: Traffic between different functional blocks in NFV architectural framework vary widely depending on many factors including, but not limited to, communication protocols, VNFs, implementation, and configurations. ETSI specifications [11, 14–17] define the interfaces, operations and information model supported by the reference points. However, the details of the operations and the communication protocols are not discussed. Herein, we assume that the various functional blocks communicate through RESTful HTTP interfaces. The data exchange format is JSON and the total header size for an HTTP request-response is set to 500 bytes. Besides, we use network traffic produced by monitoring of VNF instances in our evaluation experiments. Monitoring is an essential function in the lifecycle management that involves collecting data related to the VNF instances to analyze it and ensure they meet the desired requirements. In this context, we assume that the VNFM periodically collects the performance metrics of VNF resources from VIM [16] and the VNF indicators information from the VNF instance or EM [11]. An indicator is application-level information that provides insight into the VNF behavior [11]. We further consider that the NFVO periodically collects this information from the VNFM

to analyze it. Nevertheless, we assume that the collection period, which specifies the periodicity at which data is collected [11, 16], is shorter for the VNFM compared to the NFVO. Because of C1 VNFs requirements, we consider that the NFVO and VNFM collection periods are shorter for C1 compared to C2. We also assume that a VNF in C1 has more resource performance metrics and VNF indicators compared to C2. Further, based on the ETSI information model in [11, 16], we estimate the data size of one performance metric and one VNF indicator in JSON format to be 350 and 250 bytes, respectively. Table 3.2 presents the data collection periods and the network traffic information between different functional blocks.

3.4.2 Static MPP Experiments

In all experiments, the snapshot duration is set to one hour. Both small-scale and large-scale deployments are covered. Three different sizes of NFVI are considered in the small-scale deployments: 8, 16 and 24 PoPs. In the large-scale deployments, the NFVI is made of 64 PoPs. All PoPs in both cases are distributed across the USA, as shown in Figure 3.3. Two different PoPs are used to host the NFVO (i.e., h_p). One PoP is located in Dallas with a central location in the NFVI structure while the other is located in San Jose; which is considered an edge PoP in the NFVI structure. Moreover, two architectural options related to VNFM are considered. The first is to use VNF-specific VNFM in the system so that a distinct VNFM type manages each VNF class. The other option considers that a generic VNFM can manage both classes. The number of the VNF instances ranges from 100 to 500 in the small-scale deployments, and from 1000 to 5000 in the large-scale ones. Each VNF instance is chosen uniformly at random from the two VNF classes and placed at a PoP selected uniformly at random.

3.4.3 Dynamic MPP Experiments

In the dynamic MPP experiments, we assume that the demand is temporally distributed in three equal time slot periods during the day: morning, afternoon and night. Each time slot period represents a system snapshot of 8 hours long. We note that the choice of

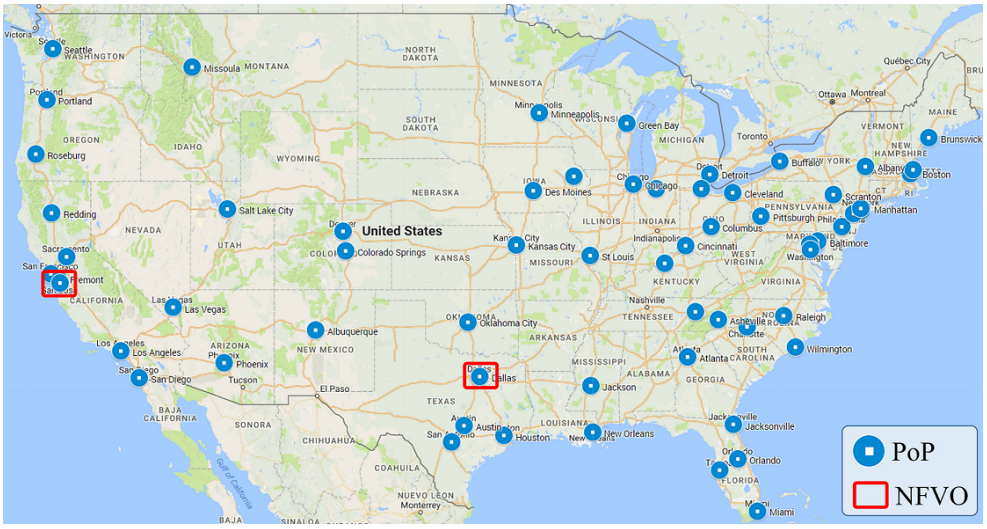


Figure 3.3: 64 PoPs distributed across the USA

the snapshot's duration has an impact on the placement decisions and the overall cost in the system over time. However, we select the period of 8 hours to capture major human dynamics. In the morning time slot, there is a total of 300 VNF instances (225 instances of C1 class and 75 instances of C2 class). At the beginning of the afternoon time slot, the VNF instances of class C1 are scaled in to 75 instances, and VNF instances of class C2 are scaled out to 225 instances. Lastly, the demand falls at the beginning of night time slot, so the system scales in the VNF instances of C1 and C2 classes to 50, i.e., total of 100 VNF instances. Further, the NFVI consists of 16 PoPs located on the east coast of the USA (e.g., same time zone). VNF instances are placed randomly at the PoPs.

3.5 Numerical Results

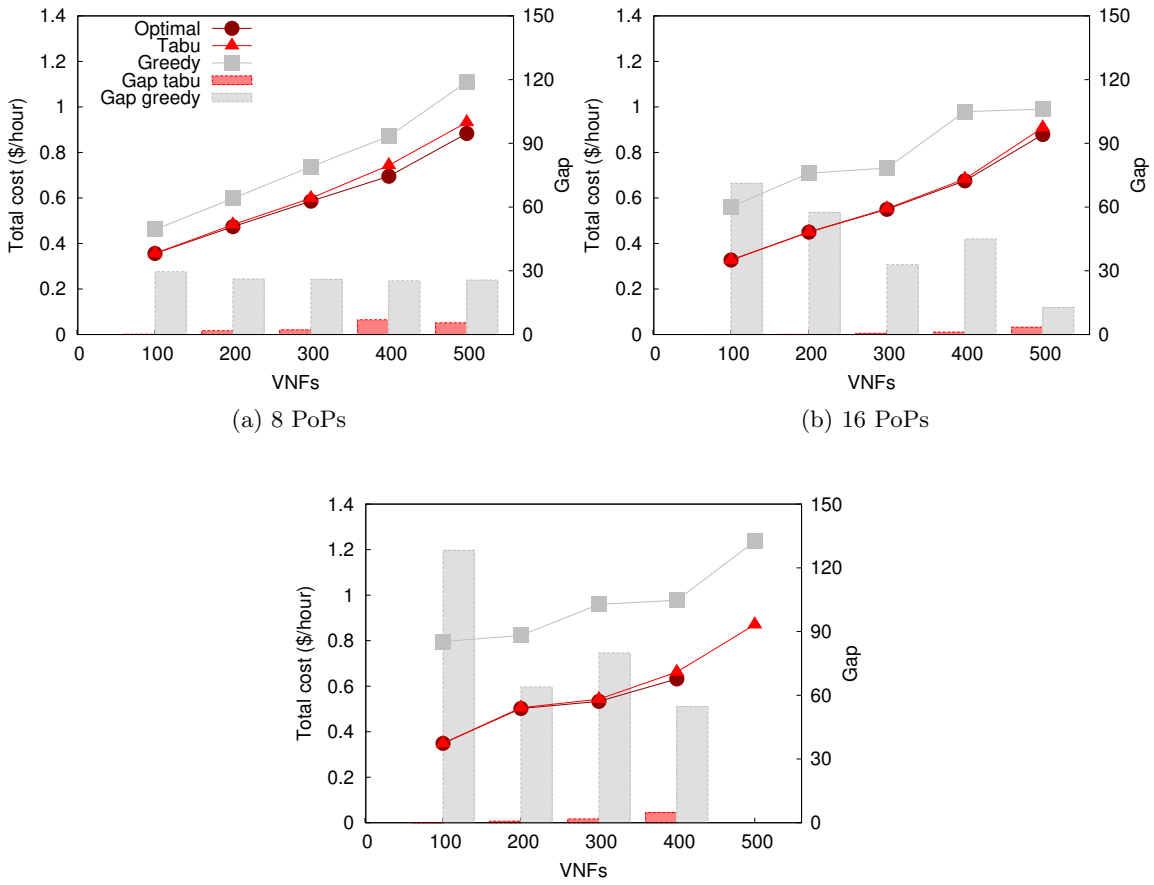
In this section, we evaluate the performance of the tabu search heuristic, by covering various aspects, as discussed in section 3.4. We compare the obtained results to those derived based on exact and first-fit greedy heuristic. The tabu search heuristic is implemented in JAVA. The ILP model in section 3.2 is implemented and solved in CPLEX 12.6.3 [69]. We exploit the first-fit greedy heuristic to assess the impact of simple VNFM placement on the operational cost. We also use it as a baseline to evaluate the performance of tabu search

heuristic when the optimal solution cannot be obtained, e.g., for the large-scale scenario. Herein, the first-fit greedy heuristic iterates over the set of VNF instances. At a specific iteration, it assigns the VNF instance to the first VNFM with adequate available capacity. Otherwise, a new VNFM is placed at the first PoP that satisfies the MPP constraints. Then, the VNF instance is assigned to that VNFM. Moreover, in all experiments, unless mentioned otherwise, all cost weights are set to 1 and a distinct VNFM type is used for each of the C1 and C2 VNF classes, e.g., VNF-specific VNFM.

3.5.1 Heuristic Performance Evaluation

We start by assessing the performance of the tabu search heuristic by evaluating the quality of the derived solution against the optimal solution obtained based on CPLEX and the greedy first-fit approach. We plot in Figure 3.4 the overall operational cost corresponding to the optimal solution, the greedy first-fit solution, as well as the average of 10 runs of the tabu search heuristic. The results are derived with respect to a varying number of VNF instances in the system for 8, 16 and 24 PoPs. The average gap between the tabu search and greedy heuristic with respect to the optimal cost are also portrayed in the figure. We observe that the tabu search heuristic leads to very high-quality solutions in most cases, and reaches optimality in many of them. In particular, we notice that the average gap remains smaller than 9 % for the 8, 16 and 24 PoPs. Instead, the gap for the greedy heuristic can reach very high values that can overpass 100 %.

Similarly, we show in Figure 3.5 the overall operational cost for the optimal, greedy first-fit and average tabu search solution, for 8, 16 and 24 PoPs, with a varying number of VNF instances in the case of $\varepsilon = 20$. There also, the results are obtained by assuming that the NFVO is located in Dallas. We observe that the tabu search heuristic again significantly outperforms the greedy heuristic, with results that are very close to optimality, with average gaps smaller than 8 %. We derive the same results by assuming the NFVO is located in San Jose. The obtained results confirm as well the high-quality solutions that are derived based on the tabu search heuristic. We show those that correspond to $\varepsilon = 1$ and $\varepsilon = 20$ in Figure 3.6 and Figure 3.7, respectively.

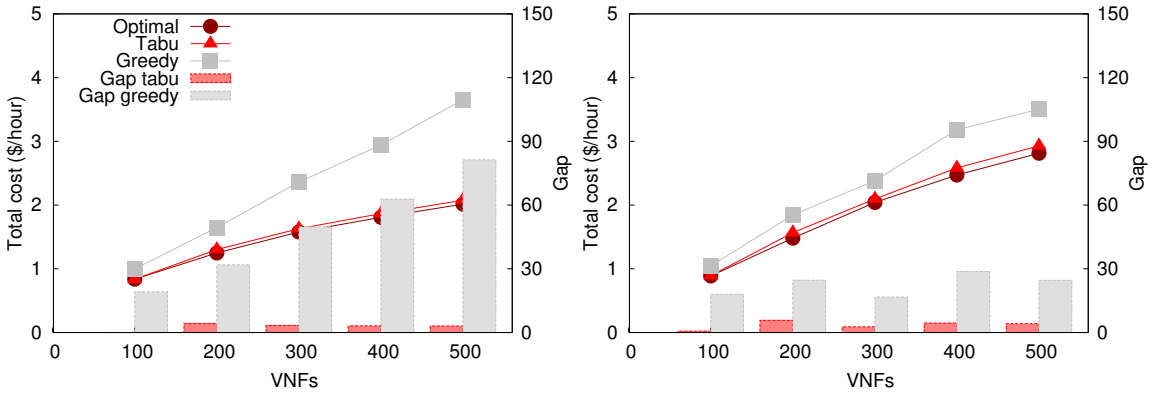


(a) 8 PoPs

(b) 16 PoPs

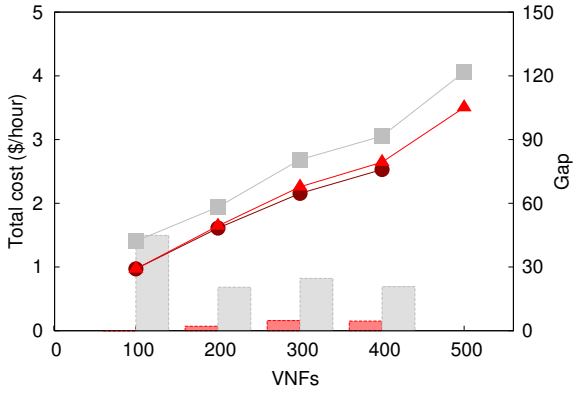
(c) 24 PoPs

Figure 3.4: Total cost for optimal, tabu and greedy solutions, together with the gap from optimality for the tabu and greedy heuristics with $\varepsilon = 1$. The results are derived by assuming the NFVO is located in Dallas



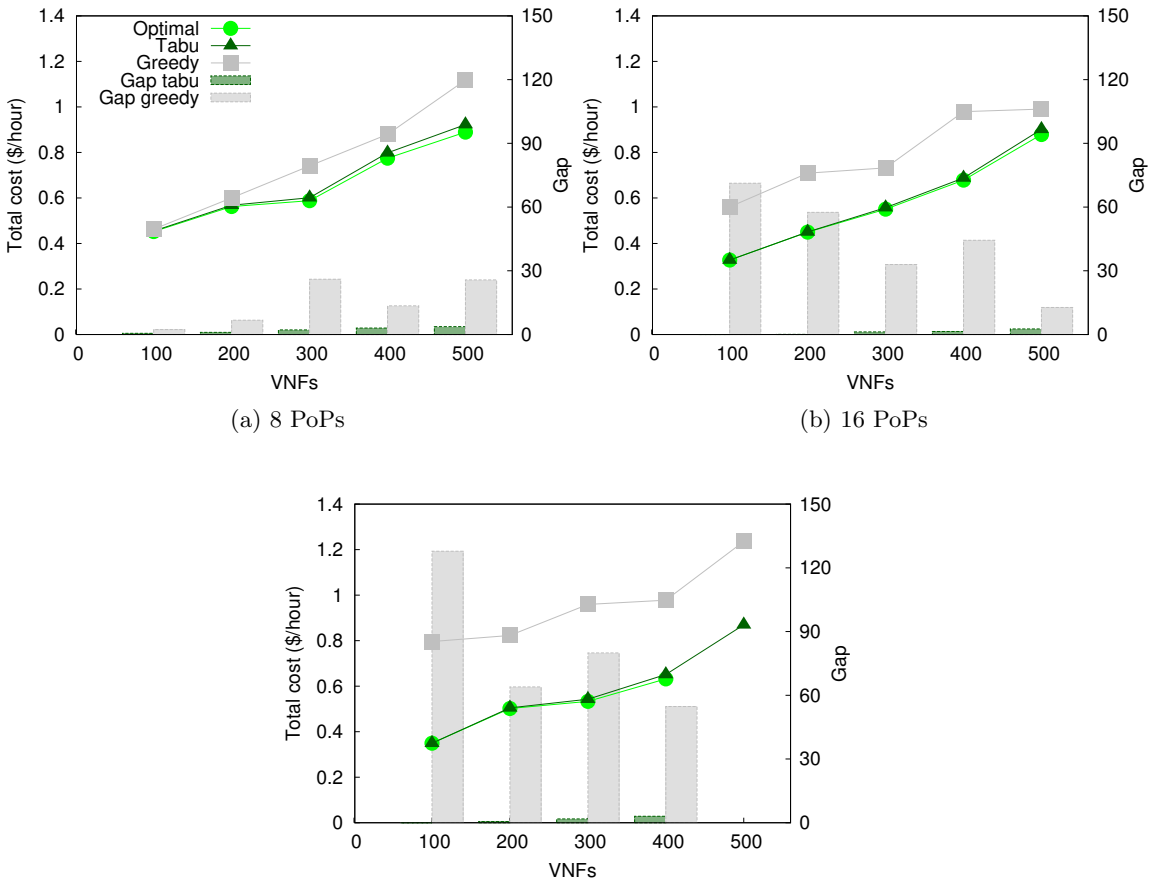
(a) 8 PoPs

(b) 16 PoPs



(c) 24 PoPs

Figure 3.5: Total cost for optimal, tabu and greedy solutions, together with the gap from optimality for the tabu and greedy heuristics with $\varepsilon = 20$. The results are derived by assuming the NFVO is located in Dallas

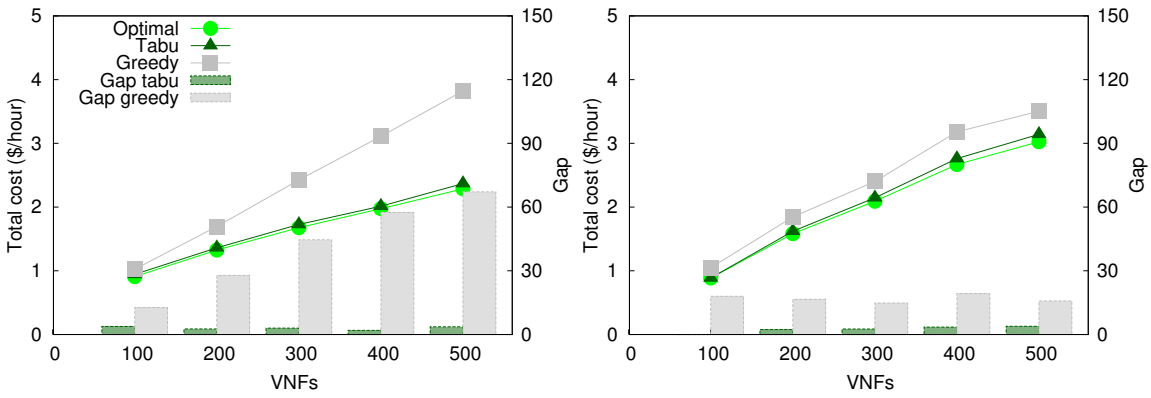


(a) 8 PoPs

(b) 16 PoPs

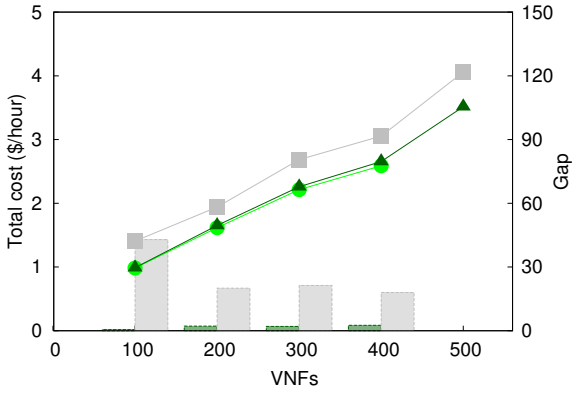
(c) 24 PoPs

Figure 3.6: Total cost for optimal, tabu and greedy solutions, together with the gap from optimality for the tabu and greedy heuristics with $\varepsilon = 1$. The results are derived by assuming the NFVO is located in San Jose



(a) 8 PoPs

(b) 16 PoPs



(c) 24 PoPs

Figure 3.7: Total cost for optimal, tabu and greedy solutions, together with the gap from optimality for the tabu and greedy heuristics with $\varepsilon = 20$. The results are derived by assuming the NFVO is located in San Jose

Table 3.3: Average Execution Time

Experiment Parameters			Execution Time (s)	
#PoPs	#VNFs	ε	CPLEX	Tabu Search
8	100	1	0.24	0.41
8	100	20	0.23	0.33
8	200	1	3.19	0.83
8	200	20	4.20	0.55
8	500	1	462.80	1.80
8	500	20	68.60	1.67
16	500	1	45 398	2.40
16	500	20	5574	1.67
24	500	1	∞	1.98
24	500	20	∞	1.42
64	1000	1	∞	5.03
64	5000	1	∞	29.80

Table 3.3 shows the average execution time of the tabu search heuristic compared to CPLEX. They were run on a server with 2×12-Core 2.20 GHz Intel Xeon E5-2650v4 CPUs and 128 GB memory. The results are for a subset of the experiments in which the NFVO is placed in Dallas. The results show that CPLEX is slightly faster for a few of the very small scale problems in which the execution time is less than one second. However, in all other cases, our heuristic significantly outperforms CPLEX by many orders of magnitude.

3.5.2 Optimization Objective Weight

Focusing on our objective function, we note that ε comes balancing two metrics: on the one hand, the compute resource cost and on the other hand the lifecycle management cost (e.g., bandwidth consumed in the life cycle management). The bigger the value of ε, the more we favor placement in the light of minimizing the lifecycle management cost. With our uniform bandwidth cost over communication links between each couple of PoPs, this translates into favoring the placement of additional VNFMs over the same PoPs of the VNF instances they are managing. Such a placement clearly results in less traffic over the communication links among PoPs, as the lifecycle management traffic would be circulating inside the same PoP,

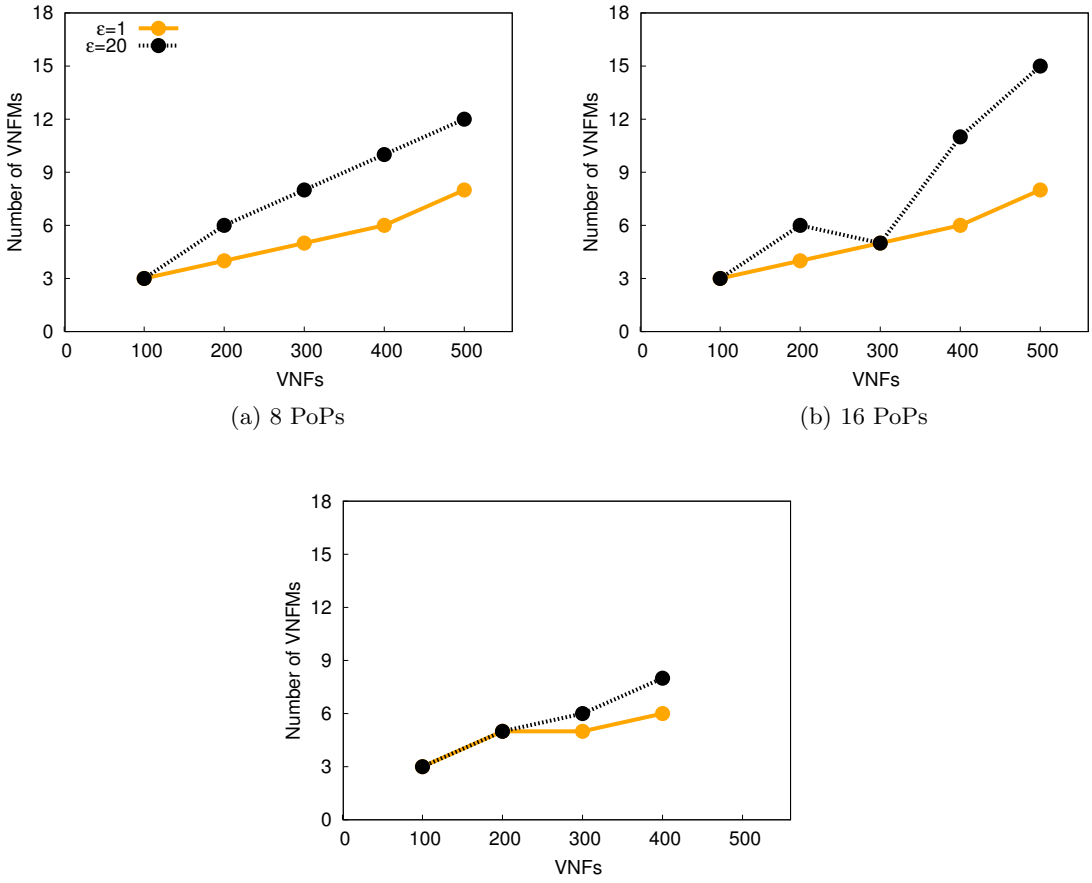
at no cost. We investigate in Figure 3.8 the number of VNFMs placed in the system, with respect to the number of VNF instances, for 8, 16 and 24 PoPs, with the NFVO placed in Dallas, considering $\varepsilon = 1$ and $\varepsilon = 20$. There, we notice a clear jump in the number of VNFMs in the case of $\varepsilon = 20$ compared to $\varepsilon = 1$. Interestingly, deeper investigations allow us to observe that the additional VNFMs are all of the C1 type. That is due to the fact that the management of C1 VNF instances consumes more bandwidth, compared to C2 VNF instances, in our evaluation scenarios.

3.5.3 Impact of NFVO Location

We now investigate how the NFVO location affects the results of the VNFM placement. We thus compare the total cost of the optimal solutions when the NFVO is placed in Dallas to the case when the NFVO is placed in San Jose. The results are plotted in Figure 3.9 and Figure 3.10 for $\varepsilon = 1$ and $\varepsilon = 20$ respectively. They are derived for 8, 16 and 24 PoPs. In both figures, we notice that when the NFVO is placed in San Jose, the total cost can record slightly higher values, with respect to the case of placing it in Dallas. Going back to the geographical distribution of PoPs over the USA in Figure 3.3, we notice that San Jose is far from many of the PoPs. This translates into less flexibility in the system, due to the imposed delay over the communication link between the VNFM and the NFVO. As a result, VNFMs are constrained to a smaller set of PoPs, preventing additional cost gains.

3.5.4 Architectural Options Related to VNFM

Figure 3.11 and Figure 3.12 depict the total operational cost for the optimal and tabu search solutions, considering the generic and VNF-specific options, for $\varepsilon = 1$ and $\varepsilon = 20$ respectively. Cost reductions of up to 34.8 % are obtained when the generic VNFM is used instead of the VNF-specific. The VNF-specific VNFM can handle one VNF class within the evaluation scenarios. Therefore, in this case, the system has to place two distinct set of VNFMs to manage the C1 and C2 classes. This leaves many VNFMs with unutilized capacity since the number of VNF instances per class is relatively small. However, when

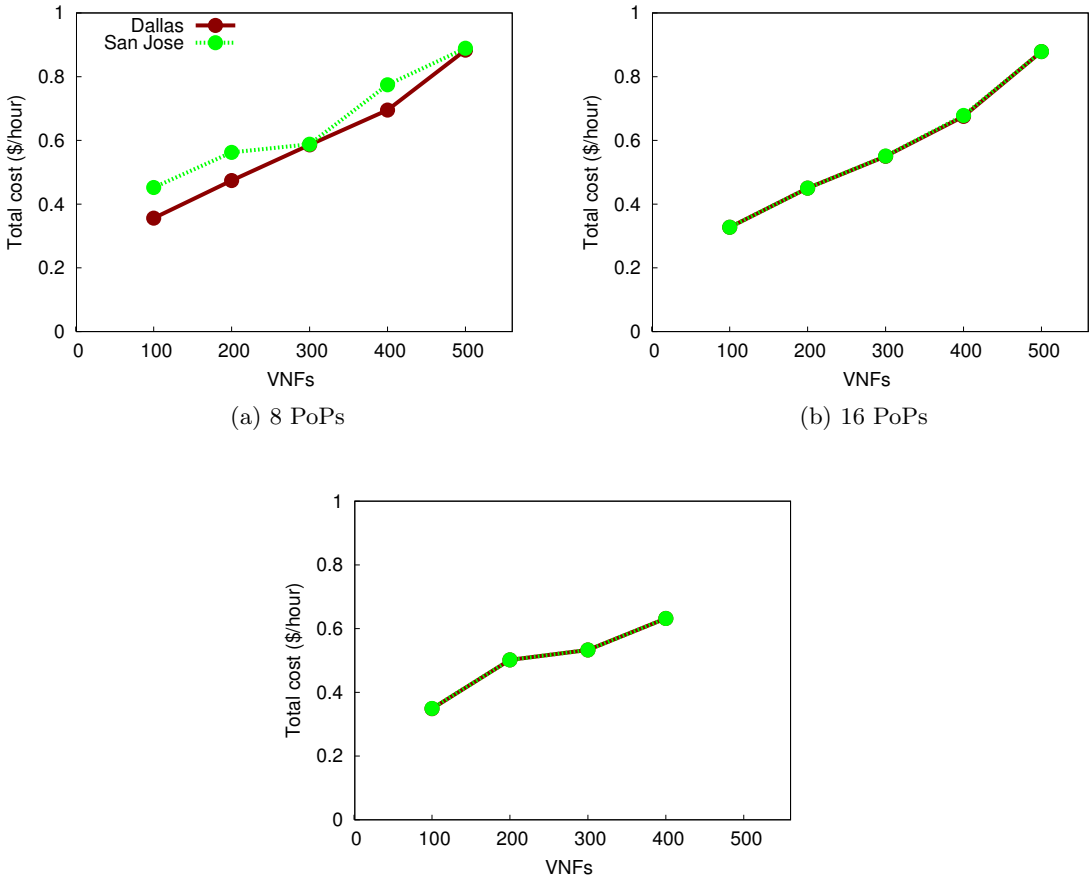


(a) 8 PoPs

(b) 16 PoPs

(c) 24 PoPs

Figure 3.8: Number of VNFMs placed in the system with respect to the number VNF instances with $\varepsilon = 1$ and $\varepsilon = 20$

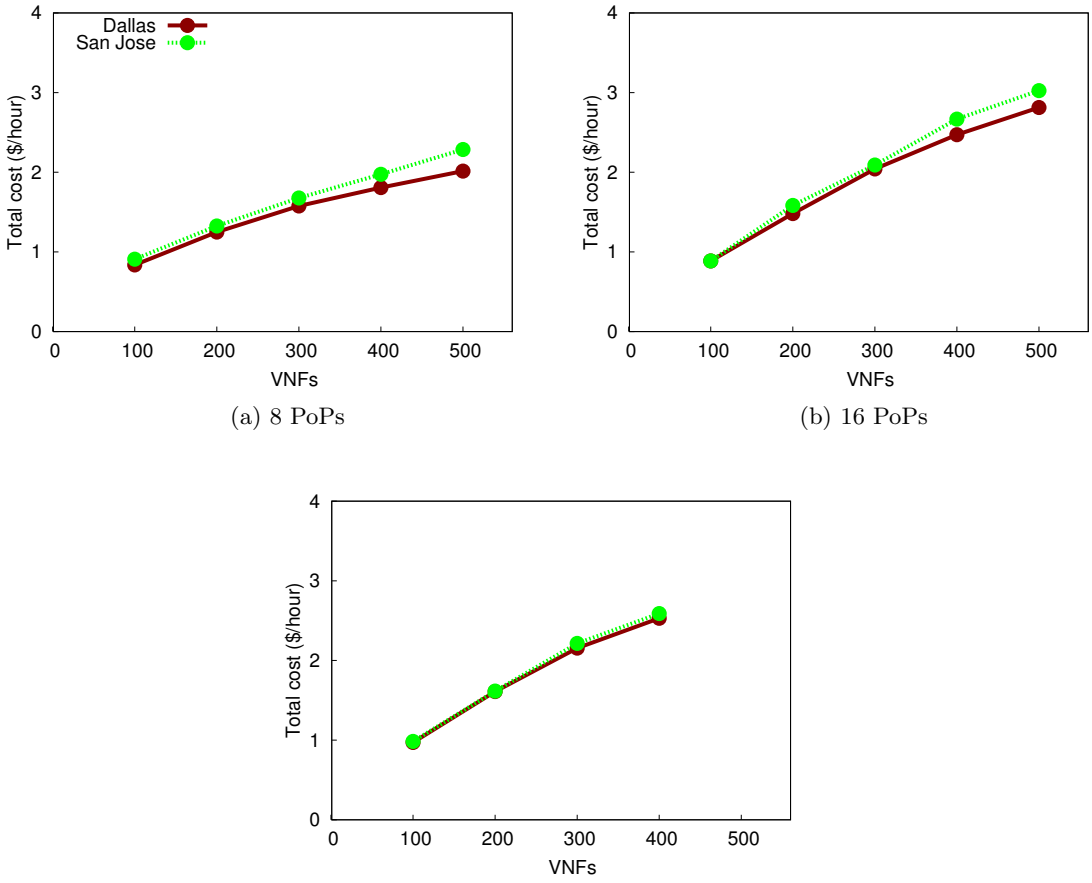


(a) 8 PoPs

(b) 16 PoPs

(c) 24 PoPs

Figure 3.9: Total cost for optimal solutions with $\varepsilon = 1$ for the NFVO placed in Dallas and NFVO placed in San Jose

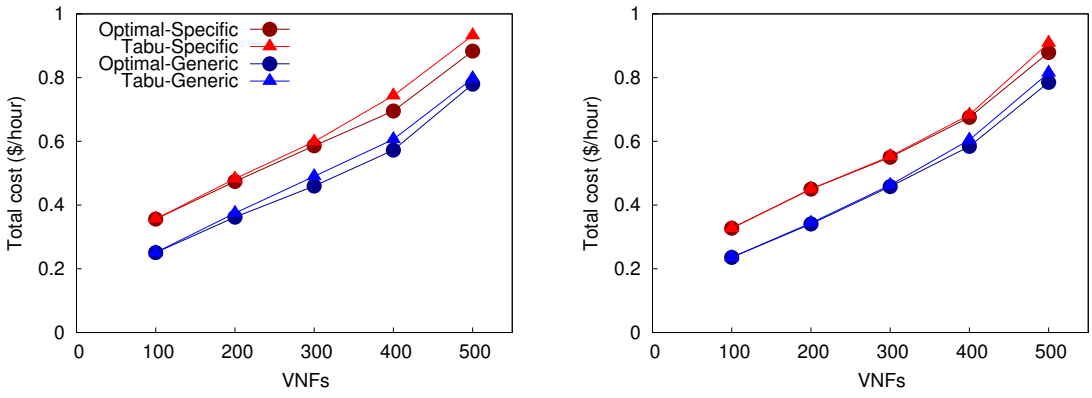


(a) 8 PoPs

(b) 16 PoPs

(c) 24 PoPs

Figure 3.10: Total cost for optimal solutions with $\varepsilon = 20$ for the NFVO placed in Dallas and NFVO placed in San Jose

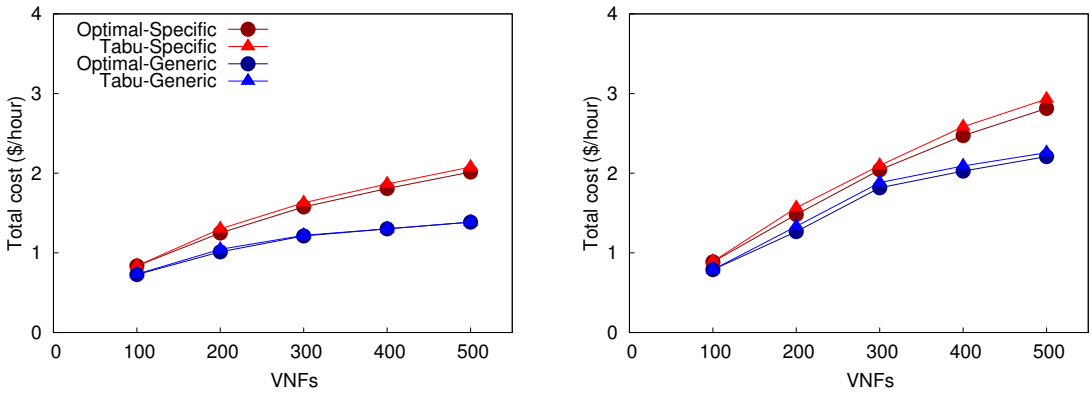


(a) 8 PoPs

(b) 16 PoPs

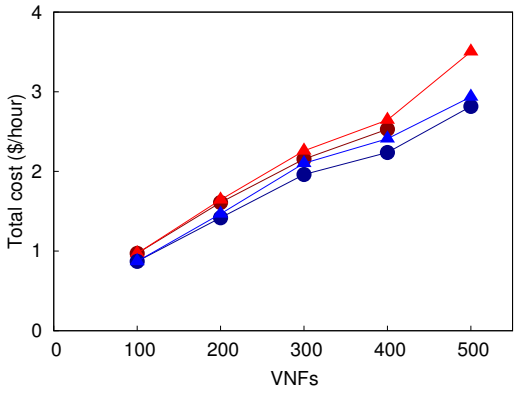
(c) 24 PoPs

Figure 3.11: Total cost for optimal and tabu solutions with $\varepsilon = 1$ for the VNFM architectural options: generic and VNF-specific. The results are derived by assuming the NFVO is located in Dallas.



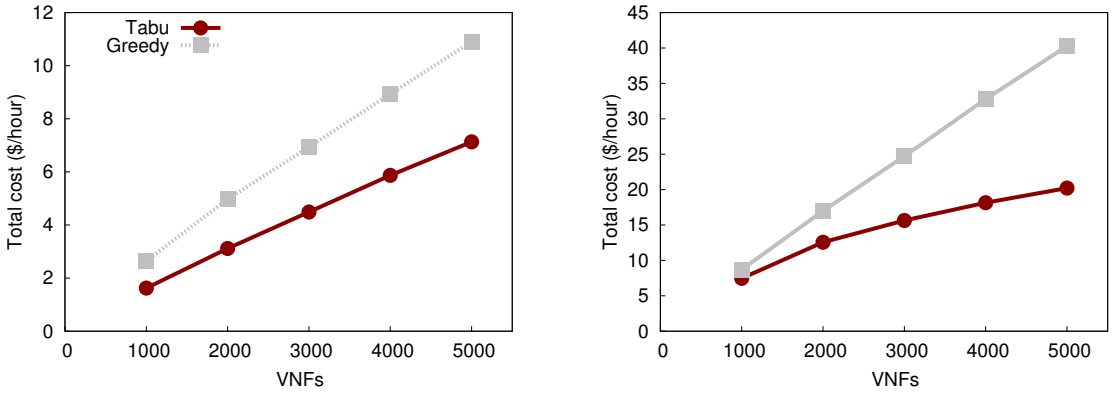
(a) 8 PoPs

(b) 16 PoPs



(c) 24 PoP

Figure 3.12: Total cost for optimal and tabu solutions with $\varepsilon = 20$ for the VNFM architectural options: generic and VNF-specific. The results are derived by assuming the NFVO is located in Dallas



(a) $\varepsilon = 1$ (b) $\varepsilon = 20$

Figure 3.13: Total cost for tabu and greedy solutions in large-scale deployments. The results are derived by assuming the NFVO is located in Dallas

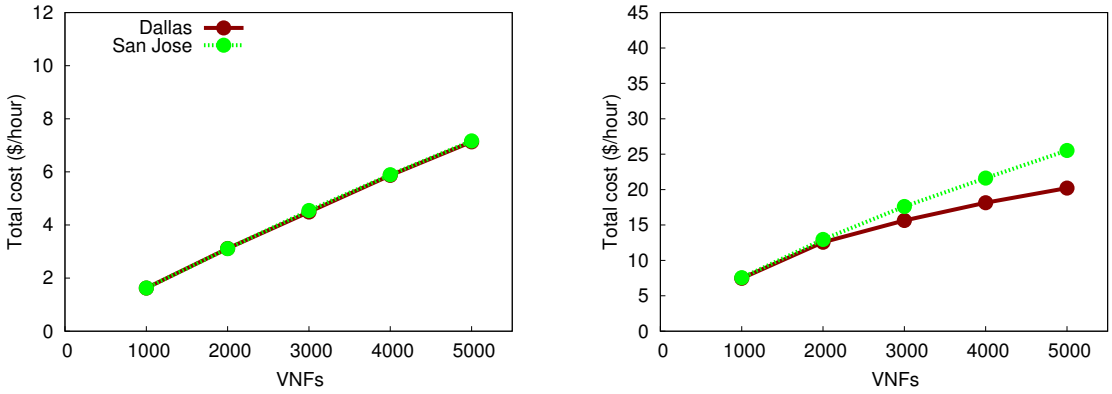
the generic VNFM is adopted, one set of VNFMs manages all VNF instances. The system maximizes the capacity utilization of the VNFMs, resulting in the placement of fewer VNFMs.

3.5.5 Large-Scale Deployment

We report the solutions obtained by tabu search and first-fit heuristics for large-scale deployments in Figure 3.13. The results are reported for $\varepsilon = 1$ and $\varepsilon = 20$ and considering the NFVO is placed in Dallas. As shown, the tabu search outperforms the first-fit and yields up to 49.8 % reduction in the operational cost. We further compare the operational cost with respect to the NFVO location in Figure 3.14. We observe that the results are similar to those obtained in the small-scale deployments. More precisely, when $\varepsilon = 20$ and the NFVO is in San Jose, a higher operational cost is obtained, with respect to the case when the NFVO is in Dallas. The difference in the cost between the two scenarios is proportional to the deployment size, e.g., the number of VNF instances in the system.

3.5.6 Dynamic Placement Evaluation

Figure 3.15 shows the operational cost for managing the VNF instances described in section 3.4.3, with respect to the snapshots. The results are derived using the static and



(a) $\varepsilon = 1$ (b) $\varepsilon = 20$

Figure 3.14: Total cost for tabu solutions in large-scale deployments for the NFVO placed in Dallas and NFVO placed in San Jose

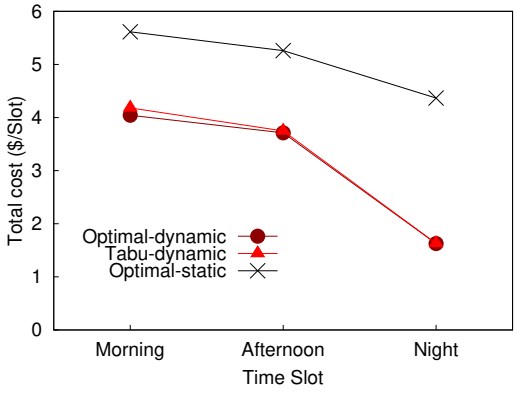


Figure 3.15: Total cost for optimal and tabu solutions of the dynamic MPP and for optimal solution of the static MPP

dynamic versions of MPP. Considering the static MPP, the system manages the VNF instances through over-provisioning of VNFMs. The problem is solved with respect to the maximum number of VNF instances in the system, e.g., 225 VNF instances for each VNF class. The depicted operational cost of static MPP is the sum of VNFM compute resource cost ($C^{com}(t)$) and lifecycle management cost ($C^{lif}(t)$) per snapshot. $C^{com}(t)$ is fixed for all snapshots whereas $C^{lif}(t)$ is incurred for a VNF instance only during its lifetime in the system. Further, the figure reports the operational cost of dynamic MPP corresponding to the optimal and tabu search-based solutions. The numerical results confirm the quality of the obtained tabu search solution in the dynamic MPP. They also show that operational

cost of the static MPP is significantly higher by 38.8 % to 168.3 % compared to the dynamic MPP over all snapshots. The superiority of the dynamic MPP emerges from its ability to adapt the system to the variation in VNFs workload. It can scale out and scale in number of VNFMs, change the VNFMs locations and change the associations between the VNFMs with VNF instances.

3.6 Conclusion

In this chapter, we introduced and studied the VNFM Placement Problem (MPP). We covered two versions of the problem: static and dynamic. We mathematically formulated the problem and proposed tabu search heuristic to solve its large instances. Our numerical results show that the tabu search heuristic leads to high-quality solutions, within an average gap of 9 % from optimality. We further investigated the impact of key aspects (NFVO location, VNFM architectural options and objectives) on the placement decisions and showed that they affect the overall operational cost. Finally, we examined the dynamic MPP and showed that it enables adapting the decisions to the changes in the system, leading to significant reductions in cost with respect to static MPP.

Chapter 4

A Scalable Architecture for NFV Management and Orchestration

4.1 Introduction

The centralized NFV MANO poses scalability concerns, since despite the NFVO capacity, a single NFVO does not scale as the deployment size grows (e.g., VNF instances, PoPs, covered geographical area, etc.) and will fail to efficiently handle all requests and events while providing the anticipated performance. Additionally, regardless of NFVO location, when the infrastructure has a large Diameter, the communication between NFVO and other functional blocks will experience a high delay, decreasing the scalability and degrading the performance.

This chapter presents a scalable and multi-orchestrator NFV management and orchestration architecture in the specific context of Virtual Network Platform-as-a-Service (VNPaaS) for 3GPP 4G and beyond. VNPaaS is analogous to the Platform-as-a-Service (PaaS) model of cloud computing, but targeting the network domain instead. In this regard, the VNPaaS service provider will provide a toolkit for the customers to provision (e.g., develop, deploy, manage and terminate) their virtual network services according to the pay-as-you-go model. These services can range from simple services, such as the Home Subscriber Server (HSS) service, to more complex services, such as IP Multimedia Subsystem (IMS) and even

Evolved Packet Core (EPC). Although VNPaaS is a PaaS for provisioning network services, it needs to include functionalities that differ from those covered in a typical PaaS, due to the differences in the requirements of network services and IT applications. For instance, PaaS encompasses centralized applications management. However, the proposed architecture employs two-layer hierarchical service orchestration in order to cope with the distributed nature of NFV scenarios. There, the PoPs are grouped into domains (or orchestration zones) and a domain orchestrator is assigned the responsibilities of the resource orchestration and service lifecycle management in a domain. A global service orchestrator maintains a global view of the entire system and performs end-to-end service orchestration across different domains. The number of the domain orchestrators can be adjusted to satisfy system capacity and performance requirements. Moreover, we present a proof-of-concept prototype to validate the feasibility of the approach. We also use the Home Subscriber Server, a functional entity used in IMS and EPC networks, as an illustrative use case. We redesign HSS for the cloud environment, in which the HSS is decomposed into VNFs with a granularity finer than what is known today. The new architecture allows the different Diameter interfaces of HSS to be deployed and scaled independently. It also enables performance isolation between these interfaces, which is further demonstrated by experimentation.

4.2 4G Mobile System Architecture: Overview and Challenges

The Evolved Packet System (EPS), which is also known as 4G, was defined by the 3rd Generation Partnership Project (3GPP) in Release 8 as the first pure IP-based mobile system. It consists of (1) the Long Term Evolution (LTE) as the radio access network, and (2) the Evolved Packet Core (EPC) as the core network of the system. The IP Multimedia Subsystem (IMS) is the 3GPP service network which was used for providing multimedia services for 3G systems and then continued to be used at the inception of 4G. Figure 4.1 shows a simplified EPS architecture.

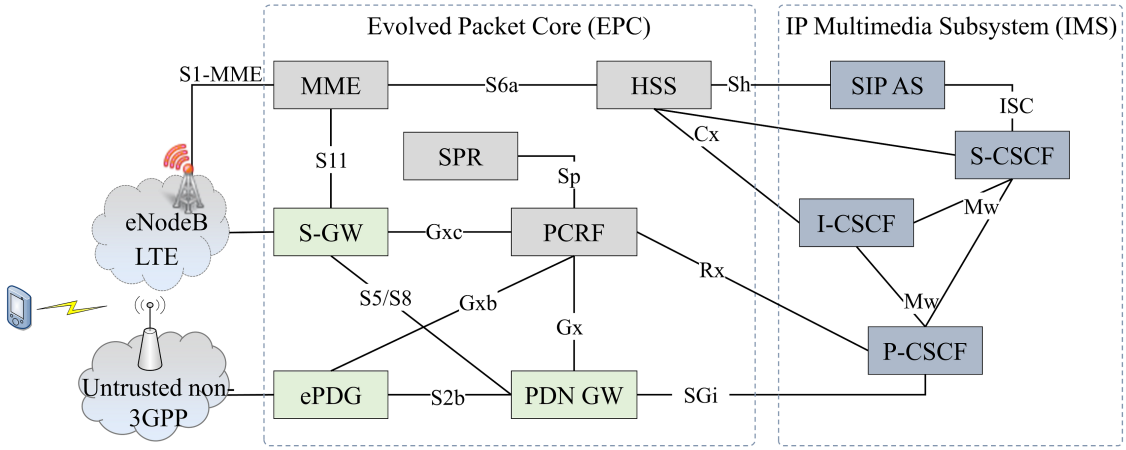


Figure 4.1: Simplified EPS architecture

1. Evolved Packet Core: The EPC [70, 71] is a flat IP-based core network for the LTE family of wireless access technologies. It can also accommodate other 3GPP access networks, such as GPRS and UTRAN, and even non-3GPP access networks, such as WiMAX and Wi-Fi. The EPC architecture enhances network performance by separating data and control planes.

In the data plane, the Serving Gateway (S-GW) and evolved Packet Data Gateway (ePDG) act as the access gateways for LTE technologies and the untrusted non-3GPP access networks respectively while the Packet Data Network Gateway (PDN-GW) provides the connectivity to the external networks and nodes. In the control plane, the Policy and Charging Rule Function (PCRF) is the decision maker in the Policy and Charging Control (PCC) system. The PCRF enables EPC to support flow-based policy control and charging. It has interfaces with data plane gateways to control the policy enforcement. It is also connected to the Subscription Profile Repository (SPR) which contains the subscription information, such as user policies.

The Mobility Management Entity (MME) handles key control functions such as mobility for LTE technologies and the selection of S-GW as well as PDN-GW. The Home Subscriber Server (HSS) is the central database of the mobile network that contains user-related information, such as subscription, location, and identification information. It has several Diameter interfaces to communicate with other functional entities in the system.

For instance, S6a and Cx interfaces are used for the interaction with MME and IMS, respectively.

In Release 9, 3GPP introduced the User Data Convergence (UDC) concept [72] which separates the application logic of the 3GPP network functional entities (e.g., HSS), the so-called Front-Ends (FEs), and the user data which is moved into a logically unique repository, referred to as User Data Repository (UDR). The FEs access the UDR via the Ud standard interface. Considering the HSS, the data, which is usually stored in HSS, will be moved to UDR whereas the logic of HSS will be implemented in a functional entity called HSS Front-End (HSS-FE).

2. IP Multimedia Subsystem: The 3GPP IMS [73] is a service network on top of an IP transport layer required for the seamless and robust provisioning of IP multimedia services to end-users. It uses the Session Initiation Protocol (SIP) to control multimedia functions. It is made up of data plane and control plane. The data plane consists of SIP Application Servers (SIP-ASs) that implement the logic of IMS services such as a presence service. The SIP-AS interacts with HSS to access the users information via the Sh reference point.

The key functional entity of the control plane is the Call State Control Function (CSCF). There are three types of CSCF: Proxy-CSCF (P-CSCF), Interrogating-CSCF (I-CSCF) and Serving-CSCF (S-CSCF). P-CSCF is the first point of contact for the IMS User Equipment (UE) within an IMS network. It acts as a stateful SIP proxy when routing SIP signaling messages going to and from an IMS UE. It is allocated to the IMS UE and does not change for the duration of the registration. I-CSCF is the first contact point for external IMS networks. It is a stateless SIP proxy that selects an S-CSCF for IMS UE and routes incoming SIP signaling messages to the selected S-CSCF. Serving-CSCF (S-CSCF) is the central node of the control plane of an IMS network. It acts as a stateful SIP registrar and proxy in an IMS network. As a SIP registrar, it registers IMS users and maintains the binding between the public user identity and the user profile. It also interacts with the HSS via the Cx reference point to obtain users profiles. As a SIP proxy, S-CSCF forwards

specific types of SIP messages to the appropriate SIP-AS.

4.2.1 Challenges for Mobile Network Cloudification

Mobile network operators have stringent performance, scalability and fault tolerance requirements on their services [74]. Today, these services are provided by manually deployed and highly reliable network function appliances that are provisioned for peak traffic. However, to deliver these network services as cloud services, the software modules of the network functions must be rebuilt as cloud applications, not just adapted for the cloud, in order to achieve operational efficiency, resource efficiency while providing the same performance and reliability. Cloud applications are designed in a way that supports automated scalability and resilient services on non-reliable hardware.

Typically, a traditional 4G functional entity contains a set of functions as one deployable and scalable unit. However, since cloud resources are provisioned and de-provisioned on-demand, the current architecture might lead to inefficiency in resource usage. Besides, 4G functional entities are often stateful which hinders elastic scalability and resiliency.

A key challenge in migrating these entities to the cloud is to redesign them as smaller and lighter functions. In other words, a good starting point is to consider the current granularity and decompose the logic of a functional entity into smaller functions. This design gives finer control over the distinct functions allowing to deploy and scale them independently, when and where needed. However, if the new functions interact with each other, then there is a need to design new interfaces. These interfaces should be very lightweight to minimize the extra cost induced by the communication. On the other hand, they also need to be reliable and scalable.

Another challenge to identify the optimal granularity of these small functions that can achieve the intended benefits. Indeed, refining that level of granularity through the splitting of the functional entities will usually lead to an additional cost (e.g., inter-function communication). These costs may (or may not) offset the gains expected from the refining. Optimal splitting, therefore, becomes key.

4.3 VNPaaS Architecture

4.3.1 Business Model

The proposed architecture relies on a business model that involves a variety of actors. Each actor might play several business roles. The key roles are NFVI-as-a-Service (NFVIaaS) provider, VNPaaS provider, VNF-as-a-Service (VNFaaS) provider, Virtual Network-as-a-Service (VNaaS) provider and Mobile Virtual Network Operator (MVNO).

The NFVIaaS provider offers VNPaaS provider the NFVI (i.e., resources) required to deploy and run VNFs and VNPaaS components. The NFVI consists of several geographically distributed PoPs, as well as the WAN connectivity between them. These PoPs might belong to one or multiple providers. They might also have different capabilities. Some of them can offer both physical (i.e., bare-metal) and virtual resources while others can offer only virtual resources.

The VNPaaS provider provides toolkits and Application Programming Interfaces (APIs) for provisioning the network services. It also provides the network service and VNF catalogs that allow VNFaaS and VNaaS providers to offer their VNFs and network services respectively, for on-demand usage. The VNFaaS provider implements network functions as VNFs and adds them to VNF catalog. This would make these VNFs available to VNaaS provider to use them on-demand. These VNFs might have the same granularity of network functions as known today. They might also have a finer granularity (i.e., decomposition) or even a coarser one (i.e., aggregation). Varying granularity would be aspired to meet particular requirements such as response time. The VNaaS provider, in turn, uses VNPaaS capabilities to provision its network services. It might use its VNFs or reuse VNFs offered in the VNF catalog. The VNaaS provider can offer its network services directly to MVNO. It can also add its network services to the network service catalog to indirectly offer them (i.e., via VNPaaS) to other VNaaS providers so that they can be used to build end-to-end network services. MVNO uses network services to provide mobile network services to end-users.

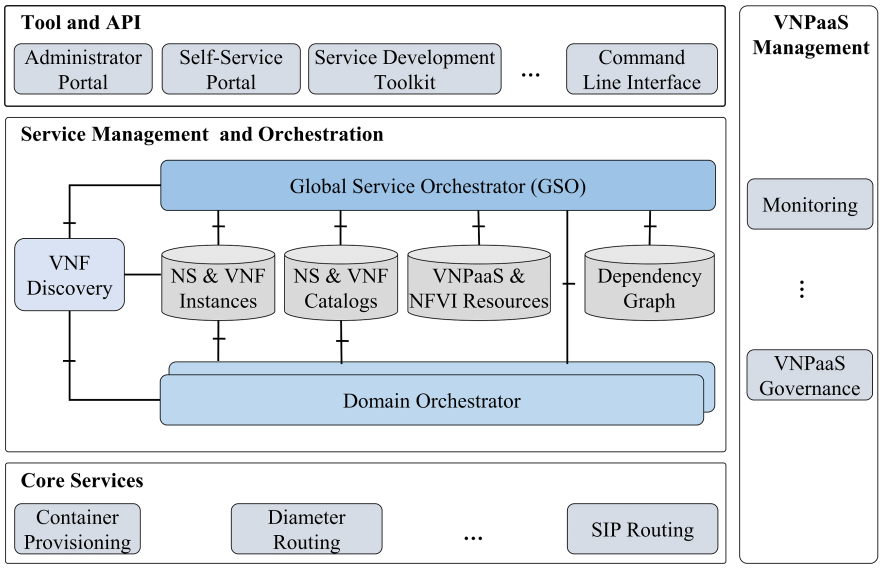


Figure 4.2: VNPaaS high-level architecture

4.3.2 Layers and Functional Components

Figure 4.2 depicts the proposed architecture of VNPaaS. It shows the key components of the architecture. Other components (e.g., logging management) that exist in regular IT PaaS are an integral part of the proposed architecture. However, we do not detail them to focus on the components related to the novel contribution. The architectural layers are:

1. Core Services: This layer contains the services hosted and managed by the VNPaaS. The container provisioning service is responsible for provisioning and scheduling the VNFs packaged as containers. Diameter and SIP are essential signaling protocols in 3GPP mobile systems. Therefore, many of the prospective VNFs will support Diameter and/or SIP interfaces. Thus, the VNPaaS includes Diameter routing and SIP routing services to manage Diameter and SIP signaling, respectively. These services can distribute signaling traffic across multiple VNF instances to enable horizontal scalability.

2. Management and Orchestration: This layer provides scalable and multi-orchestrator NFV management and orchestration. This is attained by decomposing the NFVI, which consists of multiple geographically distributed and interconnected PoPs, into logical partitions called domains (or orchestration zones) as shown in Figure 4.3. Each domain composes

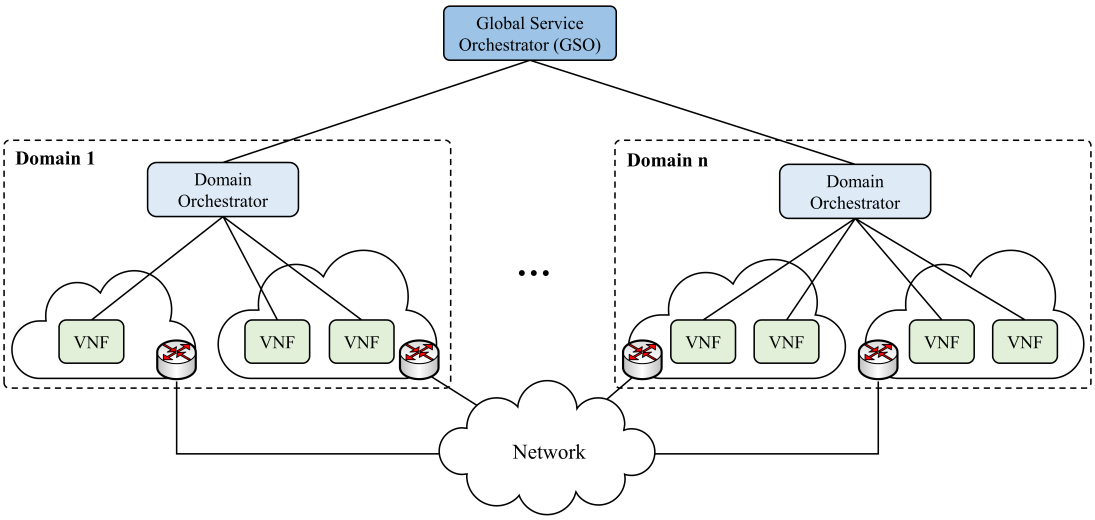


Figure 4.3: Two-layer hierarchical service orchestration

of one or more PoPs and contains a domain orchestrator that is responsible for resource orchestration and network service lifecycle management within the domain boundaries. This approach improves NFV MANO scalability for two main reasons. First, it distributes the NFV management and orchestration responsibilities among several orchestrators that provide higher processing capacity compared to the centralized MANO. Second, a single domain orchestrator would cover smaller geographical area compared to the orchestrator in centralized MANO which leads to lower communication delay and better scalability.

The number of domains, and consequently the domain orchestrators, would depend on the capacity and maximum tolerable network delay in the system. Thus, their number can be adjusted to meet the scalability and performance requirements in different deployment scenarios. In this architecture, the domain orchestrator provides the same functionality of NFVO and generic VNFM as defined in ETSI NFV MANO framework. Moreover, the Global Service Orchestrator (GSO) is responsible for the end-to-end network service orchestration across multiple domains. It is also in charge of domains management, as we will elaborate upon later in this chapter.

Our architecture uses model-driven orchestration to orchestrate and automate the end-to-end service lifecycle management. The model is based on the Topology and Orchestration

Specification for Cloud Applications (TOSCA). TOSCA [75] is an OASIS standard to describe cloud applications by means of service templates and management plans. The service template describes the structure of the service (or application) using topology that defines its components, relationships, dependencies and requirements. Further, the management plan captures the lifecycle management tasks (e.g., instantiation, configuration and scaling) as workflows, which allows the orchestrator to automate the lifecycle management. In this regard, for instance, the domain orchestrator can deploy and manage the network services and VNFs described by TOSCA service templates, under the instruction of the GSO.

Moreover, the VNF discovery engine plays a key role in composing network services from VNFs managed by different domain orchestrators (i.e., deployed in different domains). It is responsible for centralizing the information of VNF instances used across domains into a common registry and providing easy publish/discovery functionality. In the publish operation, the domain orchestrators would publish the information of their VNF instances accessed from other domains. This information includes, but is not limited to, VNF type and metadata of the connection points (e.g., IP and port). In the discovery operation, the domain orchestrators query the VNF discovery engine to get the information of VNF instances which their locally managed VNF instances will communicate with (if any). Then, they make this information available for lifecycle operations such as configuration.

The GSO and domain orchestrators use four categories of repositories to support their functions as depicted in Figure 4.2. The network service & VNF catalogs hold information (e.g., description) about network services and VNFs. The network service & VNF instances describe the network service instances and VNF instances. The VNPaaS and NFVI resources repository holds the information (e.g., description and location) about reserved and available resources. The dependency graph repository contains a graph structure that represents the relations among the main architectures components (e.g., VIMs, domain orchestrators, network service instances and VNF instances). Such a graph structure allows to easily handle large-scale dependencies between components in order to keep track of the system evolution.

3. Tool and API: This layer includes different tools and APIs to access the VNPaaS capabilities and functions.

4. VNPaaS Management: This layer interacts simultaneously with the three previous layers. It includes the components responsible for the management functions related to the VNPaaS. An example of such components is the monitoring, which is responsible for monitoring the resource consumptions, health and Key Performance Indicators (KPIs) of VNPaaS components.

4.3.3 Operational Procedures

The GSO supports multiple procedures in order to support the proposed NFV management approach. Next, we discuses the main procedures.

4.3.3.1 Network Service Deployment

In the proposed architecture, many network services might include VNFs deployed in different domains. To deploy such a network service, the GSO decomposes it into smaller subservices according to the domains where the VNFs would be deployed; each subservice is described by a service template. Then, it extends these templates by adding VNFs publish/discovery operations as required. The GSO also configures the VNF discovery engine to control the publish/discovery operations so that the composition leads to the desired network service. Lastly, the GSO would instruct the domain orchestrators responsible for the domains to deploy the subservices and compose them into the desired network service.

4.3.3.2 Domains Management

The GSO would maintain a global view of the entire system state including resources, network services and domains. It would use predefined policies and KPIs to decompose the NFVI into domains. An example of such a KPI would be an upper bound limit on the network delay between the domain orchestrators and VIMs (i.e., NFVI). Furthermore, the GSO would use the KPIs to scale out/in domains (including domain orchestrators) and

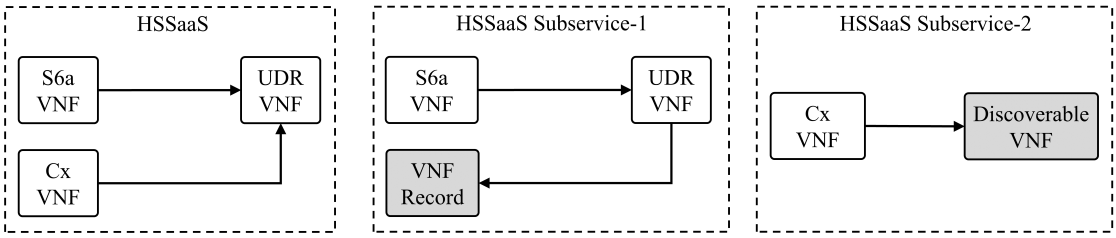
reshape them dynamically with respect to the evolving features of the VNPaaS environment (e.g., system workload). To that end, it might use, for example, the execution time of the key management operations (e.g., monitor VNF) performed by the domain orchestrators.

4.4 Illustrative Use Case: HSS-as-a-Service

4.4.1 NFV-based HSS

To provision HSSaaS using the proposed VNPaaS, the network functions of HSS should be implemented as VNFs. Considering the current granularity in HSS, each of HSS-FE and UDR would be implemented as a VNF. These VNFs can be provided by one or more VNFaaS providers. VNaaS provider will use VNPaaS to provision HSSaaS made from the composition of these VNFs. HSS-FE VNF includes all supported Diameter interfaces. These interfaces would be deployed and scaled together. However, there is no functional requirement to keep these interfaces together in one entity. In fact, they do not have direct interaction with each other. Therefore, we propose to decompose HSS-FE functions and implement them as smaller and independent VNFs according to the Diameter interfaces so that each interface is implemented as a separate VNF. These VNFs can be provided by one or more VNFaaS providers.

This proposed decomposition of HSS-FE does not introduce additional communication overhead. Meanwhile, it would bring two main benefits. First, it isolates the performance so that the traffic on different interfaces does not affect each other. This isolation becomes more important when there is a sudden surge in signaling traffic on a particular interface. The signaling storm [76] is an example of a surge in signaling traffic in mobile networks. It generates traffic on S6a of HSS-FE and does not affect other interfaces. Second, it promotes flexibility, which enables deployment optimization by allowing different management policies on different interfaces. This would help in meeting different requirements. For example, considering the response time, the decomposition offers the ability to place different interfaces at different PoPs to gain performance improvement.



(a) HSSaaS (b) HSSaaS subservices

Figure 4.4: High-level TOSCA topology templates

4.4.2 Illustrative Scenario

In this scenario, we assume that the HSSaaS supports S6a and Cx interfaces. Figure 4.4(a) shows its associated high-level topology template. It consists of three VNFs: S6a, Cx and UDR. We also assume that VNPaaS has two domains. Each domain has its container provisioning service. The S6a and UDR will be deployed in domain-1 whereas Cx will be deployed in domain-2. Yet, as another assumption, the UDR VNF will be deployed on a VM whereas S6a and Cx VNFs are packaged as containers.

Figure 4.5 shows a high-level sequence diagram for deploying the described HSSaaS. First, the GSO decomposes it according to the domains into two subservices. After that, it adds publish/discovery operations to the service templates (steps 1 and 2). The resulting subservices are depicted in Figure 4.4(b). Subservice-1 includes S6a and UDR VNFs (i.e., VNFs deployed in domain-1). The UDR VNF is connected to a TOSCA node, the so-called "VNF Record" via a special TOSCA connect-to relationship. The implementation of this node and relationship would ensure that the domain orchestrator-1 will publish the information of the UDR VNF instances by creating a record for each instance in the VNF discovery engine. In the subservice-2 template, the Cx VNF is connected to a new TOSCA node, the so-called "Discoverable VNF". This node provides an abstract view of the UDR VNF instances deployed in domain-1. The implementation of this node would ensure that the domain orchestrator-2 will query the VNF discovery engine to get the details of this service. All these new TOSCA nodes and relationship are implemented by the VNPaaS itself and do not require any changes in the VNFs.

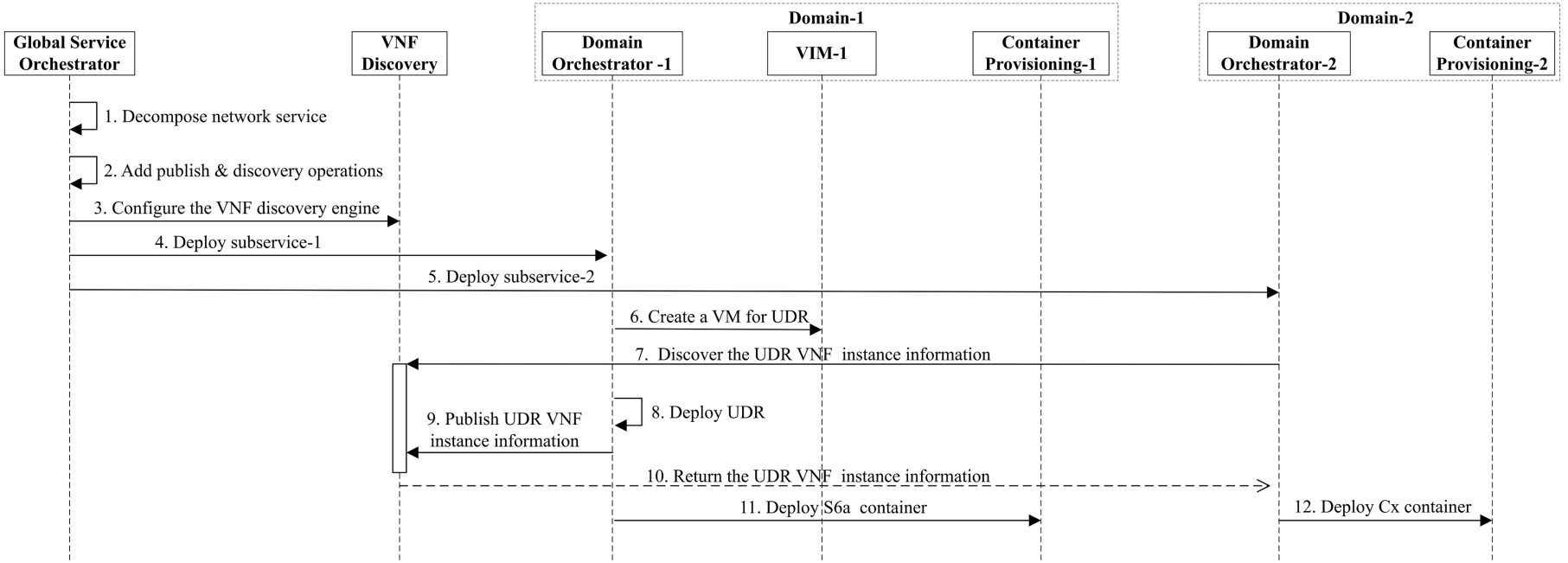


Figure 4.5: Illustrative HSSaaS deployment scenario

The GSO configures the VNF discovery engine to control the publish/discovery operations (step 3). Then, it sends to the domain orchestrators to deploy the subservices (steps 4 and 5). The domain orchestrators start to invoke the lifecycle operations of the TOSCA nodes and relationships defined in the service templates in the right order, based on their dependencies. The domain orchestrator-1 starts to deploy subservice-1 by sending a request to VIM-1 to create a VM to host UDR VNF (step 6). Simultaneously in subservice-2, since the Cx VNF is connected to "Discoverable VNF", the domain orchestrator-2 queries VNF discovery engine to get UDR VNF instance information (step 7). However, the VNF discovery engine blocks the request until the information becomes available (the request has a timeout). The domain orchestrator-1 invokes the lifecycle operation of the UDR VNF (step 8). After that, the domain orchestrator-1 publishes the information by creating a VNF record in the VNF discovery engine (step 9). As a result, the information required by the discovery operation becomes available. Therefore, the discovery operation initiated by domain orchestrator-1 returns the UDR VNF instance information (step 10). Lastly, the domain orchestrators create S6a and Cx VNF containers and configure them to connect to UDR VNF instance (steps 11 and 12).

4.5 Prototype Implementation

In order to demonstrate the feasibility of the proposed approach, the scenario presented in section 4.4.2 is implemented. The decomposition of HSSaaS into two subservices, as well as adding the publish/discovery operations are done manually in this prototype. The prototype covers the implementation of a simplified version of VNPaaS architecture and HSSaaS. More details are provided in what follows.

4.5.1 VNPaaS

Figure 4.6 shows the architecture of the VNPaaS prototype. It is deployed on OpenStack [77] as NFVI. Kubernetes [78] is used as a container provisioning service. It is an open source project for scheduling, managing and orchestrating Docker [79] containers. Furthermore, a

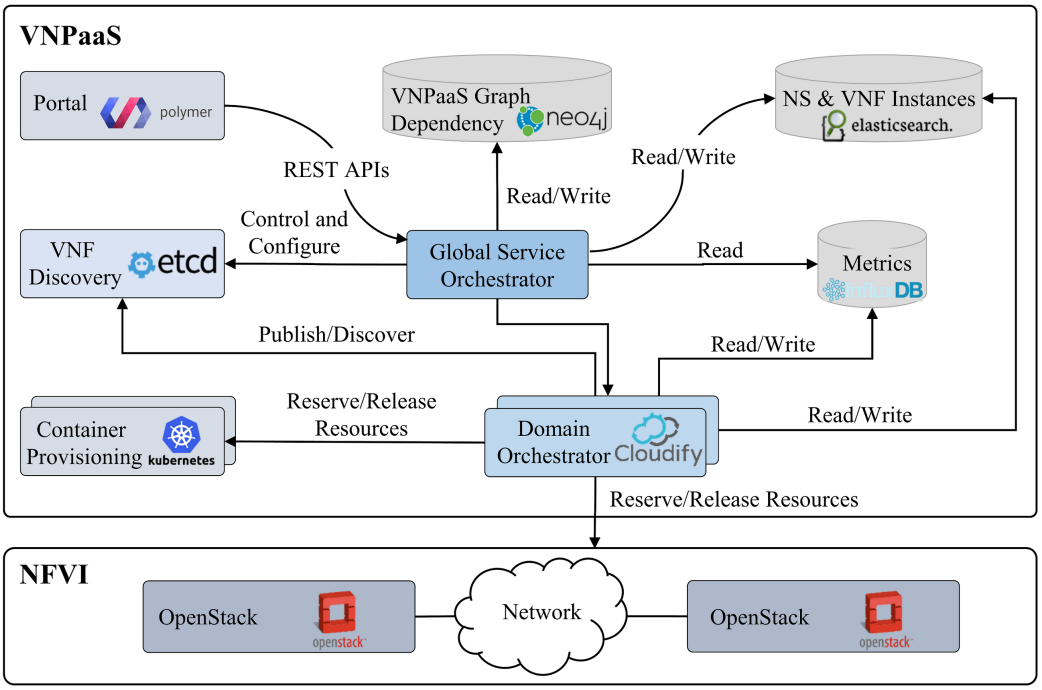


Figure 4.6: VNPaaS Prototype architecture

Diameter proxy is implemented in JAVA to offer Diameter routing service in the prototype. It supports Diameter message routing based on the Diameter application. It also supports load balancing between Diameter servers in a round-robin fashion. An extended version of Cloudify [80] plays the role of domain orchestrator. Cloudify is an open source orchestration engine to deploy and manage the applications described in TOSCA. Two plugins are implemented for Cloudify v3.1. One is used for the interaction with Kubernetes to deploy and manage the Docker containers. The other is used for the communication with etcd [81], playing the role of VNF discovery engine. etcd is a highly available key-value storage for shared configuration and service discovery.

A simple GSO is implemented as a JAVA tool. It exposes its capabilities via REST API. Neo4j [82] is a graph database that is used to hold the graph structure that models dependencies between different components in the architecture. Elasticsearch [83] is a distributed document-oriented database used to store the information of network service and VNF instances. InfluxDB [84] is a time series database that plays the role of metrics repository.

4.5.2 HSSaaS

The CHeSS distribution of HSS is used in the prototype. It is part of the OpenEPC Release 2 testbed [85]. It is implemented using the C language and uses a MySQL database to store its data. It supports the S6a, Cx and Sh interfaces. Moreover, CHeSS HSS is extended to export performance metrics (e.g., response time) through the log file. A simple monitoring agent is implemented as a JAVA tool. It can parse the log and aggregate the metrics for each request. Then, it pushes the metrics to the responsible domain orchestrator.

The CheSS HSS is decomposed based on the Diameter interface. Each interface is packaged with a monitoring agent into a Docker image so that each of them can be deployed, scaled and monitored independently. Lastly, HSSaaS is modeled using TOSCA so it can be deployed and managed by VNPaaS.

4.6 Performance Evaluation

We conduct a set of experiments to validate and evaluate the prototype, in particular, the impact of splitting the HSS-FE on the response time and resource usage. We use two of Ericsson's proprietary traffic generators, referred to as EPC and IMS generators, to generate Diameter traffic workload. The EPC generator generates S6a traffic whereas the IMS generator generates Cx and Sh traffic. These traffic generators simulate several scenarios for the interaction with the HSS in real-world mobile networks. Each of these scenarios is associated with a specific probability and includes one or more Diameter request messages, as reported in Table 4.1.

To that end, the response time and resource usage of S6a and Cx in the full (i.e., non-split HSS-FE) and split architectures of HSS-FE are evaluated. Sh interface is excluded from the study as it constitutes a very small percentage (about 2 %) of the generated traffic. In our experiments, we over-provision the resources allocated to MySQL (representing the UDR in the HSS architecture), Diameter proxy and Diameter traffic generator tools to avoid becoming a performance bottleneck. MySQL is deployed on a VM with 16 CPUs and 32 GB memory. The Diameter proxy is deployed as a Docker container with 16 CPUs and

Table 4.1: Diameter Traffic Details

Interface	Diameter Command Name	Percentage
S6a		
	Authentication Information Request (AIR)	40 %
	Purge UE Request (PUR)	30 %
	Update Location Request (ULR)	30 %
Cx		
	Location Info Request (LIR)	43 %
	Multimedia Authentication Request (MAR)	2 %
	Server Assignment Request (SAR)	8 %
	User Authorization Request (UAR)	47 %

16 GB. In the split HSS-FE case, each of S6a and Cx is deployed in a Docker container with 1 CPU and 1 GB memory. The Diameter proxy routes the Diameter traffic to S6a and Cx containers based on the Diameter application. In the full HSS-FE case, S6a and Cx interfaces are packaged in the same Docker image (single deployable and scalable unit). Two instances are deployed; each one is assigned 1 CPU and 1 GB memory. The Diameter proxy distributes the traffic between the two containers in a round-robin fashion.

Next, the response time experiments are presented first, followed by the resource usage experiments.

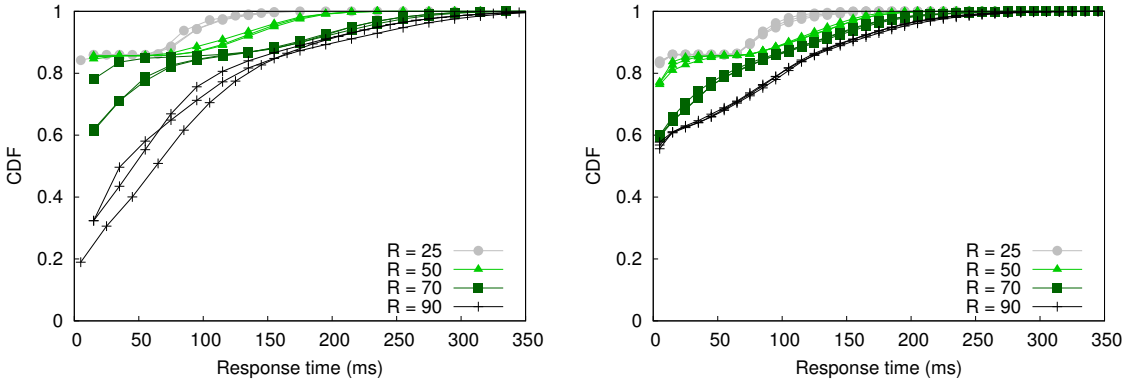
4.6.1 Response Time

In this part, the response time of S6a and Cx in the full (i.e., non-split HSS-FE) and split architectures of HSS-FE are evaluated, when the traffic workload on each interface approximately utilizes the same CPU time. Four CPU utilization rates (R) are considered during these experiments: 25 %, 50 %, 70 % and 90 %. Trial-and-error procedure is used to determine the configurations of the Diameter traffic generator tools required to generate the traffic workloads that lead to the CPU utilization rates mentioned above. For each of the CPU utilization rates, we repeat the experiment three times. The duration of each run is seven minutes. Table 4.2 shows the details of the generated Diameter traffic.

Next, we start our analysis by considering the overall performance of the HSS-FE.

Table 4.2: Number of Diameter Messages

Interface	CPU Utilization			
	25 %	50 %	70 %	90 %
S6a	19067	40749	59366	76783
Cx	21234	45102	64098	84131



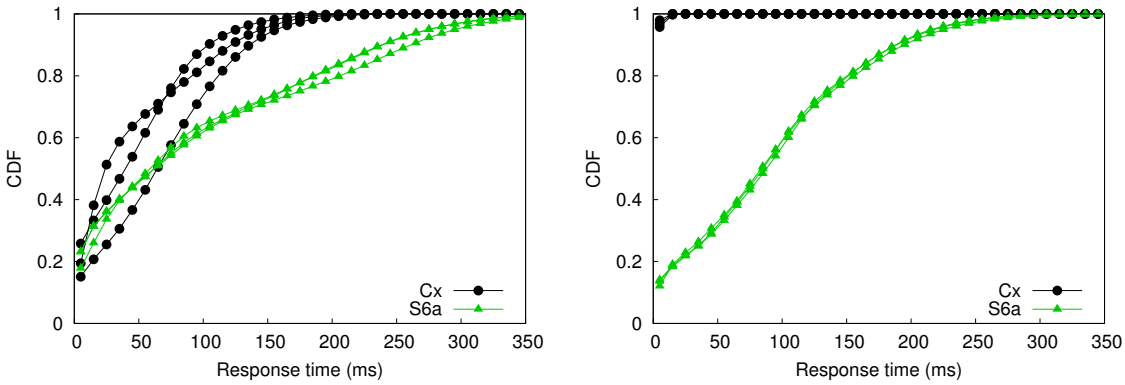
(a) Full HSS-FE (b) Split HSS-FE

Figure 4.7: Response time CDF obtained when aggregating all samples for each experiment

Then, we investigate the performance at the level of interfaces. Finally, we evaluate the performance at the finest granularity of a message.

HSS-level Analysis: We plot the Cumulative Distribution Function (CDF) of the response time for each experiment of full and split HSS-FE cases, in Figure 4.7(a) and Figure 4.7(b) respectively, for different R. By comparing the two figures, we notice that the results for low values of R are similar. However, significant differences can be noted for high values of R, with response time mainly concentrated around small values in the split case. This indicates a major shift in the performance when switching from the full HSS-FE to the split one, for high load scenarios. In the rest of the section, we focus on the R= 90%, representing the worst-case scenario.

Interface-level Analysis: Figure 4.8(a) and Figure 4.8(b) show the response time CDF when aggregating messages over the Cx and S6a interfaces separately, for the full and split HSS-FE cases, respectively. Over the Cx interface, we notice a massive shift of response time in the split HSS-FE setup towards very small values, indicating a significant improvement



(a) Full HSS-FE (b) Split HSS-FE

Figure 4.8: Response time CDF of all samples over the S6a and Cx interfaces, for R = 90% experiments

in the performance. As for the S6a interface, we notice that the portion of messages with low response time drops slightly, indicating a small decrease in performance. This lets us draw the following observation: in the case of high loads, splitting the HSS leads to a major improvement of the performance over the Cx interface that comes at the cost of a slight decrease in performance over the S6a interface. This behavior is due to performance isolation enabled through splitting the interfaces and, as a result, completely separating the corresponding traffics that present different characteristics, as we clarify next.

Message-level Analysis: In this part, we derive response time distributions, by considering for an experiment, all records for each type of message separately. We plot the results in Figure 4.9, using a candlestick representation. The candlestick shows the minimum, first quartile, average, third quartile, and maximum values. We notice the performance is better in the split case for various messages, except for AIR and PUR. This is due to the difference in the processing time of each message. In fact, ULR messages require exceptionally long processing time, greater than 100 ms, while all others require only a few ms. This is due to the implementation of ULR message which has much higher interactions with the database, compared to other messages. As a result, in the split setup, by isolating the Cx messages from ULR messages, we significantly reduce the time they spend in the queue. Over the S6a interface, this implies a slight increase in queuing time for AIR and PUR messages,

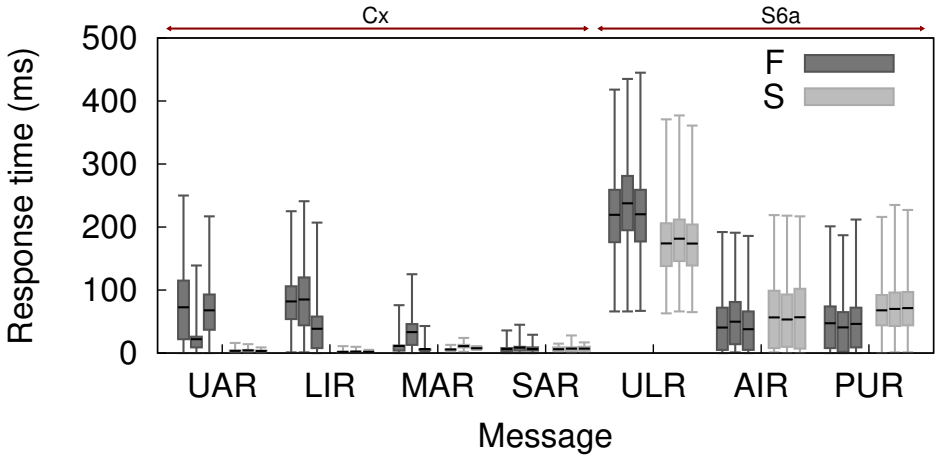


Figure 4.9: Response time distribution per message type (e), for one split (S) and one full (F) HSS-FE experiments with R = 90%

translating into a minor increase in the response time. Concerning the behavior of ULR messages, we record an unexpected improvement in their response time in the split case, due to the lower load they induce over the database compared to the full HSS-FE setup. More precisely, in the split HSS-FE setup, we have one VNF supporting the S6a interface, compared to two VNFs in the full HSS-FE setup. This is translated to a decrement in the number of ULR messages being processed in parallel, implying a lower load on the database and leading to a decrease in the response time.

4.6.2 Resource Usage

In this part, the resource usage of S6a and Cx in the full and split architectures of HSS-FE are evaluated under different workloads. Here, the workload represents the number of EPC and IMS scenarios executed by traffic generators in one second. Four workloads are considered (1) 100 IMS (i.e., 100 IMS scenario/second) / 50 EPC (i.e., 50 EPC scenario/second) (2) 200 IMS / 100 EPC, (3) 300 IMS / 150 EPC, and (4) 400 IMS / 200 EPC.

In these experiments, the traffic workload on HSS-FE is gradually increased until it reaches the desired level. When the average CPU utilization of HSS-FE violates a predefined threshold, the HSS-FE is scaled out by deploying a new container and registering it in the Diameter proxy. There is a minimum period of one minute between two consecutive scaling

Table 4.3: HSS-FE Scaling Policies

	Scaling Policy	CPU Utilization	
		Cx	S6a
Full (F) HSS-FE			
	F	60 %	60 %
Split (S) HSS-FE			
	S	60 %	60 %
	S2	80 %	60 %
	S3	60 %	80 %

operations. Four scaling policies are used as shown in Table 4.3. These policies cover two scenarios. The first scenario assumes that the S6a and Cx interfaces require the same scaling policy. For both full and split HSS-FE setups, a scaling action is triggered when the average CPU exceeds 60 % ("F" and "S" in Table 4.3). In the second scenario, the S6a and Cx interfaces require different CPU utilization thresholds for the scaling operation. In full HSS-FE, S6a and Cx interfaces are deployed and scaled as one unit. Hence, there is no way to apply different scaling policies on each interface. On the other hand, the split HSS-FE has higher flexibility, and different CPU thresholds can be applied to scale the interfaces. Thus, additional two scaling policies are considered for the split HSS-FE. In the first policy ("S2"), 80 % and 60 % average CPU are used to scale out the S6a and Cx interfaces, respectively. The second policy ("S3") considers 60 % and 80 % thresholds to scale out the S6a and Cx interfaces, respectively.

We plot the obtained results in Figure 4.10. The x-axis represents the scaling policies and workloads used in the experiments. Every four consecutive bars are the results for a particular workload, and each bar represents the number of containers used in the experiment for a certain scaling policy. The result shows that when the same CPU utilization threshold is used to scale out the S6a and Cx interfaces ('F" and "S"), the same number of containers is allocated to both full and split HSS-FE cases for three workloads whereas the full HSS-FE performs better in one case in which the split HSS-FE uses one more container (policy"S" for workload "300 IMS / 150 EPC"). We can also notice that for all workloads,

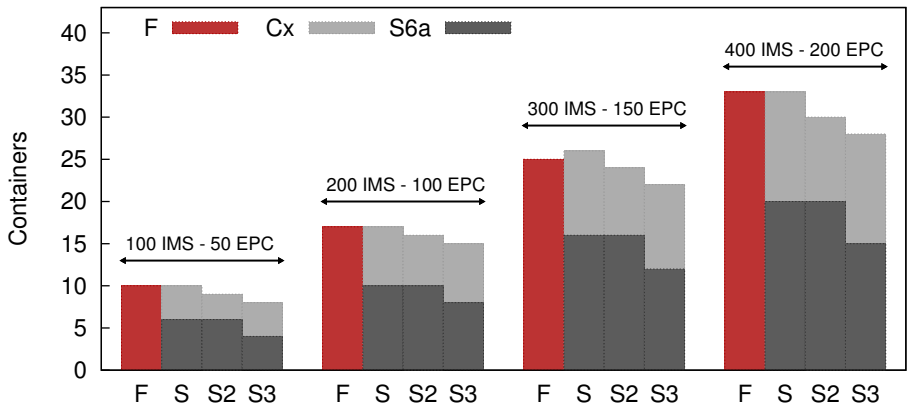


Figure 4.10: Number of deployed containers

when the scaling policies employ different CPU utilization rates to trigger the scaling operation, the split HSS-FE uses fewer containers than the full HSS-FE. The results allow us to draw the following observation. The performance isolation and flexibility obtained in the split HSS-FE may not be priceless when the same scaling policy is applied in the system as split HSS-FE may lead to allocating more resources compared to full HSS-FE. However, when different scaling policies are possible, the split HSS-FE architecture uses less resources than the full HSS-FE.

4.7 Conclusion

We proposed a novel VNPaaS architecture for provisioning 3GPP 4G and beyond network services. The architecture supports scalable and multi-orchestrator NFV MANO that employs two-layer hierarchical service orchestration approach. We also presented a realistic use case implementing HSSaaS with NFV-based architecture to validate the feasibility of MANO approach. We decomposed the HSS-FE into smaller VNFs according to the Diameter interfaces, enabling deployment optimization and performance isolation of these interfaces. Our experiments underlined the criticality of performance isolation. The numerical results showed that the traffic on S6a interface significantly degraded the performance of Cx interface at a high CPU utilization rate in the full HSS-FE architecture whereas it is not the case for the split HSS-FE.

Chapter 5

Joint Placement of NFV Orchestrator and VNF Manager: The Multi-Orchestrator Case

5.1 Introduction

Chapter 4 presented multi-orchestrator NFV management and orchestration architecture to address the scalability and performance challenges. There, the PoPs are grouped into domains and a domain orchestrator is placed in each domain to perform the resource orchestration and network service lifecycle management. We recall that the domain orchestrator supports the functions of NFVO and VNFM as defined in ETSI MANO framework. A GSO is used to deliver end-to-end network services across multiple domains. Moreover, the number of domains and domain orchestrators can be adjusted to meet the capacity and performance requirements of different deployment scenarios. However, it is crucial to find the structure of these domains as well as the number and placement of the orchestrators that provide the required capacity and performance.

In this chapter, we refine the system architecture and decompose the domain orchestrator into NFVO and VNFM functional blocks as it allows to scale the NFVO and VNFM

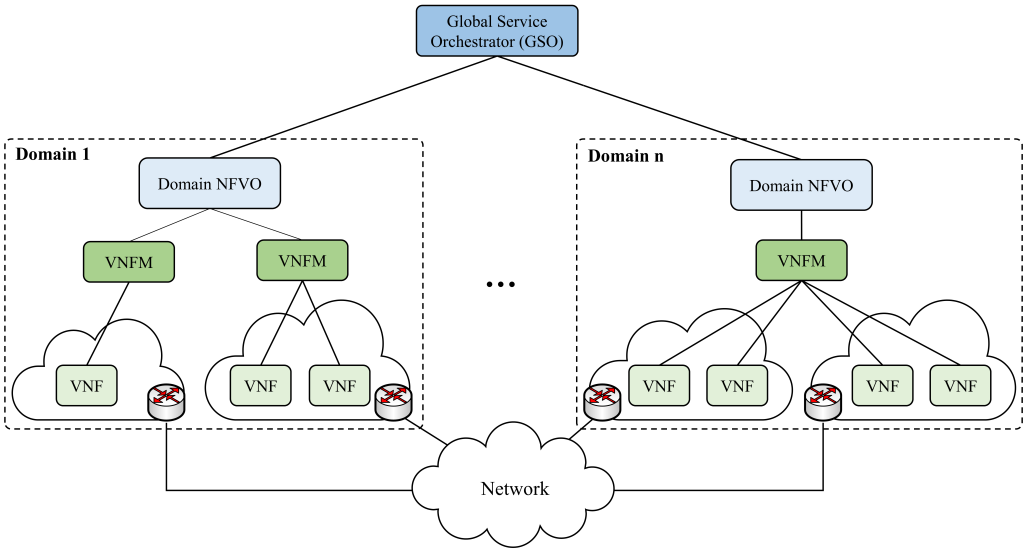


Figure 5.1: High-level system architecture

functions independently. Then, we introduce and study the joint placement of NFVO and VNFM. In particular, given the NFVI topology, a set of VNF instances, and the location of the GSO, we aim at finding the number and location of NFVOs and VNFMs needed in the system that minimizes their number, as it is a measure of the cost. We propose an ILP formulation of the problem. The formulation accounts for the capacity (e.g., NFVO and VNFM) and delays between various functional blocks to satisfy the scalability and performance requirements of the system. Besides, we propose two-step placement heuristic and evaluate it.

5.2 The Joint Placement of NFVO and VNFO

As shown in Figure 5.1, we consider a distributed NFVI consisting of multiple PoPs located in different regions. The NFVI is decomposed into a set of domains such that each domain consists of one or more PoPs. Then, in each domain, there are a domain NFVO and one or more VNFMs that perform the standard functions defined by ETSI NFV MANO framework. Besides, there is a single GSO responsible for end-to-end service orchestration across multiple domains.

The architecture allows adjusting the number of NFVOs and VNFMs to meet the scalability and performance requirements of different scenarios. However, there is still a challenge of determining their number and placement. We refer to this problem as the joint placement of NFVO and VNFM. It consists of finding (1) the optimal number and location of NFVOs needed in the system, (2) the PoPs assigned to each NFVO, i.e., PoPs in each domain, (3) the number and placement of VNFMs in each domain, and (4) the VNF instances assigned to each VNFM. Our objective is to minimize the number of NFVOs and VNFMs while fulfilling the capacity (e.g., NFVO and VNFM) and delay constraints.

Furthermore, for the sake of simplicity, we make the following assumptions: (1) a VNF instance and its EM are deployed at the same PoP, (2) a VIM manages resources for one PoP, and it is placed at that PoP.

5.2.1 System Model

Consider the NFVI modeled as a graph $G = (P, E)$ where P is the set of PoP nodes and E is the set of edges linking them, such that $E = \{(p, q) \mid p \in P, q \in P, p \neq q\}$. We use $\delta_{p,q}$ to represent the network delay of an edge $(p, q) \in E$. Let V represent the set of VNF instances in the system. The location of a VNF instance $v \in V$ is defined by $l_{v,p} \in \{0, 1\}$ such that $l_{v,p}$ equals to 1 only when v is placed at $p \in P$. We define M to represent the set of VNFMs that can be used to manage the VNF instances. We also use φ to denote the capacity of a VNFM. It represents the maximum number of VNF instances that can be managed by a VNFM. We consider that an NFVO has capacity defined in terms of the maximum number of VNF instances in its domain. We employ Φ to refer to this capacity. Moreover, we assume that the GSO is deployed at a given PoP. We define $w_p \in \{0, 1\}$ to indicate the GSO location, such that w_p is equal to 1 only if the GSO is placed at $p \in P$.

We consider that there is an upper bound on the acceptable network delay between various functional blocks to ensure predictable system performance. We use ψ and Ψ to denote the maximum acceptable delay between an NFVO on the one hand, the GSO and the VIM on the other hand. Moreover, the same VNFM can manage different VNF types (e.g., firewall) which can impose different requirements on the network delay over the

Table 5.1: Summary of Key Notations

Inputs	
$G(P, E)$	NFVI G with PoPs P and edges linking them E
E	The set of edges (i.e., logical communication links) in the network, $E = \{(p, q) \mid p \in P, q \in P, p \neq q\}$
$\delta_{p,q}$	Delay of edge $(p, q) \in E$
V	Set of VNF instances
$l_{v,p} \in \{0, 1\}$	$l_{v,p} = 1$ if VNF instance v is placed at $p \in P$
M	Set of VNFMs that can exist in the system
φ	Capacity of a VNFM
Φ	Capacity of an NFVO
$w_p \in \{0, 1\}$	$w_p = 1$ if the GSO is placed at $p \in P$
ψ	Maximum acceptable delay between GSO and domain NFVO
Ψ	Maximum acceptable delay between domain NFVO and VIM
Ω_v	Maximum acceptable delay between NFVO and VNFM managing the VNF instance v
ω_v	Maximum acceptable delay between VNF instance v and its designated VNFM
Decision Variables	
$x_p \in \{0, 1\}$	$x_p = 1$ if there is an NFVO placed at PoP $p \in P$
$r_{q,p} \in \{0, 1\}$	$r_{q,p} = 1$ if PoP $q \in P$ is assigned to an NFVO placed at $p \in P$
$x_{m,p} \in \{0, 1\}$	$x_{m,p} = 1$ if VNFM $m \in M$ is placed at PoP $p \in P$
$y_{v,m,p} \in \{0, 1\}$	$y_{v,m,p} = 1$ if VNF $v \in V$ is assigned to VNFM $m \in M$ placed at $p \in P$

VNFM reference points. Thus, we define the upper bound on network delay between the VNFM and other functional blocks per VNF instance. We use Ω_v to indicate the maximum acceptable delay between the NFVO and the VNFM assigned to VNF instance v. We also employ ω_v to denote the upper bound on the delay between the VNF instance v and its designated VNFM. Due to the assumption (2), ω_v also represents the maximum acceptable delay between the VNFM and the VIM of PoP where v is located. Table 5.1 lists the key notations used in this chapter.

5.2.2 Problem Formulation

We formally define the joint placement of NFVO and VNFM problem as follows.

Problem Definition: Given an NFVI G, a set of VNF instances V, a set of VNFMs M, and a fixed location of GSO; find the optimal number and location of NFVOs, the PoPs belong to each domain, the number and location of VNFMs in each domain, and the assignment of VNF instances to VNFMs such that the number of NFVOs and VNFMs is minimized, and without violating the capacity and delay constraints.

We mathematically formulate the problem as an ILP model. We define the decision variable h_p to represent the placement of the NFVOs.

$$h_p = \begin{cases} 1, & \text{if there is an NFVO placed at PoP } p \in P, \\ 0, & \text{otherwise.} \end{cases}$$

We also define the decision variable $r_{q,p}$ to map the PoPs to NFVOs and determine the structure of the domains.

$$r_{q,p} = \begin{cases} 1, & \text{if PoP } q \in P \text{ is assigned to an NFVO placed at } p \in P, \\ 0, & \text{otherwise.} \end{cases}$$

We further let $x_{m,p}$ to denote the location of the VNFM $m \in M$.

$$x_{m,p} = \begin{cases} 1, & \text{if VNFM } m \in M \text{ is placed at PoP } p \in P, \\ 0, & \text{otherwise.} \end{cases}$$

Finally, we denote the assignment of VNF instances to VNFMs by the decision variable define $y_{v,m,p}$.

$$y_{v,m,p} = \begin{cases} 1, & \text{if VNF } v \in V \text{ is assigned to VNFM } m \in M \text{ placed at } p \in P, \\ 0, & \text{otherwise.} \end{cases}$$

The objective function minimizes the number of NFVOs and VNFMs as defined in

equation (5.1), as it represents a measure of the operational cost of the NFV management and orchestration in the system.

$$Minimize \quad \sum_{p \in P} h_p + \sum_{m \in M} \sum_{p \in P} x_{m,p} \tag{5.1}$$

Now, we need to ensure that only one NFVO is in charge of resource orchestration of a PoP (i.e., a PoP belongs exactly to one domain). We represent this constraint as follows.

$$\sum_{p \in P} r_{q,p} = 1 \qquad \forall q \in P \tag{5.2}$$

We also should ensure that the PoP q can be assigned to the NFVO at PoP p only if there exists an active NFVO at p.

$$r_{q,p} \leq h_p \qquad \forall q, p \in P \tag{5.3}$$

Constraint (5.4) indicates that an NFVO should be placed within its domain boundaries.

$$r_{p,p} = h_p \qquad \forall p \in P \tag{5.4}$$

A VNFM can be placed only at one PoP. This constraint is expressed as follows.

$$\sum_{p \in P} x_{m,p} \leq 1 \qquad \forall m \in M \tag{5.5}$$

In constraint (5.6), we guarantee that each VNF instance is assigned to one VNFM.

$$\sum_{m \in M} \sum_{p \in P} y_{v,m,p} = 1 \qquad \forall v \in V \tag{5.6}$$

Constraint (5.7) stipulates that a VNF instance can be assigned to VNFM m placed at PoP p only when m is located at p.

$$y_{v,m,p} \leq x_{m,p} \qquad \forall v \in V, m \in M, p \in P \tag{5.7}$$

Next, we ensure that both a VNF instance and its designated VNFM exist in the same domain.

$$l_{v,q}\, y_{v,m,\acute{p}}\, r_{\acute{p},p} \leq r_{q,p} \qquad \forall v \in V, m \in M, q, \acute{p}, p \in P \tag{5.8}$$

We enforce the capacity constraints of NFVO and VNFM by (5.9) and (5.10).

$$\sum_{v \in V} \sum_{m \in M} \sum_{q \in P} y_{v,m,q}\, r_{q,p} \leq \Phi\, h_p \qquad \forall p \in P \tag{5.9}$$

$$\sum_{v \in V} y_{v,m,p} \leq \varphi\, x_{m,p} \qquad \forall m \in M, p \in P \tag{5.10}$$

Constraint (5.11) ensures that a VNFM is active only if it manages at least one VNF instance.

$$x_{m,p} \leq \sum_{v \in V} y_{v,m,p} \qquad \forall m \in M, p \in P \tag{5.11}$$

Constraints (5.12)-(5.15) enforce the delay limits in the system.

$$w_p\, h_q\, \delta_{p,q} \leq \psi \qquad \forall (p,q) \in E \tag{5.12}$$

$$r_{q,p}\, \delta_{p,q} \leq \Psi \qquad \forall (p,q) \in E \tag{5.13}$$

$$l_{v,p}\, y_{v,m,q}\, \delta_{p,q} \leq \omega_v \qquad \forall v \in V, m \in M, (p,q) \in E \tag{5.14}$$

$$y_{v,m,q}\, r_{q,p}\, \delta_{p,q} \leq \Omega_v \qquad \forall v \in V, m \in M, (p,q) \in E \tag{5.15}$$

Note that the constraints (5.8), (5.9) and (5.15) are non-linear constraints and can be linearized by replacing them with linear constraints (5.16)-(5.21) as follows.

$$l_{v,q}\, z_{v,m,\acute{p},p} \leq r_{q,p} \qquad \forall v \in V, m \in M, q, \acute{p}, p \in P \tag{5.16}$$

$$\sum_{v \in V} \sum_{m \in M} \sum_{q \in P} z_{v,m,q,p} \leq \Phi\, h_p \qquad \forall p \in P \tag{5.17}$$

$$z_{v,m,q,p}\, \delta_{p,q} \leq \Omega_v \qquad \forall v \in V, m \in M, (p,q) \in E \tag{5.18}$$

$$z_{v,m,q,p} \leq y_{v,m,q} \qquad \forall v \in V, m \in M, (p,q) \in E \tag{5.19}$$

$$z_{v,m,q,p} \leq r_{q,p} \qquad \forall v \in V, m \in M, (p,q) \in E \tag{5.20}$$

$$z_{v,m,q,p} \geq y_{v,m,q} + r_{q,p} - 1 \qquad \forall v \in V, m \in M, (p,q) \in E \tag{5.21}$$

5.3 Two-Step Placement Heuristic

We propose Two-Step Placement (TSP) heuristic to solve the overall problem, as illustrated in Algorithm 5.1. It begins by first decomposing the NFVI into one or more domains and placing a single NFVO in each domain. This step is performed through a tabu search heuristic which will be discussed later in section 5.3.1. However, in general, the heuristic aims to minimize the number of NFVOs in the system. It gives the solution of decision variables h_p and $r_{q,p}$ that satisfies the model constraints (5.2)–(5.4), (5.9), (5.12) and (5.13). Besides, this step disregards the placement of the VNFMs themselves. However, we assure that the solution would give the possibility for future VNFM placement to satisfy the VNFM delay constraints, i.e., constraints (5.14) and (5.15). To do so, we impose additional constraints on the solution to ensure that $\forall v \in V, \exists \acute{p} \in P$ such that:

$$l_{v,q}\, r_{q,p}\, r_{\acute{p},p}\, \delta_{q,\acute{p}} \leq \omega_v \qquad \forall q, p \in P \tag{5.22}$$

$$l_{v,q}\, r_{q,p}\, r_{\acute{p},p}\, \delta_{\acute{p},p} \leq \Omega_v \qquad \forall q, p \in P \tag{5.23}$$

The constraints (5.22) and (5.23) guarantee that for every VNF instance v, there exists a PoP $\acute{p}$ in the same domain where a VNFM can be placed to manage v while fulfilling the delay constraints. After that in the second step, for each domain, we place the needed VNFMs and map the VNF instances onto the VNFMs. We do that by utilizing the VNFM placement heuristic presented in chapter 3. This step provides the solution of decision variables $x_{m,p}$ and $y_{v,m,p}$. The obtained solution satisfies the model constraints (5.5)–(5.8), (5.10), (5.11), (5.14) and (5.15).

Algorithm 5.1: Two-Step Placement Heuristic

```
   /* Step One                                                              */
1  h_p, r_{q,p} ⟵ call NFVO_Placement()
   /* Step Two                                                              */
2  foreach p ∈ P do
3      if h_p = 1 then
4          compute VNFM placement inputs for this domain
5          S ⟵ call VNFM_Placement()
6          x_{m,p}, y_{v,m,p} ⟵ extract decision variables from solution S
7      end
8  end
9  return (h_p, r_{q,p}, x_{m,p}, y_{v,m,p})
```

5.3.1 NFVO Placement

We propose a tabu search heuristic to place the NFVOs and define the boundaries of their domains. In what follows, we present the key components of the proposed tabu search heuristic.

1. Initial solution: The heuristic starts with a simple initial solution where an NFVO is placed at each PoP in the system. The resulting solution may be infeasible, violating the delay constraint between GSO and NFVO. However, it provides a good enough starting solution that tabu search can improve gradually.

2. Neighborhood structure: We define two movements to transit from the current solution to a neighbor solution. The first one is to select an NFVO randomly and then invert its state, i.e., change from active to inactive and vice versa. The second movement is to draw a PoP randomly and reassign it to another NFVO chosen at random.

3. Tabu list: We use a tabu list of a fixed length to store the most recent moves made. A move in this list is called tabu move. The heuristic forbids tabu moves and does not select them while they are on the list unless certain criteria, known as aspiration criteria, are satisfied. In our work, we release a tabu move and accept it when the move leads to a solution better than the best-known solution.

4. Acceptance criteria: We relax the constraints (5.2)–(5.4), (5.9), (5.12), (5.13), (5.22) and (5.23) to allow the tabu search to explore the infeasible boundary. However, we assign

Table 5.2: Simulation Parameters

Parameter	Value
Number of PoPs ($\|P\|$)	8, 16
GSO location (w_p)	Central PoP
NFVO capacity (Φ)	20
VNFM capacity (φ)	10
Number of VNF instances ($\|V\|$)	10–60
Acceptable delay between GSO and NFVO (ψ)	80 ms
Acceptable delay between NFVO and VIM (Ψ)	60 ms
Acceptable delay between NFVO and VNFM managing VNF instances v (ω_v)	45 ms
Acceptable delay between VNFM and VNF instance v (Ω_v)	30 ms

a penalty for each solution to lead the heuristic to satisfy those constraints through the search process. We use two objectives in scoring a solution: solution penalty and number of NFVOs in the system. The heuristic aims first to minimize the solution penalty, then the number of NFVOs. We also employ a simple oscillation strategy in the solution evaluation. In each iteration, if there is a neighbor solution that has a better score than the best-found solution, then we choose it. Otherwise, we select a solution that minimizes the number of NFVOs, although it may not have the lowest penalty. Our goal is to drive the search to explore the infeasible solutions and thus induce diversification.

5. Stop criteria: The heuristic stops after $(4 \times |P|)$ consecutive iterations without an improvement in the solution. The formula allows the number of iterations to grow with respect to the number of PoPs. The multiplier 4 is adjusted experimentally.

5.4 Evaluation

In this section, we compare the performance of the TSP heuristic with the optimal solution obtained by solving the ILP model with CPLEX. In the following, we first describe the simulation setup, followed by numerical results. The simulation parameters are listed for convenience in Table 5.2.

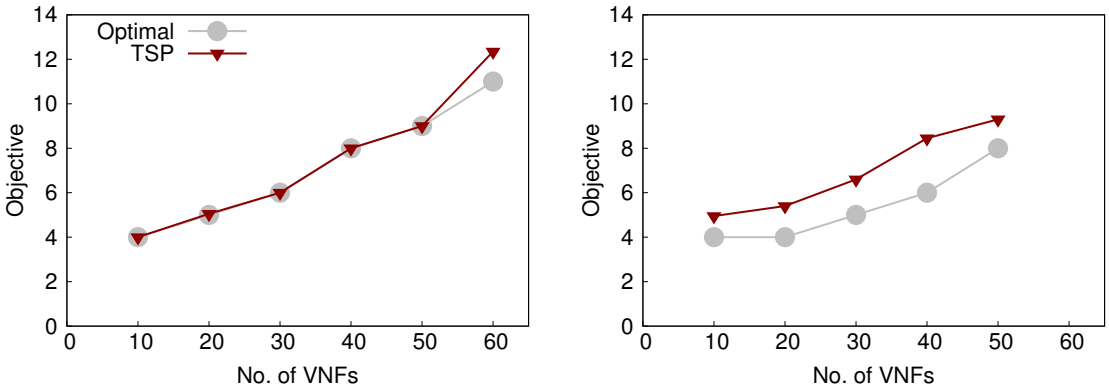
5.4.1 Simulation Setup

We simulate two different NFVIs with 8 and 16 PoPs. Each PoP represents an AT&T data center located in North America [86]. The inter-POP delays are the round-trip delay of ping packets between each pair of PoPs and obtained from public ping statistics [87]. The GSO is placed at a PoP with a central location in the NFVI structure. We also consider that the capacity of an NFVO and a VNFM are 20 and 10 VNF instances, respectively. In our experiments, the VNF instances are placed uniformly at random over PoPs, and their number varies from 10 to 60. Further, we assume that communication between the GSO and NFVO tolerates higher network delay compared to the communication between the functional blocks inside a domain. We further consider that the communication between the VNF instance and its designated VNFM is more sensitive to delay compared to the communication between other functional blocks inside a domain. Table 5.2 presents the maximum acceptable delay between various functional used in our simulation.

5.4.2 Numerical Results

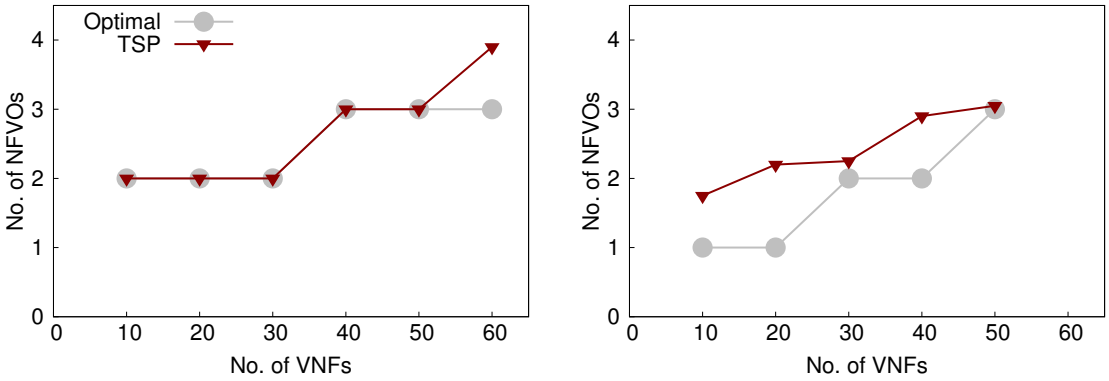
Figure 5.2 portrays the objective function value, i.e., the total number of NFVOs and VNFMs, of the optimal and TSP solutions for 8 and 16 PoPs. The TSP results are the average of 20 runs of each experiment. In Figure 5.2(a), we observe that TSP provides solutions that are very close to the optimal solution and attains the optimality in most cases. However, Figure 5.2(b) shows that TSP gives solutions that are within 1.4 times of the optimal solutions. Further, the results indicate that the number of NFVOs and VNFMs increases gradually with the increase of the number of VNF instances in the system.

For a better interpretation of the results, we provide the detailed number of NFVOs and VNFMs in Figure 5.3 and Figure 5.4 respectively. We can easily notice that the number of VNFMs grows at a higher rate compared to the number of NFVOs. The main reason is that an NFVO can accommodate more VNF instances than a VNFM as it has higher capacity. In general, the results point out that the NFV MANO capacity is adjusted to accommodate the number VNF instances in the system.



(a) 8 PoPs (b) 16 PoPs

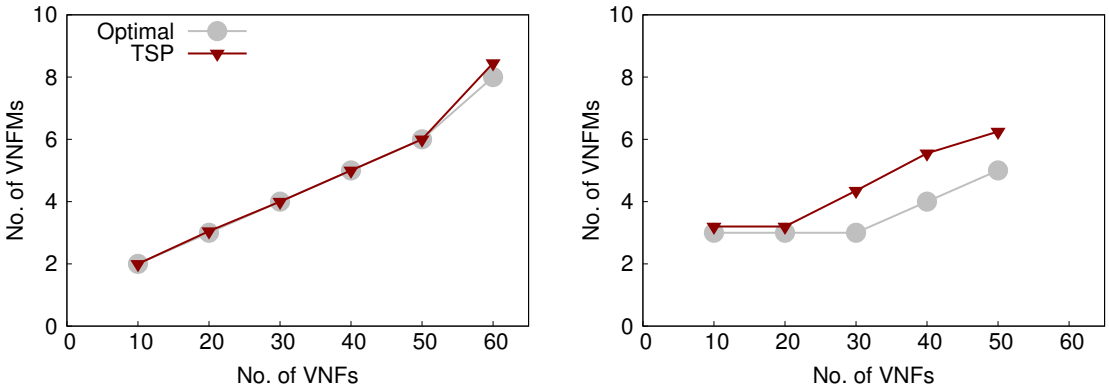
Figure 5.2: Objective function value



(a) 8 PoPs (b) 16 PoPs

Figure 5.3: Number of NFVOs

Moreover, although a single NFVO has adequate capacity to accommodate 10 VNF instances in our experiments, interestingly Figure 5.3(a) reports that two NFVOs are needed when the NFVI consists of 8 PoPs, whereas Figure 5.3(b) tells that single NFVO is sufficient for 10 VNF instances over 16 PoPs. This difference in the results is attributed to the number and location of PoPs. The system with 8 PoPs is constrained to a small number of PoPs which imposes the need of additional NFVO to satisfy the delay constraints in the system, whereas the NFVI of 16 PoPs provides the system with a higher degree of flexibility and allows fulfilling the delay constraints using one NFVO. Considering Figure 5.4, we notice that the system needs 2 and 3 VNFMs to manage 10 VNF instances over 8 and 16 PoPs respectively. We can thus conclude that number of PoPs and their location impact the



(a) 8 PoPs (b) 16 PoPs

Figure 5.4: Number of VNFMs

number of NFVOs and VNFMs needed in the system.

5.5 Conclusion

The number and location of NFVOs and VNFMs are vital to the scalability and performance of the NFV MANO. Thus, this chapter discussed the joint placement of NFVO and VNFM functional blocks in multi-orchestrator NFV MANO system. The problem consists of finding the number and placement of NFVOs, the structure of the domains, the number and location of VNFMs in each domain, and finally the assignment of the VNF instances to VNFMs. We aimed at minimizing the number of NFVOs and VNFMs as it represents a measure of the cost. We formulated the problem as ILP and proposed two-step placement heuristic. Our numerical results indicated that the number and geographical location of PoPs have an impact on the required number of NFVOs and VNFMs in the system.

Chapter 6

Joint Placement of NFV Orchestrator and VNF Manager: The Single and Multi-Orchestrator Cases

6.1 Introduction

The NFVO and VNFM communicate with each other, and with other functional blocks in ETSI NFV architectural framework (e.g., VIM, VNF and EM) to perform their functions such as network service instantiation, VNF instantiation, monitoring and fault management. Their location in the infrastructure impacts the delay experienced in their communications. A high delay increases the execution time of the MANO operations which decreases the scalability and degrades the performance. Hence, network operators are advised to plan their placement to minimize the communication overhead.

To address the above challenges, we revisit the joint placement of NFVO and VNFM

problem. We consider two cases. The first one is the centralized NFV MANO that employs one NFVO over an infrastructure of a single domain. The second case is the multi-orchestrator NFV MANO system, which is presented in this thesis, where the infrastructure is decomposed into domains and an NFVO is placed in each one. We call these problems the NFVO and VNFM functional blocks Placement in Single- and Multi-Domain environments. We refer to them by OMP-SO and OMP-MO, respectively. Our objective is to jointly optimize NFVO and VNFM placement to efficiently manage a given set of VNF instances by minimizing the total worst-case delay between various functional blocks. Precisely, given a certain NFVI and a set of VNF instances, then OMP-SO finds the optimal locations for a single NFVO and one or more VNFMs that minimize the total worst-case delay between the NFV functional blocks. On the other hand, OMP-MO seeks to select the best locations for a certain number of NFVOs and VNFMs so that total worst-case delay in the system is minimized.

The problems are formulated as Mixed Integer Linear Program (MILP) and implemented in CPLEX to find the optimal solutions. Given their complexity, we propose a multiple-walk Late Acceptance Hill-Climbing (LAHC) heuristic to solve the problems in reasonable time. LAHC is a local search metaheuristic that has been proposed recently [88]. The heuristic is hybridized with a strategic oscillation scheme to enable the diversification of search paths and go beyond the local optimum. Strategic oscillation is a diversification approach that is used in tabu search metaheuristic. Furthermore, we perform extensive simulation experiments to evaluate the proposed heuristic against the MILP implementation in terms of solution quality and execution time. We further investigate a possible impact for the number of NFVOs and VNFMs on the worst-case delay in the system.

6.2 The NFVO and VNFM Placement Problem

As depicted in Figure 6.1, we consider an NFVI that consists of several PoPs distributed in various geographical locations to provide the performance expected by different use cases. We assume that a local VIM manages the resources in each PoP. We represent the NFVI

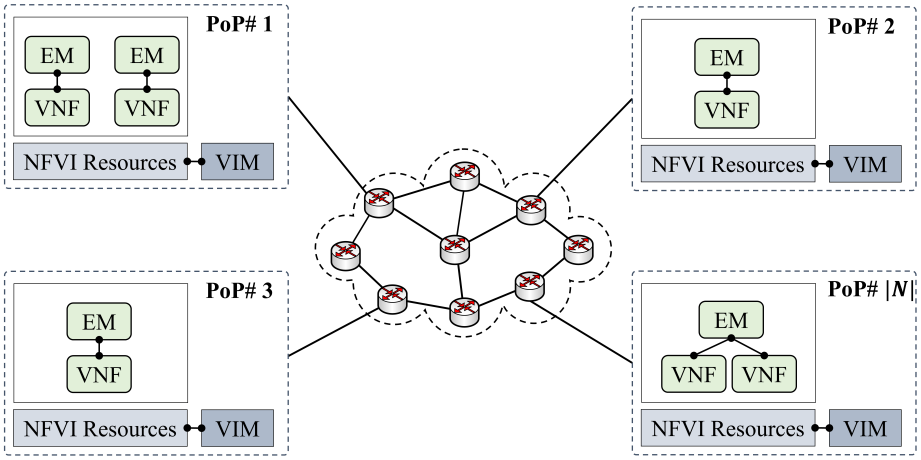


Figure 6.1: System model

as graph $G(N, L)$, where N is the set of PoP nodes and L is the set of links between them. We denote $D(n, \bar{n})$ as the delay of the shortest path between n and $\bar{n}$, for $n, \bar{n} \in N$. We use V to denote the set of VNF instances in the system. We also let $g_{v,n} \in \{0, 1\}$ be the location of the VNF instance $v \in V$.

$$g_{v,n} = \begin{cases} 1, & \text{if VNF instance } v \in V \text{ is located at PoP } n \in N, \\ 0, & \text{otherwise.} \end{cases}$$

For the sake of simplicity, we assume that a VNF and its EM are deployed at the same PoP, as illustrated in Figure 6.1. Moreover, NFVO and VNFM functional blocks usually have limited resources and hence they have limited processing capacity. We consider that the capacity of an NFVO is defined in terms of the maximum number of VNF instances in its domain. We employ Φ to refer to this capacity. We further use φ to refer to the capacity of a VNFM. It represents the maximum number of VNFs that can be assigned to a VNFM. Table 6.1 summarizes the key notation used in this chapter.

6.2.1 The Single-Orchestrator Case

In the centralized NFV MANO architecture, a single NFVO and one or more VNFMs are responsible for the lifecycle management of the network services and their constituent VNFs

Table 6.1: Summary of Key Notations

Inputs	
$G(N, L)$	NFVI G with PoPs N and links connecting them L
L	The set of links in the network, $L = \{(n, \bar{n}) \mid n \in N, \bar{n} \in N, n \neq \bar{n}\}$
$D(n, \bar{n})$	Delay of link $(n, \bar{n}) \in L$
V	Set of VNF instances
$g_{v,n} \in \{0, 1\}$	$g_{v,n} = 1$ if VNF instance $v \in V$ is placed at $n \in N$
φ	Capacity of a VNFM
Φ	Capacity of an NFVO
$w_n \in \{0, 1\}$	$w_n = 1$ if the GSO is placed at $n \in N$
$\mathcal{P}$	Number of NFVOs in multi-orchestrator case
$\mathcal{Q}$	Number of VNFMs
$\mathcal{M}$	A big enough positive constant
$\alpha, \beta, \gamma, \delta$	Weighting factors to adjust relative importance of delay components
Decision Variables	
$h_n \in \{0, 1\}$	$h_n = 1$ if there is an NFVO placed at PoP $n \in N$
$x_n \in \{0, 1\}$	$x_n = 1$ if there is any VNFM placed at $n \in N$
$y_{v,n} \in \{0, 1\}$	$y_{v,n} = 1$ if VNF $v \in V$ is assigned to a VNFM placed at $n \in N$
$r_{n,\bar{n}} \in \{0, 1\}$	$r_{n,\bar{n}} = 1$ if PoP $n \in N$ is assigned to an NFVO placed at PoP $\bar{n} \in N$
$m_n \in \mathbb{N}_0$	Number of VNFMs placed at PoP $n \in N$
$d^1 \in \mathbb{R}_0^+$	Worst-case delay between NFVO and VIM
$d^2 \in \mathbb{R}_0^+$	Worst-case delay between NFVO and VNFM
$d^3 \in \mathbb{R}_0^+$	Worst-case delay between VNFM and VNF instance
$d^4 \in \mathbb{R}_0^+$	Worst-case delay between GSO and NFVO

in an administrative domain, as shown in Figure 6.2. To manage a set of VNF instances efficiently, the OMP-SO problem seeks to select the optimal locations for an NFVO and a certain number of VNFMs as well as the assignment of VNF instances to VNFMs that minimize the total worst-case delay between the various functional blocks in the system. Three intra-domain delay components contribute to the objective: the delay between NFVO and VIM, NFVO and VNFM, and between VNFM and VNF instance.

We let $\mathcal{Q}$ be the number of VNFMs to be deployed in the system. Then, the OMP-SO can be formally defined as follows.

Problem Definition: Given an NFVI G, a set of VNF instances V, and number of VNFMs

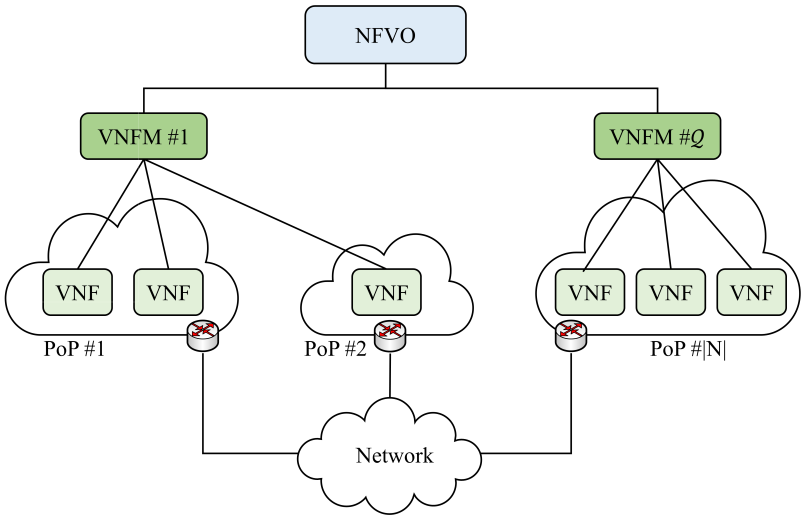


Figure 6.2: Single-orchestrator system architecture

$\mathcal{Q}$; find the optimal locations for an NFVO and $\mathcal{Q}$ VNFMs, and the assignment of VNF instances to VNFMs such that the total worst-case delay between various functional blocks is minimized, and without violating the capacity constraint of VNFM.

We formulate the OMP-SO problem as MILP. We define the decision variable h_n to represent the location of the NFVOs in the infrastructure.

$$h_n = \begin{cases} 1, & \text{if the NFVO is placed at } n \in N, \\ 0, & \text{otherwise.} \end{cases}$$

We further let the decision variable x_n to denote the location of the VNFMs.

$$x_n = \begin{cases} 1, & \text{if there is any VNFM placed at } n \in N, \\ 0, & \text{otherwise.} \end{cases}$$

We denote the number of VNFMs placed at PoP $n \in N$ by the decision variable $m_n \in \mathbb{N}_0$. We also define another variable $y_{v,n}$ to represent the assignment of the VNF instances to

the VNFMs.

$$y_{v,n} = \begin{cases} 1, & \text{if VNF instance } v \in V \text{ is assigned to a VNFM placed at } n \in N, \\ 0, & \text{otherwise.} \end{cases}$$

Finally, we define the decision variables $d^1, d^2, d^3 \in \mathbb{R}_0^+$ to represent the worst-case delay between NFVO and VIM, NFVO and VNFM, and between VNFM and VNF instance, respectively.

The objective of the OMP-SO problem is to minimize the weighted sum of the aforementioned delay components and can be expressed as follows, where α, β, γ are constants to adjust the relative significance of the different delay components.

$$\textit{Minimize} \quad \alpha\, d^1 + \beta\, d^2 + \gamma\, d^3 \tag{6.1}$$

Now, we need to guarantee that exactly one NFVO and $\mathcal{Q}$ of VNFMs are placed in the system. We represent these constraints as follows.

$$\sum_{n \in N} h_n = 1 \tag{6.2}$$

$$\sum_{n \in N} m_n = \mathcal{Q} \tag{6.3}$$

ETSI NFV MANO framework requires that every VNF instance is associated with a VNFM to manage its lifecycle. This constraint is expressed as follows.

$$\sum_{n \in N} y_{v,n} = 1 \qquad \forall v \in V \tag{6.4}$$

Next, we ensure that a VNF instance is assigned to a VNFM at PoP $n \in N$ only if there exists one at that PoP.

$$y_{v,n} \leq x_n \qquad \forall v \in V, n \in N \tag{6.5}$$

We also need to make sure that VNFMs are placed at PoP $n \in N$ only when there are

VNF instances assigned to that PoP.

$$x_n \leq \sum_{v \in V} y_{v,n} \qquad \forall n \in N \tag{6.6}$$

$$m_n \leq \mathcal{M}\, x_n \qquad \forall n \in N \tag{6.7}$$

Constraint (6.8) ensures that the VNFMs at a particular PoP have enough capacity to manage the assigned VNF instances.

$$\sum_{v \in V} y_{v,n} \leq \varphi\, m_n \qquad \forall n \in N \tag{6.8}$$

Finally, we present the delay constraints that measure the worst-case delay components, which are induced by the NFVO and VNFMs placement.

$$d^1 \geq h_n\, D(n, \bar{n}) \qquad \forall (n, \bar{n}) \in L \tag{6.9}$$

$$d^2 \geq h_n\, x_{\bar{n}}\, D(n, \bar{n}) \qquad \forall (n, \bar{n}) \in L \tag{6.10}$$

$$d^3 \geq g_{v,n}\, y_{v,\bar{n}}\, D(n, \bar{n}) \qquad \forall v \in V, (n, \bar{n}) \in L \tag{6.11}$$

We note that the constraint (6.10) is a non-linear constraint and can be linearized by replacing it with the following linear constraints.

$$d^2 \geq z_{n,\bar{n}}\, D(n, \bar{n}) \qquad \forall (n, \bar{n}) \in L$$

$$z_{n,\bar{n}} \leq h_n \qquad \forall (n, \bar{n}) \in L \tag{6.12}$$

$$z_{n,\bar{n}} \leq x_{\bar{n}} \qquad \forall (n, \bar{n}) \in L \tag{6.13}$$

$$z_{n,\bar{n}} \geq h_n + x_{\bar{n}} - 1 \qquad \forall (n, \bar{n}) \in L \tag{6.14}$$

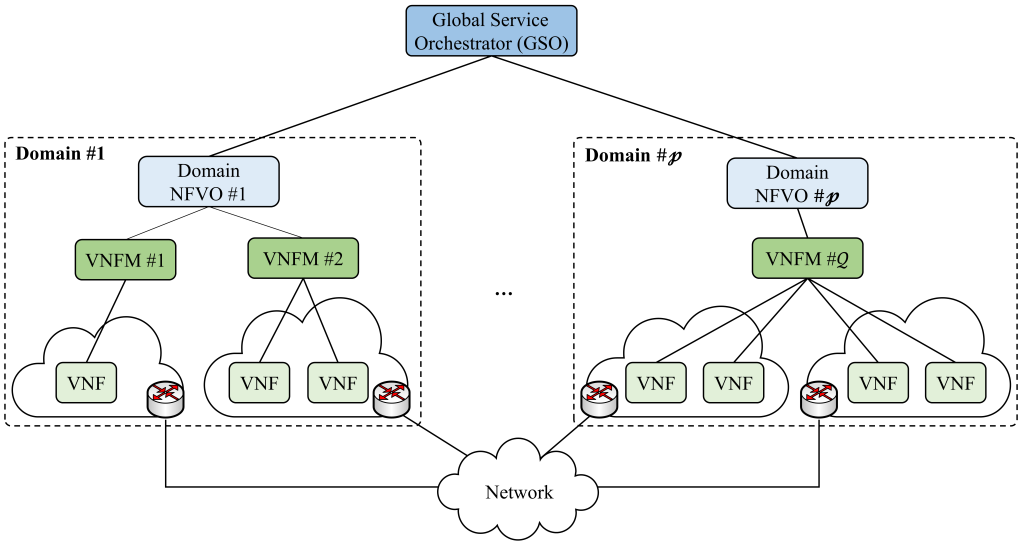


Figure 6.3: Multi-orchestrator system architecture

6.2.2 The Multi-Orchestrator Case

The multi-orchestrator case relies on the system architecture discussed in chapters 4 and 5. As shown in Figure 6.3, the infrastructure encompasses a set of domains; each domain consists of one or more PoPs and includes a single NFVO and one or more VNFMs. The GSO allows providing end-to-end service across multiple domains. We assume that the location of GSO is known and denoted by $w_n \in \{0, 1\}$.

$$w_n = \begin{cases} 1, & \text{if the GSO is placed at } n \in N, \\ 0, & \text{otherwise.} \end{cases}$$

The OMP-MO problem consists of finding the optimal structure for the domains (i.e., the PoPs belong to each domain) and selecting the locations for an NFVO and one or more VNFMs inside each domain so that the total-worst delay is minimized. The objective comprises four delay components: inter-domain delay between the GSO and NFVO of each domain, and the three intra-domain delay components considered by the OMP-SO problem, namely: delay between NFVO and VIM, NFVO and VNFM, and VNFM and VNF. We use $\mathcal{P}$ to denote the number of NFVOs, which also represents the number of domains in

the system since there is one NFVO in each domain. Further, it worth noting that in the multi-orchestrator environment, the number of VNFMs should be greater or equal to the number of NFVOs (i.e., $\mathcal{Q} \geq \mathcal{P}$).

The OMP-MO problem can be defined as follows.

Problem Definition: Given an NFVI G, a set of VNF instances V, location of GSO w_n, number of NFVOs $\mathcal{P}$, and number of VNFMs $\mathcal{Q}$; then decompose NFVI into $\mathcal{P}$ domains and find the PoPs belonging to each domain, the locations for $\mathcal{P}$ NFVOs and $\mathcal{Q}$ VNFMs, and assign VNF instances to VNFMs; such that the total worst-case delay between various functional blocks is minimized, while satisfying the capacity constraints of the NFVO and VNFM.

Before presenting our formulation, we introduce the decision variables. We define $r_{n,\bar{n}}$ to determine the structure of the domains by mapping the PoPs to NFVOs.

$$r_{n,\bar{n}} = \begin{cases} 1, & \text{if PoP } n \in N \text{ is assigned to an NFVO placed at } \bar{n} \in N, \\ 0, & \text{otherwise.} \end{cases}$$

We also define $d^4 \in \mathbb{R}_0^+$ to denote the worst-case delay between GSO and NFVO. Moreover, the objective can be stated as follows, where δ is a weighting factor to adjust the importance of the delay between GSO and NFVO.

$$\textit{Minimize} \quad \alpha\, d^1 + \beta\, d^2 + \gamma\, d^3 + \delta\, d^4 \tag{6.15}$$

The OMP-MO problem is subject to the constraints (6.3)–(6.8) defined in section 6.2.1 as well as the constraints (6.16)–(6.25) that will be presented in what follows.

The constraint (6.16) ensures that the number of NFVOs (and consequently the domains) is equal to $\mathcal{P}$.

$$\sum_{n \in N} h_n = \mathcal{P} \tag{6.16}$$

Next, we ensure that every PoP $n \in N$ belongs to one domain. In other words, only one NFVO is responsible for the resource orchestration of a particular PoP. We represent

this constraint as follows.

$$\sum_{\bar{n} \in N} r_{n,\bar{n}} = 1 \qquad \forall n \in N \tag{6.17}$$

The constraint (6.18) ensures that every PoP $n \in N$ is assigned to the NFVO at PoP $\bar{n} \in N$ only when there exists one at $\bar{n}$.

$$r_{n,\bar{n}} \leq h_{\bar{n}} \qquad \forall n, \bar{n} \in N \tag{6.18}$$

Further, we need to ensure that an NFVO is placed within its domain boundaries.

$$r_{n,n} = h_n \qquad \forall n \in N \tag{6.19}$$

We also need to guarantee that a VNF and its designated VNFM exist in the same domain.

$$g_{v,n_1}\, y_{v,n_2}\, r_{n_2,n_3} \leq r_{n_1,n_3} \qquad \forall v \in V, \forall n_1, n_2, n_3 \in N \tag{6.20}$$

The number of VNFs in each domain should not exceed the NFVO capacity. We express this constraint as follows.

$$\sum_{v \in V} \sum_{n \in N} y_{v,n}\, r_{n,\bar{n}} \leq \Phi\, h_{\bar{n}} \qquad \forall \bar{n} \in N \tag{6.21}$$

Lastly, the constraints (6.22)-(6.25) compute the worst-case delay components that contribute to the objective.

$$d^1 \geq r_{n,\bar{n}}\, D(n, \bar{n}) \qquad \forall (n, \bar{n}) \in L \tag{6.22}$$

$$d^2 \geq r_{n,\bar{n}}\, x_n\, D(n, \bar{n}) \qquad \forall (n, \bar{n}) \in L \tag{6.23}$$

$$d^3 \geq g_{v,n}\, y_{v,\bar{n}}\, D(n, \bar{n}) \qquad \forall v \in V, (n, \bar{n}) \in L \tag{6.24}$$

$$d^4 \geq w_n \, h_{\bar{n}} \, D(n, \bar{n}) \qquad \forall (n, \bar{n}) \in L \tag{6.25}$$

Note that the constraints (6.20), (6.21) and (6.23) are non-linear and can be linearized as follows.

$$g_{v,n_1} \, t_{v,n_2,n_3} \leq r_{n_1,n_3} \qquad \forall v \in V, \forall n_1, n_2, n_3 \in N \tag{6.26}$$

$$\sum_{v \in V} \sum_{n \in N} t_{v,n,\bar{n}} \leq \Phi \, h_{\bar{n}} \qquad \forall \bar{n} \in N \tag{6.27}$$

$$d^2 \geq k_{n,\bar{n}} \, D(n, \bar{n}) \qquad \forall (n, \bar{n}) \in L \tag{6.28}$$

$$t_{v,n,\bar{n}} \leq y_{v,n} \qquad \forall v \in V, (n, \bar{n}) \in L \tag{6.29}$$

$$t_{v,n,\bar{n}} \leq r_{n,\bar{n}} \qquad \forall v \in V, (n, \bar{n}) \in L \tag{6.30}$$

$$t_{v,n,\bar{n}} \geq y_{v,n} + r_{n,\bar{n}} - 1 \qquad \forall v \in V, (n, \bar{n}) \in L \tag{6.31}$$

$$k_{n,\bar{n}} \leq r_{n,\bar{n}} \qquad \forall (n, \bar{n}) \in L \tag{6.32}$$

$$k_{n,\bar{n}} \leq x_n \qquad \forall (n, \bar{n}) \in L \tag{6.33}$$

$$k_{n,\bar{n}} \geq r_{n,\bar{n}} + x_n - 1 \qquad \forall (n, \bar{n}) \in L \tag{6.34}$$

6.3 Proposed Heuristic

The problems of our interest correspond to the hierarchical facility location problem which is a generalization of uncapacitated facility location problem whose hardness is shown [89]. We propose a multiple-walk LAHC heuristic for finding a solution in a reasonable time frame. The multiple-walk is a metaheuristic parallelism approach that allows independent and concurrent explorations of the search space to improve heuristic robustness. Next, we present the LAHC heuristic in section 6.3.1, followed by the description of our parallel implementation in section 6.3.2.

6.3.1 Late Acceptance Hill-Climbing

LAHC is a new trajectory-based local search procedure that was recently proposed in [88]. It starts with a given initial solution (s_0) and iteratively modifies it to move from one solution to another in the search space. At each iteration, it identifies a candidate solution (s^*) in the neighborhood of the current solution (s_c) and evaluates it by an objective function (U) that measures the solution quality. If the objective function value meets the acceptance criteria, the candidate solution is accepted and replaces the current one. The search process stops when the termination criteria are met and returns the best solution found (s_b).

LAHC is similar to the hill-climbing algorithm but utilizes a new acceptance criterion to accept or reject a candidate solution, called late acceptance rule. This rule endeavors to escape a local optimum in the hope of finding a global one by allowing worsening moves, i.e., moves which worsen the objective function value, when a candidate solution is better than it was a number of iterations before. More precisely, LAHC tracks the search history by recording the objective function values of the previous current solutions in a fixed-length list, known as fitness array. Let $F_i = \{f_0, f_1, ..., f_{l-1}\}$ denote a fitness array of length l, where f_i is the objective function value of the current solution before i iteration. Then, at each iteration, LAHC compares the objective function value of candidate solution with the last element in the list (i.e., f_{l-1}) and accepts it if it is better. Each time a solution is accepted, the objective function value of the accepted solution is added to the beginning of the list, and the last element is removed from the end of the list. To avoid shifting all elements in the list, Burke et al. [90] propose a FIFO mechanism to maintain the list, wherein a virtual beginning of the list $\bar{v}$ at iteration i is computed by $\bar{v} = i \bmod l$. Thereafter, the candidate solution is compared to the objective function value $f_{\bar{v}}$ and if accepted the value of the candidate solution is assigned to $f_{\bar{v}}$.

Moreover, an improved variant of LAHC combines the late acceptance rule with a greedy rule that accepts a candidate solution that is similar or better than the current solution [90]. Our heuristic employs both acceptance rules and further incorporates a strategic oscillation mechanism, as we elaborate upon later. In what follows, we present the different components

Algorithm 6.1: Late Acceptance Hill-Climbing Heuristic

Input: Initial Solution s_o
Output: Best solution found s_b

$s_c \leftarrow s_0, s_b \leftarrow s_0$
$\forall q \in \{0, ..., l-1\}$: $f_q \leftarrow U(s_0)$
$i_{max} \leftarrow |N| + |V| + \mathcal{P} + \mathcal{Q} + 800$
$i_{idle} \leftarrow 0, i \leftarrow 0$
repeat
  $S \leftarrow \emptyset$
  $\bar{v} \leftarrow i \bmod l$
  repeat
    construct a candidate solution s^*
    calculate a candidate objective function solution $U(s^*)$
    if $U(s^*) < f_{\bar{v}}$ **or** $U(s^*) <= U(s_c)$ **then**
      $S \leftarrow S \cup \{s^*\}$
    end
    else if $R(s^*) < R(s_c)$ **or** $A(s^*) < A(s_c)$ **then**
      $S \leftarrow S \cup \{s^*\}$
    end
  until $|S| = \eta$
  $s_c \leftarrow$ **select** the best solution from S
  $f_{\bar{v}} \leftarrow U(s_c)$
  if $U(s_c) < U(s_b)$ **then**
    $s_b \leftarrow s_c$
    $i_{idle} \leftarrow 0$
  end
  else
    $i_{idle} \leftarrow i_{idle} + 1$
  end
  $i \leftarrow i + 1$
until $i_{idle} = i_{max}$
return s_b

of the proposed LAHC heuristic, which is also outlined in Algorithm 6.1.

1. Initial Solution: We use a greedy heuristic to generate a random initial solution (s_0). The heuristic starts by randomly placing single NFVO in the OMP-SO problem or $\mathcal{P}$ of NFVOs in the OMP-MO problem. After that, it forms the domains by assigning every PoP $n \in N$ to the closest NFVO. From there, the heuristic selects the locations for $\mathcal{Q}$ of VNFMs randomly but ensures that there is at least one VNFM in each domain. Lastly, it assigns every VNF instance $v \in V$ to the nearest VNFM inside same domain. The heuristic provides a solution that may be infeasible, violating NFVO and VNFM capacity constraints. However, it is a good enough starting solution that LAHC can improve gradually.

2. Neighborhood Structure: Our LAHC heuristic randomly selects a move among the

following types to modify the current solution and generate a candidate solution:

- **Relocate NFVO**: Select an NFVO randomly and move it to different PoP chosen at random.
- **Relocate VNFM**: Select a random VNFM and move it to different PoP chosen randomly.
- **Reassign PoP**: Randomly select a PoP and reassign it to different NFVO that is chosen at random.
- **Reassign VNF**: Choose a VNF instance at random and reassign it to different VNFM that is selected randomly.

3. Late Acceptance List: LAHC records the score of last accepted solutions in a list of fixed length l. In our heuristic, we set l to 10000. Further, since there is no search history at the beginning, the heuristic assigns the score of the initial solution to every element in the list.

4. Objective Function: The proposed heuristic seeks to minimize the objective function (U) that consists of three hierarchically (or lexicographically) structured objectives: penalty, primary objective, and auxiliary objective, as indicated by equation (6.35). We relax the constraints (6.8), (6.18), (6.19), (6.20) and (6.21) to allow the heuristic to explore a larger search space. However, the penalty function $P(s)$ penalizes the constraint violations and assigns a penalty to the solution proportional to the level of violations to drive the heuristic to satisfy the constraints during the search process. The primary objective function $R(s)$ corresponds to the model objective functions indicated by equations (6.1) and (6.15). Moreover, since many neighbor solutions score equally with respect to the primary objective, we define the auxiliary objective $A(s)$ to effectively drive the search to more interesting areas. It is defined as the total delay between every PoP $n \in N$ and the domain NFVO, and every VNF instance $v \in V$ and its designated VNFM.

$$\text{Minimize lix} \left(P(s), R(s), A(s) \right) \tag{6.35}$$

5. Acceptance Criteria: At each iteration, LAHC, as proposed in the literature [88, 90], identifies and evaluates one candidate solution. The solution is admitted and replaces the current solution when it satisfies the acceptance criteria. However, instead of replacing the current solution with the first admitted solution, our heuristic generates a small number (η) of admissible candidate solutions (i.e., solutions that meet acceptance criteria) and selects the best one to replace the current solution. The goal is to improve the quality of final solution by replacing current solution with a good-quality solution at each iteration. Furthermore, a candidate solution is accepted if it meets either the greedy rule or the late acceptance rule. In other words, the heuristic accepts a candidate solution if it is similar or better than the current solution (i.e., $U(s^*) \leq U(s_c)$) or if it is better than it was a number of iterations before (i.e., $U(s^*) < f_{\bar{v}}$). We also incorporate a simple strategic oscillation mechanism into the LAHC heuristic, which allows the search to cross the boundary between the feasible and the infeasible space and thus induce diversification. A candidate solution, which does not satisfy the greedy and late acceptance rules, is admitted if it improves the primary objective function or the auxiliary objective function, compared to the current solution (i.e., $R(s^*) < R(s_c)$ or $A(s^*) < A(s_c)$).

6. Termination Criteria: The search terminates when no further improvement can be made. Let i_{idle} denote the number of non-improving iterations, i.e., the number of iterations since the last obtained best solution. Then, our heuristic stops when i_{idle} reaches ($|N|+|V|+\mathcal{P}+\mathcal{Q}+800$) iterations. The formula allows i_{idle} to grow with respect to the infrastructure size and number of functional blocks in the system. The constant 800 is used to avoid the early termination of the heuristic.

6.3.2 Parallelization Approach

Parallel metaheuristics are increasingly being used to take advantage of the advancement in parallel computing and improve the performance, solution quality and robustness of heuristics without expensive effort in parameter tuning [91, 92]. Metaheuristic parallelization strategies are classified into two broad categories: single-walk and multiple-walk [91]. In a

Algorithm 6.2: Independent Multiple-Walk LAHC Heuristic

Input: A problem instance I, Number of search threads θ
Output: Best solution found s_b for I

```
for j ← 1 to θ do
    s_0 ← InitialSolution(I)
    create a search thread of LAHC(s_0)
end
run all search threads
wait search threads till they stop
s_b ← choose best solution found
return s_b
```

single-walk parallelization, the search procedure traverses a unique trajectory in the neighborhood graph. The search for the best neighbor at each iteration is performed in parallel, either by the parallelization of the neighborhood evaluation (i.e., objective function evaluation) or construction. This strategy seeks to accelerate the exploration of the search and reduce the running time. A multiple-walk parallelization simultaneously explores multiple trajectories, each of them by a search thread. These threads can be either independent or cooperative. In the case of independent multiple-walk, no communication takes place between the search threads whereas in the cooperative multi-walk the information collected along each trajectory is disseminated and used by other threads. This approach can improve the performance and solution quality.

Since our goal is to improve the solution quality, we adopt independent multiple-walk parallelization. We favor the independent approach due to its implementation simplicity. Algorithm 6.2 outlines the proposed multiple-walk LAHC heuristic. It launches multiple threads of the sequential LAHC (Algorithm 6.1). Each search thread starts with a different initial solution. Once all threads have stopped, the best overall solution is identified.

6.4 Numerical Results

We perform an extensive set of experiments to study how the number of NFVOs and VNFMs affect the worst-case delay in the system, and to evaluate the performance of multiple-walk LAHC heuristic against the MILP models presented in section 6.2. The heuristic is implemented in JAVA while the MILP models are implemented and solved in CPLEX

12.6.3. The experiments are executed on a server with 2×12-Core 2.20 GHz Intel Xeon E5-2650v4 CPUs and 128 GB memory. The heuristic and CPLEX are configured to use 32 threads. We run the heuristic 20 times for each experiment and report the average value. Moreover, in all experiments, unless stated differently, the weighting factors $(\alpha, \beta, \gamma, \delta)$ are set to 1. Next, we first present our simulation setup, followed by the results.

6.4.1 Simulation Setup

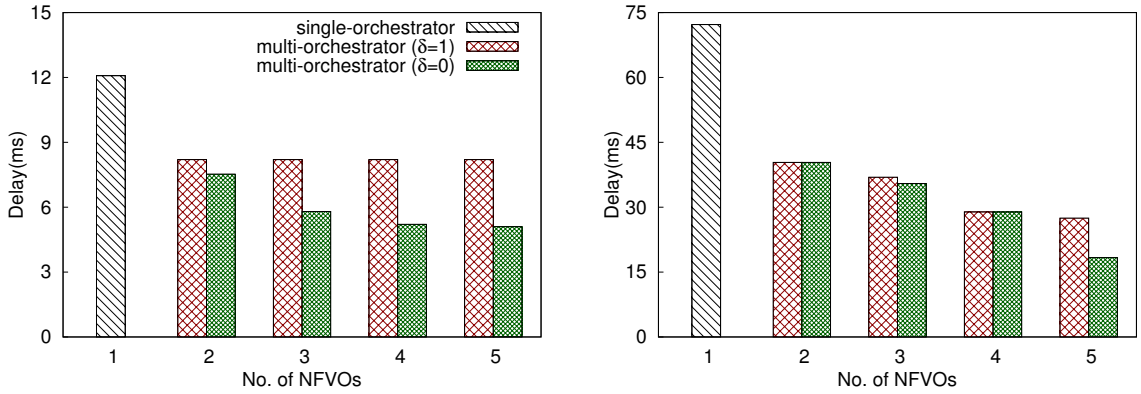
1. NFVI: Two NFVI topologies are designed using the public information available about the data centers of the AT&T and CDN77 providers. The AT&T NFVI topology consists of 26 PoPs located in North America [86] while the CDN77 topology encompasses 32 PoPs distributed among different locations around the globe [93]. Each PoP in our NFVIs represents a data center for the providers mentioned above. We use the latitude and longitude information of the cities in which the data centers are located to calculate the distance between the PoPs. The propagation delay over each link is calculated by dividing the distance between the PoPs by the speed of light over optical fiber (200,000 km/s).

2. NFV Functional Blocks: To assess the impact of the number of NFVOs ($\mathcal{P}$) on worst-case delay, we vary $\mathcal{P}$ from 2 to 5 in the multi-orchestrator problem. We also vary the number of VNFMs ($\mathcal{Q}$) to evaluate its effect on delay in both single- and multi-orchestrator problems. In the single-orchestrator case, $\mathcal{Q}$ is varied from 1 to 8 whereas in multi-orchestrator it is varied from $\mathcal{P}$ to 8. Recall that $\mathcal{Q}$ must be greater than or equal $\mathcal{P}$ in the multi-orchestrator environment. In all experiments, we fix the number of VNF instances ($|V|$) to be equal to the number of PoPs. Unlike the number of NFVOs and VNFMs, considering a larger number of VNF instances would not change the results of our experiments since it does not affect the objective function. It is also worth mentioning that these configurations are the largest that we could run the implementation of the MILP models in a manageable time. The execution time of the models in each experiment ranges from few seconds to 37 hours. Further, the GSO is placed at a PoP with a central location in the NFVI structure. The capacities of NFVO and VNFM are set to 200 and 50, respectively.

6.4.2 Impact of Number of NFVOs

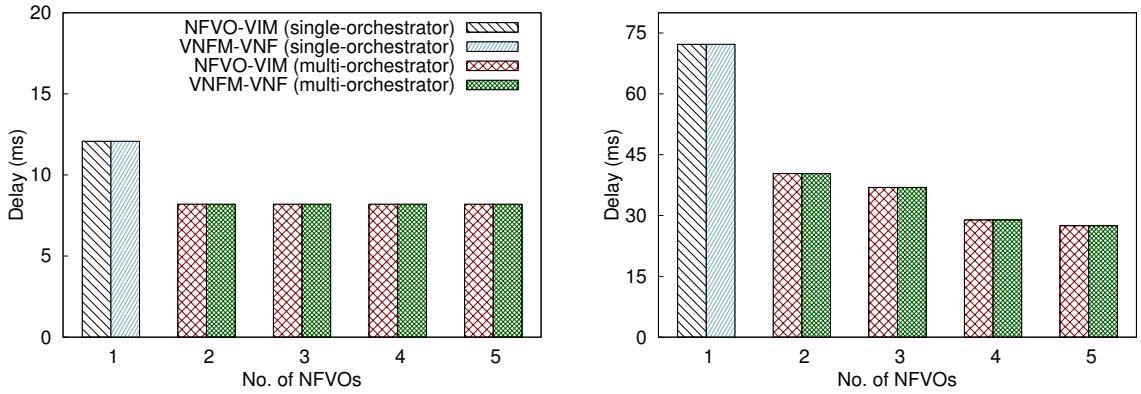
We start our analysis by evaluating the effect of the number of NFVOs on the worst-case delay. Figure 6.4 shows the worst-case delay (propagation delay) between the NFVO and VIM in the single- and multi-orchestrator problems for AT&T and CDN77 topologies. The results are illustrated with respect to the number of NFVOs considering $\delta = 1$ and $\delta = 0$. We observe that the worst-case delay is significantly higher in the single-orchestrator environment compared to multi-orchestrator, which is expected since the NFVI in the latter case is decomposed into smaller domains allowing the placement of NFVO closer to the PoPs. Moreover, when $\delta = 1$, Figure 6.4(a) indicates that two NFVOs are sufficient to yield the minimum worst-case delay and no improvement made after that. As for CDN77 topology, we notice in Figure 6.4(b) that the delay decreases gradually with increasing of number of NFVOs in the system. The reason why the worst-case delay does not improve after two NFVOs in AT&T topology is that the NFVO placement in the multi-orchestrator case is constrained by two delay components that contribute to the objective function: one with GSO and the other with VIM. The placement that minimizes one of these delay components can inversely affect the other. Consequently, and keeping in mind that the objective is to minimize the total worst-case delay, adding additional NFVOs in the system does not necessarily mean a lower delay between NFVO and VIM when $\delta = 1$. To give a more comprehensive comparison, we conduct the same set of experiments with $\delta = 0$, so that delay between GSO and NFVO does not contribute to the objective function. Now, we observe the worst-case delay continues to improve with increasing of number of NFVOs.

Furthermore, minimizing the worst-case delay between the NFVO and VIM is manifested as a lower intra-domain delay, i.e., lower worst-case delay between other functional blocks in the domain. For instance, Figure 6.5 presents a comparison of worst-case delay between NFVO and VIM, and between VNFM and VNF, with respect to the number of NFVOs. We observe that the improvement in the delay between NFVO and VIM is translated into improvement of delay between VNFM and VNF. In fact, the results demonstrate that worst-case delay between VNFM and VNF is equal to the one between NFVO and VIM.



(a) AT&T (b) CDN77

Figure 6.4: Worst-case delay between NFVO and VIM



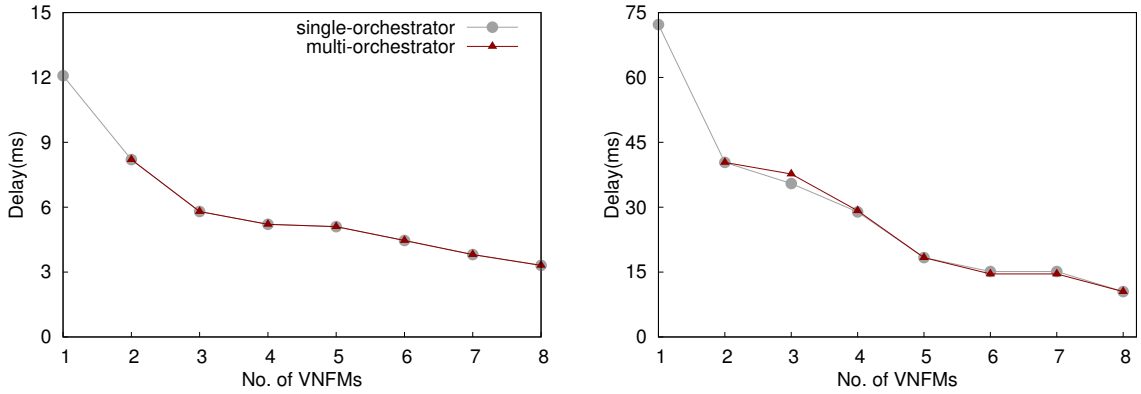
(a) AT&T (b) CDN77

Figure 6.5: A comparison of worst-case delay between NFVO and VIM, and between VNFM and VNF

The rationale is that when $\gamma = 1$, our models tend to place NFVO and VNFM functional blocks at the same PoP to minimize the objective function, which we elaborate on later.

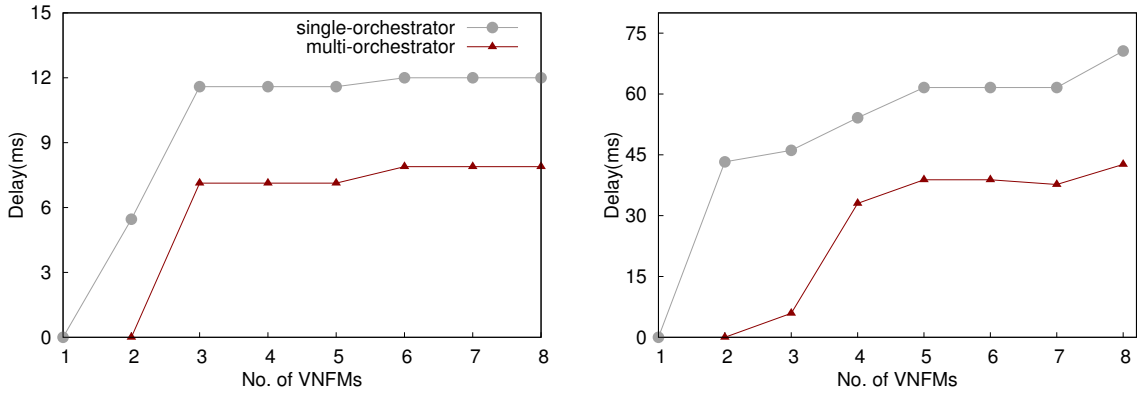
6.4.3 Impact of Number of VNFMs

Next, we look at how the number of VNFMs impacts the worst-case delay between the NFVO and VNFM, and between VNFM and VNF instance. As mentioned earlier, when $\gamma = 1$, the models always place the NFVO and VNFM at the same PoP in order to minimize the objective function, which means that the worst-case delay between NFVO and VNFM is



(a) AT&T (b) CDN77

Figure 6.6: Worst-case delay between VNFM and VNF functional blocks. The results are derived with $\gamma = 4$ and two NFVOs in the multi-orchestrator case



(a) AT&T (b) CDN77

Figure 6.7: Worst-case delay between NFVO and VNFM

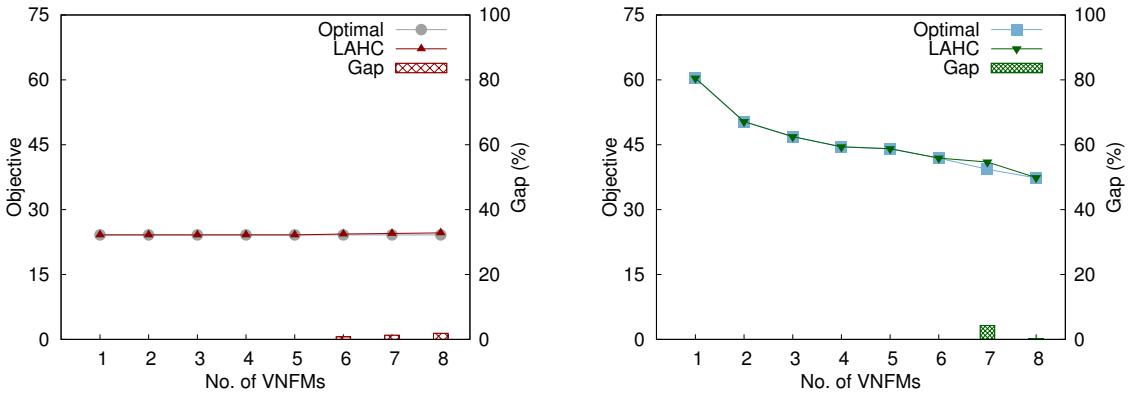
always 0, no matter how many VNFMs are deployed, and the worst-case delay between the VNFM and VNF is equal to the delay between NFVO and VIM. We thus perform another set of experiments in which we vary the number of VNFMs and set $\gamma = 4$, which is a big enough coefficient to prioritize the minimization of the delay between the VNFM and VNF instance. We plot the worst-case delay between NFVO and VNFM, and between VNFM and VNF in Figure 6.6 and Figure 6.7, respectively. The results of the multi-orchestrator experiments are obtained by considering two NFVOs in the system. In Figure 6.6, we notice that the worst-case delay decreases gradually with increasing number of VNFMs. We can also

observe that the results of single-orchestrator and multi-orchestrator experiments are equal in the majority of the experiments, which indicates that the number of NFVOs does not have a significant effect on the delay between VNFM and VNF under these configurations. However, considering Figure 6.7, the results show that the delay between NFVO and VNFM is always smaller in a multi-orchestrator compared to a single-orchestrator, which is due to the fact that the intra-domain delay is lower in the multi-orchestrator environment. The results also point out that worst-case delay increases with the increase of number VNFMs. The reason is the inverse relation between the delay between NFVO and VNFM, on the one hand, and the delay between VNFM and VNF, on the other hand.

6.4.4 Heuristic Performance

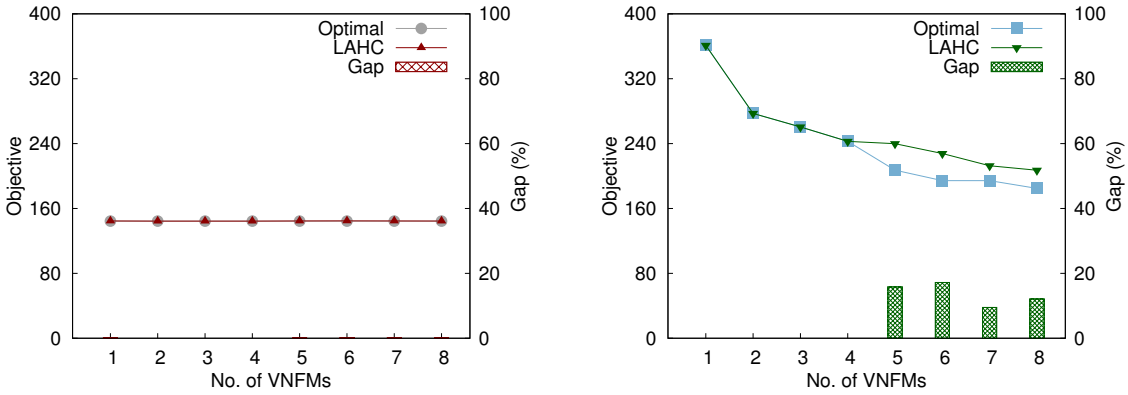
Finally, we evaluate the performance of the multiple-walk LAHC heuristic in terms of solution quality and execution time under different settings. Figures 6.8 and 6.9 depict the objective function values of the heuristic and optimal solutions for AT&T and CDN77 topologies in the single-orchestrator problem. The figures also show the average gap between the heuristic and optimal solutions. The results are derived with respect to a varying number of VNFMs for $\gamma = 1$ and $\gamma = 4$. We observe that LAHC yields high-quality solutions and reaches optimality in many experiments. In Figure 6.8, we notice that the average gap remains smaller than 4.3 %. Figure 6.9 shows that the heuristic also produces high-quality solutions and the average gap is smaller than 17.2 %. Further, Figures 6.10 and 6.11 present the objective function values in the multi-orchestrator case. The results are illustrated for a varying number of NFVOs and 8 of VNFMs. The results show that the heuristic produces solutions with an optimality gap of at most 14 %.

Now, we look at the execution time of the heuristic. Table 6.2 compares the average execution time of the heuristic and CPLEX for the CDN77 topology in multi-orchestrator case. The results are obtained considering three NFVOs in the system. We notice that the heuristic outperforms CPLEX by many orders of magnitude. From these results, we can see that the heuristic provides solutions within 82.8–100 % of optimality in several orders of magnitude faster than the models.



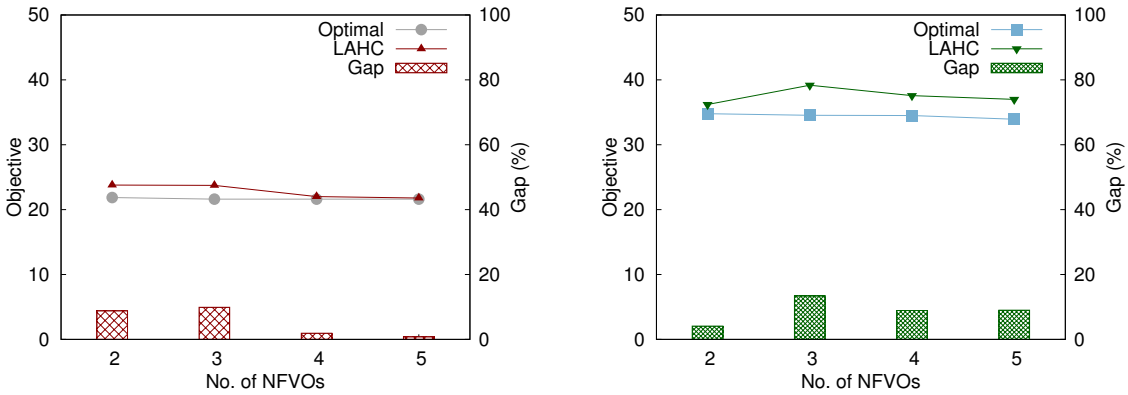
(a) $\gamma = 1$ (b) $\gamma = 4$

Figure 6.8: Objective function value for AT&T topology in single-orchestrator case



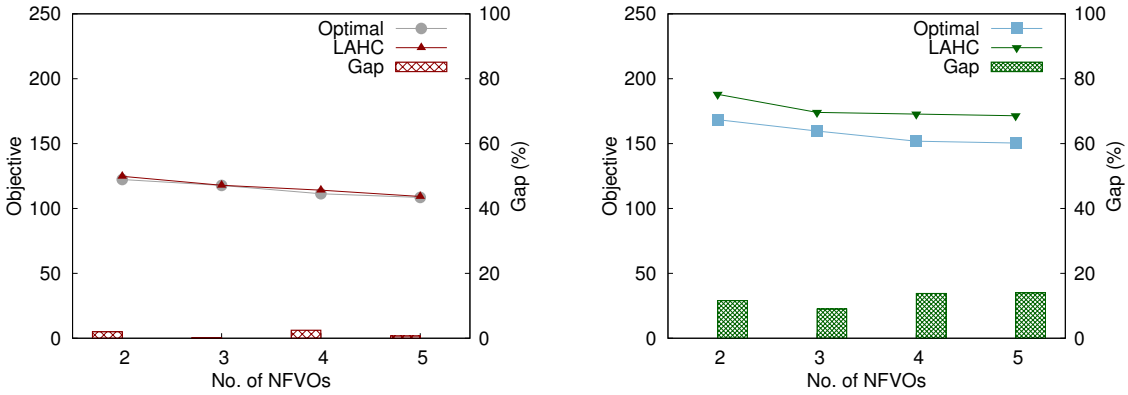
(a) $\gamma = 1$ (b) $\gamma = 4$

Figure 6.9: Objective function value for CDN77 in single-orchestrator case



(a) $\gamma = 1$ (b) $\gamma = 4$

Figure 6.10: Objective function value for AT&T topology in multi-orchestrator case



(a) $\gamma = 1$ (b) $\gamma = 4$

Figure 6.11: Objective function value for CDN77 topology in multi-orchestrator case

Table 6.2: Execution Time (s) for multi-orchestrator case

No. of VNFMs	CPLEX	LAHC
3	16 246.4	1.3
4	8214.2	1.5
5	3499.3	1.9
6	10 718.1	2.1
7	8975.7	2.1
8	3978.8	2.2

6.5 Conclusion

This chapter discussed the problem of NFVO and VNFM placement for single- and multi-orchestrator cases. Our objective was to minimize the total-worst case delay between NFV functional blocks by selecting the best locations for NFVO and VNFM over a distributed NFVI. We mathematically formulated the problems and proposed multiple-walk LAHC heuristic to solve it in a reasonable time. The LAHC heuristic was hybridized with strategic oscillation mechanism to diversify the search paths and improve solution quality. We conducted extensive simulation experiments to evaluate the proposed heuristic and evaluate the impact of the number of NFVOs and VNFMs on the worst-case delay in the system. Overall, the numerical results showed that our heuristic provided solutions within 82.8–100 % of optimality in many orders of magnitude faster than models. The results also indicated that the worst-case delay in the multi-orchestrator case is lower than in the single-orchestrator case.

Chapter 7

Conclusion and Future work

This thesis addressed multiple challenges associated with NFV MANO in distributed and large-scale deployments. Design aspects such as workload and network delay lead to scalability and performance issues that are important to consider in designing and operating NFV MANO platforms. We approached these challenges in two complementary ways. First, two placement problems are introduced targeting the NFVO and VNFM; the functional blocks responsible for the lifecycle management of network services and VNFs. The placement of these functional blocks is selected to guarantee that they have enough capacity to handle the workload and the network delay between the functional blocks in NFV system is tolerable. Second, an architecture was proposed for scalable and multi-orchestrator NFV management and orchestration system.

Chapter 3 of this thesis presented the VNFM placement problem that seeks to find the number and placement of the VNFMs, at each moment, that are required to manage a set of VNF instances. We presented a mathematical model with the objective of minimizing the operational cost. We also proposed tabu search heuristic to solve the problem. Through numerical analysis, we showed that the dynamic VNFM placement yields significant reductions in cost compared to the static placement. Moreover, we examined the impact of VNFM architectural option and NFVO location on the operational cost. The results indicated that the usage of generic-VNFM results in lower operational cost compared to the specific-VNFM approach. The results also showed that the location of NFVO might have

a notable impact on operational cost and thus it requires adequate tuning.

The centralized MANO is prone to suffer from scalability issue due to the fact of centralizing the resource and network service orchestration at a single NFVO. Motived by that, chapter 4 presented a scalable and multi-orchestrator NFV MANO architecture. There, the PoPs are grouped into domains based on predefined policy. A domain orchestrator is placed in each domain to be in charge of resource and service orchestration inside the domain. A global service orchestrator ensures the delivery of end-to-end network services across multiple domains. We presented a proof-of-concept prototype to validate the feasibility of the approach. The architecture was illustrated through HSS use case. However, the HSS, similar to other mobile network functional entities, are not designed for the cloud environment. We thus redesigned the HSS by decomposing it into smaller functional blocks. The new architecture was implemented and evaluated. The experiments demonstrated that the new architecture enables the performance isolation between the HSS diameter interfaces.

In the multi-orchestrator system, the number of orchestrators and their location must be tuned to provide the capacity and performance needed in the system. Thus, chapter 5 addressed the joint placement of NFVO and VNFVM in the multi-orchestrator system. There, we aimed at finding the number and location of NFVOs and VNFMs that minimize their number, as it is a measure of the cost. We presented a two-step placement heuristic and evaluated it. The numerical results showed that number and location of PoPs have an impact on the required number of NFVOs and VNFMs in the system.

Chapter 6 revisited the joint placement of NFVO and VNFM considering both single- and multi-orchestrator MANO systems. However, this time, we aimed at minimizing the total worst-case delay between the various functional blocks in the system. A mathematical model was established and a late acceptance hill-climbing heuristic was proposed to solve the problem. Moreover, we showed that worst-case delay is lower in the multi-orchestrator system compared to the single-orchestrator. We also showed that increasing number of VNFMs does not necessarily lead to a lower delay between the VNFM and other functional blocks.

7.1 Future Work

This thesis presented significant contributions in the area of resource management for NFV management and orchestration. Yet, there exist several research directions for the future.

7.1.1 Distributed NFV Infrastructure Design

Through the thesis, we assumed that the resources of each PoP are managed by a local VIM, which is the common approach for deploying large-centralized data centers. However, NFV use cases, such as 5G, drive the efforts to move from large-centralized data centers to smaller ones massively distributed at the edge of the network. This raises the challenge of managing the resources distributed over a large number of PoPs, since having a local VIM per PoP may not be effective as it increases the complexity of NFV management and orchestration. It would be interesting to investigate the number and the placement of VIMs needed to manage the whole infrastructure.

7.1.2 NFVO and VNFM Placement

Chapters 5 and 6 presented static (offline) placement strategies for NFVO and VNFM. However, one issue network operators will have to face is that the NFV infrastructure will eventually be updated to include more PoPs. Besides, the number of VNF instances would grow to meet the growing demand. Thus, another important study can be to consider the online placement of NFVO and VNFM that supports infrastructure expansion and demand variation.

7.1.3 Resilience of NFV Management and Orchestration

Given the criticality of NFV MANO, resilience is a key aspect to consider in designing NFV MANO platform. It is important that the system is designed to tolerate failures in network links and MANO functional blocks. The presented placement problems can be extended to improve resilience. Since the general approach to building fault-tolerant systems is redundancy, the placement can be selected to maximize network links redundancy and

minimize the effect of a link failure. The placement problem can also be used to ensure that a primary and backup NFVOs are assigned to each domain.

Bibliography

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, *et al.*, "A View of Cloud Computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[2] Q. Duan and M. Toy, *Virtualized Software-Defined Networks and Services*. Boston: Artech House, Dec. 2016.

[3] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network Function Virtualization: Challenges and Opportunities for Innovations," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 90–97, Feb. 2015.

[4] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. D. Turck, and R. Boutaba, "Network Function Virtualization: State-of-the-Art and Research Challenges," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.

[5] W. John, G. Marchetto, F. Nemeth, P. Skoldstrom, R. Steinert, C. Meirosu, I. Papafili, and K. Pentikousis, "Service Provider DevOps," *IEEE Communications Magazine*, vol. 55, no. 1, pp. 204–211, Jan. 2017.

[6] ETSI, *Network Functions Virtualization (NFV); Architectural Framework*, V1.2.1, Dec. 2014.

[7] ETSI, *Network Functions Virtualisation (NFV); Management and Orchestration*, V1.1.1, Dec. 2014.

[8] R. Mijumbi, J. Serrat, J.-L. Gorricho, S. Latre, M. Charalambides, and D. Lopez, "Management and Orchestration Challenges in Network Functions Virtualization," *IEEE Communications Magazine*, vol. 54, no. 1, pp. 98–105, 2016.

[9] C. Rotsos, D. King, A. Farshad, J. Bird, L. Fawcett, N. Georgalas, M. Gunkel, K. Shiomoto, A. Wang, A. Mauthe, N. Race, and D. Hutchison, "Network service orchestration standardization: A technology survey," *Computer Standards & Interfaces*, SI: Standardization SDN&NFV, vol. 54, no. Part 4, pp. 203–215, Nov. 2017.

[10] ETSI, *Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV*, V1.3.1, Jan. 2018.

[11] ETSI, *Network Functions Virtualisation (NFV); Management and Orchestration; Ve-Vnfm reference point - Interface and Information Model Specification*, V2.1.1, Oct. 2016.

[12] ETSI, *Network Functions Virtualisation (NFV); Management and Orchestration; Report on Architectural Options*, V1.1.1, Jul. 2016.

[13] R. Chayapathi, S. F. Hassan, and P. Shah, *Network Functions Virtualization (NFV) with a Touch of SDN*, English, 1 edition. Indianapolis, IN: Addison-Wesley Professional, Dec. 2016.

[14] ETSI, *Network Functions Virtualisation (NFV); Management and Orchestration; Or-Vnfm reference point - Interface and Information Model Specification*, V2.1.1, Oct. 2016.

[15] ETSI, *Network Functions Virtualisation (NFV); Management and Orchestration; Or-Vi reference point - Interface and Information Model Specification*, V2.1.1, Apr. 2016.

[16] ETSI, *Network Functions Virtualisation (NFV); Management and Orchestration; Vi-Vnfm reference point - Interface and Information Model Specification*, V2.1.1, Apr. 2016.

[17] ETSI, *Network Functions Virtualisation (NFV); Management and Orchestration; Os-Ma-Nfvo reference point - Interface and Information Model Specification*, V2.1.1, Oct. 2016.

[18] S. Abdelwahab, B. Hamdaoui, M. Guizani, and T. Znati, "Network function virtualization in 5G," *IEEE Communications Magazine*, vol. 54, no. 4, pp. 84–91, Apr. 2016.

[19] P. K. Agyapong, M. Iwamura, D. Staehle, W. Kiess, and A. Benjebbour, "Design Considerations for a 5G Network Architecture," *IEEE Communications Magazine*, vol. 52, no. 11, pp. 65–75, Nov. 2014.

[20] Recommendation ITU-R M.2083-0, *IMT Vision - Framework and overall objectives of the future development of IMT for 2020 and beyond*, Sep. 2015.

[21] ETSI, *Network Functions Virtualisation (NFV); Management and Orchestration*, V1.1.1, Dec. 2014.

[22] J. Garay, J. Matias, J. Unzilla, and E. Jacob, "Service Description in the NFV Revolution: Trends, Challenges and a Way Forward," *IEEE Communications Magazine*, vol. 54, no. 3, pp. 68–74, Mar. 2016.

[23] M.-A. Kourtis, M. J. McGrath, G. Gardikis, G. Xilouris, V. Riccobene, P. Papadimitriou, E. Trouva, F. Liberati, M. Trubian, J. Batallé, *et al.*, "T-NOVA: An Open-Source MANO Stack for NFV Infrastructures," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 586–602, 2017.

[24] G. Xilouris, E. Trouva, F. Lobillo, J. Soares, J. Carapinha, M. J. McGrath, G. Gardikis, P. Paglierani, E. Pallis, L. Zuccaro, *et al.*, "T-NOVA: A Marketplace for Virtualized Network Functions," in *Networks and Communications (EuCNC), 2014 European Conference on*, IEEE, 2014, pp. 1–5.

[25] S. Dräxler, H. Karl, M. Peuster, H. R. Kouchaksaraei, M. Bredel, J. Lessmann, T. Soenen, W. Tavernier, S. Mendel-Brin, and G. Xilouris, "SONATA: Service Programming

and Orchestration for Virtualized Software Networks," in *Communications Workshops (ICC Workshops), 2017 IEEE International Conference on*, IEEE, 2017, pp. 973–978.

[26] V. Sciancalepore, F. Giust, K. Samdanis, and Z. Yousaf, "A double-tier MEC-NFV Architecture: Design and Optimisation," in *Standards for Communications and Networking (CSCN), 2016 IEEE Conference on*, IEEE, 2016, pp. 1–6.

[27] B. García, M. Gallego, L. López, G. A. Carella, and A. Cheambe, "NUBOMEDIA: An Elastic PaaS Enabling the Convergence of Real-Time and Big Data Multimedia," in *Smart Cloud (SmartCloud), IEEE International Conference on*, IEEE, 2016, pp. 45–56.

[28] R. Vilalta, A. Mayoral, R. Muñoz, R. Casellas, and R. Martínez, "The SDN/NFV Cloud Computing Platform and Transport Network of the ADRENALINE Testbed," in *Network Softwarization (NetSoft), 2015 1st IEEE Conference on*, IEEE, 2015, pp. 1–5.

[29] *OpenStack Tacker*. [Online]. Available: `https://wiki.openstack.org/wiki/Tacker` (visited on 03/21/2018).

[30] *OpenMANO*. [Online]. Available: `https://osm.etsi.org/` (visited on 03/21/2018).

[31] *Open Baton*. [Online]. Available: `https://openbaton.github.io/` (visited on 03/21/2018).

[32] K. Katsalis, N. Nikaein, and A. Edmonds, "Multi-Domain Orchestration for NFV: Challenges and Research Directions," in *2016 15th International Conference on Ubiquitous Computing and Communications and 2016 International Symposium on Cyberspace and Security (IUCC-CSS)*, Dec. 2016, pp. 189–195.

[33] *Open Network Automation Platform (ONAP)*. [Online]. Available: `https://www.onap.org/` (visited on 11/27/2017).

[34] *ONAP Architecture (Amsterdam Release)*. [Online]. Available: `https://onap.readthedocs.io/en/latest/guides/onap-developer/architecture/onap-architecture.html` (visited on 11/27/2017).

[35] X. Li, R. Casellas, G. Landi, A. d. l. Oliva, X. Costa-Perez, A. Garcia-Saavedra, T. Deiss, L. Cominardi, and R. Vilalta, "5G-Crosshaul Network Slicing: Enabling Multi-Tenancy in Mobile Transport Networks," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 128–137, 2017.

[36] C. J. Bernardos, B. P. Gerö, M. Di Girolamo, A. Kern, B. Martini, and I. Vaishnavi, "5GEx: realising a Europe-wide multi-domain framework for software-defined infrastructures," *Transactions on Emerging Telecommunications Technologies*, vol. 27, no. 9, pp. 1271–1280, 2016.

[37] J. G. Herrera and J. F. Botero, "Resource Allocation in NFV: A Comprehensive Survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, 2016.

[38] S. Kim, S. Park, Y. Kim, S. Kim, and K. Lee, "VNF-EQ: dynamic placement of virtual network functions for energy efficiency and QoS guarantee in NFV," *Cluster Computing*, vol. 20, no. 3, pp. 2107–2117, 2017.

[39] D. Bhamare, M. Samaka, A. Erbad, R. Jain, L. Gupta, and H. A. Chan, "Optimal virtual network function placement in multi-cloud service function chaining architecture," *Computer Communications*, vol. 102, pp. 1–16, Apr. 2017.

[40] J. Cao, Y. Zhang, W. An, X. Chen, J. Sun, and Y. Han, "VNF-FG design and VNF placement for 5G mobile networks," *Science China Information Sciences*, vol. 60, no. 4, p. 040 302, 2017.

[41] C. Pham, N. H. Tran, S. Ren, W. Saad, and C. S. Hong, "Traffic-aware and Energy-efficient vNF Placement for Service Chaining: Joint Sampling and Matching Approach," *IEEE Transactions on Services Computing*, vol. PP, no. 99, pp. 1–1, 2017.

[42] H. Moens and F. De Turck, "VNF-P: A Model for Efficient Placement of Virtualized Network Functions," in *Network and Service Management (CNSM), 2014 10th International Conference on*, IEEE, 2014, pp. 418–423.

[43] M. Mechtri, C. Ghribi, and D. Zeghlache, "A Scalable Algorithm for the Placement of Service Function Chains," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 533–546, 2016.

[44] A. Hirwe and K. Kataoka, "LightChain: A Lightweight Optimisation of VNF Placement for Service Chaining in NFV," in *NetSoft Conference and Workshops (NetSoft), 2016 IEEE*, IEEE, 2016, pp. 33–37.

[45] L. Qu, C. Assi, K. Shaban, and M. Khabbaz, "Reliability-Aware Service Provisioning in NFV-enabled Enterprise Datacenter Networks," in *Network and Service Management (CNSM), 2016 12th International Conference on*, IEEE, 2016, pp. 153–159.

[46] M. Xia, M. Shirazipour, Y. Zhang, H. Green, and A. Takacs, "Network Function Placement for NFV Chaining in Packet/Optical Datacenters," *Journal of Lightwave Technology*, vol. 33, no. 8, pp. 1565–1570, 2015.

[47] M. Ghaznavi, A. Khan, N. Shahriar, K. Alsubhi, R. Ahmed, and R. Boutaba, "Elastic Virtual Network Function Placement," in *Cloud Networking (CloudNet), 2015 IEEE 4th International Conference on*, IEEE, 2015, pp. 255–260.

[48] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near Optimal Placement of Virtual Network Functions," in *Computer Communications (INFOCOM), 2015 IEEE Conference on*, IEEE, 2015, pp. 1346–1354.

[49] M. Bouet, J. Leguay, T. Combe, and V. Conan, "Cost-based placement of vDPI functions in NFV infrastructures," *International Journal of Network Management*, vol. 25, no. 6, pp. 490–506, 2015.

[50] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and Placing Chains of Virtual Network Functions," in *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*, IEEE, 2014, pp. 7–13.

[51] G. Wang, Y. Zhao, J. Huang, and W. Wang, "The Controller Placement Problem in Software Defined Networking: A Survey," *IEEE Network*, vol. 31, no. 5, pp. 21–27, 2017.

[52] B. Heller, R. Sherwood, and N. McKeown, "The Controller Placement Problem," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, ser. HotSDN '12, New York, NY, USA: ACM, 2012, pp. 7–12.

[53] M. T. I. ul Huque, W. Si, G. Jourjon, and V. Gramoli, "Large-Scale Dynamic Controller Placement," *IEEE Transactions on Network and Service Management*, vol. 14, no. 1, pp. 63–76, 2017.

[54] B. P. R. Killi and S. V. Rao, "Capacitated Next Controller Placement in Software Defined Networks," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 514–527, Sep. 2017.

[55] M. F. Bari, A. R. Roy, S. R. Chowdhury, Q. Zhang, M. F. Zhani, R. Ahmed, and R. Boutaba, "Dynamic Controller Provisioning in Software Defined Networks," in *Network and Service Management (CNSM), 2013 9th International Conference on*, IEEE, 2013, pp. 18–25.

[56] A. Sallahi and M. St-Hilaire, "Expansion Model for the Controller Placement Problem in Software Defined Networks," *IEEE Communications Letters*, vol. 21, no. 2, pp. 274–277, 2017.

[57] L. F. Müller, R. R. Oliveira, M. C. Luizelli, L. P. Gaspary, and M. P. Barcellos, "Survivor: an Enhanced Controller Placement Strategy for Improving SDN Survivability," in *Global Communications Conference (GLOBECOM), 2014 IEEE*, IEEE, 2014, pp. 1909–1915.

[58] S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, D. Hock, M. Jarschel, and M. Hoffmann, "Heuristic Approaches to the Controller Placement Problem in Large Scale SDN Networks," *IEEE Transactions on Network and Service Management*, vol. 12, no. 1, pp. 4–17, 2015.

[59] Q. Zhang, M. F. Zhani, S. Zhang, Q. Zhu, R. Boutaba, and J. L. Hellerstein, "Dynamic Energy-aware Capacity Provisioning for Cloud Computing Environments," in *Proceedings of the 9th International Conference on Autonomic Computing*, ser. ICAC '12, New York, NY, USA: ACM, 2012, pp. 145–154.

[60] ETSI, *Network Functions Virtualisation (NFV); Use Cases*, V1.1.1, Oct. 2013.

[61] A. Furno, D. Naboulsi, R. Stanica, and M. Fiore, "Mobile Demand Profiling for Cellular Cognitive Networking," *IEEE Transactions on Mobile Computing*, 2016.

[62] M. Vaezi and Y. Zhang, "Virtualizing the Network Services: Network Function Virtualization," in *Cloud Mobile Networks*, Springer, 2017, pp. 47–56.

[63] R. Z. Farahani and M. Hekmatfar, *Facility Location: Concepts, Models, Algorithms and Case Studies*, en. Springer Science & Business Media, Jul. 2009.

[64] N. Mladenović, J. Brimberg, P. Hansen, and J. A. Moreno-Pérez, "The p-median problem: A survey of metaheuristic approaches," *European Journal of Operational Research*, vol. 179, no. 3, pp. 927–939, 2007.

[65] S. Basu, M. Sharma, and P. S. Ghosh, "Metaheuristic applications on discrete facility location problems: A survey," *Opsearch*, vol. 52, no. 3, pp. 530–561, 2015.

[66] F. Glover, "Tabu search–part I," *ORSA Journal on computing*, vol. 1, no. 3, pp. 190–206, 1989.

[67] *WonderNetwork*. [Online]. Available: `https://wondernetwork.com/` (visited on 01/30/2017).

[68] J. Zheng, T. E. Ng, K. Sripanidkulchai, and Z. Liu, "Pacer: A Progress Management System for Live Virtual Machine Migration in Cloud Computing," *IEEE Transactions on Network and Service Management*, vol. 10, no. 4, pp. 369–382, 2013.

[69] *IBM CPLEX Optimizer*. [Online]. Available: `https://www.ibm.com/software/commerce/optimization/cplex-optimizer/` (visited on 06/05/2017).

[70] 3GPP TS 23.402, *Architecture enhancements for non-3gpp accesses*, Version 12.8.0, Release 12, Apr. 2015.

[71] 3GPP TS 23.401, *General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access*, version 12.9.0, Release 12, Jul. 2015.

[72] 3GPP TS 23.335, *LTE; User Data Convergence (UDC); Technical realization and information flows; Stage 2*, Version 13.0.0, Release 13, Jan. 2016.

[73] 3GPP TS 23.228, *IP Multimedia Subsystem (IMS); Stage 2*, Version 12.8.0, 2015.

[74] M. Vaezi and Y. Zhang, "Virtualizing the network services: Network function virtualization," in *Cloud Mobile Networks*, Springer, 2017, pp. 47–56.

[75] OASIS, *Topology and Orchestration Specification for Cloud Applications*, Version 1.0, Nov. 2013.

[76] X. Zhou, Z. Zhao, R. Li, Y. Zhou, T. Chen, Z. Niu, and H. Zhang, "Toward 5G: When Explosive Bursts Meet Soft Cloud," *IEEE Network*, vol. 28, no. 6, pp. 12–17, Nov. 2014.

[77] *OpenStack*. [Online]. Available: `https://www.openstack.org/software/` (visited on 11/24/2016).

[78] *Kubernetes*. [Online]. Available: `http://kubernetes.io/` (visited on 11/24/2016).

[79] *Docker*. [Online]. Available: `https://www.docker.com/` (visited on 11/24/2016).

[80] *Cloudify*. [Online]. Available: `http://getcloudify.org/index.html` (visited on 11/20/2016).

[81] *Etcd*. [Online]. Available: `https://coreos.com/etcd/` (visited on 11/24/2016).

[82] *Neo4j*. [Online]. Available: `https://neo4j.com/` (visited on 11/24/2016).

[83] *Elasticsearch*. [Online]. Available: `http://www.elastic.co` (visited on 11/24/2016).

[84] *InfluxDB*. [Online]. Available: `https://www.influxdata.com/` (visited on 11/24/2016).

[85] *OpenEPC*. [Online]. Available: `http://www.openepc.com/` (visited on 11/20/2016).

[86] *AT&T Data Center Locations*. [Online]. Available: `https://www.business.att.com/enterprise/Service/cloud/colocation/data-center-locations` (visited on 08/25/2017).

[87] WonderNetwork, *Global Ping Statistics*. [Online]. Available: `https://wondernetwork.com/pings` (visited on 08/25/2017).

[88] E. K. Burke and Y. Bykov, "A Late Acceptance Strategy in Hill-Climbing for Exam Timetabling Problems," in *7th international Conference on the Practice and Theory of Automated Timetabling (PATAT)*, Montreal, Canada, Aug. 2008.

[89] J. Krarup and P. M. Pruzan, "The simple plant location problem: Survey and synthesis," *European journal of operational research*, vol. 12, no. 1, pp. 36–81, 1983.

[90] E. K. Burke and Y. Bykov, "The late acceptance Hill-Climbing heuristic," *European Journal of Operational Research*, vol. 258, no. 1, pp. 70–78, Apr. 2017.

[91] V.-D. Cung, S. L. Martins, C. C. Ribeiro, and C. Roucairol, "Strategies for the Parallel Implementation of Metaheuristics," in *Essays and Surveys in Metaheuristics*, Boston, MA: Springer US, 2002, pp. 263–308.

[92] T. G. Crainic and M. Toulouse, "Parallel Meta-heuristics," in *Handbook of Metaheuristics*, M. Gendreau and J.-Y. Potvin, Eds., Boston, MA: Springer US, 2010, pp. 497–541.

[93] *CDN77 Data Center Locations*. [Online]. Available: `https://www.cdn77.com/network` (visited on 10/24/2017).