A Cost-driven Approach for Virtualized Network Services Life-cycle Management)

Nour El Houda Nouar

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ FÉDÉRALE TOULOUSE MIDI-PYRÉNÉES

Délivré par :
l'Université Toulouse 1 Capitole (UT1 Capitole)

Présentée et soutenue le *15/03/2022* **par :**

Nour El Houda NOUAR

A Cost-driven Approach for Virtualized Network Services Life-cycle Management

JURY

Isabelle BORNE	Professeur d'Université	Président du Jury
Jean-C LAPAYRE	Professeur d'Université	Rapporteur
Salima BENBERNOU	Professeur d'Université	Rapporteur
Toufik AHMED	Professeur d'Université	Rapporteur

École doctorale et spécialité :
MITT : Domaine STIC : Réseaux, Télécoms, Systèmes et Architecture
Unité de Recherche :
Laboratoire d'analyse et d'architecture des systèmes
Directeur(s) de Thèse :
Khalil DRIRA, *Noura FACI*, *Said TAZI* et *Sami YANGUI*

Acknowledgments

I first would like to thank the members of my jury for the time they spent on my work and for their helpful remarks.

I would like to thank my supervisors for these years during which they accompanied me and encouraged me in my thesis work.

I would like to thank all the present and past members of the SARA team for welcoming me there. Especially Olivier BRUN and Ali EL-AMINE, working with you has been very rewarding, both scientifically and humanly.

Thanks to all my friends, especially Raoua, with whom I shared these years, for helping me survive all the stress and for making the hours of depression of writing bearable and often even enjoyable. And especially thanks to *Fadel* for supporting me and being always present despite everything.

Lastly, my family deserves endless gratitude: my father and my mother for supporting and unconditionally loving me, and my sisters and brother, Rym, Ikram, and Hamza, you are my soulmates, I am grateful to have you in my life. To my family, I give everything, including this.

Abstract:

Network Function Virtualization (NFV) is widely expected to be a backbone of the future service providers by allowing to run Virtualized Network Functions (VNF) on top of generic Commercial-Off-The-Shelf (COTS) hardware, anytime and anywhere in the network. This revolutionary concept transforms how VNFs, along with their connectivity through the Network Service (NS) concept, are designed, deployed, and managed. Intuitively, the agility and cost-effectiveness aim to decrease CAPital EXpenditure (CAPEX) and OPrational EXpenditure (OPEX). However, in order to accomplish such goal, both technological, conceptual, and management advances are required.

At present, operational NS provisioning and management solutions are not mature enough. On the one hand, a lack of a shared understanding of VNF descriptions and automated discovery mechanisms, which can undoubtedly be observed due to the heterogeneity of technologies and providers, make the NS design phase non-efficient and time-consuming. On the other hand, following the standards specification, NS provision necessarily implies instantiating all the involved VNFs and configuring all connectivity mentioned in the description template. This rigid instantiation obliges NS providers to anticipate, compose, and configure all possible VNFs and connectivities. Moreover, optimal and automatic placement of the composed VNFs and mapping them to the available resources over the multi-domain environment while considering QoS and SLA constraints can be evenly essential.

This thesis intends to address the aforementioned practical challenges and propose using a standard and convenient domain language through ontological approaches to describe and discover NFV components. Further, we introduce a novel approach that enables dynamic and re-configurable wiring among VNF in a given NS. Besides, a lazy resources allocation solution to the Network Service Embedding problem is proposed. This solution optimizes both VNF placement for each network service along with traffic routing across VNFs while satisfying constraints like location and latency.

Keywords: ETSI-NFV; MANO; Network Service; Semantic description; Semantic discovery; Instantiation; Deployment; Management.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context

The newly introduced computing models (e.g., cloud computing, edge/fog computing) and technologies (e.g., virtualization, the fifth generation of telcos mobile network) enabled a myriad of novel applications that range from health care to intelligent transportation/manufacturing and multimedia [TeleGeography] (see Fig.1.1). This novel ecosystem relies on a specific business model where every single entity could be modeled, implemented, offered, and/or consumed as a service. Netflix, Airbnb, and Google are among the forceful examples. All of them operate massive infrastructure and content according to well-designed business processes and procedures to provide the prospective users with the required services in an agile and cost-effective fashion. For instance, Netflix is currently the first multimedia content host globally, while the company does not own any of that content. Similarly, Airbnb is currently the first hotel company in the world. It is basically the only provider that could propose accommodation in almost every single city in the world. However, Airbnb owns neither hotels nor any accommodation facilities on Earth. From operating point of view, all these companies operate according to the service computing model. From business point of view, all of them rely on the utility computing model, where features and added-value products are managed and offered as services. These companies could be then referred to as Service Providers (SP) and the prospective end-users as the Service Consumers (SC). Obviously, SP continuously aims to reduce the high Capital Expenditures (CAPEX) and Operational Expenditures (OPEX) of their investments. For instance, Netflix introduced the DevOps perspective to bring agility and automation to the services life cycle. Google made up the Site Reliability Engineering (SRE) concept [goo] to operate the large data centers it owns in an automated and sometimes autonomous way.

Against this backdrop, the concept of Network Functions Virtualization (NFV) was introduced. Broadly speaking, NFV enables the virtualization of the network and the network functions (a.k.a. middle-boxes) to benefit from the virtualization advantages (i.e., flexibility, dynamicity, cost-effectiveness) when operating the network. In the earlier ages of cloud computing, the *Everything-as-a-Service* (XaaS for short) service delivery models were implemented as IaaS for infrastructure resources, PaaS for platform resources, and SaaS for software. This stipulates that applications could be virtualized and offered as SaaS. These applications are hosted and executed over PaaS

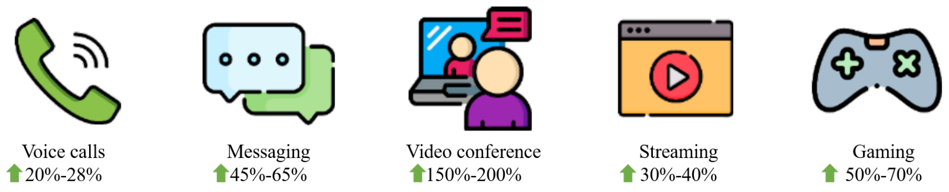


Figure 1.1: The rising demand for applications/services during last years

resources that are provisioned on IaaS appliances. However, the link between these different layers, i.e., the network, remains physical and static. With NFV, virtualization is enabled everywhere, from the applications down to the infrastructures through the connecting network and middle-boxes. In conventional networks, middle-boxes were statically provisioned in the network due to the fact that they are tightly coupled to the underlying hardware. This brings enormous CAPEX and OPEX, considering the labor-intensive manual efforts for the deployment and configuration of middle-boxes. NFV allows decoupling the network functions from the underlying hardware. This eventually enables the dynamic provisioning of middle-boxes (e.g., NAT, firewall) on top of any generic, Commercial-Off-The-Shelf (COTS) hardware, anytime and anywhere in the network.

The European Telecommunications Standards Institute (ETSI) has defined a reference architectural framework for NFV [ETSI 2012, Virtualisation 2014]. The NFV framework includes three main components: Virtualized Network Functions (VNFs), NFV Infrastructure (NFVI), and NFV Management and Orchestration (MANO). A VNF is the software implementation of a given middle-box. NFVI provides virtualized and physical resources as services. NFVI is managed by MANO, providing the required environments for VNFs to be deployed and executed. NFV MANO is also responsible for VNF and network service life cycle provisioning including deployment, execution, and management.

1.2 Motivations and Challenges

The Web Services (WS) example is the most used implementation that applies the service computing architecture. Originally, WSs were designed to be hosted and executed by service containers. Hence, the minimal required resources are packaged within their artifacts. To this end, end-users rely on the information in the description file *WSDL*[1], which is stored in a centralized and unique *UDDI* [2], to find the appropriate WSs that meet their needs. This information describes the communication between the server

[1]Web Services Description Language

[2]Universal Description, Discovery, and Integration

and the client, the transport protocol used, the WS interfaces, etc.

It is advocated that NFV is at the crossroad of networking and service computing research fields. The same principles and lifecycle management procedures could be applied to NFV resources in general, and Virtualized Network Functions (VNF) and Network Services (NS) in particular. VNFs represent the concrete implementation of the network function in NFV while NS involve a composition of a set of VNFs to implement a more complex network function.In other words, NFV aims at provisioning the NS through the VNF concept [Mijumbi 2016]. The latter could be provided in the same way as any other kind of services such as telco or Web services, indeed, Service-Oriented Architecture (SOA) principles (e.g., service abstraction, discoverability, and composability) [Thomas 2007, Ordanini 2008] could ensure the viability of network services' ecosystem that could be dynamically and flexibly provisioned, thus coping with changeable network provider (i.e., the service consumer) needs and dynamic Quality of Service (QoS) requirements along with context conditions.

Contrary to the VNFs that are first designed and developed as software before being published in appropriate repositories. VNF artifacts are standalone and need to properly incorporate all the necessary resources to execute the VNFs (serverless distribution). Before their deployment, VNFs are instantiated from the proprietary marketplaces into the target network. In fact, the end hosting nodes range from powerful computing servers to virtual machines and smartphones [Chowdhury 2010] [Tao 2017]. Since nodes have different capabilities (e.g., CPU, RAM, graphics resolution, bandwidth), this implies that additional checking of the correct matching between the non-functional requirements of the discovered VNFs and the potential hosting nodes' characteristics needs to be integrated into the discovery procedure. Following, they are configured to be integrated as part of a specific topology.

As for the NS instantiation, the web services are instantiated based on a business process, and managed during run-time following either orchestration or choreography methods [Peltz 2003]. WSs are never downloaded and instantiated in the local domain, in addition to the fact that they are process-driven. Unlike the NSs, which are mainly orchestrated by a central entity, and the constituent VNFs are arranged according to a Network Connectivity Topology (NCT) or without any connectivity specification between them. They are data-driven [Walsh 2002] and their source code need to be integrated into the host node.

When it comes to the deployment phase, WSs do not make any placement decision as they are invoked remotely. Unlike the VNFs, they need to be placed and routed according to the NCT with an optimized solution that aims to minimize the total deployment cost in terms of allocated resources while satisfying constraints like location and latency.

Last but not least, we noticed that the management of WSs and VNFs, that make up a given NS, could be slightly different. The VNFs interaction is not limited to basic operations like the case with WSs. It should also include additional sophisticated operation management dedicated to each VNF. Further, the deployed VNFs, when necessary, could dynamically be re-configured and adjusted, depending on the need (e.g., request type, data type), during run-time. Thus, the composition and placement requests issued for these VNFs should vary accordingly.

Wherefore, the existing service standards, studies, and frameworks do not address VNFs and NSs life-cycle management specificities.

1.3 Research outline

In this thesis, we intended to explore the NS provisioning life-cycle summarized in Fig. 1.2

1.3.1 Research questions

The NS life-cycle is mainly composed into 4 phases [nfv 2018, Boubendir 2018, Yangui 2016]. In the following, we detail each phase and highlight their limitations:

NS design phase. Valuable standardization initiatives (e.g., ETSI NFV [Virtualisation 2014], IETF SFC [IETF 2018], and OASIS TOSCA [Specification 2017]) focus on providing an intent framework and enabling VNF provisioning capabilities such as description, publication, and discovery mechanisms to design the NS. Similarly, several research papers (e.g., [Oliver 2018], [Bouten 2016], [Hoyos 2016]) proposed description models for VNFs. On the one hand, these proposed VNF descriptions and publication models are mostly not comprehensive. Indeed, they include details on VNF deployment but fail to cover their associated functional and non-functional specifications. Functional characteristics of VNFs refer to the business functionality that a given VNF implements, while non-functional characteristics refer to what the VNF requires for proper functioning. In other words, these characteristics aim to describe the optimal use state and/or the requirements of the VNF in terms of security, reliability, performance, maintainability, and so on. On the other hand, the existing discovery approaches remain specific to the owner providers. Each provider defines particular procedures and practices to parse VNF descriptors and select relevant information for them. Where the prospective consumers (e.g., NS provider) still need to manually select the required VNF rather than having an automated discovery mechanism. All these limitations are due to several reasons. In NFV broad landscape, a lack of a shared understanding of VNF descriptions can undoubtedly be observed because of the heterogeneity of technologies and providers. Besides, implicit knowledge leads to possible different interpretations. Consequently, consumers are obliged to parse a priori known sources to look for VNF candidates. This way of doing is time-consuming and often results in a minimal number of VNF candidates, not all relevant concerning consumers' initial needs.

NS instantiation phase. The VNFs could be arranged and chained together in a pre-defined order to provide end-to-end services [NFV 2017]. Moreover, several industrial groups are developing standards for VNF chaining. For example, ETSI has proposed a service architecture that uses network forwarding graphs to forward traffic between virtual network functions, called VNF Forwarding Graph (VNF-FG). However, despite all the promising advantages brought by NFV, there are still many fundamental challenges that need to be tackled in NFV-based chaining with more attention. For instance, in VNF-FG, the VNFs are arranged as a set of functions according to a Network Connectivity Topology (NCT) or with no connectivity specification between them. However, ETSI NFV does not provide a specific control technique that dictates the execution order and the composition logic of VNFs in a given NS. This reflects negatively on the NS provisioning methodology and procedures. Indeed, at instantiation time, developers must entirely rely on SDN, along with its inherent service function chaining concept, to design and configure the routing rules between the involved VNFs. Consequently, they have to foresee and plan all possible and practical composition scenarios in the NS. Moreover, they should instantiate, deploy, configure, and manage all these potential paths at run-time, including those that will rarely be or never be used. On top of being costly, tedious, and time-consuming, this way of proceeding contradicts NFV spirit, specifically virtualization, promoting agility and cost-effectiveness in networking applications. Yet another limitation concerns the routing path models supported by SDN. In fact, SDN allows only sequential service function chains to be provisioned, where VNFs are tied in a linear way. Therefore, more complex and sophisticated chains are still not supported in the NS.

NS deployment and management phase. In order to satisfy end-users' requirements, the network service can be instantiated and deployed over multi-administrative domains expanded over larger areas (e.g., different countries) or one compass area where its QoS can only be guaranteed by combining resources from multiple administrative operators. Furthermore, the network service instance could dynamically be reconfigured and adjusted during run-time depending on the need (e.g., request type, data type). Consequently, the resource allocation requests issued for this kind of network service should vary over time, i.e., new/existing VNFs and the corresponding virtual links are added/removed. A primary alternative to support such a procedure could be one-to-one mapping between pre-defined provisioning requests and available resources over the different NFVIs. However, this alternative remains static and fails to update the placement decisions following any perspective changes on the deployment/placement requests. For instance, changes would reflect SLAs updates, such as billing/metering variation or technical breakdown. Therefore, a Network Service Embedding (NSE) solution that provides VNF placement for each network service along with traffic routing across VNFs while satisfying constraints like location and latency is needed. This optimization aims at minimizing the network service deployment cost in terms of allocated resources.

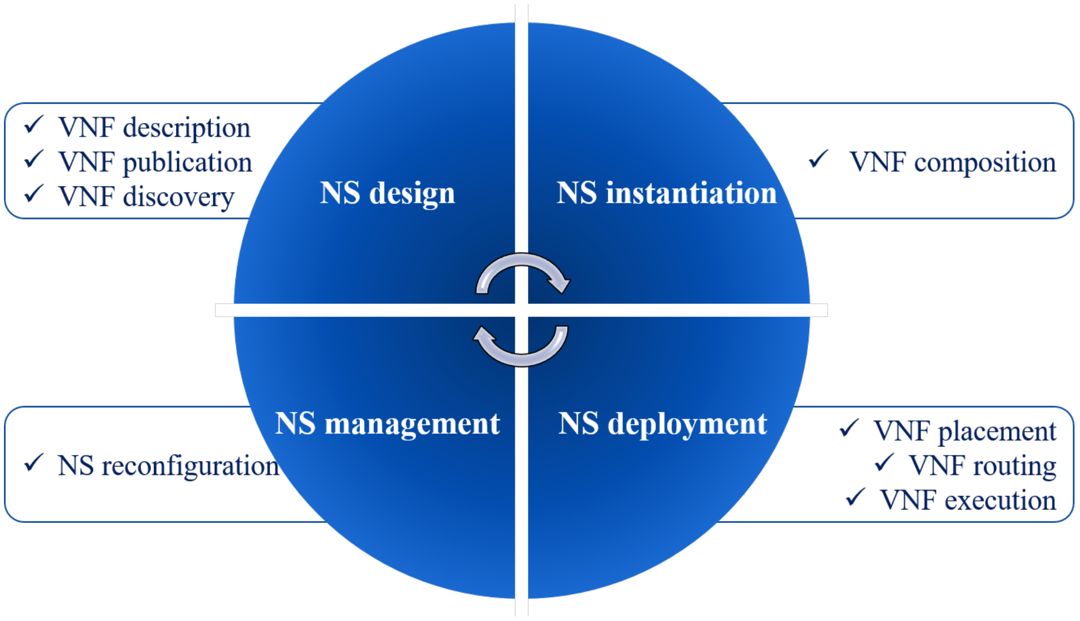


Figure 1.2: NS life-cycle

1.3.2 Objectives and contributions

Most challenges in the field of NFV are associated with management and orchestration operations, such as automated provision, configuration, and optimisation. For this purpose, the key factor towards accelerating NFV adoption is not associated with a single specific technology, but rather with a complete end-to-end provision framework which will fulfil all the key NFV-related requirements from the network operators' as well as the customers' point of view. In this context, this thesis considers.

1. **Semantic Description and Discovery of Virtual Network Functions.** This contribution introduces a domain-independent VIrtualized networK functIoN ontoloGy (called VIKING). The latter proposes a novel semantic-based

methodology to enable VNFs developers and owners to sufficiently describe the capabilities and requirements of their VNFs prior to publication. Prospective VNFs consumers use the same model to automate the discovery process, improve its precision and rely on federated repositories systems if needed. Yet another contribution is supporting the VNFs non-functional properties and user preferences during the discovery process, in addition to the classical functional properties.

2. **Agile and Dynamic Virtualized Network Functions Wiring in Network Services.** A novel approach is proposed to enable agile and dynamic NS composition as an extension to the ETSI NFV architectural framework. Broadly speaking, this contribution initiative introduces novel technical VNFs, called routing VNFs, with efficient re-configurable wiring capabilities for NSs. Furthermore, this initiative distinguishes domain-specific aspects from the connectivity ones in the NS definition. Domain-specific aspects are implemented with regular VNFs, while the routing VNFs support connectivity aspects. By doing so, developers will be able to change and update the NS's composition logic at run-time, given specific criteria (e.g., message type, data size, QoS metrics).

3. **Resource Optimization for Network Service Deployment.** This contribution introduces a lazy resources allocation solution to Network Service Embedding (NSE) across multi-administrative domains problem. The proposed solution optimizes both VNF placement for each network service along with traffic routing across VNFs while satisfying constraints like location and latency. This optimization aims at minimizing the network service deployment cost in terms of allocated resources and when necessary. Further, prior to provisioning, the NFVO, with the help of VIMs, will scrutinize the hosting capacity for each infrastructure and bandwidth for each physical link over time.

1.4 Thesis Organization

The remaining of the thesis is centered around five main chapters, and is structured as follows:

Chapter 2 provides the main concepts of the NFV system in order to provide the background information that is necessary for the understanding of this thesis work. The chapter also gives a literature overview and analysis of standardization efforts, reference frameworks, and research works related to the main research questions of this thesis. *Chapter 3* addresses the problem of describing and discovering semantic Virtual Network Functions with the primary objective of enabling VNFs developers and owners to sufficiently relate the functional and non-functional capabilities and the requirements of their VNFs before publication. Then, VNFs consumers use the same model to automate their discovery process. In *Chapter 4*, a novel concept of new routing-VNFs with re-configurable wiring capabilities that would allow agile and dynamic VNFs wiring from design-time to run-time is presented. Finally, an optimal solution for the NS embedding problem in the NFV ecosystem is proposed in *Chapter 5*. It minimizes the network service total deployment cost in terms of allocated resources while satisfying some constraints.

Chapter 2

Background and thesis positioning

2.1 Introduction

Nowadays, Service Providers expand their market offerings beyond the network connectivity with new value-added services, allowing them to enhance their service provisioning and meet their consumers' requirements. These services portfolios provide a wide range of functionalities, covering from essential internet connectivity services, such as IPTV delivery, to highly available and secure connectivity between business sites [ITU 2015]. Of course, adopting value-added services to current communication service provisioning is not that simple. Even basic value-adds, like simple format conversions, necessitate new hardware. They rely massively on deploying various chained middle-boxes, each of which operates a different network function, like network address translation (NAT), firewall, encryption, Domain Name Service (DNS), Caching, etc. Those devices are highly specialized hardware produced by several vendors, available in abundance, and dispersed over the network. However, operating these chained middle-boxes is immensely expensive and challenging since they should be physically placed in the network and manually configured. In addition, their maintenance still requires both labor and equipment spending. Moreover, middle-boxes scaling up/down cannot be performed faster, where most of them are over-provisioned and possibly overloaded when the traffic gets unexpected peaks. These operations are time-consuming and costly in terms of CAPital EXpenditures (CAPEX) and OPerating EXpenditures (OPEX).

To face those limitations, a group of organizations launched in 2012 an initiative around a new paradigm called Network Function Virtualization (NFV). They believe that the best way to deliver value-added services with greatly simplified operations and reduced costs is through NFV. Where, the network function are virtualized as a software appliance called Virtualized Network Functions (VNF). To further understand the implementation and design aspects of NFV, we must first understand its architecture and how internal components interact with each other as well as with other external elements of other systems.

The rest of this chapter is organized as follows. First, Section 2.2 provides a detailed overview of the NFV architecture, the service management in both service computing and NFV settings, and evaluates the benefits gained from virtualization. Then Section 2.3 discusses and synthesizes the state-of-the-art approaches related to network service design, instantiation, deployment, and management.

2.2 Background information

This Section introduces fundamental and background information that are necessary for the understanding of this work.

2.2.1 ETSI specification for the NFV

To further understand the implementation and design aspects of NFV, we must first understand its architecture and how internal components interact with each other external components.

ETSI NFV Industry Specification Group (ISG) defined an architectural framework [ETSI 2014] which enables flexible deployment and execution of VNFs and NSs on virtualized infrastructure, as depicted in Fig. 2.1. This framework is composed of four main interconnected functional blocks, each one with well-defined set of responsibilities and interfaces, as described below:

- **Operation/Business Support System (OSS/BSS)**. It is not a functional component of the NFV architecture, but it represents the service provider's back-end system that operates its business requests. The NFV-MANO must provide an interface for interoperability with OSS/BSS.

- **Virtualized Network Function (VNF)**. It corresponds to the individual functions of a network that have been virtualized and can be executed on the NFVI equipment. In addition to the Element Management System (EMS) which is responsible for the FCAPS (faults, configuration, accounting, performance, security management) for the functional part of the VNF.

- **NFV Infrastructure (NFVI)**. It is made up of basic hardware resources that contain compute facilities (COTS appliances), storage hardware (hard disks), and network hardware (switches/routers). This resources are partitioned and shared through a hypervisor layer which abstracts them. All of this form the virtual infrastructure that contains the virtualized resources.

- **NFV Management and Orchestration (NFV-MANO)**. It is the brain of this architecture. It is responsible for driving the deployment, execution, and operation of VNFs/NSs in the NFV infrastructure.

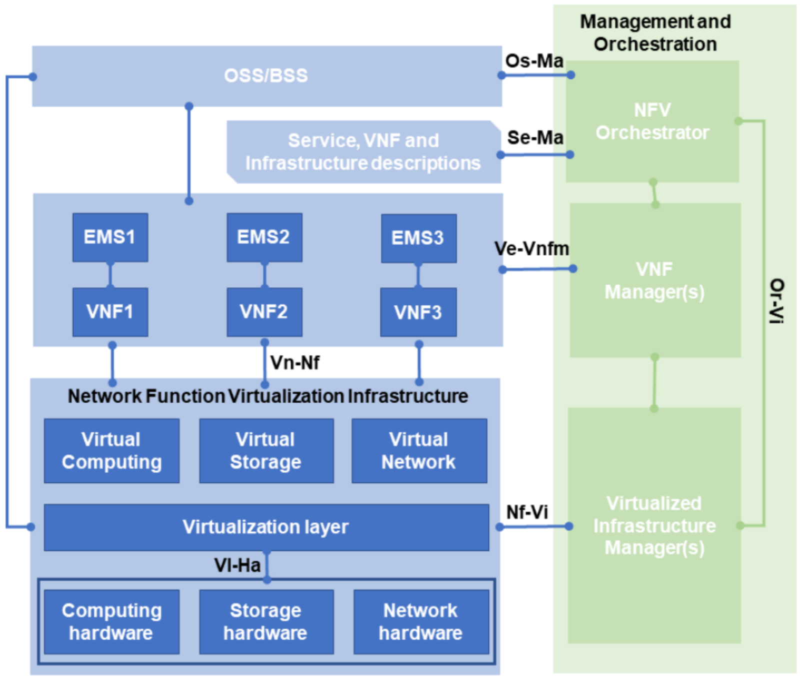


Figure 2.1: ETSI-NFV architecture overview[ETSI 2014]

NFV-MANO specifies open technologies and paradigms to orchestrate the NFVIs and manage the life-cycle of VNFs/NSs based on their descriptors. It comprises three main components, as follows:

- **NFV Orchestrator (NFVO)**. It has two primary responsibilities. The Resource Orchestration (RO) ensures an optimized allocation of the necessary resources and the connectivity between VNFs. And, the Network Service Orchestration (NSO) responsible for managing the Network Service (NS) life-cycle.

- **VNF Manager (VNFM)**. It is in charge of the life-cycle management of VNF instances and the execution of FCAPS management, under the control of the NFVO.

- **Virtualised Infrastructure Manager (VIM)**. It translates the directives of NFVO in order to controls and manages the NFVI compute, storage, and network resources in one domain. In case of multiple VIMs in an NFV architecture, each VIM manages its respective NFV Infrastructure (NFVI) domain.

In addition to these functional blocks, NFV-MANO includes **Data Repositories** (DR) these are repositories that keep different types of information in the NFV-MANO. **DR** contains four types of repositories: *VNF* and *NS catalogs*, *NFV instances*, and *NFVI resources repository*. The first two catalogs are descriptor repositories of on-boarded VNFs and NSs, respectively. VNF catalog consists of a set of templates that describe the VNFs. NS catalog represents a set of pre-defined templates of the NS which include the constituent VNFs and their connectivity. We give more details on these templates in Section 2.2.3. *NS instance repository* preserves VNF and NS instances related information. *NFVI repository* maintains available, reserved, and allocated NFVI resources and their state information. NFV-MANO functional blocks have access to these repositories to perform NS/VNF and NFVI orchestration and management.

Many communities and leading tech vendors are offering ETSI align NFV-MANO projects and products. Hereafter, we briefly present some open source industry projects.

Open Source MANO (OSM) is an ETSI-hosted open-source project launched in 2016 [osm]. The implemented architecture includes three functional blocks NFVO, VNFM, and VIM. These blocks perform the service orchestration and configuration, abstraction of VNFs, and orchestration and management of the infrastructure resources.

Open Networking Automation Platform (ONAP) [ona 2021] is a platform with orchestration capabilities, which can be applied to both physical and virtual network functions. They followed a modular implementation and supported YANG and TOSCA data models.

OpenBaton [Carella 2015] is an open-source platform align with ETSI-NFV architecture framework. It is composed of multiple components implemented in JAVA. Their orchestration design allows the integration of several VIM without changing the orchestrator logic.

Cloudify [clo 2022] is an open-source platform developed to perform cloud orchestration software. This platform permits modeling network services and automating their life-cycle management. It allows service mono and multi-domain deployment.

2.2.2 NFV paradigm and analogy with service computing

NFV is at the crossroad of networking and service computing research fields for many reasons. From one side, NFV aims to provision the network functions decoupled from their proprietary hardware appliances and run them as software through the VNF concept. The latter is packaged as virtual machines or containers and provided with its deployment description file. NFV allows the use of these VNFs to compose rich network services. The connection of the constituent VNFs is defined by network connectivity topology. On the other side, service computing refers to the loose coupling of various components of an application to its operating systems and other technologies. These components are separated into autonomous and self-describing units, called services. The services are accessible via predefined network interfaces. This allows developers to create and reuse them in developing new applications efficiently. Moreover, these components can communicate with each other by sending data in a well-defined format.

The network service falls into the definition of IT services at large [Mijumbi 2016]; so, it could be provided and managed in the same way as any other kind of services such as telco or web services. In its simplest form, the service life-cycle consists of three phases: service design, service composition, and service execution, defined as follows:

Service design. During this phase, the service provider should provide enough information about its requirements and preferences to dynamically discover the desired functions.

Service instantiation. This phase consists of transforming the service requirements to a concrete composite of functions with more connectivity aspects.

Service deployment and execution. In this phase, the constructed composite service is deployed and executed to allow its utilization by end-users.

Despite the similarities, the reader should underline that the implemented service life-cycle phases remain specific to service computing's operating procedures. For illustration purposes, we present Web-Services (WS) as the most used implementation applying the service computing architecture [Yang 2004]. Table 2.1 sums up the fundamental differences between VNFs and WSs operations for every single phase of the life-cycle. On one side, WSs are designed to be hosted and executed by service containers (e.g., Apache Tomcat, Apache Axis2). Consequently, minimal resources are packaged within the artifact. Specifically, the WSs are deployed only once over the hosting service containers and can be simultaneously invoked by several end-users. To this end, end-users rely on the information in the WSDL. The latter is stored in a centralized and unique UDDI. The discovery of the most relevant associated WSDL could be either syntactic or semantic. In all cases, web services are never downloaded and instantiated in the local domain, as they are hosted with local servers on the cloud. On the other side, VNFs are first designed and developed before being published in appropriate repositories for prospective consumers. VNF artifacts are standalone and need to properly incorporate all the necessary resources to execute the VNFs (serverless distribution). Before their deployment, VNFs are instantiated from proprietary marketplaces into the target network. The choice of the VNFs placement is based on some optimization objectives (e.g., minimize resource consumption and cost). After, they are configured to be integrated as part of a specific topology. Once deployed, VNFs are executed and, when necessary, are subject to management considerations at runtime (e.g., scale-up/down, migrate). The discrepancy between the Web services and VNFs models and the operating procedures shows clearly that it is not appropriate to recall existing approaches that manage the WSs and simply adapt or extend them to address appropriate approaches for NFV. Therefore, in the following section, we present how ETSI-NFV manage the network service life-cycle and orchestrate required resource for the proper functioning of NSs.

Table 2.1: Dissimilarities between Web services and VNFs life-cycle implementation

life-cycle phase	**Web Services**	**VNFs**
Design/ Development	Deployable that consists of artifact and source code. Web service deployable is simple code that needs to be hosted and executed within Web servers	Deployable that consists of standalone artifact capable of running in a serverless fashion
Description	Web Service Description Language (WSDL)	VNF Descriptor (VNFD)
Publication	Universal Description Discovery and Integration (UDDI)	Proprietary VNF repositories
Discovery	Manual (by user) or automatic (using matchmakers)	Manual (by user) or automatic (using matchmakers)
Instantiation	No instantiation is required. Web services are invoked as remote resources through valid Unified Resource Identifier (URI)	An instance of VNF is downloaded, installed, and configured over a virtualized infrastructure
Placement	Static placement within web servers on the cloud	Dynamic placement according to an optimization objective in a target domain within a network topology
Execution	Remote Procedure Call (RPC)	Local calls from the network domain

2.2.3 ETSI-NFV management and orchestration aspects

As mentioned in Section 2.2.1, ETSI NFV enables providers to model and compose VNFs to deliver sophisticated network services for prospective consumers. Thus, NFV-MANO relies on deployment descriptors which include the necessary resources and operations information for proper network service management and orchestration. In what follows, we provide a detailed description of the information elements needed by NFV-MANO. Then, we illustrate how NFV-MANO uses them to perform NS orchestration and management operations. A *descriptor* is a deployment and configuration template that defines the main properties and requirements of the managed objects, such as a VNF and NS.

Starting with the elementary component, the VNF. The associated *VNFD* is a deployment and operational template that encapsulates all necessary information that specifies VNF's characteristics like its internal composition and required resources [ETSI 2021], etc. A VNF could consist of several distinct VNF Components (VNFC) and comprise one or many virtual deployment units (VDU). Each VDU can support specific deployment resources and operation behavior and hosts one or more VNFC. Mainly, a VDU describes the Virtual Compute (VC), Virtual Storage (VS), and Virtual Memory (VM) resources. These resources' data are necessary for deploying a VNFC. VNFC can be linked via either connection points to local VDUs or external connection points to VDUs that belong to other VNFs. The virtual links description in the VNFD indicates how the VDUs are connected and via which connection points. The deployment flavor describes a specific template/image of a VNF with capacity and performance requirements. The NFVO, VNFM utilize this *VNFD* to instantiate the VNF and reserves its required resources and to manage its life-cycle, respectively.

NSD represents the description of the constituent VNFs with their connectivities. Fig. 2.2 depicts the core concepts describing the NS connectivity. The NS exposes service access points (i.e., specific connection endpoints) that define the NS interfaces. Similarly, the VNFs that compose the NS are bound to each other by Virtual Links (VL s) that connect their associated access points. Altogether, it represents the Network Connectivity Topology (NCT). The NCT formalizes a high-logical view of connectivity between VNFs in the NS. The reader should note that constituent VNFs can be arranged in NS with unspecified connectivity between them [NFV 2017]. NCT encompasses different VNF Forwarding Graphs (VNFFGs). A VNFFG describes NS topology. It references a set of connection points, service access points, the descriptors of its constituent VNFs, and the VLs that connect them. A VNFFG embraces one or more sequential, alternative, and/or concurrent Network Forwarding Paths (NFPs) where a given NFP implements the concrete network path for the actual traffic flows in a VL.

As reported in Section 2.2.1, to orchestrate an NS, based on the reported *NSD*, the NFVO uses both Resource Orchestrator (RO) and Service Orchestrator (SO), and it

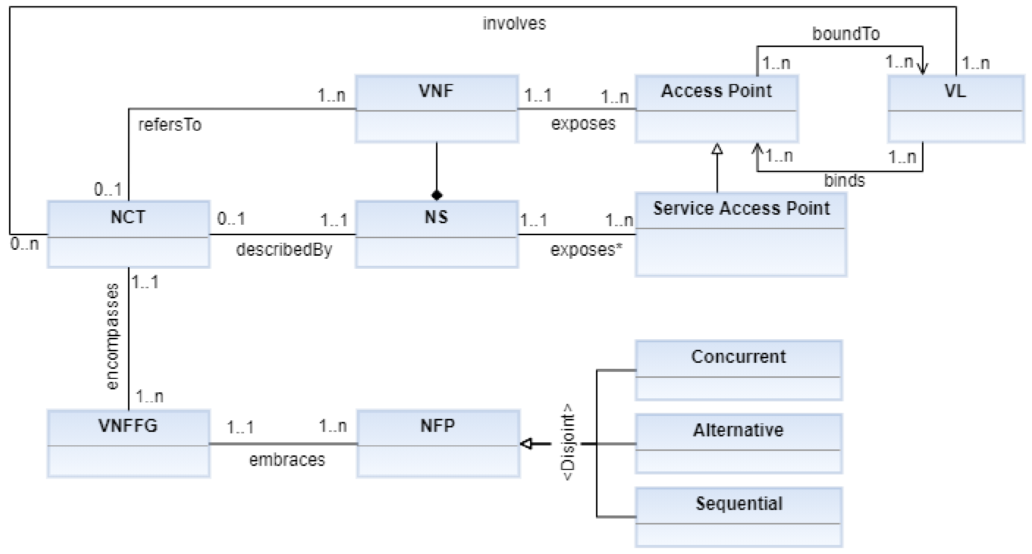


Figure 2.2: NS Connectivity management in ETSI NFV-MANO [ETSI 2021]

follows a workflow hierarchy, as follows:

Network Service design. First, the NS provider prepares the NSD, referring to the constituent VNFs and VNFFGs. Each NS provider has its proper manner of discovering and selecting the desired VNFs.

Network Service on-board. This procedure refers to submitting the prepared NSD to the NFVO in the catalog. It is worth mentioning that the NS provider is in charge of preparing and onboarding the constituent VNFDs, as well.

Network Service instantiation. When this procedure is triggered, the NFVO collects all necessary information from VNFDs and NSD. RO communicates the required resources to the appropriate VIMs, where the SO passes the VNF information to the VNFM. The latter checks the feasibility of VNFs instantiation over NFVI. Once it is validated, RO executes the resource allocation needed for the NS. Then, VNFM instantiates the VNFs with any specific life-cycle parameters.

Network Service management. During run-time, the VIMs and VNFM update the NFVO with the execution status of the VNFs and their connectivity. Based on this information, NFVO may update, delete, terminate processes on the VNFFGs, VNFs, or NSs.

2.2.4 NFV restrain cost benefits

As previous sections show, ETSI-NFV proposed the virtualization concept with its associated technology, likely leading to considerable financial gains over time. According to [tco 2015], virtualization could allow an operator to reduce up to 2/3 of its Capital Expenditure (CAPEX), which refers to the initial cost to deploy the NFV solution, and Operational Expenditure (OPEX), which refers to the operational cost to maintain the proposed solution. Thereby, SPs can gain some cost-benefit, but the initial NFV vision does not look as promising as expected from a management perspective. Indeed, the cost has several facets, including the ability to bring new services and features to market in extremely short timescales; guarantee the highest levels of service agility based on business/network policies with the lowest cost of operations; ensure service placement and routing while minimizing the total deployment cost, etc.

Unfortunately, the service providers nowadays cannot achieve the agility, dynamicity, and cost-effectiveness they desire for several reasons. First, the solutions proposed by standardization have addressed uniquely unprecedented challenges imposed by the new network service architecture. However, they lack performance analysis methods for their proposed solutions. In NFV broad landscape, a lack of a shared understanding of VNF descriptions can undoubtedly be observed due to the heterogeneity of technologies and providers. Besides, implicit knowledge leads to possible different interpretations. Consequently, consumers are obliged to parse priory known sources to look for VNF candidates. This is time-consuming and often results in a minimal number of VNF candidates, not all relevant concerning NS providers' initial needs. Furthermore, following the standards specification, provisioning the NS necessarily implies deploying all the involved VNFs and configuring the connectivity as depicted in the associated VNFFG description. This static description obliges NS providers to anticipate, compose, and configure all the possible NFPs (even those who could be rarely used). This ends with a heavy, complex, and static NCT handle during every single phase of the NS life-cycle. Besides, optimal and automatic placement of the composed VNFs, mapping these functions to the available resources over the underlying resources, and considering QoS and SLA constraints is equally essential. Thereby, it is essential for service providers to look beyond the basic virtualization of the services and seek the full potential and value of NFV. That is why traditional manual service life-cycle management processes and their multiplicity of vendor-specific components at different network layers will need to be replaced by automated, software-based management that is standardized across vendors.

2.3 The State-Of-The-Art

This section presents a detailed state-of-the-art analysis that defines the thesis scope. Each field is an essential aspect to consider when tackling thesis problems.

2.3.1 Network service life-cycle management

Valuable standardization initiatives (e.g., ETSI NFV [Virtualisation 2014], IETF SFC [IETF 2018], and OASIS TOSCA [Specification 2017]) provided an intent framework and enabled NS, with its constituent VNFs, provisioning capabilities such as design, composition, and deployment mechanisms for NSs.
Initially, to design the NS, the existing discovery approaches that rely on VNFD are still in their early ages, and much work has yet to be done for optimal VNF provisioning. These approaches remain specific to the owner providers. Where each provider defines specific procedures and practices to parse VNF descriptors and select relevant information for them. In addition, the service providers still need to manually select the required VNF rather than having an automated discovery mechanism. Further, these VNF descriptions and publication models are not comprehensive. Indeed, they do include details on VNF deployment but fail to cover their associated functional and non-functional specifications.

Furthermore, the standards allow composing complex and sophisticated NSs made up of elementary VNFs, at a higher level of abstraction. However, they do not provide a specific control that dictates the execution order and the composition logic of VNFs in a given NS. This reflects negatively on the NS provisioning methodology and procedures. Moreover, they should instantiate, deploy, configure, and manage all the potential paths at run-time, including those that will rarely be or never be used. On top of being costly, tedious, and time-consuming, this way of proceeding contradicts NFV spirit promoting agility and cost-effectiveness in networking applications. Nevertheless, another limitation concerns the routing path models. In fact, these models allow only sequential service function chains to be provisioned where VNFs are tied in a linear way. Therefore, more complex and sophisticated chains are still not supported in the NS.

Afterward, the composed network services may be deployed over a set of physical network infrastructures; each spread over mono-/multi- domains and managed by a specific VIM. This context requires the NFV-MANO to consider the trust partnerships between NS/infrastructure providers and infrastructures providers. Moreover, service providers may request extending their already deployed NS with adding/removing some VNFs without interrupting the NS. Realizing such NS deployment across multi-administrative domains is challenging, not only from the perspective of associating an NS request into a specific domain(s) but also assuring its performance while reducing the total deployment cost without saturating the resources.

2.3.2 On network service design

In this section, we review the more relevant works related to VNF description, publication, and discovery.

2.3.2.1 Semantics in networking

A plethora of studies in the service computing field investigated services' and users' queries description. Several concepts have been studied; however, the most important results were obtained when using semantics. Handling semantics in service discovery was primarily investigated from two main matching perspectives: syntactic and semantic. The first relies on graph theory, such as Resource Description Framework (RDF) [rdf 2020] and DIANE Service Description [Küster 2007]. In contrast, the second relies on ontologies, such as the W3C Web Ontology Language(OWL-S) [Martin 2004] and Web Service Modeling Ontology (WSMO) [Domingue 2005]. Many research works compare the syntactic ones, exemplified by information retrieval metrics, *versus* the semantic matching ones, illustrated by logic inference (e.g., see [Adala 2011] [Seog-Chan Oh 2006]). The latter turns out more efficient than the former in terms of precision and recall. This result is one of the reasons that led us to advocate for semantic matching for this work.

Generally speaking, in the networking domain, semantics has been widely used since the late eighties (e.g., [Sowa 1987], [Shapiro 1987]). Artificial intelligence and machine translation were the first to develop and use semantic networks. More broadly, the use of semantics in networks is done through declarative graphic representation that represents knowledge and supports automated plans for reasoning about learning. Some approaches are highly informal, but others are formally defined as logic systems. In particular, the reason behind semantics is to build and evolve network ontologies (e.g., [Voigt 2018]), retrieve information in networks (e.g., [Tang 2003] for peer-to-peer networks), and network slicing and segmentation (e.g., [Long 2015]).

When it comes to highly dynamic and/or virtualized environments such as ad-hoc networks and cloud computing (i.e., the main building blocks of NFV), OWL ontologies have been massively applied. For instance, OWL ontologies have been used for cloud environments to describe the heterogeneous multi-vendor cloud resources and users' SLA in the FP7 European mOSAIC project [Petcu 2013]. In dynamic and ad-hoc networks, we find that all of Network Description Language (NDL-OWL) [ndl 2020], Network Mark-Up Language(NML), Infrastructure and Network Description Language (INDL) [Gruber 1993], Network Innovation over Virtualized Infrastructures (NOVI) [van der Ham 2015] and Federated Infrastructure Discovery and Description Language (FIDDLE) [Willner 2015] use OWL ontologies.

2.3.2.2 Main contributions from research projects

Besides the previously discussed ETSI VNFD model, one of the most known and used approaches is Topology and Orchestration Specification for Cloud Applications TOSCA-based, namely TOSCA-NFV [Specification 2017]. TOSCA is a data model standard managed by the OASIS industry group. This data model is used to describe services' operations and requirements [Binz 2014]. It also explains how services can be deployed and managed through management plans (workflows) at runtime. TOSCA-NFV is the concrete implementation of the model applied to NFV for VNFs provisioning and management. It proposes a model to describe topologies, dependencies, and relationships between virtual applications and simplify these services' complexities rather than define VNFs capabilities and requirements. TOSCA-NFV model assumes that the VNFs are already discovered. Its main scope is to deliver orchestration and interoperability of VNFs. The same observation is valid for the IETF Service Function Chaining[14] (SFC) initiative. SFC in NFV setting relies on VNFD for VNFs description and selection. The reader should note that these procedures only support the VNFs business (functional) operations. In fact, SFC enables VNFs composition by simply matching their related operations [Mechtri 2017].

The EU-funded project T-NOVA [Xilouris 2014] provides a VNF marketplace that: (1) helps VNF developers describe and store network functions, and (2) assists the consumers when browsing and selecting the network functions that match their needs. T-NOVA extends the ETSI NFV description model by applying business aspects from the TMForum SID model [sid 2013]. Additional fields enable business interaction among actors that communicate through the T-NOVA Marketplace (e.g., SLA specification, pricing), besides deployment details needed to deploy the network services. The VNF/NS discovery process is conducted through the brokerage module [Xilouris 2014], which permits consumers to search for VNFs/NSs while specifying their specific requirement in terms of network SLA.

Cloud4NFV [Soares 2014] is a virtualized platform for VNFs provisioning. It aims to deliver NF-as-a-service to end customers. Cloud4NFV is ETSI-compliant with significant contributions to the modeling and orchestration aspects. On one side, Cloud4NFV processes a front-end database that stores collections of VNFs along with a high-level description (e.g., ID, name, description, location). On the other side, it handles a back-end database that stores specific VNF information necessary for the VNF deployment and configuration. Cloud4NFV provides only deployment and configuration information and lacks automated discovery process support.

[14] `https://tools.ietf.org/html/rfc7665`

2.3.2.3 Main contributions form academic research work

In the academic literature, Hoyos et al. [Hoyos 2016] propose an NFV Ontology called NOn and a Semantic nFV Services (SnS). NOn enables the description of NFV as a high-level framework with reusable element descriptors. As the concrete semantic application of NOn to the NFV domain, SnS can be used to create explicit service descriptors. It relies on various agents to parse and evaluate NFV services capabilities. However, NOn only considers the resources' functional capabilities. Furthermore, the reader should note that this approach imposes strong constraints on existing providers and assumes they could support these agents, which may or may not be accurate.

Oliver et al. [Oliver 2018] propose an ontology for NFV that describes the whole network resources, including VNFs, properties, and relationships (dependencies). The resources description is achieved through reusable semantic concepts used to construct additional rules for reasoning over the network. For instance, this could be useful to automate network topology design and deployment. Although this work proposes a semantic-based description model for functional VNFs operations, it mainly focuses on network engineering and integration efforts. It does not cover the VNFs discovery given specific and precise user needs.

The authors in [Bouten 2016] identify and discuss a set of affinity and anti-affinity constraints useful for virtualized network management. The validation of these rules is semantic-based. The addressed limitations are mainly related to service function chain requests. For instance, they defined a VNFs placement strategy that considers the network provider constraints and the chain request. This work assumes that the VNFs are already discovered and deployed. In [Bonfim 2019], the authors introduce Onto-NFV, an OWL-based ontology. It offers a vocabulary with its relations and constraints to describe a VNF composition (called network service) policies and the hosting NFVI policies. The policies involve information related to resource usage, VNFs precedence, and location constraints (e.g., number of CPUs, amount of memory). The authors propose NSChecker, a semantic verification system integrated into the ETSI MANO that uses Onto-NFV. The ultimate goal of this work is to detect and diagnose policy conflicts in NFV environments. For the semantic description, Onto-NFV only focuses on functional properties with no reference to non-functional properties such as security and availability. For the VNFs publication and discovery, it relies entirely on the ETSI MANO procedures.

Kim et al. [Kim 2018] use Network Service Description (NSD) data and ontology to automate VNFs management and network services generation. Network services consist of VNFs bound to each other through virtual links to implement shared and more general functionalities. The proposed solution relies on semantic annotation of NSD information according to ETSI NFV. This descriptor contains functional, non-functional, and optional information blocks. The functional block provides information related to the VNFs and their connection and dependencies. The non-functional block provides the full network service's general and profile data. Finally, the optional block provides policies and monitoring information. This work addresses one of the significant limitations of TOSCA-NFV. It models the relationship between parameters that TOSCA could not define. However, this work provides ontology only with neither investigated reasoning technology nor discovery algorithms/procedures for VNFs.

An approach to use microservices architecture for implementing VNFs is proposed in [Hawilo 2019]. To exploit the microservices adoption's full potential in NFV, they highlight some challenges like microservice discovery. To foster VNF dynamic scaling, the authors claim that a real-time automated service discovery mechanism should be developed to enable the required dynamic service chains. Indeed, in such a setting, service discovery is critical with regard to network dynamicity (e.g., relocation, auto-scaling) and frequent on-the-fly events (e.g., failures, upgrades). More in-depth details on relevant discovery patterns in the microservice context are provided in [Dzone 2018]. Basically, the authors define two patterns, i.e., client-side and server-side discovery patterns. They both assume that microservices are already known and only, their instances should be discovered. In the former pattern, the service client is in charge of determining the network locations of available service instances and defining load balancing requests across them. Specifically, the client, first, queries a service registry referring to available instances and, then, asks a load balancer to keep the best instance. A significant drawback of this pattern is that the client and the service registry are tightly coupled; each programming language used on the client-side requires a dedicated logic for service discovery. In the latter pattern, the client makes a request to a service (e.g., VNF) via a load balancer. The load balancer queries the service registry and routes each request to an available service instance. Discovery details are abstracted away from the client. Clients simply make requests to the load balancer. A major drawback of this pattern is that the load balancer should be provided by the deployment environment and, therefore, should be highly available (i.e., a single point of failure).

To our best of knowledge, there are few works on semantics in the context of microservices. In [Salvadori 2017], the authors present a framework for aligning heterogeneous ontologies in order to integrate data provided by different microservices. In line with microservices' design principles (e.g., loose coupling and independent maintenance), the design of heterogeneous ontologies to describe the same domain is *de facto*. An alignment contains a set of correspondences between entities and properties of such ontologies. Correspondences refer to semantic connections between concepts used to describe microservices data. The most important difference from the traditional ontology alignment is that equivalence statements can only be obtained at run-time. Indeed, entities are created during interactions between microservices and their consumers. Therefore, it is not possible to directly access a predefined comprehensive data-set. Instead, the proposed framework dynamically loads entities provided by registered microservices to infer alignment statements.

2.3.3 On network service instantiation and management

In the literature, several surveys (e.g., [De Sousa 2019, Medhat 2016, Bhamare 2016]) recently investigated and discussed the current challenges in NFV. Most of them argue that there is still a need for more appropriate and suitable VNF chains management. Moreover, they claim that VNF chains should be efficiently modeled, draw on dynamic business policies, and consider the network context to ensure efficient operation and better fit the evolving users' requirements. In this regard, various approaches have been proposed to tackle these challenges focusing on enabling VNF dynamic chaining. Some of them rely on SDN to ensure the dynamicity of the control plane following a service function chain update, while others propose a fully NFV-based solution.

Zhang et al. [Zhang 2017] introduces a hybrid packet processing architecture named parabox for highly latency-sensitive applications. This architecture dynamically transmits packets across VNFs in a parallel way and merges their outputs intelligently to ensure correct packet sequential processing. They implemented their solution using Berkeley Extensible Software Switch (BESS). Similarly, [Sun 2017] enables network function parallelism for NFV. They represent network operators' sequential or parallel chaining intents using a policy specification scheme.

Jmila et al. [Jmila 2019] propose a security-aware network service chaining design. Initially, the network service contains only basic VNFs. During the design phase, they may extend the network service chain to a more complex chaining model that includes security needs. Human security experts take the Network Service's VNF choice and position decisions. Based on security requirements, they proposed changing routes, splitting or mirroring the traffic to the newly added VNFs. Their solution is based on SDN paradigms to include customized VNFs and adapt the traffic routing.

The authors in [Lee 2015] propose an adaptive network service path model to recover network function failure during run-time. The forwarding nodes in their proposed solution contain a list of remote functions that could replace those that broke down. In case of failure, the forwarding nodes change the traffic path to one of these remote functions based on pre-defined rules. After the remote function finishes the processing, it returns the traffic processed to the originating function.

Callegati et al. [Callegati 2015] ensure dynamic chaining and flexible traffic routing for network function in hybrid cloud/edge networks. This approach dynamically configures forwarding rules in SDN switches (OpenFlow) based on network traffic conditions (e.g., bandwidth) and the users' SLA. This dynamic configuration is managed by means of the SDN-based control plane— the latter programs the data plane according to the desired VNF chaining.

Mohammed et al. [Mohammed 2016] enhance SDN orchestration to support dynamic VNF chaining as a preventive solution to service degradation. This prevention relies on network congestion prediction using traffic statistics. The SDN controller contains a list of all active paths in the network. In addition, it obtains regular operational statistics of all switches. Once the SDN orchestrator detects congestion of a switch, it tries to eliminate the considering paths and then creates a new path using other switches for each source/destination IP pair.

An SDN orchestrator for service chains to handle congestion events and SLA violations is introduced in [Gharbaoui 2017]. After collecting traffic statistics from switches, the orchestrator detects overloaded ones and then initiates a recovery procedure by adapting the impacted service chain paths. The recovery procedure deletes every service chain path that traverses an overloaded switch; and redirects the data flow to another switch, if available. While Liu et al. propose an orchestrator to dynamically provision and readjust VNF chains [Liu 2017] based on user mobility. This solution reuses existing VNFs to deal with the new users' requests. For cost-effectiveness purposes, the orchestrator periodically readjusts the chains by either migrating the VNFs over new datacenters or replacing them following the user requests.

Scheid et al. [Scheid 2016] proposes a policies-based Service Function Chains (SFC) management. Their proposition is based on a set of written policies which can be triggered during run-time to update the service chaining graph dynamically. The network providers write these policies in Controlled Natural Language (CNL). Then, the orchestrator interprets the written policies to create the SFC; and communicates with the traffic steering component to create the forwarding rule to steer the data flow across the corresponding VNFs..

In [Martini 2016], the authors propose an SOA-inspired NFV orchestrator, which relies on SDN capabilities for network control. On the one hand, the NFV orchestrator discovers and orchestrates VNFs. On the other hand, the SDN controller manages and delivers the traffic automatically to adjustable VNF chains. When network service degradation is raised, the proposed solution adapts active paths with pre-established ones based on resource usage (e.g., switch load). The corresponding steering rules are updated in the forwarding nodes through the OpenFlow protocol.

A framework called ESCAPE is introduced in [Csoma 2014]. It aims to build customized VNF chains in an SDN environment using Mininet, POX, ClickOS, and NetCONF tools. In addition, this framework contains a VNF catalog that serves to compose the network service. While the SDN controller decides how to chain the VNFs based on specific policies and real-time information collected from running VNF instances, OpenFlow switches steer the traffic across the VNFs. The authors in [Zsoka 2019] define some heuristic bandwidth-aware algorithm to create multiple dynamic service chains given a pre-defined set of allowed VNFs. This algorithm seeks acceptable solutions for service chaining where the candidate network links are less heavily loaded than possible.

2.3.4 On network service deployment and management

This section discusses related work on cost-driven network service embedding over multi-administrative domains.

The authors in [Katsalis 2016, Rosa 2015] analyzed the VNFs placement and routing problem across different administrative domains. They discussed theoretically the challenges and proposed research directions to solve the problem. However, no concrete algorithms for VNF deployment or routing mapping were proposed. Also, some studies such as [Baranda 2020, Valcarenghi 2018] discuss the challenges associated with incorporating trust into multi-administrative domains.

The Network service embedding problem take variant definitions and approaches, depending on several design metrics [Herrera 2016], such as: (i) Objectives (e.g., QoS, cost minimization, fault tolerance, load balancing, energy efficiency, etc); (ii) type of resources (e.g., CPU and bandwidth, processing time and buffer capacity, ternary content-addressable memory); (iii) the technology domain where Network service embedding is applied (e.g.,radio access networks, LTE/EPC, mobile core network, cloud networks); (v) single administrative domain or multi-domain approaches; (vi) solution strategy (e.g., exact, heuristic, or meta-heuristic).

Luizelli et al. in [Luizelli 2018] presented an operational cost minimization placement algorithm to guarantee a reasonable level of network performance under a distribute and gather NS placement strategies based on load balancing policy and energy-saving policy, respectively. Their solution minimizes the operational cost associated with virtual switching with the aim to provide a cost-efficient NSs deployment. The performance evaluation of their cost model shows an occurrence lower than 5% compared to actual deployment strategies.

A dynamic Virtualized Network Functions Forwarding Graph (VNFFG) extension problem is addressed in [Houidi 2020]. They proposed a seamless extension to place the newly added VNFs concerning previously deployed ones. They model their solution to match with the infrastructure providers' interests and favor the infrastructure nodes and links that are least loaded to host the requests.

An optimal service function chain embedding strategy is proposed by Pei et al. [Pei 2018]. Their model considers the dynamic VNF placement in geo-distributed cloud systems while optimizing resource utilization. The solution to place the VNFs is based on executing the shortest path over an expanding network. Then, the network is adjusted periodically according to the load variation.

Morin et al. [Morin 2020] introduced a network service embedding over public and private clouds while minimizing the total deployment cost. Their proposed model allows the NFV Orchestrator to select the best cloud provider offers based on the network services required resources. Furthermore, based on the predictions of the NFV Orchestrator, their model plans in advance long-term reservations, which reduces hourly prices.

5Grouth [Li 2021] introduce a service platform for zero-touch service and network orchestration and management. The objective is to deploy network services while ensuring established SLAs, even for shared resources with different service providers over multiple domains. Artificial Intelligence and Machine Learning solutions are used to achieve their requirement. One of the service orchestrator's responsibilities is to optimally place the VNFs and allocate the necessary resources across mono-/multi-domains. To do so, a service and resource federation needs to be established to deploy the network services. The same logic is applied within the project 5G-Transformer [Baranda 2020, Mangues-Bafalluy 2019].

The researchers in [Dwiardhika 2018] propose a virtual network embedding based on a security level. In this work, they considered only the cost of virtual network embedding, then they extended it in [Dwiardhika 2019] to consider also the revenue. Their work aims to place some security VNFs to increase the security level of substrate networks. More virtual networks can be embedded by adopting this method while respecting the security levels.

Alaluna et al. [Alaluna 2017] address the virtual network embedding problem to minimize networking and computing resource allocation cost and the overall length paths of the SFC. Their model propose two security constraints; cloud provider trustworthiness represents a measured level of its good reputation, and node and link security levels which have to be equal to or greater than virtual function security requirement. The same strategy is followed by Wang et al. [Wang 2015]. They proposed an admission process (based on node, link, and network trust levels) to help the network embedding model decide where to place the VNFs. However, Fischer et al. [Fischer 2017] consider each of nodes, links, and topology security requirements. Furthermore, they proposed a more concrete trustworthiness model for topology security and allowed the user to choose with whom to deploy their services.

A trust-aware service chain embedding problem in introduced by [Torkzaban 2019]. The researchers aim to deliver secure network service deployment on a reliable infrastructure. The constituent VNFs express security requirements via a trust value; after, their solution assign each VNF in the chain to a server that matches its trust level. Torkzaban et al. [Torkzaban 2019] extended their work in [Torkzaban 2019]. Where, they generalized the trustworthiness concept to cover service network links as well. Therefore, the service chain embedding process decides the VNF placement related to the trust value required by the VNF and the choice of substrate network paths between the VNFs based on the trust required by service network links.

Forti et al. [Forti 2020] introduce a secure deployment of IoT applications in edge and cloud infrastructure, taking into consideration the security requirement. The proposed deployment model proposes a deployment solution that guarantees a complete trust between the application and chosen infrastructure providers and between all the chosen infrastructure providers, as well. Where, each node in the infrastructure must

Table 2.2: Synthesis of related works to the NS design

Reference	Description		Publication	Discovery
	Functional	Non-Functional	Interoperability	Semantic matchmaking
ETSI VNFD	Yes	No	No	No
OASIS TOSCA NFV [Specification 2017]	Yes	No	Yes	No
IETF SFC [Mechtri 2017]	Yes	No	No	No
T-NOVA [Xilouris 2014]	Yes	Partially	Yes	No
Cloud4NFV [Soares 2014]	Yes	No	No	No
Hoyos et al. [Hoyos 2016]	Yes	No	No	No
Oliver et al. [Oliver 2018]	Yes	No	No	No
Bouten et al. [Bouten 2016]	Yes	No	No	No
Bonfim et al. [Bonfim 2019]	Yes	No	No	No
Kim et al. [Kim 2018]	Yes	Yes	No	No

provide its security capabilities and strength protection against attacks. Further, each infrastructure provider announces a trust level towards other infrastructure providers.

2.4 Synthesis

Table 2.2 sums up the most relevant studied works concerning VNF description, publication, and discovery. The literature study shows that several works (e.g., [Mechtri 2017], [Xilouris 2014]) tried to extend the VNFD proposed by ETSI with additional information using different approaches. However, only a few works (i.e., [Xilouris 2014], [Kim 2018]) succeeded in covering both the functional and non-functional properties of the VNFs in their proposed description models. The reader should note that the description of the non-functional properties in T-NOVA is limited. It only involves the business information (e.g., cost, SLA) necessary for interaction with other T-NOVA actors. Nevertheless, another observation related to the VNF description is the popularity of OWL as the most used semantic language to describe VNFs in the literature (i.e., [ndl 2020] [Gruber 1993] [van der Ham 2015] [Willner 2015] [Bonfim 2019]). Regarding VNF publication, the study highlights that most of the existing models require VNF publication in dedicated and proprietary repositories. T-NOVA is the only approaches that do not impose any compatibility constraints on the provider side and enable NFV repositories federation. Since it relies on generic and unified semantic models, this eliminates dependencies related to technologies used when offering the VNFs to prospective consumers.

In addition, this study shows that all the existing work either did not address the discovery process or propose simplistic procedures for the discovery phase. These procedures are often characterized by manual VNF selection or automated syntactic-based matchmaking between the offered and required VNFs. In other cases, the studied work entirely relies on ETSI MANO to discover and deploy the VNFs. Thus, they

Table 2.3: Synthesis of related works to the NS instantiation and management

Reference	NS instantiation	Dynamic re-configuration	Technology-based
Liu et al. [Liu 2017]	Linear	Yes	NFV
Zhang et al. [Zhang 2017]	Linear & parallel	No	NFV & SDN
Sun et al. [Sun 2017]	Linear, parallel & load balacing	No	NFV & SDN
Lee et al. [Lee 2015]	Linear	Yes	NFV & SDN
Mohammed et al. [Mohammed 2016]	Linear	Yes	NFV & SDN
Scheid et al. [Scheid 2016]	Linear	Yes	NFV & SDN
Martini et al. [Martini 2016]	Linear	Yes	NFV & SDN
Jmila et al. [Jmila 2019]	Linear & load balancing	Yes	SDN
Csoma et al. [Csoma 2014] Callegati et al. [Callegati 2015] Gharbaoui et al. [Gharbaoui 2017]	Linear	Yes	SDN

all suffer from the same issues highlighted in Section 2.3.1. Generally speaking, the studied discovery approaches require solid domain knowledge, are time-consuming, and are inefficient. These discovery procedures considerably decrease the agility and cost-effectiveness that one may expect from a virtualized network ecosystem. When it comes to the case of VNFs implemented as microservices, the studied papers emphasize VNF instances discovery rather than microservices discovery. To our best of knowledge, there is still no such ontology for microservices discovery.

Table 2.3 sums up the most relevant studied work concerning the NS instantiation and management problem. The literature review highlights that most existing work relies on SDN to ensure a dynamic control plane following the update of service function chaining. Furthermore, it is remarkable that all the reviewed research work have tried to bring agility and dynamicity in VNF chains using various and different approaches (e.g., QoS monitoring, statistics, heuristics) and at several phases of their life-cycle (e.g., deployment and execution). None of the proposed solutions cover the entire life-cycle (i.e., from design to *on-the-fly* management at run-time). Moreover, they all consider very simplistic VNF chains where VNFs are sequentially composed and do not tackle more complex and more sophisticated composition scenarios, as this might be the case in advanced NSs. Table 2.4 provides a comparative overview of the discussed related works about NS embedding. The comparison was based on the set of design metrics: the dynamic re-configuration of NSE during run-time, deployment across multi-administrative domains, the share of trust-partnership between the different actors, traffic equilibration over the available resources, and optimizing the deployment cost as an objective function.

Overall, all the reviewed works about NS embedding are performance-oriented, optimizing for cost or resource utilization on multi-domain infrastructures, such as [Luizelli 2018, Pei 2018]. Similar to the objective of our work, these works aim to min-

Table 2.4: Synthesis of related works to the NS deployment and management

Reference	Minimize deployment cost	Multi-domain	Security constraint	Dynamic re-configuration	Requests equilibration
Luizelli et al. [Luizelli 2018]	Yes	No	No	No	Yes
Houidi et al. [Houidi 2020]	Yes	No	No	Yes	Yes
Pei et al. [Pei 2018]	Yes	Yes	No	Yes	Yes
Morin et al. [Morin 2020]	Yes	Yes	No	No	No
5Grouth [Li 2021] 5G-Transformer [Baranda 2020]	Yes	Yes	Yes	No	No
Dwiardhika et al. [Dwiardhika 2019]	Yes	Yes	Yes	No	No
Alaluna et al. [Alaluna 2017] Fischer et al. [Fischer 2017] Wang et al. [Wang 2015] Torkzaban et al. [Torkzaban 2019]	Yes	No	Yes	No	No
Forti et al. [Forti 2020]	Yes	Yes	Yes	No	No

imize the total deployment cost concerning service performance requirements. However, They mistreated clearly the multi-provider aspect. This limitation is addressed by [Morin 2020]. In their model, they considered placing the VNFs in a hybrid geo-distributed environment with multiple providers. Regarding dynamic re-configuration of NSE, both [Liu 2017, Houidi 2020] consider updating the VNF chaining during run-time in order to satisfy user requirements. However, contrary to [Liu 2017], Houidi et al. [Houidi 2020] focus on applying the extensions required without interrupting the already deployed service, which is similar to our objective.

Although the virtualized infrastructure imposes specific security threats that need to be treated, security consideration receives only a few interests in the NSE problem. For example, the 5Growth and 5G-transformer European projects [Li 2021, Baranda 2020] introduced the trust-partnership aspect between service and infrastructure providers to achieve full service and resource federation. They assure an entire federation among providers that are using their orchestrator before any NS deployment. Similarly, [Torkzaban 2019, Torkzaban 2020] treat the trust aspect to only protect the service provider by allowing him to require security levels within their requests. Nevertheless, these works provide a single security degree for the service provider. All network service providers have to share the same security policy and priorities. Other works such as [Forti 2020, Alaluna 2017] allow the infrastructure providers to protect themselves by according a security level with other infrastructure providers.

2.5 Summary

This chapter introduced an overview of the background landscape necessary for understanding the thesis scope. After, we studied the research areas related to network service life-cycle management. Further, we have analyzed the state-of-the-art and underlined the research issues that need to be tackled.

In this context, it is fundamental to introduce a cost-driven approach for NS provisioning to address the limitations mentioned above. To this end, designing VNF discovery need to rely on a comprehensive and generic description. In addition, efficient re-configurable wiring capabilities for NSs should be defined to allow on-demand updates dynamically. Last but not least, the deployment of network service across multi-administrative domains should optimize both VNF placement for each network service along with traffic routing across VNFs while satisfying constraints like location and latency.

In the following chapters, we present our contributions to these primary challenges.

Chapter 3

A Semantic Virtualized Network Functions Description and Discovery Model

3.1 Introduction

Specific functions providers offer VNFs to prospective network service providers; Their VNFs are published in dedicated marketplaces where network providers search and instantiate them according to a pre-established service-level agreement. On top of being proprietary and specific to the functions providers, the existing VNF description models include details on VNF deployment but fail to fit VNF functional and non-functional specifications. This description alters an efficient selection of the most relevant VNFs and prevents full automation of the VNFs provisioning.

This chapter introduces a novel approach for VNFs description, publication, and discovery, to address the aforementioned limitations. The proposed approach inspired from service-oriented computing principles. The main contributions are twofold: 1) design of a domain-independent VIrtualized networK functIoN ontoloGy (VIKING for short) that enables a comprehensive and generic description of the VNF capabilities from functional and non-functional perspectives, and 2) development of a semantic-based matchmaker that relies on VIKING to ensure the best matching between requested VNFs and published ones. As for validation, we refine VIKING for the Content Delivery Networks (CDN) domain through an illustrative use case where VNF description and discovery are realized. The implemented prototype, called Mastermyr chest, fully automates and simplifies the VNFs discovery and instantiation procedures. Furthermore, it enables cooperation and federation between heterogeneous and proprietary providers in the NFV landscape. The performed experiments highlight that the proposed VNFs discovery algorithm is accurate and precise. Moreover, they also show that our algorithm can discover and select the most relevant VNFs with reasonable delays and overhead. Our initiative thus constitutes an important step for paving the way to NFV use in the novel and next-generation networks such as Content Delivery Networks (CDN), Internet of Things (IoT), and the fifth-generation (5G) of mobile telco networks, and consequently, fills a considerable gap in this emerging and promising area.

The rest of the chapter is organized as follows. Section 3.2 introduces VIKING for VNF description and its related matchmaker for semantic-based discovery. The associated Mastermyr Chest prototype is presented in Section 4.3. Section 4.4 details the performed experiments and discusses the obtained results. Finally, Section 4.5 concludes the chapter.

3.2 A semantic approach for VNF description and discovery

This Section first details VIKING ontology for semantically describing VNFs' capabilities from functional and non-functional perspectives. Then, it presents our VIKING-based matchmaking algorithm to discover the most relevant VNFs given specific network needs.

3.2.1 VIKING ontology for VNF description model

VIKING is an OWL-based (Ontology Web Language) ontology that allows describing VNFs. To design VIKING, we first determine what domain VIKING will cover (namely, network function virtualization), for what VIKING will be used (namely, VNF description, publication, and discovery), and for what types of queries VIKING should provide answers (namely, similarity and correlation). We then tackle the abstraction exercise by identifying the main common concepts shared by various application domains like CDNs, IoT, telco, and 5G networks. *Concepts* are organized as a class hierarchy, where abstract concepts will be refined with more concrete ones specific to each domain application. They are also described with properties and connected to other concepts with semantic relations. We tried not to reinvent the wheel, so we further reuse existing ontologies mainly related to VNF deployment (e.g., [Hoyos 2016]) and billing (e.g., [Afify 2017]). To assist VNF providers when creating comprehensive and consistent VNF descriptors, VIKING relies on OWL's reasoning principles. Fig. 3.1 depicts VIKING's high-level skeleton that consists of two interrelated ontologies, namely VIKING-F and VIKING-NF, related to VNF's functional and non-functional properties, respectively. On the one hand, VIKING-F refers to the formal specification of what precisely the VNF can do. It revolves around two dimensions known as **Business** and **Model**. **Business** denotes the VNF's type, inputs (i.e., details about the content upon which the VNF will take effect along with other necessary information), and outputs (i.e., details about the changes that will take place in the content). **Model** indicates the set of operations that ensure these inputs' conversion into outputs along with the related techniques and/or standards. On the other hand, VIKING-NF refers to the formal specification of what precisely the VNF needs/requires for proper function-

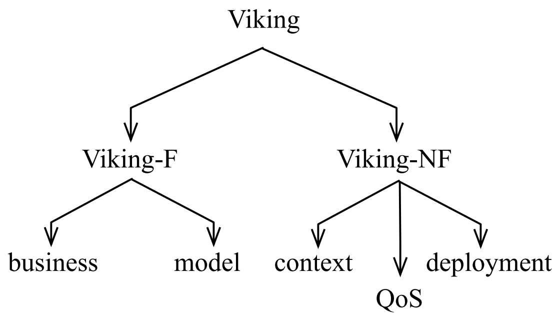


Figure 3.1: A high-level view of VIKING design

ing. It revolves around three dimensions known as **Context**, **QoS**, and **Deployment**. **Context** refers to the necessary runtime information (e.g., operating system, specific libraries, and/or system packages), as well as, device types (e.g., smartphones, TVs, desktops) upon which the VNF's outputs can be readable. **QoS** specifies common quality features offered by the VNF (e.g., response time, operation cost) and can be refined with specific-domain ones (e.g., surrogate servers locations for CDN, the bandwidth for 5G applications). Finally, **Deployment** involves VNF's artifact and configuration parameters that are needed for VNF's execution. Fig. 3.2 shows a more detailed view of

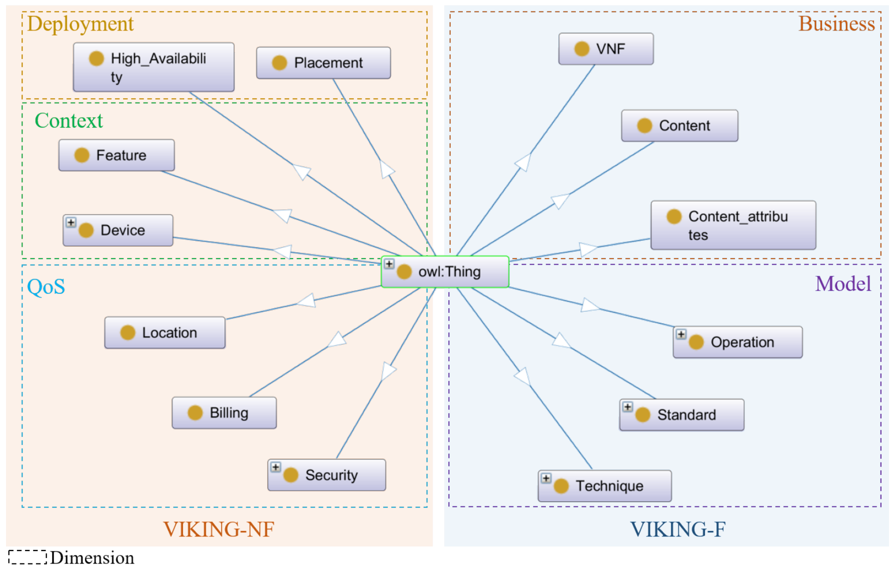


Figure 3.2: VIKING's core concepts

VIKING dimensions. Each dimension encompasses abstract conceptual areas that are instantiated using concrete concepts, producing a dedicated VIKING-F and VIKING-

NF ontologies. These concepts, as well as, the relations between them are discussed in-depth in the rest of this Section.

3.2.1.1 VIKING-F ontology

As mentioned earlier, VNF's functional properties are specialized into **Business** and **Model** dimensions, described as follows.
Business. This dimension relies on existing classification standards (e.g., ISO/IEC[4], ETSI NFV[5]) and leading service providers. Obviously, VNF design is always related to a target application domain. The VNF business description consists of three main concepts, namely, *VNF*, *Content*, and *Content-Attribute*, along with their semantic relations. *VNF* describes all necessary details on VNFs for advertisement and query-building purposes. Basically, *VNF* will be refined into concrete virtualized network functions for a given application domain. Since these functions share common concepts and semantic relations but also have their own technical specificities, they should be considered concepts rather than concept instances. *Content* refers to different domain-related artifact types manipulated by the VNFs. *Content-Attribute* indicates the type of content(s) supported by the VNF. More specifically, this concept represents the content's technical specification (e.g., required/supplied *Resolution* and *Quality*). It is worth noticing that *VNF*, *Content*, and *Content-Attribute* are semantically connected with relations, namely, ***delivers*** between *VNF* and *Content*, and ***requires/supplies*** between *VNF* and *Content-Attribute.* The first relation states that any VNF provides some content, while the second relation captures the input/output attributes upon which the VNF will act for specific content. Besides, for consistency purposes, cardinality restrictions (e.g., *at least one*) and axioms (e.g., *disjoint*) are specified, so that concept instances are related to the right instance(s) and belong to the right concepts. To ensure a consistent instantiation of concepts, Semantic Web Rule Language (SWRL) rules (including axioms) help enforce restrictions on attribute values and semantic relations, as well. Hereafter, we only exemplify SWRL rules referring to concepts, while those referring to instances will be discussed in Appendix **??**. For example, Equation 3.1 formally reflects the following statement: "*Any VNF (?x) that requires content-attribute (?y) should deliver specific content (?z)*".

$$\begin{aligned} &VNF(?x) \wedge \\ &\boldsymbol{requires}(?x, content\_attribute(?x, ?y)) \\ &\rightarrow \boldsymbol{delivers}(?x, content(?x, ?z)) \end{aligned} \tag{3.1}$$

[4]https://www.iso.org/standard/68291.html

[5]https://www.etsi.org/technologies-clusters/technologies/nfv

Table 3.1: Relations in VIKING-F with the concept *VNF*

Dimension	Relation	(Target) Concept
Business	***delivers***	*Content*
	requires/supplies	*Content_attribute*
Model	***implements***	*Operation*
	supports	*Standard*
	applies	*Technique*

Model. Technical aspects are relevant when making content exchangeable and adaptive in heterogeneous networks and devices (e.g., be able to read video in one digital encoding format different from the original video format). We thus rely on these aspects to identify three main concepts related to **Model**, namely, *Operation*, *Standard*, and *Technique* linked to *VNF* through ***implements***, ***supports***, and ***applies*** relations, respectively. Specifically, *Operation* refers to how a VNF changes on some content(s) described in **Business**. *Standard* contains different standard(s) in the target application domain to foster content exchanges. *Technique* encompasses methods and procedures that a specialized VNF applies to make the necessary changes to the content.
Furthermore, as in **Business**, restrictions and axioms such as "Any VNF can apply some techniques" might be defined. Also, in some cases, mapping **Business** onto **Model**, or vice versa, is required (e.g., matching VNF requests with VNF advertisement). To this end, SWRL rules are defined to infer new semantic relations between instances during concept instantiation. For instance, Equation 3.2 formally reflects the following statement: "*Any VNF (?x) that applies technique (?y) should implement a specific operation (?u)*".

$$\begin{aligned} & \mathit{VNF}(?x) \wedge \\ & \boldsymbol{applies}(?x, \mathit{technique}(?x, ?y)) \\ \rightarrow\ & \boldsymbol{implements}(?x, \mathit{operation}(?z, ?u)) \end{aligned} \tag{3.2}$$

Table 3.1 sums up the defined relations between the *VNF* concept and the rest of the VIKING-F concepts.

3.2.1.2 VIKING-NF ontology

As mentioned earlier, VNF's non-functional properties are specialized into **QoS**, **Context**, and **Deployment** description parts, described as follows.

QoS. This dimension consists of three concepts: *Location*, *Billing*, and *Security* linked to *VNF* through ***locates***, ***costs***, and ***ensures*** relations, respectively. *Location* refers to VNF's placement (e.g., network domain). *Billing* contains pricing models similar to those defined in cloud environments (e.g., time-based, volume-based, flat rate) [Mazrekaj 2016]. Last but not least, *Security* is related to *VNF* regardless of security mechanisms provided by the hosting platform. Indeed, the VNF should not depend on the hosting platform that can be itself a source of threats (e.g., malicious orchestrator or administrator) and thus ensure its own security compliance to ETSI NFV SEC recommendation [nfv 2015, Lal 2017]. Many existing security ontologies have been proposed in the literature, each for a specific purpose like eliciting security requirements [Souag 2015], certifying security claims [Trabelsi 2007], and determining cyber-attack goals [Doynikova 2018], to cite just a few. In this work, we deem to encompass some certification of the VNF's security capabilities into VNF discovery so that the VNF prospective consumers trust the VNF. To this end, we define *Security* with three other concepts, namely, *Security Goal*, *Security Requirement*, and *Security Property*, as depicted in Fig. 3.3. The first specifies what the VNF should prevent, while the second describes what should happen in some specific situation. A *Security Goal* can also be associated with *CIA* (stands for *Confidentiality*, *Integrity*, and *Availability*) properties. Each *CIA* property refers to protection mechanisms (e.g., cryptography, signature, and redundancy) for the VNF's capabilities along with raw/processed data against intrusions. It is worth noticing that *Security Requirement* as a constraint contributes to the satisfaction of a *Security Goal*. To meet *Security Requirement*, the VNF puts in place different *Defense* types like *Authorization*, *Authentication*, and *Trust-based*. *Authorization* refers to control access to the VNF and its data along with capabilities, respectively, by an authorized entity in an authorized manner (e.g., Role- and Identity-based mechanisms). *Authentication* refers to verification mechanisms (e.g., public key, certification, and password) for checking a source's identity, including traffic provenance. *Trust-based* refers to evaluation mechanisms (e.g., direct and collaborative trust-based) to establish trust relationships between VNFs. Finally, we refine *Security Property* into *Auditability* and *Accountability*. *Auditability* refers to VNF examination techniques (e.g., knowledge- and behavior-based), while *Accountability* refers to internal tracking mechanisms (e.g., logging) to monitor the VNF's activities. Note that this ontological model for capturing the VNF's **QoS** aspects, including security, can be easily enriched with more sophisticated ones based on the application domain. For instance, one might consider extending the **QoS** dimension with additional attributes like *performance* and *adaptability* to cite a few.

Context. This dimension encompasses two main concepts, namely *Device* and *Feature*. *Device* refers to additional details related to surrounding/target appliances (e.g., hosting machine), and *Feature* refers to options provided by the VNF (e.g., resize multimedia content in CDN, and switch communication protocol in IoT). To illustrate SWRL rules, Equation 3.3 formally reflects the following statement: "*Any VNF (?x) that implements operation (?y) and covers some device (?z) should supply a specific res-*

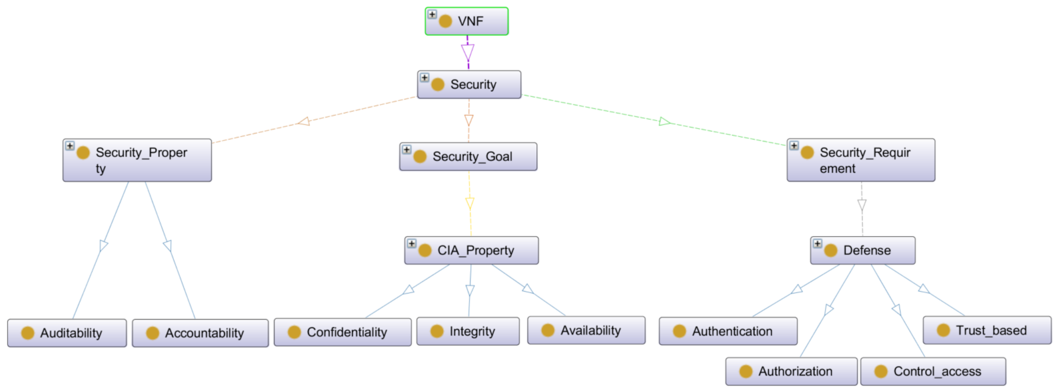


Figure 3.3: Security concept

olution (?u)".

$$\begin{aligned} \mathit{VNF}(?x) \wedge \\ \boldsymbol{implements}(?x, \mathit{operation}(?x, ?y)) \wedge \\ \boldsymbol{covers}(?x, \mathit{device}(?x, ?z)) \\ \rightarrow \ \boldsymbol{supplies}(?x, \mathit{resolution}(?z, ?u)) \end{aligned} \tag{3.3}$$

Deployment. This dimension integrates the already existing ETSI VNFD. It enricheses it with additional/complementary details on the resources to be allocated for VNF's hosting and execution (e.g., number of required CPUs, amount of RAM), as well as its high-availability. Specifically, *Placement* involves a URI of a remote enriched ETSI VNFD. Undeniably, *Placement* is a mandatory property during VNF discovery. Last but not least, *High-Availability* refers to attributes like what type of redundancy, how much redundancy, and resource requirements for redundancy as per ETSI recommendations [ETSI 2015]. Since ensuring high-availability improves the VNF's security, specifically availability (e.g., [Casazza 2017] and [Lin 2017]), we deem to link *High-Availability* and *Availability* (CIA property) with **increases** relation.

Table 3.2 sums up the defined relations between the *VNF* concept and the rest of the VIKING-NF concepts.

3.2.2 VNF discovery model

We propose a novel discovery model based on VIKING. VNF discovery process consists of two main steps: user request building and semantic matchmaking. First, this process starts with assisting the user (i.e., a network provider) build his/her VNF requests in terms of what VNF capabilities are required. Afterward, it calls for a semantic matchmaking algorithm to seek candidate VNFs offered by the providers in the ap-

Table 3.2: Relations in VIKING-NF with the concept *VNF*

Dimension	Relation	(Target) Concept
QoS	***locates***	*Location*
	costs	*Billing*
	ensures	*Security*
Context	***covers***	*Device*
	offers	*Feature*
Deployment	***refers_to***	*Placement*
	warrants	*High-Availability*

propriate repositories. Finally, the discovery process provides the user with the most relevant VNFs based on their preferences. The algorithm and methodology that implement each one of these steps are detailed as follows.

3.2.2.1 User request building

We define functional/non-functional requirement ($\mathcal{REQ}^F/\mathcal{REQ}^{NF}$) as a set of concepts in VIKING-F/VIKING-NF requested by the user ($\mathcal{U}_i$). Formally, Equation 3.4 represents the syntax used for specifying $\mathcal{REQ}^F$.

$$\mathcal{REQ}_i^F = VNF(?x)\ [\wedge\ Concept_j(?x,y)]_{j=1..n} \tag{3.4}$$

where

- $Concept_j \in$ VIKING-F such as *Operation* and *Technique*.
- $?x$ corresponds to the *VNF* instance(s) to be retrieved. Note $\mathcal{U}_i$ can refine *VNF* into concrete concepts related to a specific domain (see Section **??**).

Below, Equation 3.5 reflects the following $\mathcal{REQ}_i^F$: "*Any VNF (?x) that should implement some Operation (y) and require some Content_Attribute(z)*".

$$\mathcal{REQ}_i^F = VNF(?x) \wedge Operation(?x,y) \wedge Content\_Attribute(?x,z) \tag{3.5}$$

To specify $\mathcal{REQ}_i^{NF}$, we proceed as with $\mathcal{REQ}_i^F$ where VIKING-F is replaced with VIKING-NF. In accordance with the Web semantics principles, users can also define preferences among functional and/or non-functional requirements to select the most appropriate discovered services. To this end, we deem first to split $\mathcal{REQ}_i^F/\mathcal{REQ}_i^{NF}$ into $\{\mathcal{REQ}_{i,j}^F\}/\{\mathcal{REQ}_{i,k}^{NF}\}$ where $\mathcal{REQ}_{i,j}^F/\mathcal{REQ}_{i,k}^{NF}$ refers to $Concept_j(?x,y)/Concept_k(?x,w)$, respectively, as per Equation 3.4. Then, $\mathcal{U}_i$ defines her preference values for all $\mathcal{REQ}_{i,j}^F/\mathcal{REQ}_{i,k}^{NF}$. For the sake of simplicity,

Table 3.3: Notation

Symbol	Description
$\mathcal{U}_i$	User i
$\mathcal{REQ}_i^F$	$\mathcal{U}_i$'s functional requirement
$\mathcal{REQ}_{i,j}^F$	$\mathcal{REQ}_{i,j}^F \in \mathcal{REQ}_i^F$
$\mathcal{REQ}_i^{NF}$	$\mathcal{U}_i$'s non-functional requirement
$\mathcal{REQ}_{i,k}^{NF}$	$\mathcal{REQ}_{i,k}^{NF} \in \mathcal{REQ}_i^{NF}$
$\mathsf{Pref}(\mathcal{REQ}_i^F)$	User preference associated with $\mathcal{REQ}_i^F$
$\mathsf{Pref}(\mathcal{REQ}_i^{NF})$	User preference associated with $\mathcal{REQ}_i^{NF}$
$\mathsf{M}\text{-}\mathcal{CL}$	Mandatory-preference cluster
$\mathsf{H}\text{-}\mathcal{CL}$	High requested-preference cluster
$\mathsf{O}\text{-}\mathcal{CL}$	Optional-preference cluster
$\mathsf{Pref}(\mathcal{CL})$	User preference value associated with the cluster $\mathcal{CL}$
$[\![\mathcal{REQ}_{i,j}^F, \mathcal{CL}_{i,j}]\!]$	$\mathcal{REQ}_{i,j}^F$ is labeled with the preference cluster $\mathcal{CL}_{i,j}$
$[\![\mathcal{REQ}_{i,k}^{NF}, \mathcal{CL}_{i,k}]\!]$	$\mathcal{REQ}_{i,k}^{NF}$ is labeled with the preference cluster $\mathcal{CL}_{i,k}$

$\mathcal{REQ}_{i,j}^F/\mathcal{REQ}_{i,k}^{NF}$ will be classified into three preference clusters, namely, mandatory (M- $\mathcal{CL}$), high-requested (H- $\mathcal{CL}$), and optional (O- $\mathcal{CL}$). For readability purposes, Table 3.3 contains the notation used to formalize user requirements and preferences. After specifying all $\mathcal{REQ}_{i,j}^F$ and $\mathcal{REQ}_{i,k}^{NF}$ and labeling each requirement with a preference cluster $[\![\mathcal{REQ}_{i,j}^F, \mathcal{CL}_{i,j}]\!]$ and $[\![\mathcal{REQ}_{i,k}^{NF}, \mathcal{CL}_{i,k}]\!]$, $\mathcal{U}_i$ will define all preference values, namely, $\mathsf{Pref}(\mathcal{REQ}_i^F)$, $\mathsf{Pref}(\mathcal{REQ}_i^{NF})$, and $\mathsf{Pref}(\mathcal{CL})$. Note that all mandatory $\mathcal{REQ}_{i,j}^F$ and $\mathcal{REQ}_{i,k}^{NF}$ will serve to discard irrelevant VNFs. Note that $\mathsf{Pref}(\mathcal{REQ}_i^F) + \mathsf{Pref}(\mathcal{REQ}_i^{NF}) + \mathsf{Pref}(\mathsf{H}\text{-}\mathcal{CL}_i) + \mathsf{Pref}(\mathsf{O}\text{-}\mathcal{CL}_i) = 1$.
To sum up, the user request ($\mathcal{UR}_i$) is a 2-tuple defined as follows:

$$\begin{aligned} \mathcal{UR}_i =< \{[\![\mathcal{REQ}_{i,j}^F, \mathcal{CL}_{i,j}]\!]\}_{j=1,n}, \{[\![\mathcal{REQ}_{i,k}^{NF}, \mathcal{CL}_{i,k}]\!]\}_{k=1,m}, \\ \mathsf{Pref}(\mathcal{REQ}_i^F), \mathsf{Pref}(\mathcal{REQ}_k^{NF}), \mathsf{Pref}(\mathsf{H}\text{-}\mathcal{CL}_i), \mathsf{Pref}(\mathsf{O}\text{-}\mathcal{CL}_i) > \end{aligned} \tag{3.6}$$

3.2.2.2 Semantic matchmaking

Algorithm 1 reflects the matchmaking logic used to return the relevant set of candidate VNFs (Cand). It relies on VIKING when matching VNFs provided in a given repository (Rep) with user requests.

Algorithm 1 consists of two types of matching, namely, `matchAll` (Line 2) and `matchSome` (Line 7). On one hand, since the set of all mandatory requirements (i.e., $\{[\![\mathcal{REQ}_{i,j}^F, \mathsf{M}\text{-}\mathcal{CL}]\!]\}$ and $\{[\![\mathcal{REQ}_{i,k}^{NF}, \mathsf{M}\text{-}\mathcal{CL}]\!]\}$) **should** be fulfilled, `matchAll` checks if the VNF exactly matches this set (i.e., true or false). Indeed, any VNF should be either kept or discarded in/from Cand depending on the result provided by `matchAll`. On the other hand, `matchSome` is applied to the rest of the user request. For each VNF in Cand, `matchSome` returns a set of matched capabilities (M-Cap) that could be empty if there is no matching at all. Finally, the Cand list will be ranked based on VNF scores.

Algorithm 1 VNF matchmaking

```
1: VNF-MATCHMAKING($\mathcal{UR}_i$, Rep)
2: for all $VNF_i \in$ Rep do
3:   if matchAll($VNF_i$, $\{[\![\mathcal{REQ}^F_{i,j}, \text{M-}\mathcal{CL}]\!]\}$, $\{[\![\mathcal{REQ}^{NF}_{i,k}, \text{M-}\mathcal{CL}]\!]\}$) then
4:     append($VNF_i$, Cand)
5:   end if
6: end for
7: for all $VNF_i \in$ Cand do
8:   score(matchSome($VNF_i$, $\{[\![\mathcal{REQ}^F_{i,j}, \text{H-}\mathcal{CL}]\!]\}$, $\{[\![\mathcal{REQ}^{NF}_{i,k}, \text{H-}\mathcal{CL}]\!]\}$,
         $\{[\![\mathcal{REQ}^F_{i,j}, \text{O-}\mathcal{CL}]\!]\}$, $\{[\![\mathcal{REQ}^{NF}_{i,k}, \text{O-}\mathcal{CL}]\!]\}$), M-Cap)
9: end for
10: rank(Cand)
```

3.3 Mastermyr Chest architecture and implementation

For illustration purpose, we instantiated VIKING with the CDN case study. This results into the domain-specific VIKING-CDN ontology reported in Appendix ??.

3.3.1 Proof-of-concept architecture

The developed PoC is called the Mastermyr Chest. Its name refers to the tool chest found in Mastermyr[9] on the Gotland island, Sweden, in 1936. This chest box contained more than two hundred objects used by Viking carpenters. Similarly, our Mastermyr Chest prototype has several instruments useful for VNFs description, publication, discovery, and so on. Fig. 3.4 depicts the Mastermyr Chest tools, as well as the main interactions between them. The reader should note that the Mastermyr Chest was designed and implemented in a modular fashion to be easily extended with additional tools in the future. VIKING-CDN was implemented with Protégé 2000 ontology editor[10], while Mastermyr Chest tools were developed with Java. The associated source code is available on a GitHub repository[11]. The *description tool*[12] assists VNF developers (possibly,

[9] `https://en.wikipedia.org/wiki/M%C3%A4stermyr_chest`
[10] `https://protege.stanford.edu/`
[11] `https://github.com/NourelhoudaNouar/VNF-Description-Discovery`

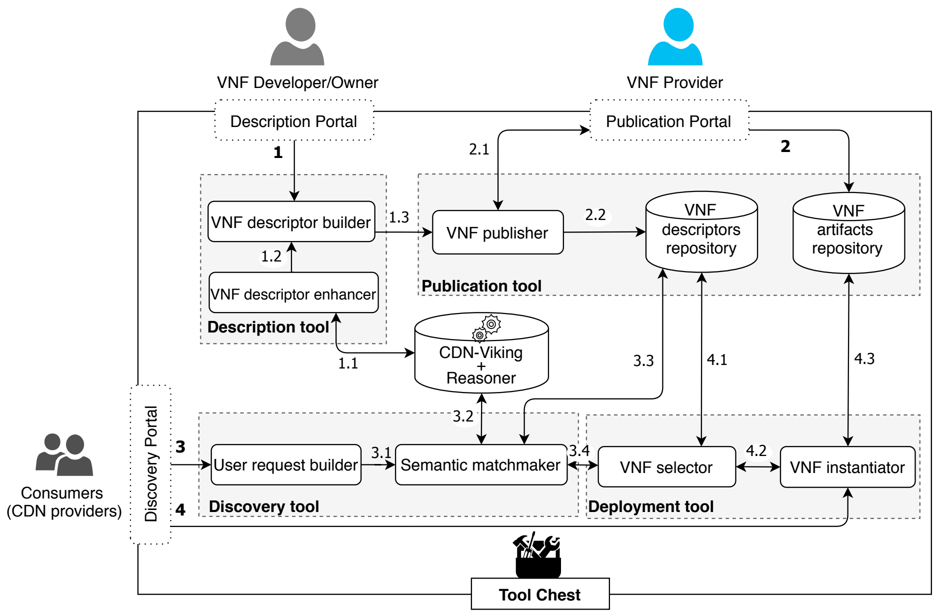
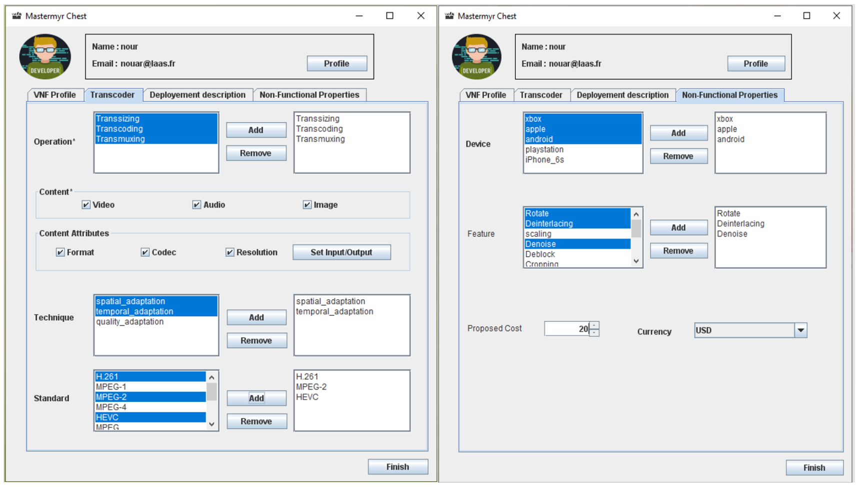


Figure 3.4: The Mastermyr chest tool architecture

VNF owners) to semantically describe the VNFs that are relevant to the CDN context (action 1). In accordance with the model introduced in Section 3.2.1, some information are mandatory, and others are not. VNF descriptors can be enriched with QoS details using the *VNF descriptor enhancer* (action 1.1). For instance, VNF developers can specify *Location* details about the VNFs that implement the CDN's surrogate servers. This would help CDN placing the popular multimedia content in the closest servers with regard to the end users location to reduce the delivery time. Afterwards, the *VNF descriptor builder* generates the VIKING-CDN-compliant descriptors of the VNFs (action 1.2) and forwards them to the *VNF publisher* (action 1.3). The VNF descriptors are implemented as OWL files. Snapshots of the *description tool* are shown in Fig. 4.6. The *publication tool* enables publishing the VNF artefacts (deployables) in the *VNF artefacts repository* to make them available to CDN providers (action 2). The *VNF publisher* requests the VNF artefacts' Unified Resource Identifier (URI) (action 2.1). After that, it annotates the VNF descriptor file with this URI and saves it in the *VNF descriptors repository* (action 2.2). For the current prototype, we did consider concrete VNF artefacts that implement one of the following middleboxes:

- **A multimedia mixer** that enables mixing several multimedia contents and returns a resulting content (e.g., adding voice to a video, adding ads banner to an image/video),

[12] A demo is available at: `https://drive.google.com/drive/folders/1ocJgxdP_oEVdPmQMFQlvNft7jsx7IhKn?usp=sharing`



Name : nour
Email : nouar@laas.fr

Operation*: Transsizing, Transcoding, Transmuxing — selected: Transsizing, Transcoding, Transmuxing

Content*: [✓] Video [✓] Audio [✓] Image

Content Attributes: [✓] Format [✓] Codec [✓] Resolution

Technique: spatial_adaptation, temporal_adaptation, quality_adaptation — selected: spatial_adaptation, temporal_adaptation

Standard: H.261, MPEG-1, MPEG-2, MPEG-4, HEVC, MPEG — selected: H.261, MPEG-2, HEVC

Device: xbox, apple, android, playstation, iPhone_6s — selected: xbox, apple, android

Feature: Rotate, Deinterlacing, scaling, Denoise, Deblock, Cropping — selected: Rotate, Deinterlacing, Denoise

Proposed Cost: 20

Currency: USD

Figure 3.5: Snapshots of the *description tool* interfaces

- **A multimedia compressor** that enables compressing the size and quality of multimedia content (e.g., degrading a high-definition video quality to save storage space or to decrease delivery time),
- **A multimedia transcoder** that converts original multimedia content to other formats using appropriate codecs (e.g., converting MP4 video to AVI).

The FFmpeg[13] open-source solution was used to implement these three middleboxes as VNFs. FFmpeg involves a suite of codecs, libraries and programs to handle video, audio, and other multimedia files and streams. Several and various FFmpeg instances with different configurations and packaging are implemented, according to the characteristics and capabilities mentioned in the VNF descriptors. In turn, VNF providers store their instances into the *VNF artefacts repository* as Ubuntu-based virtual machine appliances. The *discovery tool*[14] allows the VNF consumers (i.e., CDN providers in this specific case) to build their requests to calculate the matchmaking between required and offered VNFs (action 3). First, the *user request builder* assists VNF consumers to define a formal and VIKING-CDN-compliant request based on their functional and non-functional needs and preferences. The request is then forwarded as a required VNF descriptor to the *semantic matchmaker* (action 3.1). This *matchmaker* uses Algorithm 1 (Section 3.2.2.2) where `matchAll` and `matchSome` rely on VIKING-CDN's reasoner (Section ??) that infers relevant relationships between concepts and instances (action 3.2). Then, the *semantic matchmaker* calculates the matching scores of the requested VNF with regard to the offered VNFs descriptors published in the *VNF*

[13]`https://www.ffmpeg.org/`

[14]A demo is available at: `https://drive.google.com/drive/folders/1ocJgxdP_oEVdPmQMFQlvNft7jsx7IhKn?usp=sharing`.

descriptors repository (action 3.3). Finally, the *semantic matchmaker* transmits the obtained ranked list to the *VNF selector* (action 3.4) (e.g., see the snapshot in Fig. 3.15). The *semantic matchmaker* relies on OWL API[12] and Jena[13] plug-ins to parse OWL files and perform the OWL reasoning. The *deployment tool* enables providing a published VNF in a target network topology (action 4). First, the *VNF selector* downloads and parses its VNF descriptor. Obviously, following a discovery procedure, it selects and processes the VNFs descriptor with the highest matching score (action 4.1). Then, it forwards its URI to the *VNF instantiator* (action 4.2). The latter is responsible for downloading the VNFs, deploying them in the target CDN network, configuring, and integrating them into the existing topology (action 4.3).

3.3.2 Mastermyr Chest integration to ETSI NFV MANO

For dissemination and normalization purposes, Fig. 3.6 depicts the integration plan for the Mastermyr Chest tool built around VIKING into ETSI NFV standards. As mentioned in Section 1, the ETSI MANO framework supports VNF provisioning according to the procedures defined by ETSI and described in [Virtualisation 2014]. MANO mainly consists of three key components namely, NFV Orchestrator (NFVO), VNF Manager (VNFM), and Virtualized Infrastructure Manager (VIM). Broadly speaking, NFVO is responsible of instantiating, deploying and executing VNFs according to the strategies established by the OSS/BSS. To each provisioned VNF, MANO associates a dedicated VNFM that manages the VNF lifecycle at runtime (e.g. starting, scaling, migrating, and terminating). VNFM closely interacts with VIM that maintains the necessary compute and storage appliances from the NFV Infrastructure (NFVI) for the proper functioning of the VNFs. The reader should note that all the tools of the Mastermyr Chest tool box were developed with respect to ETSI NFV specification and procedures. Specifically, the VIKING descriptor format is compliant with the VNFD information model specification described in [Nguyenphu 2018]. The data types and communication protocols supported by Mastermyr chest tool box are in line with the ETSI specification described in [ets 2020]. Indeed, all the tools are RESTful and compliant with the ETSI policy management generic interface. To integrate and connect the Mastermyr Chest tools to the MANO components, we propose the following configuration. First, the *description tool* is standalone. It will assist VNF developers specify VIKING-compliant VNF descriptors. When it comes to the VNFs publication, the *publication tool* provides the NFVO with the VIKING-compliant descriptors and their related VNF artifacts (action 1). As is presently the case for regular VNFD storage in the MANO, the NFVO stores the VIKING descriptors in appropriate VNF catalogue (e.g., document-based database) and the associated atefacts in

[12]`http://owlapi.sourceforge.net/`

[13]`https://jena.apache.org/documentation/ontology/`

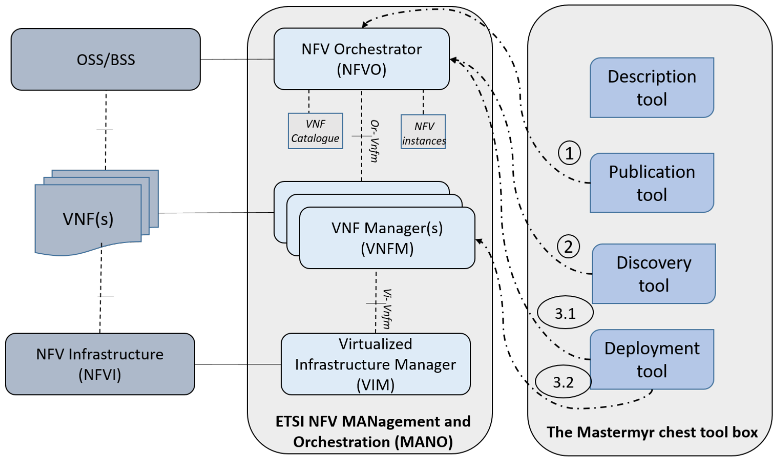


Figure 3.6: The Mastermyr Chest integration to ETSI NFV MANO

the NFV repository (e.g., VM image). As for the VNFs discovery, the ETSI standard procedure is kept. Specifically, the *discovery tool* first extracts a valid VNFD from a VIKING-compliant request and forwards it to the NFVO. The latter processes the request according to the standard VNFD discovery procedure supported by the MANO and sends back, to the *discovery tool*, a list of VNF candidates (action2). This list is refined thanks to our semantic matchmaker implemented within the *discovery tool* (eee Section 6.1).

The most relevant VNF with regard to the request is forwarded to the *deployment tool*. The latter first interacts with the NFVO to trigger the required VNF instantiation and the initialization of its corresponding VNFM (action 3.1). Then, the *deployment tool* communicates with the VNFM to acknowledge the creation of the VNF and start it (action 3.2).

3.4 Validation and evaluation

This Section discusses the performed experiments to evaluate and validate our findings. First, it describes the considered test collection and the comparative study metrics. Then, it presents the obtained measurements in terms of performance and robustness.

3.4.1 Test collection

To conduct experiments on VNF semantic discovery, we first proceed with the test collection creation. This collection includes three items:

1. A comprehensive set of valid VIKING-compliant VNFDs ($\mathcal{D}$) that covers conversion, mixing, and/or compression functions in the CDN domain,

2. A set of test queries ($\mathcal{Q}$) that challenges our semantic matchmaker in terms of false positive/negative outcomes,

3. A set of relevant VNFs per query ($\mathcal{V}_{\mathcal{Q}}$) that denotes all true positive outcomes.

These three items are detailed in the rest of this Section.

3.4.1.1 Illustrative VNFDs for the CDN use case

To achieve a comprehensive coverage for $\mathcal{D}$ (i.e., all possible valid VNFDs), we proceed as follows. First, we define the association rules listed in Table 3.4. These rules map VIKING's tree structure with semantic parsing onto VIKING's hypergraph structure (see Fig. 3.7) with syntactic parsing.

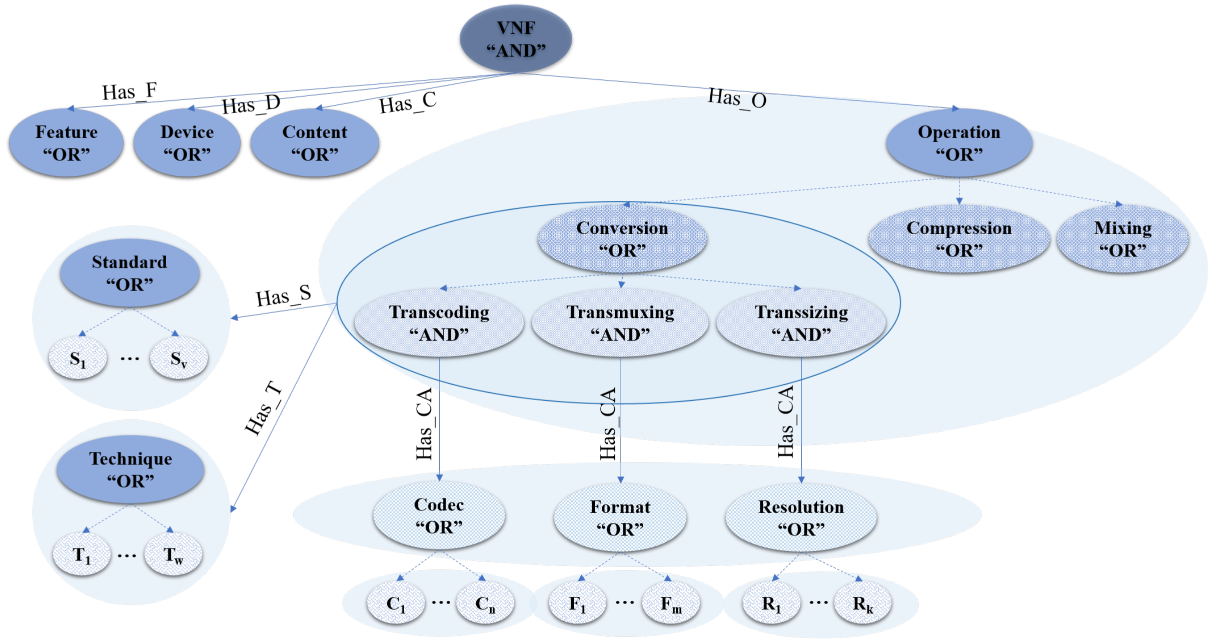


Figure 3.7: VIKING's partial syntactic representation

Formally, this hypergraph $\mathcal{G}$ is defined as a 3-tuple $<$ T, NT, H $>$ where

- T denotes the multiset of terminal nodes that correspond to VIKING's instances where each set (T(c)) refers to specific concept c.

- NT denotes the set of non-terminal nodes corresponding to VIKING's abstract and concrete concepts for the CDN domain (e.g., Operation and Transmuxing). We refine non-terminal into OrNode and AndNode to represent inheritance (INH) and object properties (OP) among concepts (c_j) respectively. $\{\mathsf{R}_i\}_{i=1,3}$ reported in Table 3.4 indicate when and how to create OrNode and AndNode.

- H represents the multiset of labeled hyperedges that refers to a set of OrNode and AndNode. Formally, H is defined as a 2-tuple $< \mathsf{NT}, \mathsf{L}, 2^{\mathsf{NT}} >$ where L refers to a set of labels like OP and/or OR (Fig. 3.7). $\{\mathsf{R}_i\}_{i=4,6}$ reported in Table 3.4 indicate when and how to create HyperEdge.

To generate $\mathcal{D}$, we adapt the well-known Depth First Search (DFS) so that possible valid VNFDs are built incrementally during visiting nodes. Algorithm 2 reflects this adaptation. This algorithm associates each visited node with some partial VNFD template where field names refer to all the parent nodes' terms. In Lines 5-11, the algorithm splits OrNode's children into a set of nodes, each corresponding to some combination of children. In Lines 14-17, it concatenates AndNode's children partial VNFD templates. As a result, $\mathcal{D}$ contains 695 VIKING-complaint VNFDs for our CDN use case.

Algorithm 2 $\mathcal{D}$'s generation

```
1: Adapted-DFS($G, queue, current_t, template, D$)
2:               // $\mathcal{G}$ is the hypergraph
3:               // $queue$ is initialized to $\mathcal{G}.root$
4:               // $current_t$ refers to the current node in $\mathcal{G}$
5:               // $template$ refers to a certain VNFD template
6:               // $\mathcal{D}$ is the set of all VNFD produced
7: if $queue \neq \emptyset$ then
8:    $current_t = queue$.pop()
9:    $template$.add($current_t$)
10:   if leafNode($current_t$) then
11:      $\mathcal{D}$.add($template$)
12:   else if OrNode($current_t$) then
13:      for all $subset\_t \in current_t$ do
14:         $queue$.push($subset\_t$)
15:         adapted-DFS($\mathcal{G}, queue, current_t, template, \mathcal{D}$)
16:      end for
17:   else if AndNode($current_t$) then
18:      for all $child\_t \in current_t$ do
19:         $queue$.push($child\_t$)
20:         adapted-DFS($\mathcal{G}, queue, current_t, template, \mathcal{D}$)
21:      end for
22:   end if
23: end if
```

[1]For AndNode, same as OrNode_2.

Table 3.4: Association rules

Rule	Condition	Action
R_1	$\mathsf{c} \in \mathsf{NT}$ & $\mathsf{T}(\mathsf{c}) \neq \emptyset$	$\mathsf{createOrNode}_1(x,\ \mathsf{T}(x))$
R_2	$\lvert\{\mathsf{INH}(x, y_i)\}\rvert \geq 2$	$\mathsf{createOrNode}_2(x,\{y_i\})$
R_3	$\lvert\{\mathsf{OP}[x, y_i]\}\rvert \geq 2$	$\mathsf{createAndNode}(x,\ \{\mathsf{OP}, y_i\})$
R_4	$\mathsf{OP}[x, \mathsf{OrNode}_1(y_i, \{y_{i,j}\})]$	$\mathsf{createHyperEdge}(x,\ \{y_{i,j}\})$
R_5	$\mathsf{OrNode}(x, \mathsf{OrNode}_2(y_i, \mathsf{OP}, \{y_{i,j}\}))$	$\mathsf{createHyperEdge}(x,\ \{y_{i,j}\})$
R_6	$\mathsf{AndNode}(x, \{\mathsf{OP}, \mathsf{OrNode}_2(y)\})$[1]	$\mathsf{createHyperEdge}(x,$ $\{\mathsf{OP}, \mathsf{OrNode}_2(y)\})$

3.4.1.2 Sample queries

To challenge our semantic matchmaker presented in Section 3.2.2.2, we build $\mathcal{Q}$ by using two types of query ambiguity introduced by Song et al. [Song 2007]. These authors classify Web queries into broad but clear and ambiguous. The former refers to queries that cover diverse "subtopics but a narrow topic," and the latter relates to queries with more than one meaning. For experimentation purposes, we refine broad-but-clear and ambiguous as follows. In the first, user requirements are specified with implicit (or hidden) terms that can be inferred by VIKING-CDN's reasoner, only. In the second, user requirements include the same naming for different instances, but only the user can remove ambiguity. Table 3.5 depicts 10 sample queries, and each described with $\mathcal{REQ}_i^F$ and/or $\mathcal{REQ}_i^{NF}$. For clarity purposes, we omit preferences from Table 3.5. On top of this, the queries are classified into either broad but clear or ambiguous depending on the source of ambiguity (Table 3.6).

We, thus, expect that our semantic matchmaker with broad-but-clear/ambiguous user queries as inputs and little knowledge about the user preferences will provide VNFs candidates that would correspond to probably inconsistent interpretations (i.e., false positive/negative outcomes).

3.4.1.3 VNFD relevance

Prior to proceeding with the set of relevant VNFDs per query ($\mathcal{V}_{\mathcal{Q}}$), we build some VNF template (t_j) per $\mathcal{Q}_j$ referring to expected VNF properties required to satisfy this query (Table 3.7). To obtain $\mathcal{V}_{\mathcal{Q}}$, we automatically annotate $\mathcal{D}$ with binary relevance values. A VNF$_i$'s relevance ($\mathcal{R}$) to a certain query ($\mathcal{Q}_j$) refers to what extent this VNF's VNFD$_i$ would be compliant with t_j (i.e., user satisfaction degree). Formally, Equation 3.7 computes $\mathcal{R}$ as follows.

Table 3.5: Sample queries ($\mathcal{Q}$)

Query	$\mathcal{REQ}_i^F$ and/or $\mathcal{REQ}_i^{NF}$
$\mathcal{Q}_1$	$VConverter(?x) \wedge C\_A(aac) \wedge C\_A(mp3) \wedge Device(Zune)$
$\mathcal{Q}_2$	$VConverter(?x) \wedge C\_A(aac) \wedge C\_A(wav) \wedge Device(iRiver)$
$\mathcal{Q}_3$	$VConverter(?x) \wedge C\_A(avc) \wedge C\_A(wmv) \wedge Device(ppc)$
$\mathcal{Q}_4$	$VConverter(?x) \wedge C\_A(mp4) \wedge C\_A(wmv) \wedge Device(pmp)$
$\mathcal{Q}_5$	$VConverter(?x) \wedge C\_A(mp3) \wedge Device(iPhone\_6s)$
$\mathcal{Q}_6$	$VConverter(?x) \wedge C\_A(mp4) \wedge Device(Galaxy_s3)$
$\mathcal{Q}_7$	$VConverter(?x) \wedge C\_A(mkv) \wedge C\_A(mp3)$
$\mathcal{Q}_8$	$VConverter(?x) \wedge C\_A(avi) \wedge C\_A(gif)$
$\mathcal{Q}_9$	$VMixer(?x) \wedge Content(video) \wedge Content(image)$
$\mathcal{Q}_{10}$	$VMixer(?x) \wedge Content(image) \wedge Content(audio)$

$C\_A : Content\_Attribute$

Table 3.6: Query classification

Query type	Query	Source
ambiguous	$\mathcal{Q}_1$	$Content\_attribute$
	$\mathcal{Q}_2$	
	$\mathcal{Q}_3$	
	$\mathcal{Q}_4$	
broad-but-clear	$\mathcal{Q}_5$	$Device$
	$\mathcal{Q}_6$	
	$\mathcal{Q}_7$	$Technique$
	$\mathcal{Q}_8$	
	$\mathcal{Q}_8$	$Operation$
	$\mathcal{Q}_{10}$	

Table 3.7: Expected VNF properties per query

Query	VNF template
$\mathcal{Q}_1$	$Operation(transcoding) \wedge Content(audio) \wedge C\_A(codec)$
$\mathcal{Q}_2$	$Operation(transmuxing) \wedge Content(audio) \wedge C\_A(format)$
$\mathcal{Q}_3$	$Operation(transcoding) \wedge Content(video) \wedge C\_A(codec)$
$\mathcal{Q}_4$	$Operation(transmuxing) \wedge Content(video) \wedge C\_A(format)$
$\mathcal{Q}_5$	$Operation(transsizing) \wedge Content(audio) \wedge C\_A(resolution)$
$\mathcal{Q}_6$	$Operation(transsizing) \wedge Content(video) \wedge C\_A(resolution)$
$\mathcal{Q}_7$	$Operation(transcoding) \wedge Content(video) \wedge Content(audio) \wedge Technique(audio\_separation)$
$\mathcal{Q}_8$	$Operation(transcoding) \wedge Content(video) \wedge Content(image) \wedge Technique(image\_sequence)$
$\mathcal{Q}_9$	$Operation(video\_image\_mixing) \wedge Content(video) \wedge Content(image)$
$\mathcal{Q}_{10}$	$Operation(image\_audio\_mixing) \wedge Content(image) \wedge Content(audio)$

$C\_A : Content\_Attribute$

$$\mathcal{R}^{\mathcal{Q}_j}(VNF_i) = \begin{cases} 1 & \text{if } |\{t_j.(\mathsf{c}_k|\mathsf{inst}_p) \in \mathsf{VNFD}_i\}| \geq \sigma_{\mathsf{c}_k},\ \forall c_k \in t_j \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

where

- $t_j.(\mathsf{c}_k|\mathsf{inst}_p)$ refers to p^{th} instance of the concept c_k in the template t_j.
- σ_{c_k} denotes the minimum number of c_k's instances that should be included in any satisfactory VNF$_i$.

This annotation was performed on every VNFD in $\mathcal{D}$ for all test queries ($\{\mathcal{Q}_j\}_{j=1,10}$). Fig. 3.8 shows how VNFs annotated with 1 (see Equation 3.7) are distributed over $\mathcal{Q}$. We can observe a non-uniform distribution over $\mathcal{D}$. For instance, the set of generated VNFs satisfying queries related to conversion ($\{\mathcal{Q}_j\}_{j=1,8}$) is more represented than the other sets. This distribution is due to a significant number of possible conversion-related capabilities compared to other operations like mixing.

3.4.2 Performance analysis

To assess the proposed approach's performance, we use 2 metrics namely, completeness and efficiency. The former describes how well our VIKING-based matchmaker (Mastermyr chest) identifies the relevant VNFs compared with the total number of such VNFs that exist in the test collection. The latter describes how well Mas-

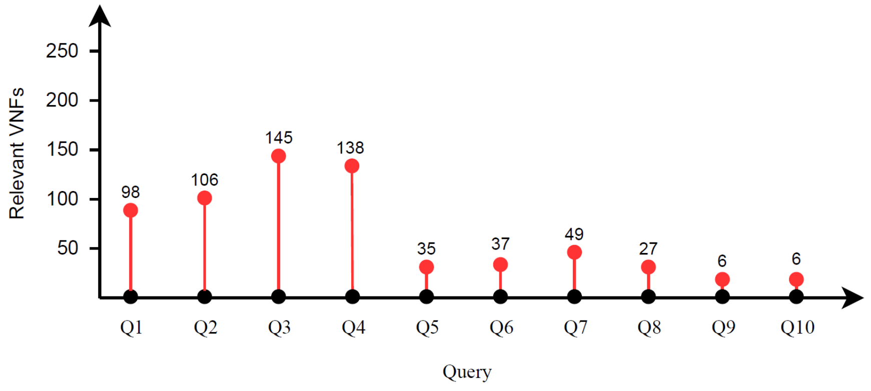


Figure 3.8: Probability distribution in $\mathcal{D}$ over $\mathcal{Q}$

termyr chest identifies only those relevant VNFs, by comparing the number of target VNFs identified with the total number of VNFs retrieved.

3.4.2.1 Performance metrics

First, we classify the retrieved VNFs provided by each matchmaker into three sets, namely True Positive (TP), False Positive (FP), and False Negative (FN) where

- TP contains the **retrieved** VNFs that are relevant as per Section 3.4.1.3,
- FP contains the **retrieved** VNFs that are not relevant, and
- FN contains the **relevant** VNFs that are not retrieved (i.e., discarded by the matchmaker).

Once the sets mentioned above are established, we use two well-known performance measurements in the semantic Web and Machine Learning communities (e.g., [Powers 2011]), namely, recall ($\mathcal{R}$) and precision ($\mathcal{P}$) that implement completeness and efficiency metrics (e.g., [Pace 2012]), respectively, and are defined as follows:

- $\mathcal{R}$ refers to the ratio between the number of true positive VNFs and the number of **relevant** VNFs, including true positive VNFs and false negative VNFs, as well (Equation 3.8).

$$\mathcal{R} = \frac{\mathsf{TP}}{\mathsf{TP} + \mathsf{FN}} \tag{3.8}$$

- $\mathcal{P}$ refers to the ratio between the number of true positive VNFs and the total number of **retrieved** VNFs, including true positive and false positive VNFs (Equation 3.9).

$$\mathcal{P} = \frac{\mathsf{TP}}{\mathsf{TP} + \mathsf{FP}} \tag{3.9}$$

It happens that $\mathcal{R}$ and $\mathcal{P}$ can be inversely related. On the one hand, lower $\mathcal{R}$ increases the risk to miss relevant VNFs and, therefore, would penalize VNF providers. On the other hand, lower $\mathcal{P}$ denotes a significant number of irrelevant VNFs and, thus, would mislead the end-users. To estimate to what extent VNF providers/end-users should trust our matchmaker, we rely on $\mathcal{F}$-measure that reflects the right balance between $\mathcal{R}$ and $\mathcal{P}$. Formally, $\mathcal{F}$-measure can be defined as follows:

$$\mathcal{F}\text{-measure} = 2 \times \frac{\mathcal{R} \times \mathcal{P}}{\mathcal{R} + \mathcal{P}} \tag{3.10}$$

Note that Equation 3.10 considers $\mathcal{R}$ and $\mathcal{P}$ as equally important.

On top of the aforementioned metrics, we deem appropriate to measure the overhead in terms of response time ($\mathcal{RT}$) defined as the amount of time necessary to get a response from the matchmaker following a discovery request sent by an end-user.

3.4.2.2 Measurement and discussion

We run experiments that challenge the *discovery tool* of the Mastermyr chest with the test collection including $\mathcal{Q}$. To demonstrate our matchmaker's completeness and efficiency, we first consider another sample of queries ($\mathcal{Q}'$) obtained from $\mathcal{Q}$'s (partial or full) disambiguation by adding new VNF property per $\mathcal{Q}_i$, as depicted in Table 3.8. After, we measured *discovery tool* performance in terms of precision, recall, $\mathcal{F}$-measure, and response time. The reader should note that we consider all matching outcomes obtained by our semantic matchmaker, given some $\mathcal{Q}_i$. The obtained results are discussed in the rest of this Section.

Fig. 3.9 depicts $\mathcal{R}$ rates for all $\mathcal{Q}_i$ and their corresponding $\mathcal{Q}'_i$. We can observe that our matchmaker with queries $\mathcal{Q}'_5$, $\mathcal{Q}'_6$, $\mathcal{Q}'_9$, $\mathcal{Q}'_{10}$ provides better $\mathcal{R}$ rates than with $\mathcal{Q}_5$, $\mathcal{Q}_6$, $\mathcal{Q}_9$, $\mathcal{Q}_{10}$, up to 33%. For instance, for $\mathcal{Q}'_9$, $Operation(video\_image\_mixing)$ helps retrieving VNFs that implement mixing capabilities without specific *Content*. As for queries $\mathcal{Q}'_1$ to $\mathcal{Q}'_4$ and $\mathcal{Q}'_7$ to $\mathcal{Q}'_8$, we can observe constant $\mathcal{R}$ rates compared to their corresponding $\mathcal{Q}_i$. Indeed, VIKING-CDN's reasoner through SWRL rules help the matchmaker identify relevant VNFs described with minimal required VNF properties. For instance, for $\mathcal{Q}'_8$, $Content(image)$ does not help retrieving VNFs that support $Technique(image\_sequence)$ due to the SWRL rule (Equation 3.1) instantiated with gif and $image$.

Fig. 3.10 depicts $\mathcal{P}$ rates for all $\mathcal{Q}_i$ and their corresponding $\mathcal{Q}'_i$ with respect to the relevant VNFs for $\mathcal{Q}_i$ as reported in Fig. 3.8. Overall, we can observe that our matchmaker achieves better $\mathcal{P}$ rate with queries $\mathcal{Q}'_1$ to $\mathcal{Q}'_6$ than with $\mathcal{Q}_1$ to $\mathcal{Q}_6$, respectively, with

Table 3.8: Sample queries ($\mathcal{Q}'$)

Query	$\mathcal{REQ}_i^F$ and/or $\mathcal{REQ}_i^{NF}$
$\mathcal{Q}'_1$	$VConverter(?x) \wedge C\_A(aac) \wedge C\_A(mp3) \wedge Device(Zune) \wedge C\_A(audio\_codec)$
$\mathcal{Q}'_2$	$VConverter(?x) \wedge C\_A(aac) \wedge C\_A(wav) \wedge Device(iRiver) \wedge C\_A(audio\_format)$
$\mathcal{Q}'_3$	$VConverter(?x) \wedge C\_A(avc) \wedge C\_A(wmv) \wedge Device(ppc) \wedge C\_A(video\_codec)$
$\mathcal{Q}'_4$	$VConverter(?x) \wedge C\_A(mp4) \wedge C\_A(wmv) \wedge Device(pmp) \wedge C\_A(video\_format)$
$\mathcal{Q}'_5$	$VConverter(?x) \wedge C\_A(mp3) \wedge Device(iPhone\_6s) \wedge Content(audio)$
$\mathcal{Q}'_6$	$VConverter(?x) \wedge C\_A(mp4) \wedge Device(Galaxy_s3) \wedge Content(video)$
$\mathcal{Q}'_7$	$VConverter(?x) \wedge C\_A(mkv) \wedge C\_A(mp3) \wedge Content(audio)$
$\mathcal{Q}'_8$	$VConverter(?x) \wedge C\_A(avi) \wedge C\_A(gif) \wedge Content(image)$
$\mathcal{Q}'_9$	$VMixer(?x) \wedge Content(video) \wedge Content(image) \wedge Operation(video\_image\_mixing)$
$\mathcal{Q}'_{10}$	$VMixer(?x) \wedge Content(image) \wedge Content(audio) \wedge Operation(image\_audio\_mixing)$

$C\_A : Content\_Attribute$

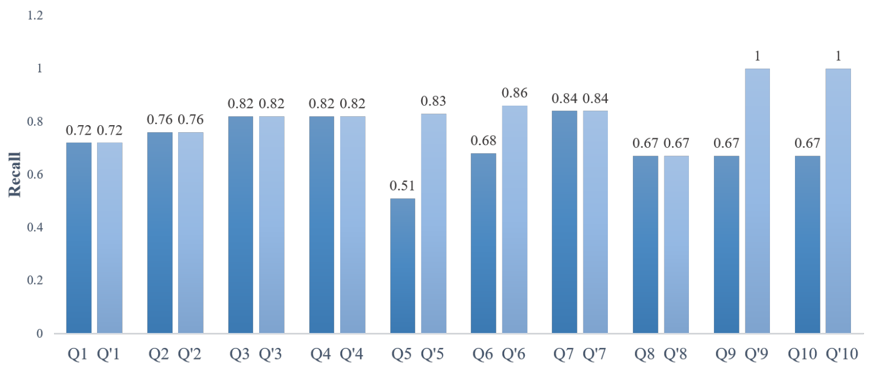


Figure 3.9: Recall rates

an approximate increase up to 15%. For instance, for $\mathcal{Q}_1$, the required VNF property $C\_A(audio\_codec)$ helps overcoming the ambiguity raised by $C\_A(aac)$ as an audio format. As for $\mathcal{Q}_7$ to $\mathcal{Q}_{10}$, considering additional VNF properties do not help discarding irrelevant VNFs. For instance, $Content(audio)$ in $\mathcal{Q}'_7$ does not permit to exclude the VNFs with $Operation(transcoding)$ on $Content(video)$ **and/or** $Content(audio)$ while relevant VNFs should implement $Operation(transcoding)$ on $Content(video)$ **to** $Content(audio)$.

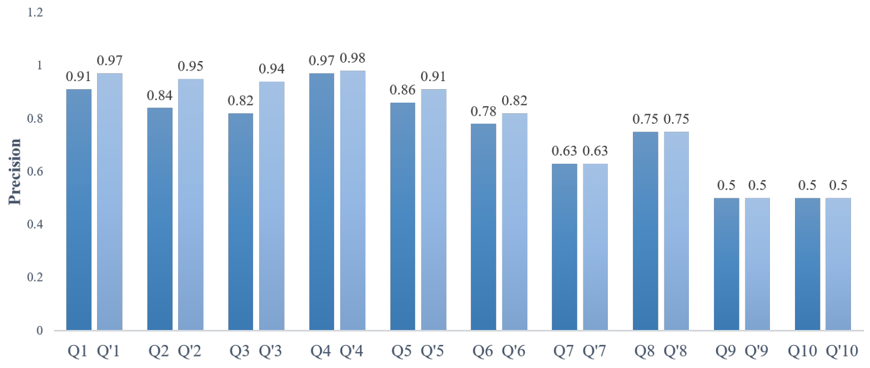


Figure 3.10: Precision rates

Since improving $\mathcal{R}$ typically reduces $\mathcal{P}$ and *vice-versa*, a decision threshold (δ) should be defined to determine a good trade-off between $\mathcal{R}$ and $\mathcal{P}$. Fig. 3.11 illustrates the way the discovered VNFs for each query (i.e., $\mathcal{Q} \cup \mathcal{Q}'$) are partitioned. We, then, compute $\mathcal{R}$ and $\mathcal{P}$ at each δ_j per query along with the recall and precision averages for $\mathcal{Q}$ and $\mathcal{Q}'$, respectively, at δ_j. Fig. 3.12 depicts the trade-off as the ratio between $\mathcal{R}$ and $\mathcal{P}$. The end-user fixes either $\mathcal{R}_j$ and $\mathcal{P}_j$, or both and obtains the corresponding δ_j. For instance, whether the end-user seeks for the most relevant VNFs, only (i.e., $\mathcal{P} = 1$), δ_1 should not exceed 10% of discovered VNFs. We can observe that when $\mathcal{R}$ is improved, $\mathcal{P}$ remains satisfactory (i.e., $\approx$ 0.75 and 0.8 for $\mathcal{Q}$ and $\mathcal{Q}'$, respectively).

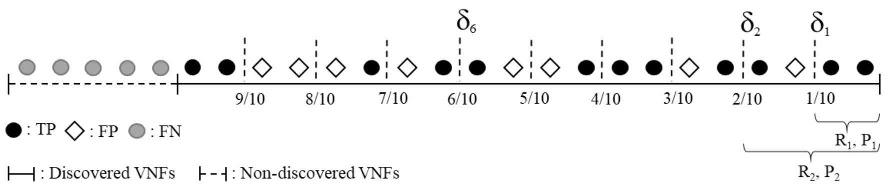


Figure 3.11: Decision threshold

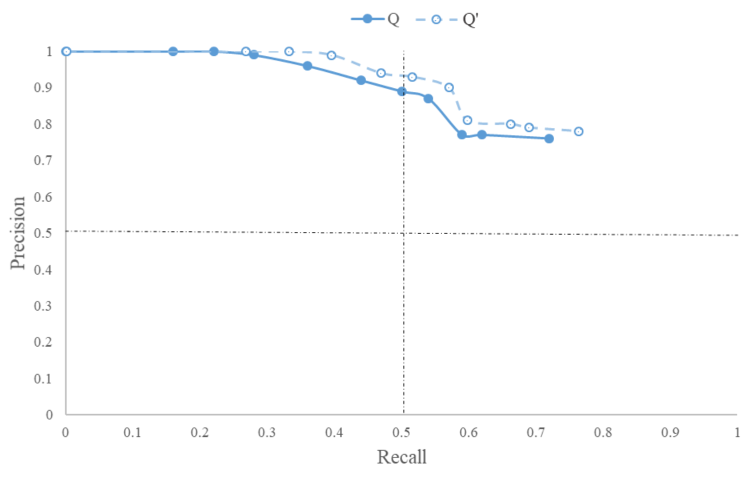


Figure 3.12: $\mathcal{R}/\mathcal{P}$ ratio

Fig. 3.13 shows F-measures for $\mathcal{Q}$ and $\mathcal{Q}'$. Note that larger F-measures indicate better overall results. Globally, the obtained F-measures vary over [0.57, 0.89] and [0.67, 0.89] for $\mathcal{Q}$ and $\mathcal{Q}'$, respectively. This indicates satisfactory results for the matchmaker from both provider and end-user perspectives. However, some improvements are still needed in terms of additional SWRL rules related to missing relationships between *Content* and *Operation* as highlighted for $\mathcal{Q}_9$ and $\mathcal{Q}_{10}$, for instance.

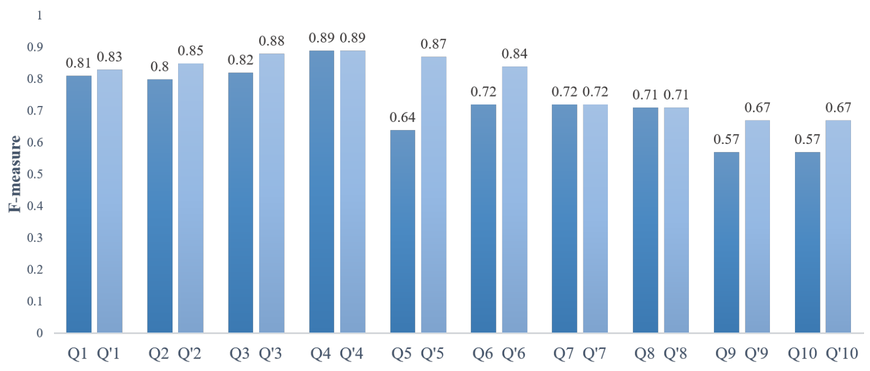


Figure 3.13: F-measures

Fig. 3.14 depicts the overhead ($\mathcal{RT}$) for $\mathcal{Q}$ and $\mathcal{Q}'$. We notice that the overhead varies according to the query. For instance, converter queries (i.e., $\mathcal{Q}_1$ to $\mathcal{Q}_8$) takes longer time than mixer queries ($\mathcal{Q}_9$ and $\mathcal{Q}_{10}$). This can be explained by complexity underlying converter's semantics (i.e., more concepts and instances along with semantic relations) compared to mixer. This complexity, thus, induces additional time for matching. However, we observe minor increase in $\mathcal{RT}$ for $\mathcal{Q}_i$. Compared to having better $\mathcal{R}$ and $\mathcal{P}$ obtained for $\mathcal{Q}'_i$, the overhead remains acceptable (i.e., $\approx$ 60ms).

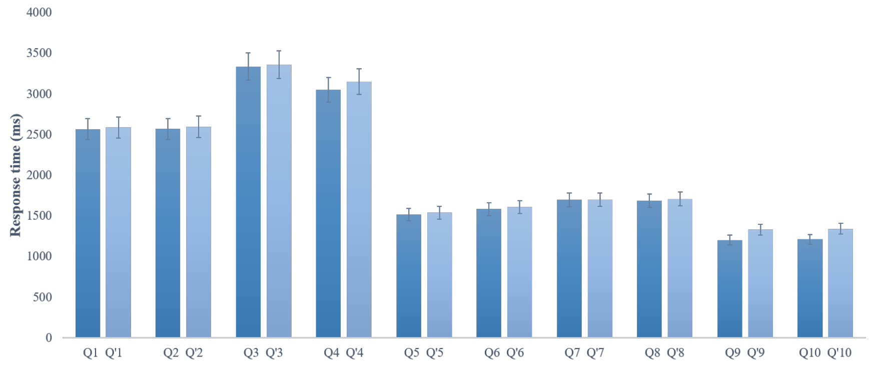


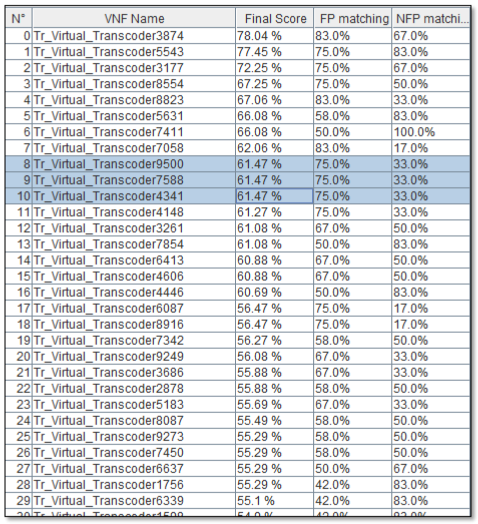
Figure 3.14: Overhead

3.4.3 Robustness evaluation

We also evaluate our matchmaker robustness in terms of consistency. For a given query, we examine and validate the obtained VNFs and their calculated matching scores considering a specific predefined set of published VNFs. For instance, the matching results for the VTranscoder are shown in Fig. 3.15. We observe that VNFs ranked from 8 to 10 have the same final score. This raking would mean that these VNFs are either identical or equivalent in terms of satisfying the query and its preferences. Let us parse the following capabilities:

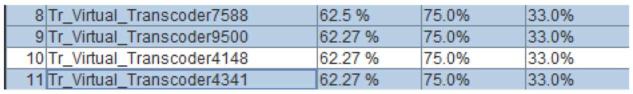
- $\{\mathsf{CAP}_{vt7588}{}^{\mathsf{F}}\}^H$={`O-Transmuxing`} & $\{\mathsf{CAP}_{vt7588}{}^{\mathsf{F}}\}^O$={`Std-MPEG_4`}
- $\{\mathsf{CAP}_{vt9500}{}^{\mathsf{F}}\}^H$={`O-Transcoding`} & $\{\mathsf{CAP}_{vt9500}{}^{\mathsf{F}}\}^O$={`Std-MPEG_2`}
- $\{\mathsf{CAP}_{vt4341}{}^{\mathsf{F}}\}^H$={`O-Transcoding`} & $\{\mathsf{CAP}_{vt4341}{}^{\mathsf{F}}\}^O$={`Std-MPEG_4`}

We note that `O-Transmuxing` and `O-Transcoding` are both functional requirements and are associated with the same preferences. This description explains the equivalence between these VNFs. To evaluate robustness, we modify the preference values for `O-Transcoding` to O- $\mathcal{CL}$. Based on the updated list of scores shown in Fig. 3.16, we notice that the final score changes. The list is refined to better satisfy the H- $\mathcal{CL}$'s requirements. Contrary to the previous list, we see that *Tr_Virtual_Transcoder7588* is ranked before *Tr_Virtual_Transcoder9500* in the updated list.



N°	VNF Name	Final Score	FP matching	NFP matchi...
0	Tr_Virtual_Transcoder3874	78.04 %	83.0%	67.0%
1	Tr_Virtual_Transcoder5543	77.45 %	75.0%	83.0%
2	Tr_Virtual_Transcoder3177	72.25 %	75.0%	67.0%
3	Tr_Virtual_Transcoder8554	67.25 %	75.0%	50.0%
4	Tr_Virtual_Transcoder8823	67.06 %	83.0%	33.0%
5	Tr_Virtual_Transcoder5631	66.08 %	58.0%	83.0%
6	Tr_Virtual_Transcoder7411	66.08 %	50.0%	100.0%
7	Tr_Virtual_Transcoder7058	62.06 %	83.0%	17.0%
8	Tr_Virtual_Transcoder9500	61.47 %	75.0%	33.0%
9	Tr_Virtual_Transcoder7588	61.47 %	75.0%	33.0%
10	Tr_Virtual_Transcoder4341	61.47 %	75.0%	33.0%
11	Tr_Virtual_Transcoder4148	61.27 %	75.0%	33.0%
12	Tr_Virtual_Transcoder3261	61.08 %	67.0%	50.0%
13	Tr_Virtual_Transcoder7854	61.08 %	50.0%	83.0%
14	Tr_Virtual_Transcoder6413	60.88 %	67.0%	50.0%
15	Tr_Virtual_Transcoder4606	60.88 %	67.0%	50.0%
16	Tr_Virtual_Transcoder4446	60.69 %	50.0%	83.0%
17	Tr_Virtual_Transcoder6087	56.47 %	75.0%	17.0%
18	Tr_Virtual_Transcoder8916	56.47 %	75.0%	17.0%
19	Tr_Virtual_Transcoder7342	56.27 %	58.0%	50.0%
20	Tr_Virtual_Transcoder9249	56.08 %	67.0%	33.0%
21	Tr_Virtual_Transcoder3686	55.88 %	67.0%	33.0%
22	Tr_Virtual_Transcoder2878	55.88 %	58.0%	50.0%
23	Tr_Virtual_Transcoder5183	55.69 %	67.0%	33.0%
24	Tr_Virtual_Transcoder8087	55.49 %	58.0%	50.0%
25	Tr_Virtual_Transcoder9273	55.29 %	58.0%	50.0%
26	Tr_Virtual_Transcoder7450	55.29 %	58.0%	50.0%
27	Tr_Virtual_Transcoder6637	55.29 %	50.0%	67.0%
28	Tr_Virtual_Transcoder1756	55.29 %	42.0%	83.0%
29	Tr_Virtual_Transcoder6339	55.1 %	42.0%	83.0%

Figure 3.15: The total ranked list of the discovered VNFs



8	Tr_Virtual_Transcoder7588	62.5 %	75.0%	33.0%
9	Tr_Virtual_Transcoder9500	62.27 %	75.0%	33.0%
10	Tr_Virtual_Transcoder4148	62.27 %	75.0%	33.0%
11	Tr_Virtual_Transcoder4341	62.27 %	75.0%	33.0%

Figure 3.16: Excerpt of a list of relevant VNFs following a requirement change

3.5 Summary

This chapter introduced a novel semantic-based methodology to describe, publish and discover Virtualized Network Functions (VNFs). It proposes a domain-independent VIrtualized networK functIoN ontoloGy (called VIKING) that enables VNFs developers and owners to sufficiently describe the capabilities and the requirements of their VNFs prior to publication. Prospective VNFs consumers use the same model to automate the discovery process, improve its precision and rely on federated repositories system if needed. Yet another contribution is supporting the VNFs non-functional properties and user preferences during the discovery, in addition to the classical functional properties.

As for validation, an extensive and real-life use case was designed and implemented. The considered use case explores VIKING in the context of Content Delivery Networks (CDN). A validating chest tool called Mastermyr Chest is developed. It consists of several tools that enable describing, publishing, discovering, and instantiating VNFs as middleboxes for CDN providers. The performed experiments on the discovery tool of Mastermyr Chest show that our semantic-based approach is accurate, precise, and with moderate delays.

Chapter 4

Agile and Dynamic Virtualized Network Functions Wiring in Network Services

4.1 Introduction

According to ETSI NFV, an NS is a composition of VNFs. These VNFs are arranged as a set of functions according to a Network Connectivity Topology (NCT) or without any connectivity specification between them [NFV 2017]. The NS concept enables developers to provision complex and sophisticated network functions, made up of elementary VNFs, at a higher level of abstraction. Furthermore, coupled with SDN, NFV provides higher flexibility in NS management. SDN enables flexible data forwarding among the VNFs. It places forwarding rules in network elements to transmit data through one of the possible paths prepared in advance by NS providers at design time [Kreutz 2015]. This capability ensures dynamic control of VNF chains by facilitating data forwarding across them. The forwarding elements route the traffic to follow an overlay path so that several VNFs are visited.

However, ETSI NFV does not provide a specific control that dictates the execution order and the composition logic of VNFs in a given NS. This reflects negatively on the NS provisioning methodology and procedures. Indeed, at design time, developers must entirely rely on SDN, along with its inherent service function chaining concept, to design and configure the routing rules between the involved VNFs. Consequently, they have to foresee and plan all possible and practical composition scenarios in the NS. Moreover, they should instantiate, deploy, configure, and manage all these potential paths at run-time, including the ones that will rarely be or never be used. On top of being costly, tedious, and time-consuming, this way of proceeding contradicts NFV spirit, specifically virtualization, promoting agility and cost-effectiveness in networking applications. Yet another limitation concerns the routing path models supported by SDN. In fact, SDN allows only sequential service function chains to be provisioned where VNFs are tied in a linear way. Therefore, more complex and sophisticated chains are still not supported in the NS.

This chapter proposes a novel approach that enables agile and dynamic NS provisioning as an extension to the ETSI NFV architectural framework. Broadly speaking, this research initiative introduces novel technical VNFs, called routing VNFs, with efficient re-configurable wiring capabilities for NSs. This initiative distinguishes domain-specific aspects from the connectivity ones in the NS definition. Domain-specific aspects are implemented with regular VNFs, while connectivity aspects are supported by the routing VNFs. By doing so, developers will be able to change and update the NS's composition logic at run-time, given specific criteria (e.g., message type, data size, QoS metrics).

The rest of the chapter is structured as follows. Section 4.2 introduces the proposed approach, as well as, the associated ETSI NFV-based architecture. Section 4.3 describes the proof-of-concept and the developed running prototype. Section 4.4 elaborates on the performed experiments. Finally, Section 4.5 discusses the lessons learned and concludes the paper.

4.2 A novel dynamic and re-configurable VNF wiring in NS

This Section details the motivating use case and describes the proposed approach to overcome the limitations. Then, we present our approach's high-level architecture.

4.2.1 Motivating use case

Platooning technology made significant advances to mitigate traffic congestion and reduces vehicle emissions. For a safe and green journey, a platoon consists of a *leader* vehicle that controls the speed and direction and *follower* vehicles that fit with the *leader*'s movement in terms of acceleration and braking. The *leader* also communicates with the networking infrastructure to retrieve details about signalization and road conditions. Platooning relies on vehicle-to-infrastructure (V2I) and vehicle-to-vehicle (V2V) communications [Boban 2018] [Cao 2016]. Notably, the fifth-generation (5G) of mobile telco networks turns out an excellent candidate to achieve this kind of communication. Indeed, the 5G network is expected to deliver ultra-low latency and ultra-high reliability to enable efficient, safe, and reliable car platooning. To reinforce mobility and allow V2X communications, ETSI-compliant 5G specifications recommend NFV, SDN, Multi-access Edge Computing (MEC), and Next-Generation Protocols (NGP) as key concepts [Raissi 2019]. On the one hand, they advocate for slicing the network into many logic and functional entities that could be virtualized and provisioned as VNFs to enable agile and cost-effective operating. On the other hand, SDN will handle the control plane to support the dynamic reshaping of the traffic.

Fig. 4.1 depicts the data plane and the signaling & control plane for car platooning. The former refers to V2V and V2I communications. The latter involves three Network

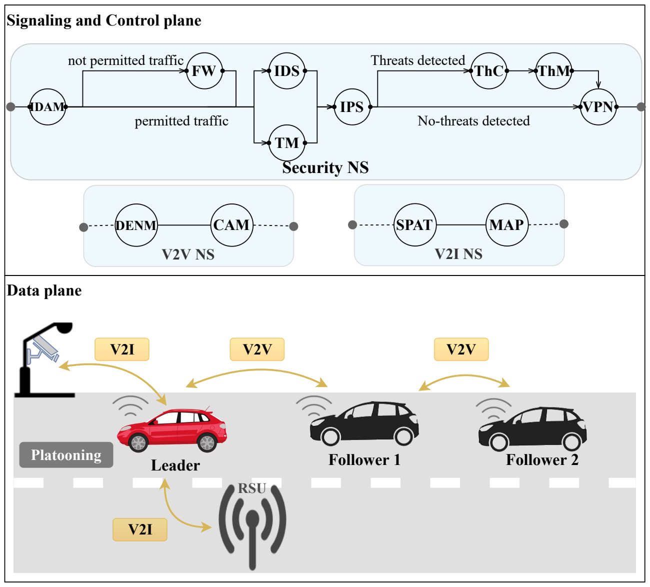


Figure 4.1: Network services and connectivity for vehicles platooning in 5G

Services (NS), namely, V2V, V2I, and security. The V2V NS ensures the communication services between the vehicles and encompasses VNFs such as Decentralized Environmental Notification Messaging (DENM) and Cooperative Awareness Messaging (CAM). The V2I NS ensures communications between vehicles and infrastructure, and it encompasses VNFs such as Signal Phase and Timing Information (SPAT), road sign information, and road topology information (MAP). Finally, the security NS secures the whole system, including the V2V and V2I communications. While the V2I and V2V NSs reflect sequential "execution" flows, the security NS provides several prospective "execution" flows depending on the case study and the data/request types to be secured. In practical terms, the traffic arrives first to the VNF named IDentity and Access Management (IDAM) that is responsible for verifying the vehicle's identity and distinguishes if the source vehicle is trustful (e.g., already part of the platoon) or not. The untrusted traffic must necessarily pass by the FireWall (FW) VNF before traveling through the rest of the NS. The FW VNF filters the network packets based on specific criteria (i.e., source, destination addresses, ports) and blocks the suspicious traffic. Next, the traffic simultaneously goes through the Intrusion Detection System (IDS) VNF and the Traffic Monitor (TM) VNF. The IDS VNF compares the received packets to a knowledge database to identify potential threats. In parallel, the TM VNF parses the network packets to detect any threats and/or malicious content. The execution results are then aggregated by the Intrusion Prevention System (IPS) VNF. When no threats are detected, the traffic is forwarded to the Virtual

Private Network (VPN) VNF that makes safe tunneling with the network destination. Otherwise, the traffic should first travel through a specific network branch that consists of Threats Classifier (ThC) VNF and Threats Mitigation (ThM) VNF before reaching the VPN VNF. The ThC VNF classifies the intrusion within pre-defined classes/families of threats while the ThM VNF reduces the extent of the intrusion by either isolating or containing it until the problem is fixed.

Following the ETSI specification, provisioning the security NS necessarily implies deploying all the involved VNFs and configure the connectivity as depicted in the associated execution workflow. Therefore, NS developers should anticipate and cover all the execution alternatives. Specifically, this means that developers need to design, deploy and configure all the possible NFP and end up with heavy, complex, and static NCT handle during every single phase of the network entities' life-cycle. Wiring constituent VNFs could be easily done for simple NS but can turn out tedious and costly tasks for complex NSs with a considerable number of VNFs and sophisticated workflows. Furthermore, depending on the traffic, some of the pre-deployed/pre-configured wires will be rarely or never used. To address all these limitations, we advocate for deploying a single execution alternative that could dynamically be reconfigured and adjusted, depending on the need (e.g., request type, data type), during run-time.

4.2.2 Requirements and foundations

As part of our study of the ETSI NFV framework, we identified a set of requirements that represent the necessary actions/characteristics for NS provisioning (i.e., design, deploy, execute, and manage). In what follows, we describe the associated requirement to each phase of the NS life-cycle when considering MANO:

1. **NS description phase:** When designing the VNFFG (introduced in Chapter 2), all the necessary NFPs among a given number of VNFs need to be prepared in advance for a prospective operation. Assuming a complete/comprehensive knowledge is unfortunately unrealistic.

2. **NS deployment phase:** All the necessary elements, such as the virtual links that make up the previously mentioned NFPs, need to be established appropriately. This includes the virtual links that eventually will never be used and, thus, leads to excessive resource consumption.

3. **NS execution and management phases:** Not all the management operations are possible when maintaining the service delivery (e.g., adding new VNFs to the NS descriptor at run-time). Only the pre-defined paths that connect VNFs in the NS's descriptor can be subject to updates during these phases. This would overlook newly added paths.

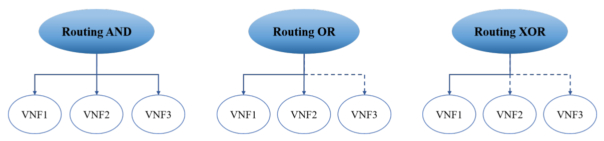


Figure 4.2: Network traffic representation per *routing* logic

To address the aformentioned requirements/limitations, we advocate for flexible and dynamic wiring among VNFs in a given NS using the separation-of-concerns design principle to separate domain-specific aspects from connectivity aspects. We introduce three novel concepts, namely, *routing-AND*, *routing-OR*, and *routing-XOR*. Each concept supports wiring capabilities that enable the NS designer to define a certain partial order among the domain-specific VNFs (i.e., all, some, or single). These novel wiring capabilities allow us to obtain non-linear (i.e., tunable) NCTs. The prospective traffic when crossing VNFs triggers a particular *routing* logic as represented in Fig. 4.2:

- *routing-AND* triggers **all** the "execution" branches by forwarding the entry traffic to all the outgoing VNFs.
- *routing-OR* triggers **some** of the "execution" branches based on traffic conditions, either conveyed in the traffic itself or reported by NS consumer, by forwarding the entry traffic to selected outgoing VNFs.
- *routing-XOR* triggers a **single** "execution" branch based on traffic conditions, either conveyed in the traffic itself or reported by NS consumer, by forwarding the entry traffic to the selected outgoing VNF.

Algorithm 3 implements the *routing* logic as follows. This algorithm first takes *routing* VNF id, NS descriptor, triggering condition, and traffic as inputs. Then, it parses the NS descriptor to retrieve all NFPs ($\{\text{NFP}_k\}_{k=1,r}$) associated with the *routing* VNF id. Afterward, $\{\text{NFP}_k\}$ are split into 2 disjoint sets namely, $\{\text{NFP}_i\}_{i=1,r'}$ and $\{\text{NFP}_j\}_{j=r'+1,r}$ corresponding to desired wiring capabilities to enable when the triggering condition is true/false (lines 3-6 & 8-11, respectively). It is worth noticing that r' would be equal to r in case of *routing-AND*. Finally, the algorithm duplicates traffic as many as either $\{\text{NFP}_i\}$ or $\{\text{NFP}_j\}$. This results in multiple traffic to forward through these NFPs.

In NFV setting, we advocate for specific types of VNFs that would implement the afore-mentioned *routing* concepts. Fig. 4.3 depicts a conceptualized view of VNF types. VNFs are specialized into *operative* for domain-specific and *routing* for connectivity. *Routing* VNFs could be implemented as either *swing* or *proxy* VNFs, both discussed in Section 4.3. Provisioning NSs endowed with *routing* VNFs would make

Algorithm 3 Routing logic

```
Require: VNF id, NS descriptor, condition, traffic
 1: $\{\text{NFP}_k\}$ = parse(VNF id, NS descriptor)
 2: if check(condition) then
 3:     if $\{\text{NFP}_i\} \equiv \{\text{NFP}_k\} \vee \{\text{NFP}_i\} \subset \{\text{NFP}_k\}$ then
 4:         for all $\text{NFP}_i$ do
 5:             forward(duplicate(traffic), $\text{NFP}_i$)
 6:         end for
 7:     else if $|\{\text{NFP}_i\}| = 1$ then
 8:         forward(traffic, $\text{NFP}_i$)
 9:     end if
10: else if $|\{\text{NFP}_j\}| = 1$ then
11:     forward(traffic, $\text{NFP}_j$)
12: else
13:     for all $\text{NFP}_j$ do
14:         forward(duplicate(traffic), $\text{NFP}_j$)
15:     end for
16: end if
```

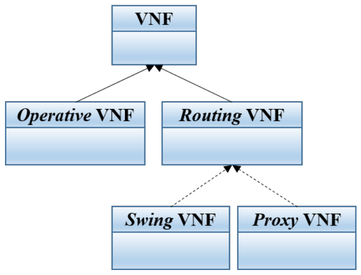


Figure 4.3: Considered VNF types

their associated NCT generic and customizable for modeling all the possible connections in a given VNFFG. Thus, the NCT can be modified at run-time following user requirements, business policies, and/or network context. In the following section, we discuss the way this NCT is defined, deployed, and managed at run-time.

4.2.3 High-level architecture

Fig. 4.4 depicts an overview of the system architecture. This architecture aims to address the requirements discussed in Section 4.2.2. It ensures the proper end-to-end provisioning of NS made up of *operative* and *routing* VNFs. The proposed design consists of 3 main layers: `NS design`, `NS deployment`, and `NS management`. Each layer implements a given phase of the NS life-cycle:

- `NS design phase`: The NS Modeler enables designing a given NS with a set of graphical elements to represent the NS's execution workflow, including *operative* VNFs, *routing* VNFs, as well as, the connections between them (Fig. 4.4, 1.1). The output is the NS Abstract Descriptor (NS-AD) that will be forwarded to the NS Builder (1.2). The latter parses the files and selects, from the VNFs Repository, the needed domain-specific VNFs that are bound to *operative* VNFs (1.3). Then, it produces the NS Concrete Descriptor (NS-CD) that refers to the concrete deployment details (e.g., VNF images), associated NFPs for a given VNFFG along with the constituent VNFs, and policies related to traffic management such as the classifiers. Finally, the NS Builder forwards the NS-CD to the NS Deployer (1.4) while saving a copy of it in the VNF Descriptors Repository (1.5).

- `NS deployment phase`: The NS deployment relies on the ETSI MANO. It uses its three components (i.e., NFVO, VNFM, and VIM) described in Section **??**. Considering a given NS-CD, NFVO receives queries from the NS Deployer (2.1). Then, NFVO requests the VIM for necessary resources to instantiate both *operative* and *routing* VNFs (2.2). VIM communicates with NFVI to allocate and instantiate the requested resources (2.3). Moreover, the NFVO also asks VIM for the required networking resources to create and maintain virtual links (VLs) for all associated NFPs. VNFM will proceed with the concrete deployment of the VNFs over the NFVI resources (2.4). VNFM also interacts with NFVO to manage them.

- `NS management phase`: The architectural components for this phase ensure the service delivery and support the necessary on-the-fly updates. Indeed, at run-time, the NS designer or NS consumer may request functional changes such as adding/removing VNFs to a running NS, or simply modifying the wiring within a running NS (3.1). The NS Updater forwards the NS-AD associated with these requested changes to the NS Builder (3.2). The latter retrieves the corresponding NS-AD from the VNF Descriptors Repository (3.3) and revises it according to the necessary modifications. The revised NS-AD is then forwarded to the NS Adapter (3.4). The NS Adapter applies the new changes on the NS (e.g., add/delete NFPs) before notifying the NS Deployer to communicate with the ETSI MANO and reflect these changes on the running NS (3.5).

4.3 Dyvine architecture and implementation

This Section presents the developed Proof-of-Concept (PoC), as well as, the running prototype that simulates the platooning use case.

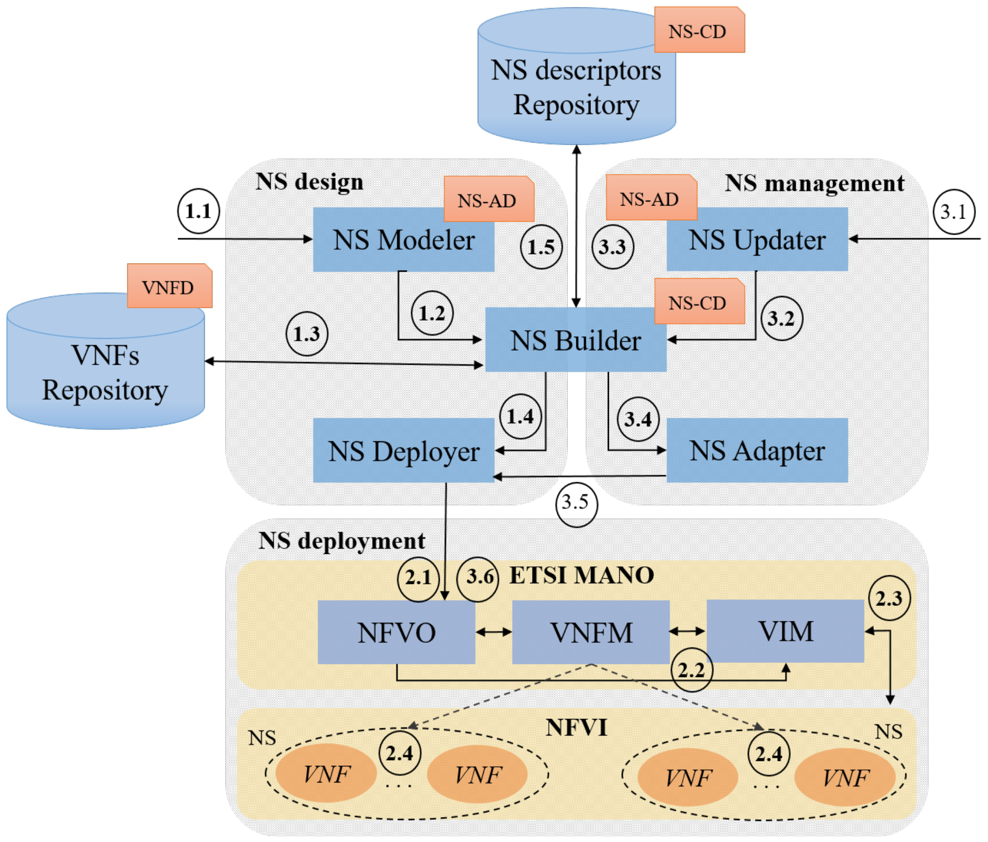


Figure 4.4: System architecture overview

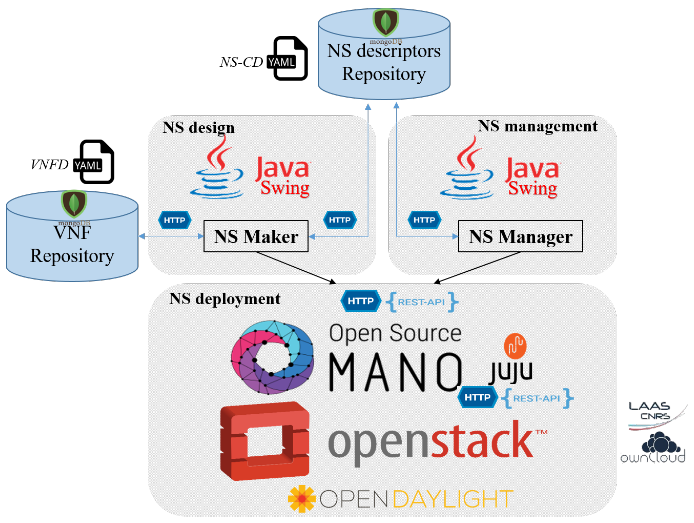


Figure 4.5: DYVINE architecture

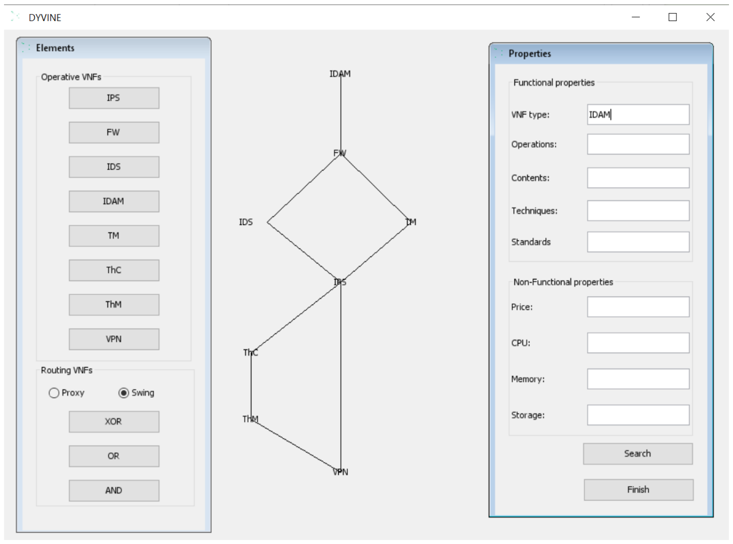
4.3.1 Proof-of-Concept

To prove the feasibility of our approach, we developed a PoC called DYVINE (stands for DYnamic forwarding graphs for VIrtual NEtwork functions). DYVINE's source code is available on a GitHub repository[1]. Fig. 4.5 depicts DYVINE's technical architecture that implements the system architecture discussed in Section 4.2.3. Basically, DYVINE tool enables NS provisioning while supporting the three main phases (i.e., design, deploy, and manage) of their life-cycle. DYVINE is coded with JAVA and incorporates two main modules, namely, NS Maker and NS Manager.

On the one hand, the NS Maker implements both NS Modeler and NS Builder at the NS Design layer. Thanks to the JAVA Swing framework, this module provides the NS developers with graphical interfaces that enable setting up NCT by drawing a given NS structure composed of *operative* VNFs and *routing* VNFs. Fig. 4.6 shows a snapshot of DYVINE with the security NS's perspective representation for our case study. The generated descriptors are implemented with YAML and stored in MongoDB, a NoSQL database.

On the other hand, the NS Manager implements the architectural components at the NS Management layer namely, the NS Updater and the NS Adapter. This module displays the graphical representation of the NS to be updated from the NS descriptors repository and applies changes requested by the NS developer, to the corresponding NCT structure.

[1]`https://github.com/NourNouar/DYVINE`



Routing VNFs: Proxy [✓ Swing]

VNF type: IDAM

Figure 4.6: A snapshot of DYVINE - the security NS design

In addition to these modules, we also did integrate OSM[2], an open-source MANO implementation, to DYVINE to handle the concrete NS deployment, orchestration, and management. Specifically, we used the VNF Configuration Adapter (VCA) module of OSM to perform Day-2 configurations and support the dynamic (at run-time) reconfiguration of the *routing* VNFs. In line with the foundations and the requirements discussed in Section 4.2.2, the VCA enables:

- Replacing the existing NFP identifier with a new one for *routing-XOR* VNFs,
- Updating a list of NFP identifiers for *routing-AND* VNFs, and
- Add/deleting one or several NFP identifiers for *routing-OR* VNFs.

As for the infrastructure, we use both OpenStack[3] platform to implement our NFVI and OpenDaylight[4] with Open vSwitch (OVS) [Pfaff 2015] to support the dynamic reshaping of the network control plane. OSM, OpenStack and, OpenDaylight are deployed over our laboratory resources. More specifically, the NS design and NS management layers along with OSM were executed over a single laptop featured with Intel(R) Core(TM) i7-8650U CPU 1.90GHz under Ubuntu 18.04 LST. Regarding Openstack and Opendaylight, both were hosted on our Lab's private cloud. We, also, use

[2]https://osm.etsi.org/
[3]https://www.openstack.org/
[4]https://www.opendaylight.org/

OVS's version 2.13.1 to support the Network Service Header (NSH) encapsulation with VXLAN tunnels.

4.3.2 Running prototype

Our prototype simulates the use case reported in Section 4.2.1. *Leader* vehicle and *attackers* were implemented as data streaming providers while the 4 *follower* vehicles play the role of consumers, all deployed over OpenStack VMs. To enable V2V communications, we create 1 virtual network mapped onto our Lab's cloud network. Any network service including VNFs deployed over OpenStack VMs is connected to this virtual network through connection points. Note that any VM has the following features: 1 CPU, 1 GB memory, and 10 GB storage.

When it comes to the security NS implementation, we did explore 3 different options. While Option 1 relies on *routing* VNFs implemented through the so-called *swing* VNFs, Option 2 considers *proxy* VNFs. As a baseline, Option 3 consists of linear service function chaining and refers to a fully SDN-based approach for VNF wiring within the NS. In the following, we refer to Option 3 as *fully-SDN* option. The VNFs source code associated to the security NS implementation are available on GitHub repository[5]. A live demo of the running prototype is available through the following link: `https://youtu.be/7emJ6tnpvLE`.

On one side, *swing* VNFs support a static and particular logic. Each *swing* VNF implements a given *routing* gateway (i.e., *routing-XOR*, *routing-AND*, or *routing-OR*). With *swing* VNFs, depending on the network traffic conditions, (a) particular "execution" branch(es) will be triggered in the NS execution path. Specifically, *operative* VNFs coupled with their respective intrusive *swing* VNF are both encapsulated in the same VM. These particular VNFs are represented with a double ring in Fig. 4.7a.

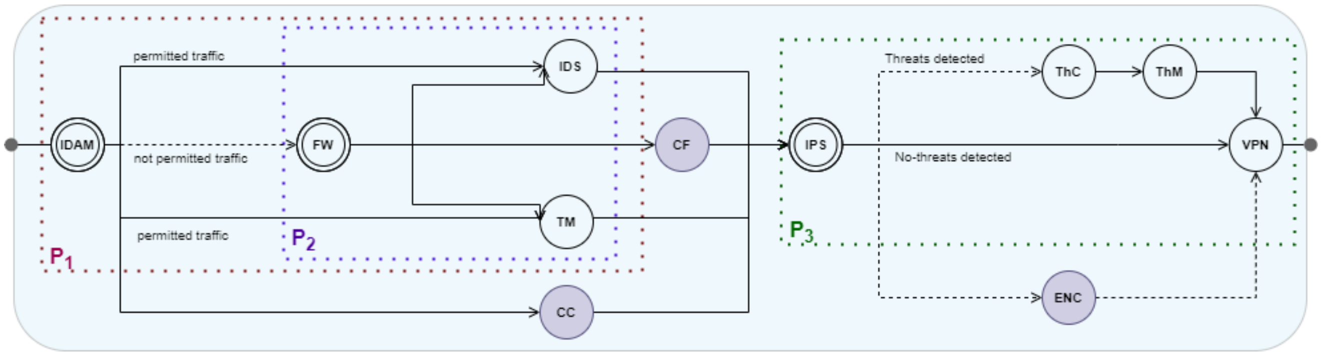
On the other side, *proxy* VNFs are dynamic and implement abstract wiring mechanisms that could be reconfigured at will during run-time. Specifically, *proxy* VNFs encompass the 3 *swing* VNF types' logic and need to be configured to select the one to be used at deployment/running time (Fig. 4.7b). Packet exchanges between *operative* VNFs and *proxy* VNFs refer to Context Header 4 field's NSH Metadata (Type 1) [Quinn 2018]. This offers higher flexibility to NS at run-time.

Finally, for the need of benchmarking, we also implemented security NS's wiring with *fully-SDN* option (Fig.4.7c). This solution was developed using OpenDayLight.

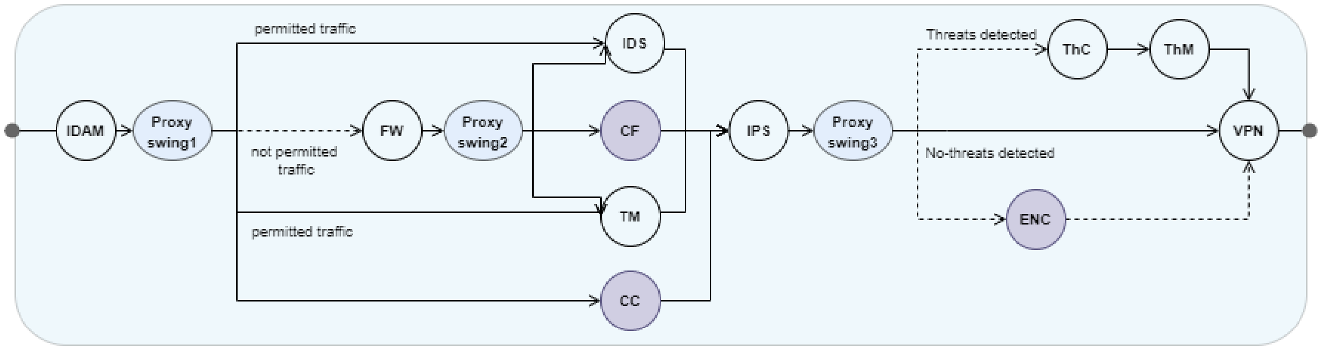
4.4 Validation and evaluation experiments

This Section describes the testbed settings and discusses the 2 performed sets of experiments. The first set aims to observe and validate the agile and adaptive operation

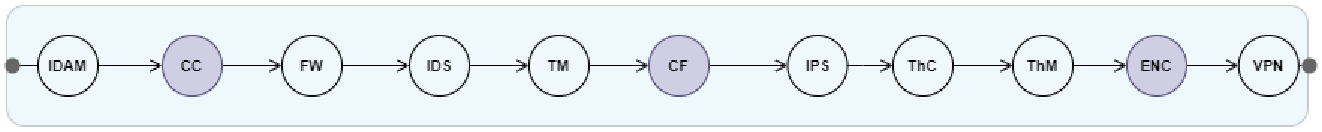
[5]`https://github.com/ieeecloudAnonym/Routing-VNFs`



(a) Option 1 - Intrusive *swing* VNFs



(b) Option 2 - Non-intrusive *proxy* VNFs



(c) Option 3 - Service function chaining (*fully-SDN*)

Figure 4.7: The 3 implemented options for wiring VNFs within the security NS in the car platoon case study

of the newly introduced VNFs, while the second set focus on cost and performance evaluation of the 3 implemented options presented in Section 4.3.

4.4.1 Testbed settings

To define our experimental scenarios, we rely on Segata et al.'s paper [Segata 2013]. This work discusses communication strategies in platooning. It reports realistic communication rates between the *leader* and any *follower*, as well as, between successive *followers* (i.e., 10 packets per second) with packet size varying over [200B, 1200B].

Furthermore, the considered experimental scenarios also refer to Denial of Service (DoS) attacks against *followers*. Any *attacker* floods the target *follower* with excessive data packets over time (i.e., 20 packets per second). The number of *attackers* varies over [1, 10], depending on the conducted experiment.

4.4.2 Performance results and analysis

The first evaluation experiment aims to estimate the average time related to one-way latency $\mathcal{L}$ (i.e., from the first bit sent to the last bit received) through a given NS. To measure $\mathcal{L}$, we first simulate the network traffic as a fixed number of packets (i.e., 100) with different sizes varying between 200B and 1200B including a certain rate of attacks varying from 10% to 100%. This traffic passes through the 3 options implementing the security NS. To avoid biases, we repeat the experiment 10 times and then, calculate $\mathcal{L}$. Fig. 4.8 represents the curves associated with $\mathcal{L}$(Option 1), $\mathcal{L}$(Option 2) and $\mathcal{L}$(*fully-SDN*) when increasing the rate of malicious packets. Overall, we notice that $\mathcal{L}$(Option 1) and $\mathcal{L}$(Option 2) achieve better performance than $\mathcal{L}$(*fully-SDN*) by $\approx$ 78% and $\approx$ 27%, respectively. When there are no attacks, $\mathcal{L}$(Option 1), $\mathcal{L}$(Option 2) and $\mathcal{L}$(*fully-SDN*) have $\approx$ 6.8ms, 8.2ms, 14ms, respectively. The difference is due to the fact that the packets need to go through various VNFs from one option to another. In fact, with regard to Option 1, the additional *proxy* VNFs in Option 2 increases latency. However, the reader should note that the latency is still less substantial for these 2 options with regard to *fully-SDN* option. Indeed, in Options 1& 2, the packets go through the specific and selected path (i.e., limited number of VNFs), while in *fully-SDN* option, packets necessarily need to go through all VNFs in the NS, one after the other. When increasing the rate of attacks, we notice that $\mathcal{L}$(Option 1) increases slightly to reach $\approx$ 9ms whereas $\mathcal{L}$(Option 2) rises significantly to reach $\approx$ 13ms. This is due to additional processing for attack packets by intrusive *swing* VNFs and *swing* proxies, where, the latter dedicates more time to process wiring information communicated by *operative* VNF in packet meta-data. Also, we notice that $\mathcal{L}$(*fully-SDN*) remains constant because *fully-SDN* option deals with both normal and attack packets in the same way.

The second evaluation experiment aims at demonstrating our approach's efficiency in terms of effective service continuity during on-the-fly network service up-

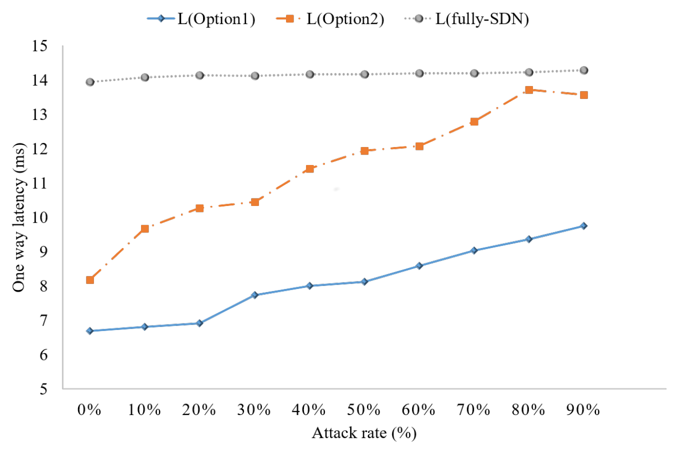


Figure 4.8: One-way latency while increasing the rate of Attacks

date. Due to some changes in user requirements at run-time, existing/new VNFs can be removed/added from/to the network service. Let us consider any vehicle that requires additional defenses against tampering attack (i.e., false safety messages), malware (i.e., unreadable data format), and cipher attack (i.e., known plaintext) [Sumra 2014]. As counter-measures, the security NS should be updated with 3 new VNFs namely, Coordination Control (CC), Content Filter (CF), and Encryption (Enc), each against one of the afore-mentioned attacks, as per Fig. **??**. To enable this update, the security NS's options 1 & 2 proceed as described in Section 4.3. Regarding the security NS's *fully-SDN* option, any NS update requires OSM to re-deploy this NS and SDN to reconnect all the VNFs. To quantify the Mean Time-To-Operation (MTTO), we measure deployment time for each new VNF in case of Options 1&2 and redeployment time for all VNFs in case of *fully-SDN* option. During this measurement, we vary the number of new VNFs to be added to the security NS at once. In Fig. 4.9, we observe that the deployment/redeployment time over the 3 Options rises in a linear way. We, also, note that MTTO_1 and MTTO_2 are relatively equal. The ratio between $\mathsf{MTTO}_{i=1,2}$ and MTTO_3 approximates 72% (i.e., 6.5 times more). This can be explained by the fact that only the newly added VNFs are deployed while *wiring* VNFs operate to establish connectivity with these VNFs.

We, also, deem appropriate to assess the time required to create a new NFP for the different options. To this end, we compute the cumulative moving average (CMA) over 30 iterations per update when the new aforementioned VNFs are incrementally added to the NS. In Fig. 4.10, we can observe that CMA weakly increases. We, also, compare CMA(Option 1) and CMA(Option 2) with CMA(*fully-SDN*) using ratio. The first approximates 7% while the second is close to 16%. Therefore, Option 1 gives acceptable CMA contrarily to Option 2.

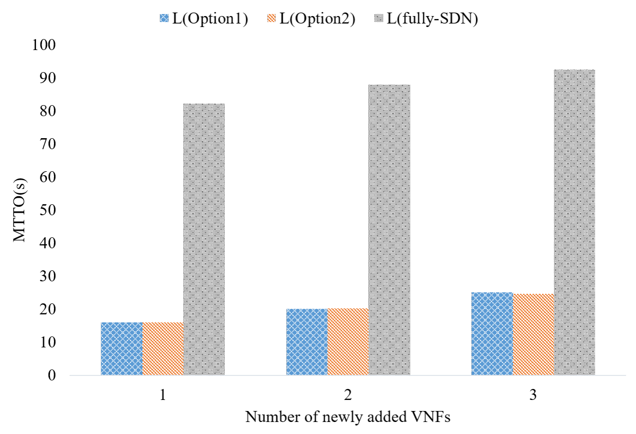


Figure 4.9: Mean Time-To-Operation

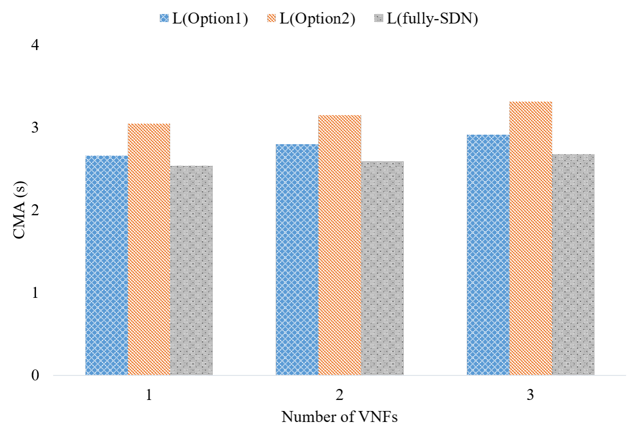


Figure 4.10: Cumulative moving average

4.4.3 Effectiveness validation

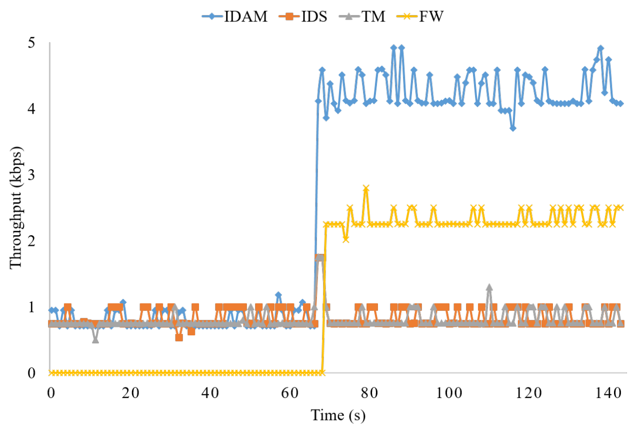
The first experiments aim at demonstrating our approach's effectiveness to support flexible NSs with wiring capabilities. To this end, we measure throughput over the security NS in case of attacks issued by a single *attacker*. For illustration purposes, we focus on Option 1, only. To perform fine-grained analysis, we deem necessary to probe throughput over NS's 3 partitions (dashed lines in Fig. 4.7a) namely, P_1, P_2, and P_3, each delimited with intrusive *swing* VNFs. Since P_i contains a set of VNFs, we measure distinct throughput over each VNF. We define 2 time-windows ([0, t_1[& [t_1, t_2]) referring to pre- and post-attack, respectively. Note that t_1 refers to start-time of attacks while t_2 indicates the end of both data streaming.

In Fig. 4.11a, during [0, t_1[, we observe that P_1's VNFs have the same throughput except for FW where the traffic is null. During [t_1, t_2], we notice that throughput over IDAM and FW increases while it remains constant over IDS and TM. This demonstrates that IDAM enriched with *swing-OR* capability works properly by redirecting the *attacker*'s traffic to FW, only. In Fig.4.11b, we observe null throughput for P_2's VNFs during pre-attack whereas it increases significantly during post-attack. This reveals that FW enriched with *swing-AND* capability works properly by passing on *attacker*'s traffic to both IDS and TM. Finally, Fig.4.11c depicts throughput over P_3's VNFs where it is similar for both IPS and VPN and null for ThC during [0, t_1[. We observe during post-attack that throughput over IPS and ThC increases significantly while it remains constant for VPN. This denotes that IPS enriched with *swing-XOR* capability fulfills its duty by sending *attacker*'s traffic to ThC. Finally, the 3 experiments highlight that the necessary switching time for the traffic route between pre- and post-attacks is instantaneous, confirming the agility and the dynamicity of our approach.

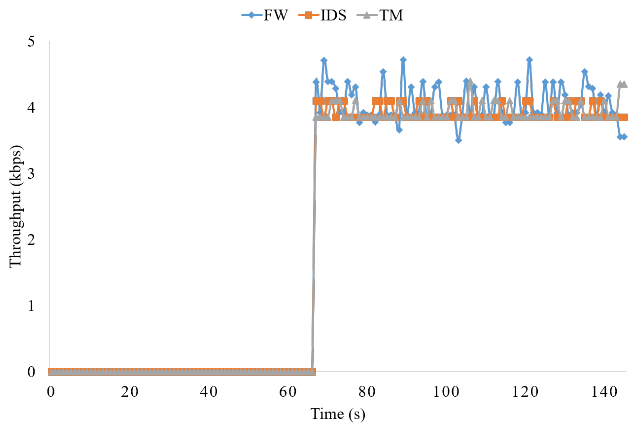
4.4.4 Discussion

The performed experiments led to several important lessons learned.

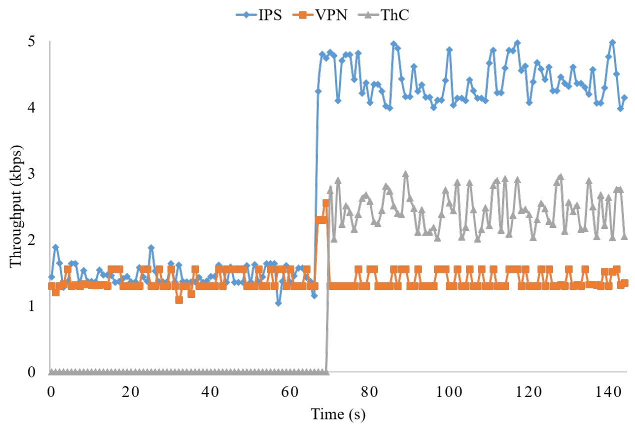
The first lesson learned is that *routing* VNFs bring added value to NS provisioning procedures. Compared to the service function chain in SDN, the use of *routing* VNFs considerably reduces complexity in NFV connectivity management. Unlike SDN procedures, where developers need to handle all the network routes, they will be henceforth able to focus on particular branches of the same network. This induces a decrease in the network's operating cost. Generally speaking, the *routing* VNF concept fits well with virtualization principles. Actually, it brings cost-effectiveness and agility to the network. Also, it enables the dynamic update of the NS's composition logic given specific criteria (e.g., message type, data size, QoS metrics). This capability seems to be appropriate to tackle the emerging needs of the next-generation service providers (e.g., mobile communication systems) with challenging constraints on the network business model ecosystem. Examples of constraints are the mobil-



(a) P_1



(b) P_2



(c) P_3

Figure 4.11: Throughput variation during pre- and post-attack (Option 1)

ity support (e.g., autonomous vehicles [Raissi 2019]), dynamic QoS management (e.g., adaptive and multicast streaming [Araniti 2017]), and the tactile Internet (e.g., haptic communications [Antonakoglou 2018]), to cite just a few.

The second lesson learned refers to the way the *routing* VNFs can be implemented and the subsequent trade-off between flexibility and performance. At run-time, *swing* VNFs (Option 1) show better performance in terms of end-to-end latency and the necessary update time with regard to *proxy* VNFs (Option 2). Nevertheless, *proxy* VNFs are easier to adopt and integrate during design time. The advantage of Option 2 is to take over the routing logic management and to not require any adaptation efforts on the network provider side. Regarding *swing* VNFs, their use would impose to integrate the routing logic (i.e., *routing-XOR*, *routing-OR*, *routing-OR*) within the *operative* VNFs. This adds more complexity to NS developers that will be in charge of injecting the routing logic within *operative* VNFs. Furthermore, this practice is not always possible because NS developers do not necessarily own all the VNFs that compose the NS. By contrast, *proxy* VNFs are compliant with the separation of concerns principle since domain-specific and connectivity aspects are decoupled in the NS. Also, this alleviates the NS design burden by delegating the *proxy* VNFs operation to the network provider. Although in line with the virtualized ecosystems' business model, this implementation option imposes upstream integration of this specific kind of VNFs as part of the technical provider's capabilities. To conclude with this, we believe that both alternatives are useful and it would be up to the NS developers to decide about the most suitable option for their specific needs (e.g., considered use case, supported data format, and SLA).

4.5 Summary

According to ETSI, NFV's objective is to transform the way that network operators architect networks by evolving standard IT virtualization. This research work fits within this vision, taking advantage of the virtualization benefits such as agility, dynamicity, and cost-effectiveness. We propose a novel way to provision and operate Network Services (NSs) in the NFV setting. Our approach supports flexible and dynamic wiring among VNFs in a given NS. Basically, we introduce 2 novel concepts namely, *operative* VNFs and *routing* VNFs with re-configurable wiring capabilities. *Operative* VNFs implement the regular (functional) network functions while *routing* VNFs implement the control logic among *operative* VNFs like concurrency (i.e., *routing-AND*), exclusivity (i.e., *routing-XOR*), or inclusivity (i.e., *routing-OR*). By doing so, we address the major limitation related to the non-support of dynamic NFPs changes at run-time and considerably reduce the provisioning time and effort of both NS provisioning and operation. Also, the proposed architecture is in line with the ETSI NFV framework.

Chapter 5

Resource Optimization for Network Service Deployment

5.1 Introduction

As reported in Chapter 2, NFV-MANO allows multiple network services to be deployed over the same set of NFVIs, each managed by a specific VIM and spread over mono-/multi-domains. From a business perspective, NFVI providers associate fixed or variable operating prices with computing resources required to execute VNFs. From a technical perspective, NFVO will decide about (i) VNF placement per network service across the different NVFIs, and (ii) data traffic routing across VNFs that compose the network services.

In the previous Chapter 4, we advocated for deploying the network services as single execution alternatives that could dynamically be reconfigured and adjusted, depending on the need (e.g., request type, data type), during run-time. Consequently, the resources allocation requests issued for this kind of network service should vary over time, i.e., new/existing VNFs along with the corresponding virtual links are added/removed. Understandably, a basic alternative to support such a procedure could be one-to-one mapping between pre-defined provisioning requests and available resources over the different NFVIs. However, this alternative remains static and fails in updating the placement decisions following any prospective changes on the deployment/placement requests. For instance, changes would reflect SLAs updates such as billing/metering variation or technical breakdown.

In this work, we introduce a lazy resources allocation solution to Network Service Embedding (NSE) problem. This solution optimizes both VNF placement for each network service along with traffic routing across VNFs while satisfying constraints like location and latency. This optimization aims at minimizing the network service deployment cost in terms of allocated resources and when necessary. Prior to provisioning, the NFVO, with the help of VIMs, will scrutinize the hosting capacity for each infrastructure and bandwidth for each physical link over time.

The rest of this chapter is organized as follows. Section 5.2 describes the resources model along with NS embedding requests. Sections 5.3 and 5.4 present the optimal and near-optimal problem-solving approaches to the NSE problem, respectively. Section 5.5 defines the dynamic pricing scheme used by the NFVI providers for resource allocation. Section 5.6 compares and evaluates the both proposed approaches using different performance metrics. Finally, Section 5.7 concludes this Chapter.

5.2 Resources model and NS embedding requests

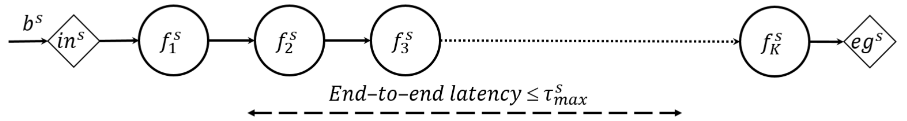
We consider a physical network $\mathcal{N}$=($\mathcal{V}$,$\mathcal{E}$) that encompasses N interconnected NFV infrastructures ($\mathcal{N}_i \subset \mathcal{N}$). Each infrastructure ($\mathcal{N}_i$) can be operated by a distinct infrastructure provider (i). Further, all nodes may belong to either the same infrastructure or different interconnected ones located in different domains ($\mathcal{D}$) like Radio Area Network (RAN), core network (cloud), or edge network (fog). In this setting, we distinguish 2 node types namely, *forwarding* and *function.* While the *forwarding* nodes (e.g., switches and routers) transmit data traffic, the *function* nodes (e.g., servers and data centers) provide compute resources that are processing (CPU), memory (RAM), and storage (HD), as well as, any specialized resources (e.g., VMs and containers) bound to nodes.

Let $\chi \subset \mathcal{V}$ be the set of all *function* nodes in $\mathcal{N}$ and $\mathcal{R}$ be the set of resource types namely, CPU, RAM and HD. Each $v \in \chi$ provides the amount of available resources per type (r), represented as a vector $\vec{p}_v = (p_v^r)_{r\in\mathcal{R}}$. It is worth noticing that the *forwarding* nodes ensure traffic routing between *function* nodes and therefore, do not support any significant computing capabilities. From business perspective, infrastructure providers define a set of fixed or variable operating prices for resources, per type r, per unit, and per node v ($\vec{\psi}_v$ =$(\psi_v^r)_{r\in\mathcal{R}}$). We, also, characterize any communication link $e \in \mathcal{E}$ with bandwidth (b_e), propagation delay (τ_e), and cost per unit (ϕ_e). For readability purpose, Table 5.1 summarizes the notations for the resource model explained above.

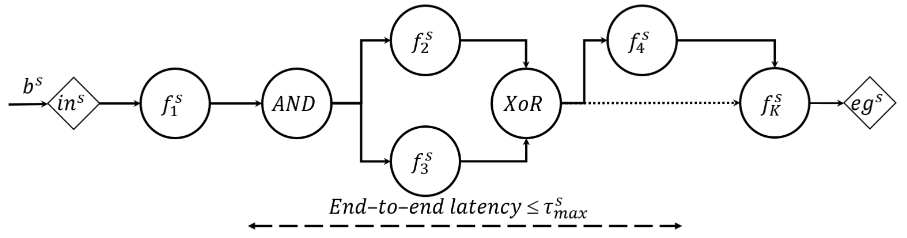
Table 5.1: Features associated with infrastructures

Symbol	Definition
$\mathcal{N}$=($\mathcal{V}$, $\mathcal{E}$)	Physical network, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of communication links
$\chi_i \subset \mathcal{V}$	Set of *function* nodes on NFVI i
$\vec{p}_v$	Amount of available resources provided by the *function* node $v \in \chi$
ψ_v^r	Cost of resource unit $r \in \mathcal{R}$ at function node $v \in \chi$
b_e	Bandwidth requested by link $e \in \mathcal{E}$
τ_e	Propagation delay of link $e \in \mathcal{E}$
ϕ_e	Cost per resource unit of link $e \in \mathcal{E}$

To deploy on-the-fly the set of network services ($\mathcal{S}$) over the physical network for a given application ($\mathcal{A}$), the NFVO seeks to determine the placement of VNFs and the corresponding routing in network services, known as online Network Service Em-



(a) Network service without *routing* VNFs



(b) Network service with *routing* VNFs

Figure 5.1: Notations used to describe network services

bedding (NSE) problem. The NFVO's objective will be to minimize $\mathcal{S}$'s deployment cost under some constraints like latency and location. More specifically, each network service $s \in \mathcal{S}$ will be deployed between 2 predefined *forwarding* nodes, *a.k.a.*, service ingress/egress (in^s/eg^s) points. We, also, denote by (b^s) the required bandwidth between these two points for a network service s.

As previously described in Chapter 4, the network connectivity topology (NCT) can be either sequential (i.e., containing operative VNFs, only) or not (i.e., containing operative and routing VNFs) as depicted in Fig. 5.1 (a) and (b), respectively; where the VNFs will be placed on *function* nodes. Let F^s be the set of VNFs associated to s and connected with the set of virtual links (L^s). Any Virtual Link $\ell \in L^s$ connected 2 VNFs (f_1, $f_2 \in F^s$) will be mapped onto the set of communication links and *forwarding* nodes between the *function* nodes where the VNFs are placed. Also, we denote by τ^s_{max} the end-to-end maximum latency for s.

Any VNF $f \in V^s$ requires some amount of resources for its execution represented as $\vec{c}_f = \left(c^r_f\right)_{r\in\mathcal{R}}$, and some bandwidth ($\lambda_f = \sum_{s\in\mathcal{S}} b^s$), along with introducing a per-packet processing delay (γ_f). Note that $\mathcal{A}$ might call for the same VNF instance f in different network services in $\mathcal{S}$. For security and/or latency reasons, VNFs should be deployed over specific domains ($\{\chi_i(f)\}$) with particular infrastructure providers. To sum-up, Equation 5.1 formally defines the NS embedding request.

$$NSE\text{-}Req(s) = [\{\langle\vec{c}_f, \lambda_f, \gamma_f\rangle\}, L^s, b^s, \tau^s_{max}] \quad | \quad f \in F^s \tag{5.1}$$

For readability purpose, Table 5.2 summarizes the notations for the NS embedding requests.

As mentioned in Chapter 4, the service owners (i.e., application developers) define their respective network services containing discovered VNFs. At deploy-

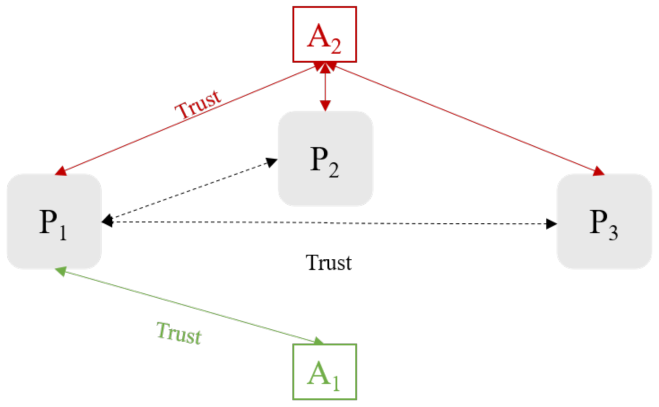
Table 5.2: Features associated with network service embedding requests

Symbol	**Definition**
s=(F^s, L^s)	Network service, where F^s is the set of VNFs and L^s is the set of virtual links in $s \in \mathcal{S}$
in^s	Ingress node of network service s
eg^s	Egress node of network service s
b^s	Required bandwidth by network service s
τ^s_{max}	End-to-end maximum latency for s
$\chi(f) = \{\chi_i(f)\}$	Set of potential infrastructures i in particular domains where VNF f should be deployed
$\vec{c}_f$	Amount of resources required by VNF f
λ_f	Total bandwidth required for VNF f
γ_f	Per-packet processing delay of VNF f

ment time, the NFVO might decide to place constituent VNFs over *function* nodes administrated by distinct providers. For high-security and/or reachability reasons, the service owner establishes trust partnerships with individual providers, only. Therefore, trust propagation among providers will be conditioned by the service owner. Indeed, this depends on whether the service owner's data traffic requires to be protected against attacks like jamming and scrambling. Furthermore, to mitigate packet loss and bottleneck risks, the network providers should jointly ensure high-connectivity to the network services. To this end, providers that would provision the same network service(s) including VNFs, with resources, should trust each other so that Service-Level-Agreement (SLA) between the service owner and all these providers will be respected. For illustration purposes, let $\mathcal{A}_1$,$\mathcal{A}_2$ be 2 applications featured with sensitive and non-sensitive data traffics, respectively, and p_1, p_2, p_3 be 3 infrastructure providers. Fig. 5.2 depicts a trust partnership between $\mathcal{A}_1$ and p_1 (i.e., 1 in the trust matrix 1) while p_1 has a trust partnership with p_2 and p_3. So, $\mathcal{A}_1$ will not trust neither p_2 and p_3 (i.e., 0 in the trust matrix 2). Contrarily, $\mathcal{A}_2$ featured with some pre-established trust partnership with p_1 will trust all the providers.

5.3 NSE problem formulation

Upon receipt of NS embedding requests ($\{NSE\text{-}Req(s)\}$), the NFVO will be in charge to determine the set of VNF placements and trust-based VL deployments requested by $\mathcal{S}/\ \mathcal{A}$ over time. Prior to provisioning, the NFVO should scrutinize the hosting capacity for each NFVI $\mathcal{N}_i$ and bandwidth for each physical link with the help of VIMs over time. As depicted in Fig. 5.3, each $\mathcal{N}_i$ maintains the VNF placement queue ($\mathcal{Q}_i$) when existing/new VNFs leave/arrive at the NFVI leading to resource release/consumption, respectively.



Trust matrix 1

Trust A ↔ P	P_1	P_2	P_3
A_1	1	0	0
A_2	1	1	1

Trust matrix 2

Trust P ↔ P	P_1	P_2	P_3
P_1	-	1	1
P_2	1	-	0
P_3	1	0	-

Figure 5.2: Trust relationships between applications and infrastructure providers

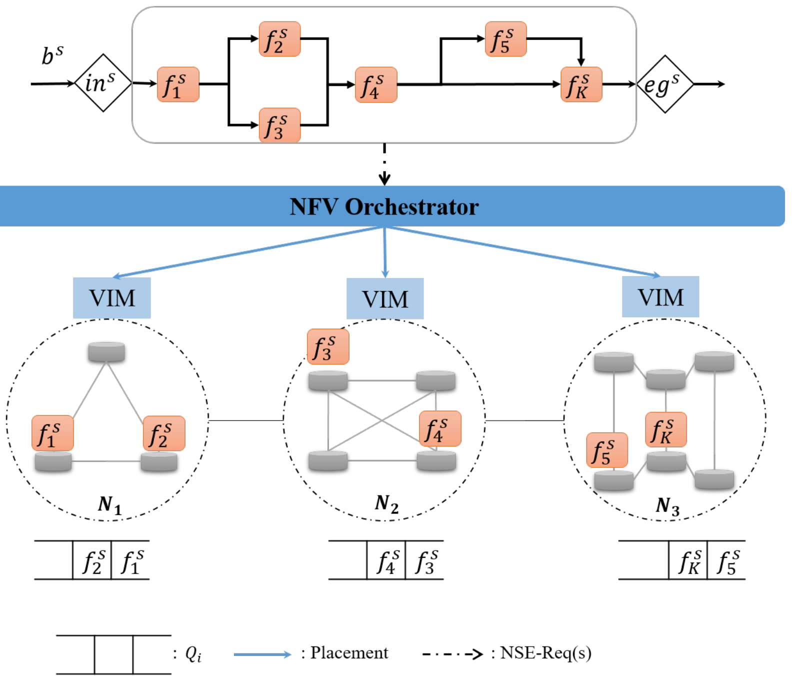


Figure 5.3: NSE orchestration

Let $\{NSE\text{-}Req(s)\}$ be the set of NS embedding requests to be satisfied. The NSE problem can be formulated as follows:

$$NSE = \{< \tau_f, t_f, \vec{c}_f, ?v \in \chi_{[?]i}(f), \{\psi_v^{r\in\mathcal{R}}\}, ?e \in \mathcal{E}_i, \phi_e, ?cost_s >\}$$

where

- f has some duration (τ_f) on any *function* node,
- t_f indicates f's arrival time on any *function* node,
- $?v$ corresponds to the *function* node to be determined for f's deployment and execution.
- $[?]i$ refers to the NFVI that can be either predefined as requirement or to be identified.
- ?e denotes the set of physical links to be decided for mapping the corresponding virtual link ℓ in L^s.
- $?cost_s$ represents the total cost to be paid by $\mathcal{A}$ for embedding s. Formally, Equation 5.2 computes this cost:

$$cost_s = \sum_{e\in\mathcal{E}} \phi_e y_e + \sum_{v\in\chi}\sum_{r\in\mathcal{R}} \psi_v^r \left(\sum_{f\in\mathcal{F}} y_{v,f}\, c_f^r\right) \tag{5.2}$$

In the following, we will consider the NSE problem as integer linear programming problem where the objective function is to minimize the total cost ($\sum cost_s$) that $\mathcal{A}$ should pay under constraints like those specified in $\{NSE\text{-}Req(s)\}$ (e.g., b^s and τ^s_{max}) but, also, those related to trust as per Section 5.2. Hereafter, we proceed with modelling NS embedding including constraints and the objective function, as well.

Placement constraints. Placement decisions are modelled with binary variables $x_{v,f}$, where $x_{v,f}$'s value will be equal to 1 if VNF $f \in \mathcal{F}^s$ will be deployed at the *function* node $v \in \chi(f)$, and to 0, otherwise. Equation 5.3 expresses that each VNF $f \in \mathcal{F}^S$ has to be mapped to one and only one node $v \in \chi(f)$.

$$\sum_{v\in\chi(f)} x_{v,f} = 1 \qquad \forall f \in \mathcal{F}^S \tag{5.3}$$

Routing constraints. Each virtual link $\ell = (f, f') \in L^s$ will be assigned with the set of binary variables $y_\ell^{e,s}$, whose value is equal to 1 if the physical link $e \in \mathcal{E}$ is part of the path between the nodes $(v,\, v')$ that would host $(f,\, f')$, and to 0 otherwise. Hence, Equation 5.5 represents the flow conservation constraints to enforce the existence of a path between v and v' where $Out(n)$ and $In(n)$ denote the set of outgoing/incoming links from/to the node n.

$$\sum_{e \in Out(n)} y_\ell^{e,s} - \sum_{e \in In(n)} y_\ell^{e,s} = \begin{cases} x_{v,f}^s & \text{if } n = v \\ -x_{v',f'}^s & \text{if } n = v' \\ x_{v,f}^s - x_{v',f'}^s & \text{if } n = v \text{and } n = v' \\ 0 & \text{otherwise} \end{cases} \tag{5.4}$$

$$\forall n \in \mathcal{V},\ \forall v \in \chi(f),\ \forall v' \in \chi(f')\ |v' \neq v,\ \forall \ell = (f, f') \in L^s,\ \forall s \in \mathcal{S}$$

Trust constraints. Let z_v^s be the binary variable whose value is equal to 1 if node $v \in \mathcal{V}$ is used to map network service $s \in \mathcal{S}$, and to 0 otherwise. We impose the equality $z_v^s = 1$ for $v \in \{in^s, eg^s\}$. For the other nodes $v \in \mathcal{V} \setminus \{in^s, eg^s\}$, the constraints on z_v^s's value are defined as follows:

$$M\, z_v^s \geq \sum_{e \in In(v)} \sum_{\ell \in L^s} y_\ell^{e,s} \tag{5.5}$$

$$z_v^s \leq \sum_{e \in Out(v)} \sum_{\ell \in L^s} y_\ell^{e,s} \tag{5.6}$$

where M is a large constant

To enforce the trust relationship ($\mathcal{T}$) between NFVI providers, we define an additional constraint as follows:

$$z_v^s + z_{v'}^s \leq 1 \tag{5.7}$$

$$\forall v \in \mathcal{N}_i,\ \forall v' \in \mathcal{N}_j,\ \mathcal{T}(\mathcal{N}_i, \mathcal{N}_j) \neq 1$$

Latency constraints. Since the end-to-end delay associated with s corresponds to $\sum_{f \in F^s} \gamma_f$ and should not exceed τ_{max}^s, it yields the following constraint:

$$\sum_{e \in \mathcal{E}} \sum_{\ell \in L^s} y_\ell^{e,s} \tau_e \leq \tau_{max}^s - \sum_{f \in F^s} \gamma_f \qquad \forall s \in \mathcal{S} \tag{5.8}$$

Capacity constraints. To enforce capacity constraints on the links, we define the following constraint:

$$\sum_{s \in \mathcal{S}} \sum_{\ell \in L^s} y_\ell^{e,s} b^s \leq b_e \qquad \forall e \in \mathcal{E} \tag{5.9}$$

Note that the same link e may appear multiple times in the end-to-end path from in^s to eg^s.

We, also, represent the constraint on *function* nodes' capacities as follows:

$$\sum_{f \in F^{\mathcal{S}}} x_{v,f} \lambda_f \vec{c}_f \leq \vec{p}_v \qquad \forall v \in \chi \tag{5.10}$$

Objective function. As previously mentioned, the aim is to minimize the total cost of the NSE decisions. So, this function (OF) can be expressed as follows:

$$OF = \sum_{e \in \mathcal{E}} \phi_e \left(\sum_{g \in \mathcal{S}} \sum_{\ell \in L^s} y_\ell^{e,s} b^s \right) + \sum_{v \in \chi} \sum_{r \in \mathcal{R}} \psi_v^r \left(\sum_{f \in F^{\mathcal{S}}} x_{v,f} \lambda_f c_f^r \right) \tag{5.11}$$

5.4 Heuristic NSE problem-solving

To reduce time of execution, we define a near-optimal solution based on candidate paths using heuristic search.

5.4.1 Heuristic search for candidate paths

To determine candidate paths, our 2-steps heuristic search solution first prunes $\mathcal{N}$ using *trust* constraint. Then, it incrementally builds the candidate paths satisfying *location* and *end-to-end latency* constraints, according to VNF ordering reported in s's forwarding graph.

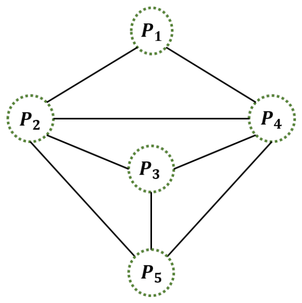
Let $\mathcal{G} = (P, T)$ be the trust graph where P represents the set of NFVI providers (P_i) and T corresponds to the set of trust relationships as per Fig. 5.4. The first step consists of building $\mathcal{N}$'s pruned networks ($\{\mathcal{N}'_{k=1,m}\}$) based on $\mathcal{G}$. To this end, we determine all maximal cliques in $\mathcal{G}$ (i.e., complete trust sub-graphs), relying on Bron–Kerbosch (BK) algorithm [Johnston 1976]. Algorithm 4 depicts BK's pseudo-code featured with recursive backtracking. It takes three disjoint sets X, Y, and Z in $\mathcal{G}$ as inputs where i) X stands for the currently growing clique and was initiated with $\{P_1, P_2\}$ [1], ii) Y refers to prospective $\{P_{j=\{3,\ldots,q\}}\}$ connected to all $P_{i \neq j} \in X$, and iii) Z contains the set of "visited" $\{P_i\}$. It is worth mentioning that X may include also any NFVI required by s. As output, BK returns $\{c_{k=1,m}\}$ maximal cliques, as reported in Fig. 5.4. Finally, each pruned network $\mathcal{N}'_k$ will be built with respect to c_k.

The second step proceeds with the pruned networks $\{\mathcal{N}'_{k=1,m}\}$. To map all virtual links $\ell \in L^s$ onto the corresponding set of physical links and intermediate *forwarding* nodes, we get inspired by Nguyen et al's work [Nguyen 2017]. First, $\mathcal{N}'_k$ will be

[1] P_1 and P_2 are the NFVI providers that own in^s and eg^s, respectively.

Algorithm 4 Bron-Kerbosch Algorithm

```
Require: $X$, $Y$, $Z$
1: if $Y = \emptyset$ and $Z = \emptyset$ then
2:     $c.add(X)$
3: end if
4: for each vertex $p \in Y$ do
5:     Call Bron-Kerbosch($X \cup \{p\}$, $Y \cap N(p)$, $Z \cap N(v)$)
6:     $Y \leftarrow Y \setminus \{p\}$
7:     $Z \leftarrow Z \cup \{p\}$
8: end for
```



Clique	NFVI
1	$\{P_1, P_2, P_5\}$
2	$\{P_1, P_3, P_4, P_5\}$

Figure 5.4: Network N' with two different cliques

artificially duplicated in $\{\mathcal{N}'_{k_i}\}_{i=0,|L^s|-1}$ and progressively connected with weighted artificial links as depicted in Fig. 5.5. Each $\mathcal{N}'_{k_i}$ will host f^s_{i+1} over some *function* node $v \in \chi(f_{i+1})$ considered as *exit* node that will be artificially linked to the same duplicate node considered as *entry* node in $\mathcal{N}'_{k_{i+1}}$ with the weight $\gamma_{f^s_{i+1}}$. It is worth noticing that *in* and *eg* correspond to *entry* and *exit* nodes in $\mathcal{N}'_{k_i}$ with i equals to 0 and $|L^s|$-1, respectively. For each $\mathcal{N}'_{k_i}$, we determine the set of all possible paths ($\pi^s_{\mathcal{N}'_{k_i}}$) between *entry* and *exit* nodes. The set of candidate paths π^s between in^s and eg^s refers to all possible combinations between adjacent $\pi^s_i \in \pi^s_{\mathcal{N}'_{k_i}}$.

Let $\tau(\pi^s)$ represent the end-to-end delay incurred by packets in s whether VNFs are placed and (physically) linked as per π^s. To get feasible solutions, we shall keep the paths π^s that satisfy *latency constraints* (i.e., $\tau(\pi^s) \leq \tau^s_{max}$).

5.4.2 Adjusting NSE formulation

In the following, we consider the set of candidate paths for S as $\Pi = \{\pi^s\}_{s\in\mathcal{S}}$. Initially, we calculate the traffic y_e passed over the link $e \in \mathcal{E}$, and expressed as:

$$y_e = \sum_{s\in\mathcal{S}} \theta^e_{\pi^s}\, b^s \tag{5.12}$$

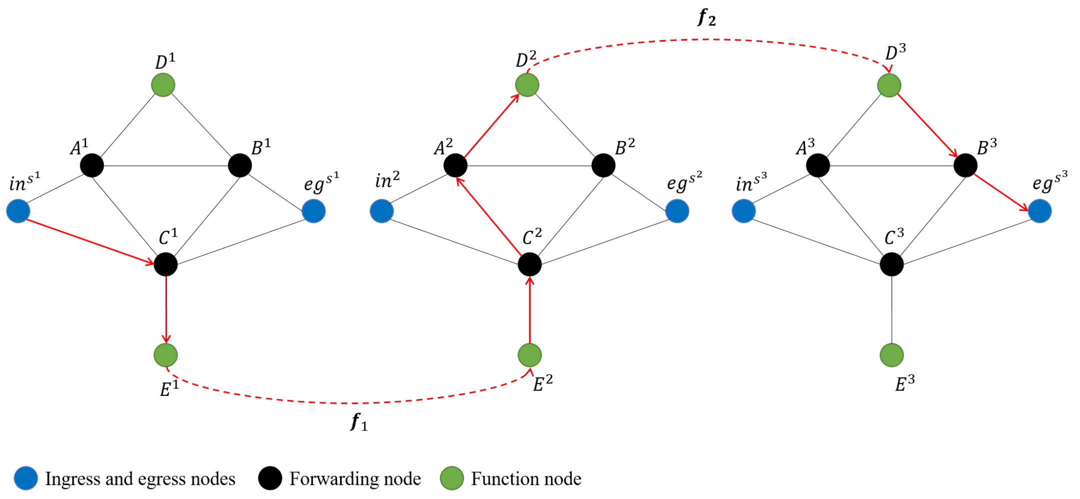


Figure 5.5: Expanded network for NS

where $\theta^e_{\pi^s}$ is defined as the number of times e in path π^s appears over all $\mathcal{N}'_{k_i}$.

Similarly, Equation 5.13 computes the total traffic processed by VNF f at node $v \in \chi(f)$ as follows:

$$y_{v,f} = \sum_{s \in \mathcal{S}} \theta^{v,f}_{\pi^s} \, b^s \tag{5.13}$$

Here, $\theta^{v,f}_{\pi^s}$ represents the number of possible artificial links associated to f.

$\boldsymbol{\pi}^s$ is said *feasible* if constraints on the capacity of links and *function* nodes are satisfied. Among the set of feasible paths, we look for some that minimizes the linear cost defined as follows:

$$\sum_{e \in \mathcal{E}} \phi_e y_e + \sum_{v \in \chi} \sum_{r \in \mathcal{R}} \psi^r_v \left(\sum_{f \in \mathcal{F}} y_{v,f} \, c^r_f \right) \tag{5.14}$$

Hereafter, we proceed with modelling NS embedding including the constraints and the objective function ($H(\Pi)$), as well.

$$H(\Pi) = min(\sum_{e\in\mathcal{E}} \phi_e \, y_e + \sum_{v\in\chi} \sum_{r\in\mathcal{R}} \psi_v^r \sum_{f\in\mathcal{F}} y_{v,f} \, c_f^r) \tag{5.15}$$

subject to:

$$y_e = \sum_{s\in\mathcal{S}} \sum_{\pi^s\in\Pi} \theta_{\pi^s}^e \, b^s \, x_{\pi^s}^s \qquad \forall e \in \mathcal{E} \tag{5.16}$$

$$y_{v,f} = \sum_{s\in\mathcal{S}} \sum_{\pi^s\in\Pi} \theta_{\pi^s}^{v,f} \, b^s \, x_{\pi^s}^s \qquad \forall v \in \chi(f) f \in \mathcal{F} \tag{5.17}$$

$$y_e \leq b_e \qquad \forall e \in \mathcal{E} \tag{5.18}$$

$$\sum_{f\in\mathcal{F}} y_{v,f} \, \vec{c}_f \leq \vec{p}_v \qquad \forall v \in \chi \tag{5.19}$$

$$y_e \geq 0 \qquad \forall e \in \mathcal{E} \tag{5.20}$$

$$y_{v,f} \geq 0 \qquad \forall v \in \chi(f) f \in \mathcal{F} \tag{5.21}$$

$$x_{\pi^s}^s \in \{0,1\} \qquad \forall \pi^s \in \Pi s \in \mathcal{S} \tag{5.22}$$

5.5 Dynamic pricing scheme for resource allocation

As per Sections 5.3 and 5.4, the NFVO obtains the NS embedding for $\mathcal{A}$'s requests over time. As per Equation 5.2 and 5.15, the deployment result heavily depends on the computing and networking resources' prices. Like any multi-provider market, the resource prices can be affected over time due to different reasons such as fair traffic load distribution at the infrastructure level, as illustrated in Fig. 5.6. Indeed, providers should satisfy their respective SLAs to avoid any penalties in case of violations. To this end, $\mathcal{N}_i$'s provider will dynamically adjust ψ_v^r and ϕ_e based on v's and e's load including in-going and out-going data traffic in order to prevent $\mathcal{N}_i$ from resource congestion/saturation. When prices increase/decrease, the NFVO will fairly allocate resources over less-saturated nodes.

Based on Table 5.1, $\rho_v^r \Phi_v^r$ and $\rho_e \Phi_e$ correspond to the utilization cost of r at v and e in the physical network, where ρ_v^r and ρ_e denote the amount of used computing/networking resources at v/e, respectively. Among existing cost functions reported in the literature, we deem appropriate to choose the Kleinrock function $\Phi_e(\rho_e) = 1/(1 - \rho_e)$.

The Kleinrock function assumes that the total traffic never exceed the resource and that the cost per resource unit grows unboundedly as the former approaches the latter. As the NFVO's objective is to prevent the NFVIs from resource congestion, we tend to compute the costs ϕ_e and ψ_v^r, which are publicly advertised, as follows:

- The cost per unit capacity ϕ_e of link $e \in \mathcal{E}$ is computed as

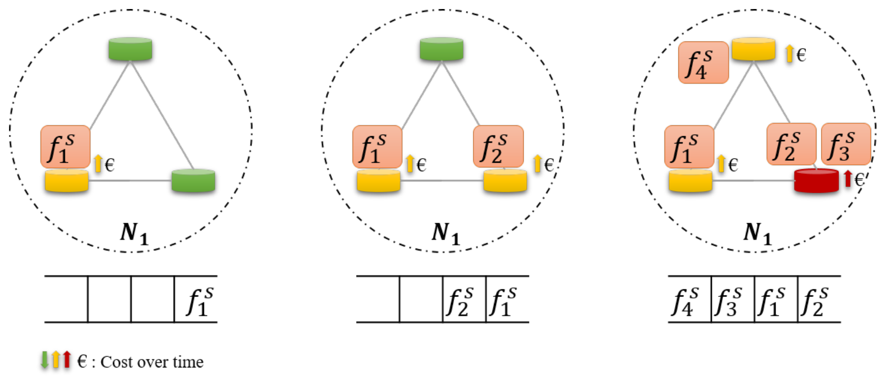


Figure 5.6: Changing costs based on resource utilization rates for Provider 1

$$\phi_e = \frac{1}{b_e}\ [\Phi_e(\rho_e) + \rho_e \Phi'_e(\rho_e)] \tag{5.23}$$

- The cost per unit capacity ψ_v^r of resource r at node v is computed as

$$\psi_v^r = \frac{1}{p_v^r}\ [\Phi_v^r(\rho_v^r) + \rho_v^r\, \Phi_v^{r\prime}(\rho_v^r)] \tag{5.24}$$

These costs per unit capacity are of course updated each time a new network services is embedded in the network.

5.6 Simulation and evaluation

To validate the proposed solutions, we developed an event-driven stochastic simulator in Python using Gurobi optimization tool ([gur 2021]).

5.6.1 Parameter settings for simulation

In the following, we describe the chosen parameter settings for the conducted simulation.

Simulated network. To generate the simulated network $\mathcal{N}$, we rely on characteristics of real networks in terms of domain and NFVI as follows:

- *Domain.* As per Section 5.2, we consider 4 domains namely, RAN, edge, transport and core, that each encompasses 3 NFVIs, one per specific provider.

- *NFVI*. To simulate the aforementioned NFVIs, we associate them with details collected from existing configurations provided by 2 libraries namely, the survivable fixed telecommunication network design [snd] and Internet zoo topology [Knight 2011]. It is worth noticing that the same NFVI's nodes are connected with intra-links. All NFVIs contain 3 ingress nodes and 3 egress nodes. The way of interconnecting NFVIs, more specifically their nodes, relies on the uniform distribution featured with $\alpha = 0.1$.

On top of this, we simulate congestion situations in $\mathcal{N}$ that make resource allocation challenging. In the literature [Nguyen 2017], there are 4 main congestion types namely, *bandwidth-limited*, *node-limited*, *both-limited* and *low*. To determine the restrictive value per congestion type for the node's resource amount and link's bandwidth, we scrutinize the rate of rejected NSE requests when running the proposed NSE solutions and varying their respective capacities until this ratio reaches 20%.

Simulated NSE requests. Let us consider 3 application types ($\mathcal{A}_{1..3}$) along with their respective requirements and the set of corresponding VNFs ($F^{\mathcal{A}_i}$) as depicted in Tables 5.3 and 5.4.

Table 5.3: Network Service requirements

Application	**Bandwidth**	**Latency**	**Trust level**
$\mathcal{A}_1$- Enhanced Media Streaming	40-340 Mbps	10-30ms	1
$\mathcal{A}_2$- Vehicle-To-Everything	150-1000 Mbps	$>$ 10ms	2
$\mathcal{A}_3$- Critical communication	5 Gbps	>5ms	3

Table 5.4: Virtual Network Functions set for each application

Application	**Operative VNFs**	**Routing VNFs**
$\mathcal{A}_1$	Content acquisition function; Compressor; Transcoder; Cache; Decoder	Proxy & Swing VNFs (AND, OR, XoR)
$\mathcal{A}_2$	IDentity and Access Management; Distributed denial of service; Traffic Monitor;Firewall; Network address translation; Virtual Private Network	
$\mathcal{A}_3$	Firewall, Malware Scanner, Intrusion Detection System; Deep packet inspection; Virtual Private Network Threats Classifier; Threats Mitigation;	

$\{NSE\text{-}Req(s \in \mathcal{S})\}$ associated with $\mathcal{A}_i$ are generated as follows:

- For each s, we randomly build $F^s \subset F^{\mathcal{A}_i}$ while avoiding routing VNFs to come after each other along with L^s that connects VNF pairwises.
- in^s and eg^s are randomly mapped onto NFVIs' ingress and egress nodes.

- As per $\mathcal{A}_i$'s requirements, we rely on the uniform distribution to produce $\vec{c}_f$, b^s, and τ_{max}.
- $\chi(f)$ [2] contains the set of potential locations where VNF f should be deployed.

Regarding trust constraints, we define 3 trust levels, each associated with 1 $\mathcal{A}_i$. At the first level, $\mathcal{A}_i$ trusts any provider. At the second/third, only one/two randomly selected provider per domain can be considered as untrustworthy.

Simulated arrival rate. For each iteration, $\{NSE\text{-}Req(s)\}$ issued by all $\mathcal{A}_i$ are deployed and pseudo-executed over $\mathcal{N}$ till completion as follows:

- *NSE-Req* arrives at t_f following Poisson distribution with parameter δ.
- τ_f denotes the pseudo-execution time generated from exponential distribution with average $\beta = 10$.

To avoid biases, we conducted our simulation for 400 time slots repeated 5 times. During each iteration, after any s's pseudo-completion, this network service will release the allocated resources.

5.6.2 Performance and quality metrics

To assess our proposed optimal and heuristic solutions' performance and quality from the both application and provider perspectives, we define the set of metrics as follows.

- *Average execution time* (T_{exec}) refers to the time needed to find placement and routing solution for an NS request. This metric reflects the ability of the algorithm to scale with problem size. The average execution time T_{exec} is defined as follows:

$$T_{exec}(s) = \frac{\sum_{j \in arrivedRequests} T^j_{exec}}{|arrivedRequests|} \tag{5.25}$$

  Where T_j represents the execution time of the j-th NSs request and is calculated by Equation

$$T^j_{exec} = T^j_{end} - T^j_{start} \tag{5.26}$$

  Where T^j_{start} and T^j_{end} denote the starting time and the completion time for mapping the j-th request, respectively.

- *Relative quality gap* (*Gap*) refers to the percentage of relative gap between the optimal solution and Heuristic NSE solution. Formally, Equation 5.27 computes

[2] This set could be empty

Gap as follows:

$$Gap(\%) = \frac{H() - O()}{H()} * 100 \tag{5.27}$$

We, also, calculate the relative standard deviation $RSD(\pm)$ reported in [rsd 2021] that measures the dispersion of the gap between the optimal and heuristic costs.

- *NSE blocking ratio* (B) represents the percentage of rejected NSE requests due to unavailability of computing/network resources.

$$B = \frac{|blockedRequests|}{|arrivedRequests|} \tag{5.28}$$

where $|blockedrequests|$ and $|Arrivedrequests|$ denote the numbers of rejected NSE requests and the total arrived NSs requests, respectively.

- Maximum resource utilization rate refers the percentage of the maximum consumed resources at nodes (U_v) and links (U_e).

$$U_v = max(\sum_{f \in F^s} \frac{y_{v,f} c_f^r}{p_v}) \tag{5.29}$$

$$U_e = max(\sum_{e \in \mathcal{E}} \frac{y_e b^s}{b_e}) \tag{5.30}$$

- *Path length* (P) indicates the average of number of hops in a path, and it is calculated as follows:

$$P = |hops| \tag{5.31}$$

5.6.3 Benchmarking

The first series of experiments compare the both optimal and near-optimal solutions in term of the relative quality gap and the execution time computed as per Equation 5.27 and 5.25 , respectively. We proceed as follows.

For each application type $(\mathcal{A}_{1..3})$, we vary the number δ of $\{NSE\text{-}Req(s)\}$ over $[2, 10]$ with step of 2, where the corresponding NSs are deployed over the 4 congestion types. Table 5.5 depicts the obtained results for the gap. We can observe that the average gap does not exceed 0.54% reflecting satisfying results for our heuristic NSE solution. The worst gap is 1.96% with A_1's 10 NSs deployed over "Both-limited" congestion type. This can be explained by the fact that as A_1 belongs to the application category featured with low trust constraints leading to a high number of candidate paths. Since the number of considered candidate paths is limited to $k = 10$, better solutions can be discarded/missing. It might happen that the choice of k affects the

Table 5.5: Relative quality gap ($\% \pm \mathrm{std}$) with $k = 10$

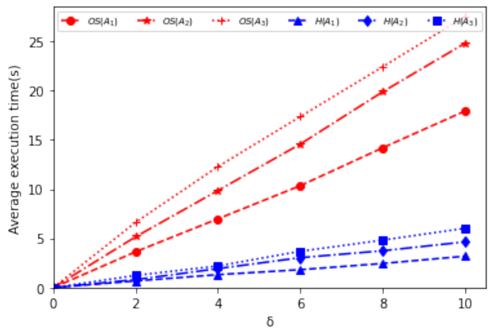
	A_1			
δ	Low	Both-limited	Node-limited	Bandwidth-limit
2	0.61 ±0.01	1.00 ±0.01	0.09 ±0.01	1.10 ±0.01
4	0.70 ±0.01	1.18 ±0.01	1.00 ±0.01	1.13 ±0.01
6	1.05 ±0.01	1.22 ±0.01	1.00 ±0.01	1.26 ±0.02
8	1.25 ±0.01	1.27 ±0.01	1.00 ±0.01	1.40 ±0.02
10	1.29 ±0.02	1.96 ±0.02	1.00 ±0.01	1.40 ±0.02

	A_2			
δ	Low	Both-limited	Node-limited	Bandwidth-limit
2	0.14 ±0.01	0.20 ±0.01	0.15 ±0.01	0.35 ±0.01
4	0.15 ±0.01	0.20 ±0.01	0.20 ±0.01	0.56 ±0.01
6	0.23 ±0.01	0.25 ±0.01	1.13 ±0.01	0.25 ±0.01
8	0.25 ±0.01	0.35 ±0.01	1.15 ±0.02	0.20 ±0.01
10	0.41 ±0.01	0.56 ±0.01	1.17 ±0.02	0.20 ±0.01

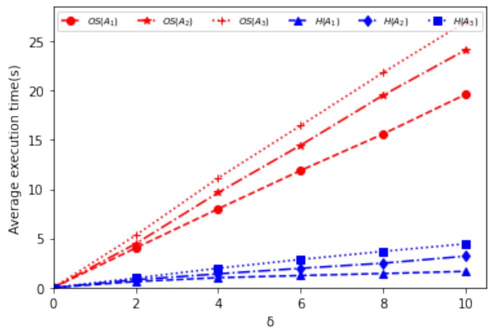
	A_3			
δ	Low	Both-limited	Node-limited	Bandwidth-limit
2	0.28 ±0.01	0.10 ±0.01	0.22 ±0.01	0.11 ±0.01
4	0.36 ±0.01	0.22 ±0.01	0.47 ±0.01	0.26 ±0.01
6	0.26 ±0.01	0.18 ±0.01	.31 ±0.01	0.15 ±0.01
8	0.32 ±0.01	0.11 ±0.01	0.26 ±0.01	0.41 ±0.01
10	0.40 ±0.01	0.19 ±0.01	0.30 ±0.01	0.24 ±0.01

quality of results. However, it is worth mentioning that including more candidate paths guarantees better solutions but will take more execution time. The corresponding standard deviation varies between 0.01 and 0.02 revealing quite a stable behavior for our heuristic solution on all congestion types and with the three application types.

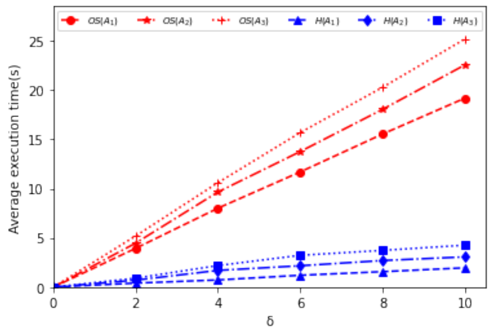
For each experiment, we calculate the average execution time $T_{exec}(s)$ using Equation 5.25. As depicted in Fig. 5.7, we observe that the heuristic NSE (H) solution is ≈ 6 times faster than the optimal one (OF). When δ augments, the average execution time linearly increases and reaches ≈ 28 seconds for the optimal approach contrary to ≈ 5 seconds for the heuristic one, in worst results. Also, we notice that T_{exec} obtained in case of our heuristic NSE solution does not increase drastically regardless of congestion types. However, it requires more execution time for network services belonging to A_3 due to strong trust constraints as explained before.



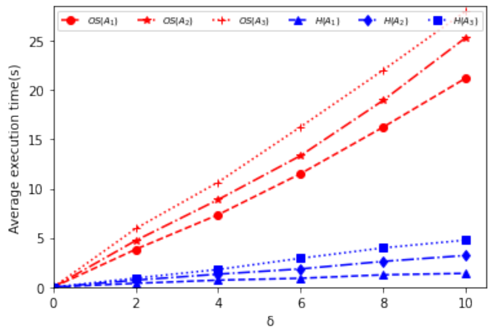
(a) Low congestion network



(b) Both-limited network



(c) Node-limited network



(d) Bandwidth-limited network

Figure 5.7: Average execution time (seconds) with k= 10

5.6.4 Dynamic pricing impact analysis

In this experiment, we run the simulation to evaluate some realistic metrics, such as the blocking ratio, the resource utilization and the average length of NS path, in a dynamic pricing system. The objective is to study the impact of the dynamic pricing. So, we compared the Heuristic NSE and optimal solution with/without dynamic pricing (H^*/H and OF^*/OF, respectively).

Fig. 5.8a illustrates the blocking ratio (B, Equation 5.28) based on load δ. Overall, we observe that H/H^* both obtain almost the same blocking ratio as the OF^*/OF solutions with an average difference less than 0.7%. We also notice that B reaches only 2.5% for all $H/H*$ and $OF/OF*$ and thus, is pretty good when $\delta <= 8$. However, B increases when δ is assigned to higher values. The important finding is that H^* reduces B with $\approx 2\%$ where H attains 19%.

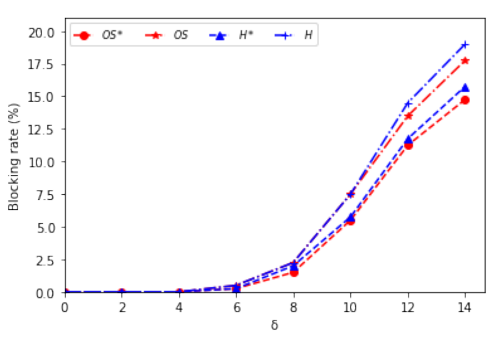
During this experiment, we also measure the node resource utilization rate (U_r, Equation 5.29). As depicted in Fig. 5.8b, H^* reduces U_r about 5%. For $\delta = 14$, H saturates at $\approx 43\%$ of some nodes' capacity. However, H^* reduces it up to 39%.

As depicted in Fig. 5.8c and Fig. 5.8d, we measure the networking-resource utilization rate (U_e, Equation 5.30) for intra-link and inter-link, respectively. We can observe that intra-links are saturated quickly compared to inter-links as expected with respect to the inter-link capacity that would be higher than the inter-link capacity (Section 5.6.1). And, H^* reduces the utilization rate up to 11%. With $\delta = 14$, 18% of inter-links are consumed, where H^* decreases it to $\approx 15\%$.

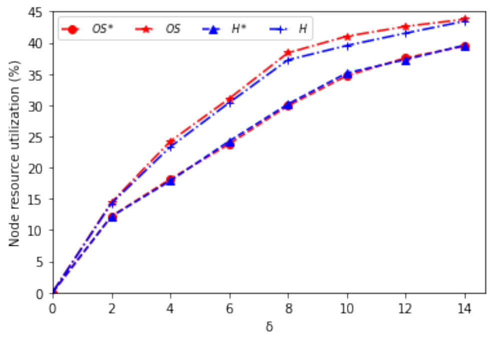
Since H^* represents an overhead due to more longer paths to build when applying the dynamic pricing, the path length (P, Equation 5.31) increases by 4 hops, only. Since the resources get expensive over time, H^* will choose longer paths with less expensive node and link resources with the option of converging to an expensive path, as well. This explains why the path length obtained in case of $H/H*$ and OF/OF^* does not increase drastically.

5.7 Summary

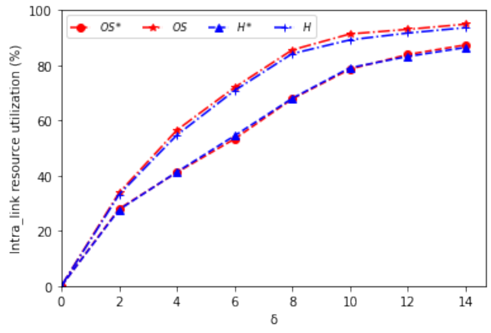
In this chapter, we proposed 2 alternatives to Network Service Embedding (NSE) problem referred to as optimal and near-optimal solutions that both support a lazy resources allocation strategy. The first consists of searching the optimal placements and paths, as well, for NSE requests while the second first seeks for candidate placements and corresponding paths prior to building the near-optimal ones. For evaluation purposes, we built event-driven simulations near to realistic NFV dynamic ecosystem along with various experimental scenarios. We, also, defined different performance and quality metrics to benchmark both solutions and analyze the impact of dynamic pricing on both.



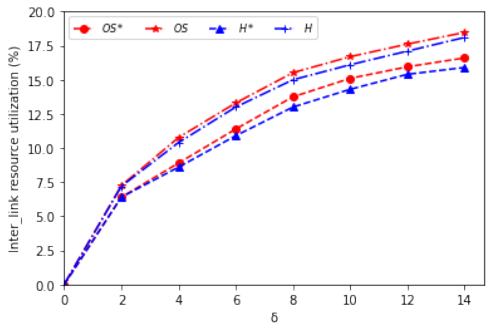
(a) Blocking ratio



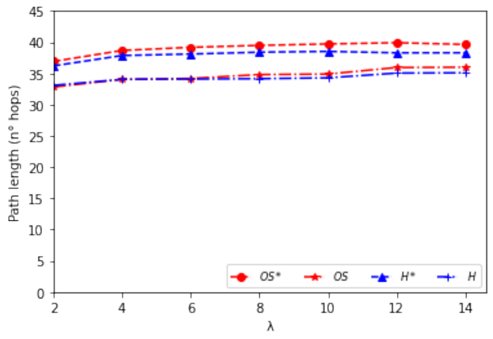
(b) Node resource utilization rate



(c) Intra-Link utilization rate



(d) Inter-Link utilization rate



(e) Path length

Figure 5.8: Dynamic pricing's impact

Chapter 6

Conclusion and perspectives

6.1 Conclusion

Service Computing refers to a distributed and dynamic computing architecture that packages functionality as a suite of interoperable entities that can be operated over multi-domains ecosystem. NFV enables the network and the network functions (VNFs) virtualization and enables their provisioning according to the very same life cycle phases (i.e., design, develop, describe, publish, discover, instantiate, configure, deploy, execute, and manage) of the service computing model. However, considering the characteristics and specificities of the network operation with regard to regular services (e.g., Web services), there is a need for a novel and appropriate procedures to be able to provision the NFV resources in a proper way.

In this thesis, we did focus on designing, implementing and evaluation appropriate procedures that support the former phases (i.e. design, instantiation, deployment, and management) of the NFV resources. We did consider the VNFs resources, as well as, the NS resources that compose a set of VNFs to support more complex and sophisticated functionalities. Specificaly, We addressed in this thesis the lack of an approach that can handle:

- The heterogeneity of NS technologies' and providers' which prevents the full automation of the NS designing
- The rigid instantiation scenarios and sequential connectivity in the NS which induce cost, tedious, and time-consuming NS provisioning
- Network service deployment and management while satisfying constraints like location, latency, and trust in multi-administrative domains.

6.2 Overview on contributions

The summary of the thesis contributions is presented below.

1. Beyond the classic VNFD proposed and standardized by ETSI-NFV, in our first contribution, we designed a domain-independent VIKING ontology that enables a comprehensive and generic description of the VNF's capabilities from functional

and non-functional perspectives. In addition, we implemented an automated semantic-based discovery mechanism that relies on VIKING to ensure the best VNFs discovery and matching between user requirements and VNF properties.

2. To achieve agile and dynamic network service provisioning, our second contribution consists of a novel approach that extends the ETSI NFV architectural framework with novel technical VNFs, called *routing VNFs* with efficient re-configurable wiring capabilities for NSs. This initiative distinguishes domain-specific aspects from the connectivity ones in the NS definition. Domain-specific aspects are implemented with regular VNFs, while the routing VNFs support connectivity aspects. By doing so, developers will be able to change and update the NS instantiation logic at run-time, given specific criteria (e.g., message type, data size, QoS metrics).

3. In the third contribution, we turn to deploy on-the-fly network services that could dynamically be reconfigured and adjusted, depending on the need (e.g., request type, data type), during run-time. We introduced a lazy resources allocation solution to the network service embedding problem. This solution optimizes both VNF placement for each network service along with traffic routing across VNFs while satisfying constraints like location, latency, and trust, among others. This optimization aims at minimizing the total network service deployment cost.

6.3 Obtained results and lessons learned

Altogether, the above contributions represent a cost-driven approach for virtualized network services life-cycle management. Firstly, the semantic VNF descriptor covering both functional and non-functional VNF's capabilities and the deployment requirement, as well, has undoubtedly improved VNF interpretation and, subsequently, the discovery process by identifying and selecting the appropriate ones. Further, the use of *routing* VNFs concept within NS provisioning fits well with virtualization principles and brings cost-effectiveness and agility to the network. Indeed, they considerably reduce complexity in NFV connectivity management and decrease the network's operating cost. Last but not least, the proposed lazy resources allocation solution, with the dynamic pricing model, to the network service embedding problem seems appropriate to tackle the emerging needs of the next-generation service providers with challenging constraints on the network business model ecosystem.

6.4 Potentials future work

In the near term, we are considering the following avenues of research.

Semantic NS composition. We plan to use semantics to describe and build VNF chains automatically. Current NS chains are represented through ETSI VNF Forwarding Graphs (VNF-FG) descriptors. These descriptors are manually designed by network administrators when using a straightforward model and sequential execution of VNFs. We believe that enhancing VNF-FG descriptors with semantics could enable the automatic build of more complex and sophisticated VNF chains.

BPEL4VNF. As seen in Chapter 2, it was beneficial to inspire (with revisitation) from SOA principles, developed in more than one decade, to target dynamic and flexible NS provisioning. To this end, we plan to extend Business Process Execution Language (BPEL) to automate NS provisioning. This could be done by extending BPEL gateways (e.g., parallel, inclusive, pick) and control flows (e.g., if, repeat until, for each) to interconnect the VNFs that make up the network services and support sharing data between them. Specifically, appropriate "BPEL4VNF" descriptors could be embedded in the NS connectivity and executed by an extended version of the MANO, the same way that BPEL engines (e.g., Apache ODE, Bonita) execute BPEL orchestration plans.

Implementation of a complete proof of concept prototype. For dissemination and normalization purposes, we plan to integrate into ETSI NFV open source MANO the proposed approaches. The Mastermyr Chest tools for VNF description, publication, and discovery, and DYVINE tools for NS instantiation are already implemented and integrated with OSM[1], an open-source MANO implementation; using both OpenStack as a VIM and OpendayLight as SDN-controller. However, the proposed heuristic NSE approach was implemented and evaluated through theoretical analysis only, which is insufficient to prove the real performances. Therefore, we plan to extend the NFVO of OSM with our heuristic NSE approach, to complete the Mastermyr Chest and DYVINE components.

In the long-term, we are considering the following avenues of research.

Zero-touch network and Service Management: The goal is to integrate our proposed *Routing VNFs* in Chapter 4 within the results of the newly Zero-touch network and Service Management (ZSM)[ETSI] ETSI working group. The ZSM working group focuses on describing automation in network management and aims to deliver policy-driven automation, intent-based automation, as well as intent-based service orchestration. We believe that this research work meets the objectives and the topics of interest of the ETSI ZSM working group.

Intent-Based Networks (IBN). IBN can be defined as a novel concept that incorporates: (i) SDN to manage the network control plane and

[1]https://osm.etsi.org/

(ii) machine learning and artificial intelligence to automate administration tasks across the network [Campanella 2019]. Recently published IBN-based work aim at automating network routes management during run-time (e.g., see [Han 2016], [Kiran 2018], [Paganelli 2017]). However, to the best of our knowledge, there are still no studies today that evaluate and compare the business and operational costs of integrating IBN into the ecosystem of network providers. IBN is still at its early stages, while NFV is becoming more broadly adopted nowadays by network providers (e.g., CISCO Systems, Netflix, etc.).

Mobility-aware NS management. Users' mobility causes significant challenges for NS management. Principally, these challenges are due to the traffic density and handovers for different access networks. A promising research area is to extend the current ETSI-MANO architecture to manage a non-homogeneous and resource-constrained set of mobile nodes (mobile devices, lamppost, any IoT sensors). It will be able to use mobile devices like our smartphones to deploy some VNFs, which may increase the limited resources and reduce the handovers. However, it may require more effort to manage *mobile* network service deployed over *mobile* infrastructure.

Scientific production

Journal

- Nour El Houda Nouar, Sami Yangui, Noura Faci, Khalil Drira, Said Tazi. A Semantic virtualized network functions description and discovery model. Computer Networks, vol. 195, 2021.

Conference

- Nour El Houda Nouar, Sami Yangui, Noura Faci, Khalil Drira, Said Tazi. Agile and Dynamic Virtualized Network Functions Wiring in Network Services. IEEE International Conference on Cloud Computing (CLOUD), pages 322-332, 2021.

Tutorial

- Nour El Houda Nouar, Sami Yangui, Noura Faci, Khalil Drira, Said Tazi. Semantic Virtualized Network Functions Description Publication and Discovery in Content Delivery Networks. 5th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA), 2018.

Bibliography

[Adala 2011] Asma Adala, Nabil Tabbane and Sami Tabbane. A Framework for Automatic Web Service Discovery Based on Semantics and NLP Techniques. Advances in Multimedia, vol. 2011, pages 1–7, 2011. (Cited in page 19.)

[Afify 2017] Yasmine M Afify, Nagwa L Badr, Ibrahim F Moawad and Mohamed F Tolba. A comprehensive business domain ontology for cloud services. In 2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS), pages 134–143. IEEE, 2017. (Cited in page 32.)

[Alaluna 2017] Max Alaluna, Luıs Ferrolho, José Rui Figueira, Nuno Neves and Fernando MV Ramos. Secure virtual network embedding in a multi-cloud environment. arXiv preprint arXiv:1703.01313, 2017. (Cited in pages 26 and 29.)

[Antonakoglou 2018] Konstantinos Antonakoglou, Xiao Xu, Eckehard Steinbach, Toktam Mahmoodi and Mischa Dohler. Toward haptic communications over the 5G tactile Internet. IEEE Communications Surveys & Tutorials, vol. 20, no. 4, pages 3034–3059, 2018. (Cited in page 74.)

[Araniti 2017] Giuseppe Araniti, Massimo Condoluci, Pasquale Scopelliti, Antonella Molinaro and Antonio Iera. Multicasting over emerging 5G networks: Challenges and perspectives. Ieee network, vol. 31, no. 2, pages 80–89, 2017. (Cited in page 74.)

[Baranda 2020] Jorge Baranda, Josep Mangues-Bafalluy, Ricardo Martínez, Luca Vettori, Kiril Antevski, Carlos J Bernardos and Xi Li. 5G-TRANSFORMER meets Network Service Federation: design, implementation and evaluation. In 2020 6th IEEE Conference on Network Softwarization (NetSoft), pages 175–179. IEEE, 2020. (Cited in pages 25, 26, and 29.)

[Bhamare 2016] Deval Bhamare, Raj Jain, Mohammed Samaka and Aiman Erbad. A survey on service function chaining. Journal of Network and Computer Applications, vol. 75, pages 138–155, 2016. (Cited in page 23.)

[Binz 2014] Tobias Binz, Uwe Breitenbücher, Oliver Kopp and Frank Leymann. TOSCA: portable automated deployment and management of cloud applications. In Advanced Web Services, pages 527–549. Springer, 2014. (Cited in page 20.)

[Boban 2018] M. Boban, A. Kousaridas, K. Manolakis, J. Eichinger and W. Xu. Connected Roads of the Future: Use Cases, Requirements, and Design

Considerations for Vehicle-to-Everything Communications. IEEE Vehicular Technology Magazine, vol. 13, no. 3, pages 110–123, 2018. (Cited in page 58.)

[Bonfim 2019] Michel Bonfim, Fred Freitas and Stênio Fernandes. A Semantic-Based Policy Analysis Solution for the Deployment of NFV Services. IEEE Transactions on Network and Service Management, vol. 16, no. 3, pages 1005–1018, 2019. (Cited in pages 21 and 27.)

[Boubendir 2018] Amina Boubendir, Emmanuel Bertin and Noemie Simoni. Flexibility and dynamicity for open network-as-a-service: From VNF and architecture modeling to deployment. In NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium, pages 1–6. IEEE, 2018. (Cited in page 4.)

[Bouten 2016] N. Bouten, M. Claeys, R. Mijumbi, J. Famaey, S. Latré and J. Serrat. Semantic validation of affinity constrained service function chain requests. In 2016 IEEE NetSoft Conference and Workshops (NetSoft), pages 202–210, June 2016. (Cited in pages 4, 21, and 27.)

[Callegati 2015] Franco Callegati, Walter Cerroni, Chiara Contoli and Giuliano Santandrea. Dynamic chaining of virtual network functions in cloud-based edge networks. In Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft), pages 1–5. IEEE, 2015. (Cited in pages 23 and 28.)

[Campanella 2019] A. Campanella. Intent Based Network Operations. In 2019 Optical Fiber Communications Conference and Exhibition (OFC), pages 1–3, 2019. (Cited in page 98.)

[Cao 2016] Hanwen Cao, Sandip Gangakhedkar, Ali Ramadan Ali, Mohamed Gharba and Josef Eichinger. A 5G V2X testbed for cooperative automated driving. In 2016 IEEE Vehicular Networking Conference (VNC), pages 1–4. IEEE, 2016. (Cited in page 58.)

[Carella 2015] Giuseppe Antonio Carella and Thomas Magedanz. Open baton: a framework for virtual network function management and orchestration for emerging software-based 5g networks. Newsletter, vol. 2016, page 190, 2015. (Cited in page 12.)

[Casazza 2017] M. Casazza, P. Fouilhoux, M. Bouet and S. Secci. Securing Virtual Network Function Placement with High Availability Guarantees. CoRR, 2017. (Cited in page 37.)

[Chowdhury 2010] N.M. Mosharaf Kabir Chowdhury and Raouf Boutaba. A survey of network virtualization. Computer Networks, vol. 54, no. 5, pages 862–876, apr 2010. (Cited in page 3.)

[clo 2022] Cloudify orchestration platform - multi cloud, cloud native amp; edge, Jan 2022. (Cited in page 12.)

[Csoma 2014] Attila Csoma, Balázs Sonkoly, Levente Csikor, Felicián Németh, Andràs Gulyas, Wouter Tavernier and Sahel Sahhaf. ESCAPE: Extensible service chain prototyping environment using mininet, click, netconf and pox. ACM SIGCOMM Computer Communication Review, vol. 44, no. 4, pages 125–126, 2014. (Cited in pages 24 and 28.)

[De Sousa 2019] Nathan F Saraiva De Sousa, Danny A Lachos Perez, Raphael V Rosa, Mateus AS Santos and Christian Esteve Rothenberg. Network service orchestration: A survey. Computer Communications, 2019. (Cited in page 23.)

[Domingue 2005] John Domingue, Dumitru Roman and Michael Stollberg. Web service modeling ontology (WSMO)-An ontology for semantic web services, 2005. (Cited in page 19.)

[Doynikova 2018] E. Doynikova and I. Kotenko. Approach for determination of cyber-attack goals based on the ontology of security metrics. IOP Conference Series: Materials Science and Engineering, vol. 450, page 052006, Nov 2018. (Cited in page 36.)

[Dwiardhika 2018] Dhanu Dwiardhika and Takuji Tachibana. Cost efficient VNF placement with optimization problem for security-aware virtual networks. In 2018 IEEE 7th International Conference on Cloud Networking (CloudNet), pages 1–3. IEEE, 2018. (Cited in page 26.)

[Dwiardhika 2019] Dhanu Dwiardhika and Takuji Tachibana. Virtual network embedding based on security level with VNF placement. Security and Communication Networks, vol. 2019, 2019. (Cited in pages 26 and 29.)

[Dzone 2018] Dzone, 2018. (Cited in page 22.)

[ets 2020] Network Functions Virtualisation (NFV) Release 3; Protocols and Data Models; RESTful protocols specification for the Policy Management Interface. ETSI GS NFV-SOL 012 V3.4.1. techreport, October 2020. (Cited in page 43.)

[ETSI] ETSI. ZSM. Zero Touch Network and Service Man. agement (ZSM). (Cited in page 97.)

[ETSI 2012] ETSI. Network Functions Virtualisation—Introductory White Paper. techreport, ETSI, 2012. (Cited in page 2.)

[ETSI 2014] Network Functions Virtualisation ETSI. ETSI GS NFV 002 V1.2.1Network Functions Virtualisation (NFV);Architectural Framework. Management and Orchestration, vol. V1.2.1, 2014. (Cited in pages ix, 10, and 11.)

[ETSI 2015] ISGNFV ETSI. ETSI GS NFV-REL 001 V1. 1.1: Network Functions Virtualisation(NFV); Resiliency Requirements, 2015. (Cited in page 37.)

[ETSI 2021] Network Functions Virtualisation ETSI. ETSI GS NFV-IFA 011, Network Functions Virtualisation (NFV) Release 4. vol. V4.2.1, 2021. (Cited in pages ix, 15, and 16.)

[Fischer 2017] Andreas Fischer, Ramona Kühn, Waseem Mandarawi and Hermann de Meer. Modeling security requirements for VNE algorithms. In proceedings of the 10th EAI International Conference on Performance Evaluation Methodologies and Tools on 10th EAI International Conference on Performance Evaluation Methodologies and Tools, pages 149–154, 2017. (Cited in pages 26 and 29.)

[Forti 2020] Stefano Forti, Gian-Luigi Ferrari and Antonio Brogi. Secure cloud-edge deployments, with trust. Future Generation Computer Systems, vol. 102, pages 775–788, 2020. (Cited in pages 26 and 29.)

[Gharbaoui 2017] Molka Gharbaoui, S Fichera, Piero Castoldi and Barbara Martini. Network orchestrator for QoS-enabled service function chaining in reliable NFV/SDN infrastructure. In 2017 IEEE Conference on Network Softwarization (NetSoft), pages 1–5. IEEE, 2017. (Cited in pages 24 and 28.)

[goo] What is site Reliability Engineering (SRE). (Cited in page 1.)

[Gruber 1993] Thomas R. Gruber. A translation approach to portable ontology specifications. Knowledge Acquisition, vol. 5, no. 2, pages 199 – 220, 1993. (Cited in pages 19 and 27.)

[gur 2021] The Fastest Solver, Dec 2021. (Cited in page 86.)

[Han 2016] Y. Han, J. Li, D. Hoang, J. Yoo and J. W. Hong. An intent-based network virtualization platform for SDN. In 2016 12th International Conference on Network and Service Management (CNSM), pages 353–358, 2016. (Cited in page 98.)

[Hawilo 2019] H. Hawilo, M. Jammal and A. Shami. Exploring Microservices as the Architecture of Choice for Network Function Virtualization Platforms. IEEE Network, vol. 33, pages 202–210, 2019. (Cited in page 22.)

[Herrera 2016] Juliver Gil Herrera and Juan Felipe Botero. Resource allocation in NFV: A comprehensive survey. IEEE Transactions on Network and Service Management, vol. 13, no. 3, pages 518–532, 2016. (Cited in page 25.)

[Houidi 2020] Omar Houidi, Oussama Soualah, Wajdi Louati and Djamal Zeghlache. Dynamic vnf forwarding graph extension algorithms. IEEE Transactions on Network and Service Management, vol. 17, no. 3, pages 1389–1402, 2020. (Cited in pages 25 and 29.)

[Hoyos 2016] L. C. Hoyos and C. E. Rothenberg. NOn: Network function virtualization ontology towards semantic service implementation. pages 1–6, Nov 2016. (Cited in pages 4, 21, 27, and 32.)

[IETF 2018] IETF. Service Function Chaining (SFC) Operation, Administration and Maintenance (OAM) Framework. draft-ietf-sfc-oam-framework-05. techreport, IETF, September 2018. (Cited in pages 4 and 18.)

[ITU 2015] ICT Facts and Figures: The World in 2015. May 2015. (Cited in page 9.)

[Jmila 2019] Houda Jmila and Gregory Blanc. Designing security-aware service requests for NFV-enabled networks. In 2019 28th International Conference on Computer Communication and Networks (ICCCN), pages 1–9. IEEE, 2019. (Cited in pages 23 and 28.)

[Johnston 1976] HC Johnston. Cliques of a graph-variations on the Bron-Kerbosch algorithm. International Journal of Computer & Information Sciences, vol. 5, no. 3, pages 209–238, 1976. (Cited in page 82.)

[Katsalis 2016] Kostas Katsalis, Navid Nikaein and Andy 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), pages 189–195. IEEE, 2016. (Cited in page 25.)

[Kim 2018] Sang Il Kim and Hwa Sung Kim. Semantic Ontology-Based NFV Service Modeling. In 2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN), pages 674–678. IEEE, 2018. (Cited in pages 21 and 27.)

[Kiran 2018] Mariam Kiran, Eric Pouyoul, Anu Mercian, Brian Tierney, Chin Guok and Inder Monga. Enabling intent to configure scientific networks for high performance demands. Future Generation Computer Systems, vol. 79, pages 205–214, 2018. (Cited in page 98.)

[Knight 2011] Simon Knight, Hung X Nguyen, Nickolas Falkner, Rhys Bowden and Matthew Roughan. The internet topology zoo. IEEE Journal on Selected Areas in Communications, 2011. (Cited in page 87.)

[Kreutz 2015] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky and S. Uhlig. Software-Defined Networking: A Comprehensive Survey. Proceedings of the IEEE, vol. 103, no. 1, pages 14–76, 2015. (Cited in page 57.)

[Küster 2007] Ulrich Küster, Birgitta König-Ries, Mirco Stern and Michael Klein. DIANE: an integrated approach to automated service discovery, matchmaking and composition. In Proceedings of the 16th international conference on World Wide Web, pages 1033–1042, 2007. (Cited in page 19.)

[Lal 2017] Shankar Lal, Tarik Taleb and Ashutosh Dutta. NFV: Security threats and best practices. IEEE Communications Magazine, vol. 55, no. 8, pages 211–217, 2017. (Cited in page 36.)

[Lee 2015] Seung-Ik Lee and Myung-Ki Shin. A self-recovery scheme for service function chaining. In 2015 International Conference on Information and Communication Technology Convergence (ICTC), pages 108–112. IEEE, 2015. (Cited in pages 23 and 28.)

[Li 2021] Xi Li, Andres Garcia-Saavedra, Xavier Costa-Perez, Carlos J Bernardos, Carlos Guimarães, Kiril Antevski, Josep Mangues-Bafalluy, Jorge Baranda, Engin Zeydan, Daniel Corujoet al. 5Growth: An End-to-End Service Platform for Automated Deployment and Management of Vertical Services over 5G Networks. IEEE Communications Magazine, vol. 59, no. 3, pages 84–90, 2021. (Cited in pages 26 and 29.)

[Lin 2017] P. Lin, C. Wu and P. Shih. Optimal Placement of Network Security Monitoring Functions in NFV-Enabled Data Centers. In International Symposium on Cloud and Service Computing (SC2), pages 9–16, Los Alamitos, CA, USA, nov 2017. IEEE Computer Society. (Cited in page 37.)

[Liu 2017] Junjie Liu, Wei Lu, Fen Zhou, Ping Lu and Zuqing Zhu. On dynamic service function chain deployment and readjustment. IEEE Transactions on Network and Service Management, vol. 14, no. 3, pages 543–553, 2017. (Cited in pages 24, 28, and 29.)

[Long 2015] J. Long, E. Shelhamer and T. Darrell. Fully convolutional networks for semantic segmentation. In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3431–3440, June 2015. (Cited in page 19.)

[Luizelli 2018] Marcelo Caggiani Luizelli, Danny Raz and Yaniv Sa'ar. Optimizing NFV chain deployment through minimizing the cost of virtual switching. In IEEE INFOCOM 2018-IEEE Conference on Computer Communications, pages 2150–2158. IEEE, 2018. (Cited in pages 25, 28, and 29.)

[Mangues-Bafalluy 2019] Josep Mangues-Bafalluy, Jorge Baranda, Iñaki Pascual, Ricardo Martínez, Luca Vettori, Giada Landi, Arturo Zurita, David Salama, Kiril Antevski, Jorge Martín-Pérezet al. 5G-TRANSFORMER Service Orchestrator: design, implementation, and evaluation. In 2019 European Conference on Networks and Communications (EuCNC), pages 31–36. IEEE, 2019. (Cited in page 26.)

[Martin 2004] David Martin, Mark Burstein, Hobbset al. OWL-S: Semantic markup for web services. W3C member submission, vol. 22, no. 4, 2004. (Cited in page 19.)

[Martini 2016] Barbara Martini and Federica Paganelli. A service-oriented approach for dynamic chaining of virtual network functions over multi-provider software-defined networks. Future Internet, vol. 8, no. 2, page 24, 2016. (Cited in pages 24 and 28.)

[Mazrekaj 2016] Artan Mazrekaj, Isak Shabani and Besmir Sejdiu. Pricing Schemes in Cloud Computing: An Overview. 2016. (Cited in page 36.)

[Mechtri 2017] Marouen Mechtri, Chaima Ghribi, Oussama Soualah and Djamal Zeghlache. NFV orchestration framework addressing SFC challenges. IEEE Communications Magazine, vol. 55, no. 6, pages 16–23, 2017. (Cited in pages 20 and 27.)

[Medhat 2016] Ahmed M Medhat, Tarik Taleb, Asma Elmangoush, Giuseppe A Carella, Stefan Covaci and Thomas Magedanz. Service function chaining in next generation networks: State of the art and research challenges. IEEE Communications Magazine, vol. 55, no. 2, pages 216–223, 2016. (Cited in page 23.)

[Mijumbi 2016] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck and Raouf Boutaba. Network Function Virtualization: State-of-the-Art and Research Challenges. IEEE Communications Surveys & Tutorials, vol. 18, no. 1, pages 236–262, 2016. (Cited in pages 3 and 13.)

[Mohammed 2016] AA Mohammed, Molka Gharbaoui, Barbara Martini, Federica Paganelli and Piero Castoldi. SDN controller for network-aware adaptive orchestration in dynamic service chaining. In 2016 IEEE NetSoft Conference and Workshops (NetSoft), pages 126–130. IEEE, 2016. (Cited in pages 23 and 28.)

[Morin 2020] Cedric Morin, Géraldine Texier, Christelle Caillouet, Gilles Desmangles and Cao-Thanh Phan. Optimization of Network Services Embedding Costs over Public and Private Clouds. In 2020 International Conference on Information Networking (ICOIN), pages 360–365. IEEE, 2020. (Cited in pages 25 and 29.)

[ndl 2020] NDL OWL, January 2020. (Cited in pages 19 and 27.)

[nfv 2015] NFV, Network Functions Virtualisation. ETSI GS NFV-SEC 009 V1. 1.1 (2015-12). 2015. (Cited in page 36.)

[NFV 2017] Management NFV Release 2 and Network Service Templates Specification Orchestration. ETSI GS NFV-IFA 014 V2.3.1 (2017-08), 2017. (Cited in pages 5, 15, and 57.)

[nfv 2018] Network Functions Virtualisation (NFV). ETSI GR NFV-IFA 012 V3.1.1 (2018-10). 2018. (Cited in page 4.)

[Nguyen 2017] Thi Minh Nguyen. Optimizing resource allocation in infrastructure networks based on network function virtualization. PhD thesis, Université Pierre et Marie Curie-Paris VI, 2017. (Cited in pages 82 and 87.)

[Nguyenphu 2018] Thinh Nguyenphu. SOL001:VNF Descriptor (VNFD) Overview. techreport, ETSI, May 2018. (Cited in page 43.)

[Oliver 2018] I. Oliver, S. Panda, K. Wang and A. Kalliola. Modelling NFV concepts with ontologies. pages 1–7, Feb 2018. (Cited in pages 4, 21, and 27.)

[ona 2021] Sep 2021. (Cited in page 12.)

[Ordanini 2008] Andrea Ordanini and Paolo Pasini. Service co-production and value co-creation: The case for a service-oriented architecture (SOA). European Management Journal, vol. 26, no. 5, pages 289–297, 2008. (Cited in page 3.)

[osm] OSM release. `https://osm.etsi.org/`, journal=OSM. (Cited in page 12.)

[Pace 2012] N. M. Pace and L. Zakaras. Moving Beyond Eyes-On Review: The Promise of Computer-Categorized Review. In Where the money goes: Understanding litigant expenditures for producing electronic discovery, chapter 5, pages 59–69. Rand, 2012. (Cited in page 50.)

[Paganelli 2017] F. Paganelli, F. Paradiso, M. Gherardelli and G. Galletti. Network service description model for VNF orchestration leveraging intent-based SDN interfaces. In 2017 IEEE Conference on Network Softwarization (NetSoft), pages 1–5, 2017. (Cited in page 98.)

[Pei 2018] Jianing Pei, Peilin Hong, Kaiping Xue and Defang Li. Efficiently embedding service function chains with dynamic virtual network function placement in geo-distributed cloud system. IEEE Transactions on Parallel and Distributed Systems, vol. 30, no. 10, pages 2179–2192, 2018. (Cited in pages 25, 28, and 29.)

[Peltz 2003] Chris Peltz. Web services orchestration and choreography. Computer, vol. 36, no. 10, pages 46–52, 2003. (Cited in page 3.)

[Petcu 2013] Dana Petcu, Beniamino Martino, Salvatore Venticinque, Massimiliano Rak, Tamás Máhr, Gorka Lopez, Fabrice Brito, Roberto Cossu, Miha Stopar, Svatopluk Šperka and Vlado Stankovski. Experiences in building a mOSAIC of clouds. Journal of Cloud Computing: Advances, Systems and Applications, vol. 2, no. 1, page 12, 2013. (Cited in page 19.)

[Pfaff 2015] Ben Pfaff, Justin Pettit, Teemu Koponen, Ethan Jackson, Andy Zhou, Jarno Rajahalme, Jesse Gross, Alex Wang, Joe Stringer, Pravin Shelaret al. The Design and Implementation of OpenvSwitch. In 12th USENIX symposium on networked systems design and implementation (NSDI 15), pages 117–130, 2015. (Cited in page 66.)

[Powers 2011] Powers. Evaluation: From precision, recall and f-measure. Journal of Machine Learning Technologies, vol. 2, no. 1, pages 37–63, 2011. (Cited in page 50.)

[Quinn 2018] Paul Quinn, Uri Elzur and Carlos Pignataro. Network service header (NSH). In RFC 8300. RFC Editor, 2018. (Cited in page 67.)

[Raissi 2019] F. Raissi, S. Yangui and F. Camps. Autonomous Cars, 5G Mobile Networks and Smart Cities: Beyond the Hype. In 2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), pages 180–185, 2019. (Cited in pages 58 and 74.)

[rdf 2020] Online, February 2020. (Cited in page 19.)

[Rosa 2015] Raphael Vicente Rosa, Mateus Augusto Silva Santos and Christian Esteve Rothenberg. MD2-NFV: The case for multi-domain distributed network functions virtualization. In 2015 International Conference and Workshops on Networked Systems (NetSys), pages 1–5. IEEE, 2015. (Cited in page 25.)

[rsd 2021] Relative standard deviation formula: RSD Calculator (excel template), Mar 2021. (Cited in page 89.)

[Salvadori 2017] I. Salvadori, B. Oliveira, A. Huf, E.C. Inacio and F. Siqueira. An Ontology Alignment Framework for Data-Driven Microservices. In Proceedings of the 19th International Conference on Information Integration and Web-Based Applications & Services, 2017. (Cited in page 22.)

[Scheid 2016] Eder J Scheid, Cristian C Machado, Ricardo L dos Santos, Alberto E Schaeffer-Filho and Lisandro Z Granville. Policy-based dynamic service chaining in Network Functions Virtualization. In 2016 IEEE Symposium on Computers and Communication (ISCC), pages 340–345. IEEE, 2016. (Cited in pages 24 and 28.)

[Segata 2013] Michele Segata, Renato Lo Cigno and Falko Dressler. Towards communication strategies for platooning, 2013. (Cited in page 69.)

[Seog-Chan Oh 2006] Seog-Chan Oh, H. Kil, Dongwon Lee and S. R. T. Kumara. Algorithms for Web Services Discovery and Composition Based on Syntactic and Semantic Service Descriptions. In The 8th IEEE International Conference on E-Commerce Technology and The 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE'06), pages 66–66, 2006. (Cited in page 19.)

[Shapiro 1987] Stuart C Shapiro and William J Rapaport. SNePS considered as a fully intensional propositional semantic network. In The knowledge frontier, pages 262–315. Springer, 1987. (Cited in page 19.)

[sid 2013] TM Forum Information framework (SID). November 2013. (Cited in page 20.)

[snd] Sndlib. `http://sndlib.zib.de/home.action`. Accessed: 2022-01-15. (Cited in page 87.)

[Soares 2014] Joao Soares, Miguel Dias, Jorge Carapinha, Bruno Parreira and Susana Sargento. Cloud4nfv: A platform for virtual network functions. In 2014 IEEE 3Rd international conference on cloud networking (cloudnet), pages 288–293. IEEE, 2014. (Cited in pages 20 and 27.)

[Song 2007] R. Song, Z. Luo, J-R. Wen, Y. Yu and H-W. Hon. Identifying ambiguous queries in web search. In Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007, pages 1169–1170, 2007. (Cited in page 47.)

[Souag 2015] A. Souag, C. Salinesi, Raúl Mazo and I. Comyn-Wattiau. A security ontology for security requirements elicitation, pages 157–177. Springer International Publishing, 2015. (Cited in page 36.)

[Sowa 1987] John F. Sowa. Semantic Networks, 1987. (Cited in page 19.)

[Specification 2017] Committee Specification. TOSCA Simple Profile for NetworkFunctions Virtualization (NFV) Version 1.0. May 2017. (Cited in pages 4, 18, 20, and 27.)

[Sumra 2014] Irshad Ahmed Sumra, Halabi Bin Hasbullahet al. Effects of attackers and attacks on availability requirement in vehicular network: a survey. In 2014 International Conference on Computer and Information Sciences (ICCOINS), pages 1–6. IEEE, 2014. (Cited in page 70.)

[Sun 2017] Chen Sun, Jun Bi, Zhilong Zheng, Heng Yu and Hongxin Hu. NFP: Enabling network function parallelism in NFV. In Proceedings of the Conference of the ACM Special Interest Group on Data Communication, pages 43–56, 2017. (Cited in pages 23 and 28.)

[Tang 2003] Chunqiang Tang, Zhichen Xu and Sandhya Dwarkadas. Peer-to-peer information retrieval using self-organizing semantic overlay networks. In Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications - SIGCOMM '03. ACM Press, 2003. (Cited in page 19.)

[Tao 2017] Xiaofeng Tao, Yan Han, Xiaodong Xu, Ping Zhang and Victor C. M. Leung. Recent advances and future challenges for mobile network virtualization. Science China Information Sciences, vol. 60, no. 4, page 040301, Mar 2017. (Cited in page 3.)

[tco 2015] vmware: total Cost of ownership Study Virtualizing the Mobile Core, 2015. (Cited in page 17.)

[TeleGeography] TeleGeography. COVID-19 Network Impact. (Cited in page 1.)

[Thomas 2007] Erl Thomas. SOA principles of service design. Boston: Prentice Hall, vol. 37, pages 71–75, 2007. (Cited in page 3.)

[Torkzaban 2019] Nariman Torkzaban, Chrysa Papagianni and John S Baras. Trust-aware service chain embedding. In 2019 Sixth International Conference on Software Defined Systems (SDS), pages 242–247. IEEE, 2019. (Cited in pages 26 and 29.)

[Torkzaban 2020] Nariman Torkzaban and John S Baras. Trust-aware service function chain embedding: A path-based approach. In 2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), pages 31–36. IEEE, 2020. (Cited in page 29.)

[Trabelsi 2007] S. Trabelsi, L. Gomez and Y. Roudier. Context-aware security policy for the service discovery, volume 1, pages 477–482. 2007. (Cited in page 36.)

[Valcarenghi 2018] Luca Valcarenghi, Barbara Martini, Kiril Antevski, CJ Bernardos, Giada Landi, Marco Capitani, Josep Mangues-Bafalluy, Ricardo Martínez, Jorge Baranda, Iñaki Pascualet al. A framework for orchestration and federation of 5G services in a multi-domain scenario. In Proceedings of the Workshop on Experimentation and Measurements in 5G, pages 19–24, 2018. (Cited in page 25.)

[van der Ham 2015] Jeroen van der Ham, József Stéger, Sándor Laki, Yiannos Kryftis, Vasilis Maglaris and Cees de Laat. The NOVI information models. Future Generation Computer Systems, vol. 42, pages 64 – 73, 2015. (Cited in pages 19 and 27.)

[Virtualisation 2014] Network Functions Virtualisation. Management and Orchestration (GS/NFV-MAN-001), 2014. (Cited in pages 2, 4, 18, and 43.)

[Voigt 2018] Shaun Voigt, Catherine Howard, Dean Philp and Christopher Penny. Representing and Reasoning About Logical Network Topologies. In Madalina Croitoru, Pierre Marquis, Sebastian Rudolph and Gem Stapleton, editors, Graph Structures for Knowledge Representation and Reasoning, pages 73–83, Cham, 2018. Springer International Publishing. (Cited in page 19.)

[Walsh 2002] Aaron E Walsh. Uddi, soap, and wsdl: the web services specification reference book. Prentice Hall Professional Technical Reference, 2002. (Cited in page 3.)

[Wang 2015] Yang Wang, Phanvu Chau and Fuyu Chen. A framework for security-aware virtual network embedding. In 2015 24th International Conference on Computer Communication and Networks (ICCCN), pages 1–7. IEEE, 2015. (Cited in pages 26 and 29.)

[Willner 2015] A. Willner, R. Loughnane and T. Magedanz. FIDDLE: Federated Infrastructure Discovery and Description Language. In 2015 IEEE International Conference on Cloud Engineering, pages 465–471, 2015. (Cited in pages 19 and 27.)

[Xilouris 2014] Georgios Xilouris, Eleni Trouva, Felicia Lobillo, João M Soares, Jorge Carapinha, Michael J McGrath, George Gardikis, Pietro Paglierani, Evangelos Pallis, Letterio Zuccaroet al. T-NOVA: A marketplace for virtualized network functions. In 2014 European Conference on Networks and Communications (EuCNC), pages 1–5. IEEE, 2014. (Cited in pages 20 and 27.)

[Yang 2004] Jian Yang and Mike P Papazoglou. Service components for managing the life-cycle of service compositions. Information Systems, vol. 29, no. 2, pages 97–125, 2004. (Cited in page 13.)

[Yangui 2016] Sami Yangui, Roch H. Glitho and Constant Wette. Approaches to end-user applications portability in the cloud: A survey. IEEE Communications Magazine, vol. 54, no. 7, pages 138–145, 2016. (Cited in page 4.)

[Zhang 2017] Yang Zhang, Bilal Anwer, Vijay Gopalakrishnan, Bo Han, Joshua Reich, Aman Shaikh and Zhi-Li Zhang. Parabox: Exploiting parallelism for virtual network functions in service chaining. In Proceedings of the Symposium on SDN Research, pages 143–149, 2017. (Cited in pages 23 and 28.)

[Zsoka 2019] Zoltan Zsoka and Khalil Mebarkia. Layered Solutions for Dynamic Service Chaining. In 2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), pages 292–296. IEEE, 2019. (Cited in page 24.)