Network Service Chaining by using Mashups Technology in NFV

Trabajo de Grado

Julian Andrés Fuentes Vidal
Juan David Salazar Idrobo

Advisor: PhD. Oscar Mauricio Caicedo Rendón

Departamento de Telemática
Facultad de Ingeniería Electrónica y Telecomunicaciones
Universidad del Cauca
Popayán, Cauca, 2017

Network Service Chaining by using Mashups Technology in NFV

Julian Andrés Fuentes Vidal

Juan David Salazar Idrobo

Trabajo de grado presentado a la Facultad de Ingeniería Electrónica y Telecomunicaciones de la Universidad del Cauca para obtener el título de: Ingeniero en Electrónica y Telecomunicaciones

Advisor: PhD. Oscar Mauricio Caicedo Rendón

Departamento de Telemática
Facultad de Ingeniería Electrónica y Telecomunicaciones
Universidad del Cauca
Popayán, Cauca, 2017

Acknowledgements

First and foremost, we would like to thank God for our lives. We express our sincere gratitude to our families, parents and brothers for their continuous support and love throughout the years. They have dedicated their lives to our education, and we would like to dedicate this work to them. We would like to thank our academic advisor, Professor Oscar Mauricio Caicedo Rendón, for his support directing our research with innovative ideas and helping solve some problems.

Abstract

Network Service Chaining (NSC) is introduced by Network Function Virtualization (NFV) to give the possibility to select Network Services (NSs) according to the needs or requests of a user. Currently, there are problems that reduce the flexibility of NSC and cause delays, traffic with unnecessary routes and few control options for administrators. Although there are different types of NSC (*e.g.* dynamic, and static) and some solutions (*e.g.* automatic algorithms, and frameworks) that allow choosing the one that suits the best the needs of the user, they do not give a full solution to the existing problem. In this undergraduate work, we propose an architecture for semi-automatic NSC, using advantages of mashups technology to achieve a solution to the current lack of flexibility of the NSC. With proof-of-concept based on Software Defined Networks (SDN), we were able to demonstrate that our architecture allows solutions of low consumption in response time (<350 ms) and bandwidth (<3 KB), giving full control to the administrator to choose the network services that are executed and avoiding the need to have advanced knowledge.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement

In the current technological evolution around the computer networks and communications highlights the importance of NFV. The main function of NFV is to decouple the network services from the specialized hardware to locate them in commodity hardware [2]. Such decoupling allows Telecommunications Service Providers (TSPs) to offer more network services (NSs) to users and performing a faster deployment of NSs. In addition, with the use of NFV, TSPs reduce Capital Expenditures (CAPEX) and Operating Expenses (OPEX) due to the integrated management option [3].

The current networks have several shortcomings and one of them is the lack of flexibility in the customization of NSs, *i.e.*, the possibility of selecting NSs according to the needs or request of a user. In this sense, NFV introduces a fundamental concept, Network Service Chaining (NSC). The authors of [1] define NSC as an provider-level process for the continuous delivery of services based on Network Functions (NF) associations.

Currently, the NSC concept is evolving due to multiple problems in the delivery of NSs such as delays and unnecessary routes, which damage the throughput of services and network resources are saturated. In addition, addressing the NSC challenge requires taking into account that it can be analyzed from three points of view, which are: static, semi-automatic and dynamic. Static NSC is used in traditional computer networks and, basically, it consists of deploying NSs using specialized hardware and predetermined service chains. Therefore, when a new NS (*e.g.* firewall, Deep Packet Inspection, Network Address Translation) is requested by a user, the acquisition and installation of specialized hardware is necessary for the new NS. Related to dynamic NSC, there are several works in which the authors propose solutions based on frameworks [4], algorithms [5] and models [6]. Although such solutions obtain significant improvement percentages (*e.g.* response times, throughput, error reduction and delay) do not define an integral solution that increases the NSC flexibility and customization.

Concerning to semi-automatic NSC, there is a information and work lack that develop it alternative solution. In this way, it must be taken into account that performing semi-automatic NSC includes human intervention (to a lesser extent than static NSC) and, therefore, it is necessary to add a concept that facilitates human-computer interaction. One such concept, which has been used in different domains, is mashups. Mashups are web applications that integrate several resources available on the Internet [7], which allows the composition of new services and, in the specific case of our work, would facilitate networks administrator the NSC.

The mashups technology has not been used in NSC but there are works done around networking. Some of these works are: [7] in which the authors introduce a mashup ecosystem, named Mashment Ecosystem (Mashment are mashups aimed at addressing network situations), which allows providing solutions to overcome situations of network management. In addition, in [8], the authors demonstrate that mashups technology can be used to carry out the monitoring of virtual nodes. Furthermore, [9] proposes a model focused on cloud services mashups, named QoS Aware Services Mashup (QASM), which facilitates the selection of service instances and data flow paths to satisfy with Quality of Service (QoS) requirements. Finally, [10] defines a mashups-based approach for network management that facilitates network administrators to build their own solutions based on requirements of the elements that make up an SDN.

To sum up, in the mentioned works have been proposed solutions that are not integral to the NSC, none of which take advantage of the mashups technology. Moreover, the projects developed with mashups technology have not yet been focused on NSC, which allows defining the following research question:

How to carry out NSC by using mashups?

In order to answer this research question, we present the following objectives.

1.2 Objectives

1.2.1 General

- Performing the network service chaining by using mashups.

1.2.2 Specifics

- To adapt the mashups-based architecture proposed in [8] [11] to support the network services chaining.
- To implement a proof-of-concept of adapted architecture.
- To evaluate proof-of-concept throughput in an NFV environment.

1.3 Research Contributions

The main contributions provided in this work are mentioned below:

- An adaptation of the mashups-based architecture, proposed in works [8] [11] to support the network service chaining.

- A proof-of-concept of the architecture proposed for the network service chaining.
- Late evaluation of the proposed architecture for the network service chaining in terms of bandwidth, flexibility and time consumption.

1.4 Methodology and Activities

The activities developed in our undergraduate work were organized using the concept of Work Breakdown Structure (WBS). WBS defines a hierarchical structure by levels in which the initial level or root defines the most abstract activity and, the following levels define the tasks to be developed for its fulfillment [12]. Figure 1.1 shows the hierarchical schema of the work packages.

We use Scrum [13] as an agile development methodology that will be used for software engineering processes. The activities to be carried out in each work package are shown below.

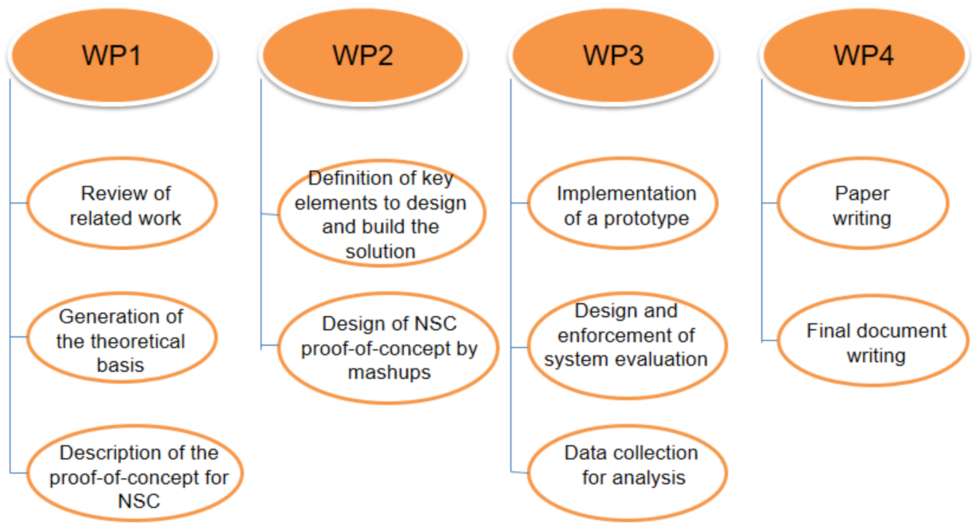

WP1
Review of related work
Generation of the theoretical basis
Description of the proof-of-concept for NSC
WP2
Definition of key elements to design and build the solution
Design of NSC proof-of-concept by mashups
WP3
Implementation of a prototype
Design and enforcement of system evaluation
Data collection for analysis
WP4
Paper writing
Final document writing

Figure 1.1: Work packets.

WP1. Initial knowledge basis generation

- Review of related work.
- Generation of the theoretical basis.
- Definition of the proof-of-concept for NSC.

WP2. Proof-of-concept design for network service chaining

- Definition of key elements to design and build the solution.
- Design of NSC proof-of-concept by mashups.

WP3. Prototype building and proof-of-concept evaluation

- Implementation of a prototype.
- Design and enforcement of system evaluation.
- Data collection for analysis.

WP4. Publishing

- Paper writing.
- Final document writing.

1.5 Publications

Our NSC proposal presented in this final document will be reported to the scientific community through paper submission to renowned journal.

- Number 1, **Julian Andrés Fuentes Vidal, Juan David Salazar Idrobo,** Oscar Mauricio Caicedo Rendon. **An approach based on mashups for Network Service Chaining: A SDN case study**
  - Status: Under reviewing

1.6 Document Structure

This document has been divided into chapters described below.

- Chapter 1 presents the **Introduction** that contains the Problem Statement, Objectives, Research Contributions, Methodology and Activities, Publications and the structure of this document.
- Chapter 2 presents the **Background** about the relevant topics concerning our research. These topics include NFV, NSC and mashups.
- Chapter 3 presents the **Related Work** that describes the researches works closer to our proposal.
- Chapter 4 introduces the mashups-based architecture for **Semi-automatic NSC**, an approach that allow conforming NSs by using mashups. To do this, we describe an overview of the operation of our solution by a Finite State Machine (FSM). Then, we present our motivating scenarios and, finally, we describe each one of the elements and layers of our architecture.
- Chapter 5 exposes and analyses the **Proof-of-concept**, where we present a prototype of our approach. Then, we explain an Operational Example.

Subsequently, we evaluate and analyze our approach by using different metrics (*i.e.,* bandwidth, time-consuming and flexibility).

- Chapter 6, we present **Conclusions** and **Future work**, where we provide the main conclusions of our work and important implications for future works.

Chapter 2

Background

In this chapter, we present a description of the technologies and concepts necessary for the development of our work. In the first description, we expose NFV and its structure, after we introduce the NSC concept and we highlight the viability of the semi-automatic NSC and finally, we present mashups technology and its main approach.

2.1 Network Functions Virtualization

Around the computer and communications networks large changes are being investigated and incorporated. In this sense, international standardization and research groups such as the European Telecommunications Standard Institute (ETSI) have defined concepts as NFV.

The main function of NFV is to decouple the network functions from dedicated hardware devices. Thus, the network functions can be executed in commodity hardware and, in this way, the TSPs can use them as software instances. Therefore, a service (*e.g.* firewall, load balancer and middleware) can be decomposed into Virtual Network Functions (VNFs) located on different network equipment [2].

NFV allows reducing CAPEX and OPEX to large industries and TSPs due to the decrease of specialized network devices [14]. It also facilitates the deployment of new services more efficiently and in less time. Consequently, NFV allow taking advantage of commodity hardware reducing the network complexity [3].

2.1.1 Fundamental concepts

ETSI divides the NFV architecture (see Figure 2.1) into three main components: Network Function Virtualization Infrastructure (NFVI), VNFs and NFV Management and Orchestration) [15]. NFVI refers to the combination hardware and software resources that conform the deployment environment for VNFs. The hardware includes Commercial-Off-the-Shelf (COTS) resources of computational, storage and network hardware that provide treatment, storage and connectivity to VNFs. Virtual resources are abstractions of computing, storage and network resources. A virtualization layer separates virtual resources from fundamental physical resources allowing abstraction [2].

NFV MANO provides the required functionality for provisioning VNFs and related operations. It includes instrumentation and life cycle management of the hardware and software resources that support the virtualization infrastructure and the life cycle management of VNFs. In essence, it focuses on all tasks required for specific virtualization management [16], *e.g.*, coordination with the Operations Support System (OSS) and Business Support System (BSS).

VNFs are an implementation of a NF deployed on virtual resources, *e.g.*, a virtual machine. VNF concept is explained in more detail in the next subsection.

2.1.2 Virtual Network Functions

A VNF is the implementation of a NF in virtual resources and can be composed for different elements that can be deployed, each element, in a different virtual machine. The number of VNFs and the order of execution of the VNFs depend directly on the service requirements and requests. Virtualization should not affect the throughput of

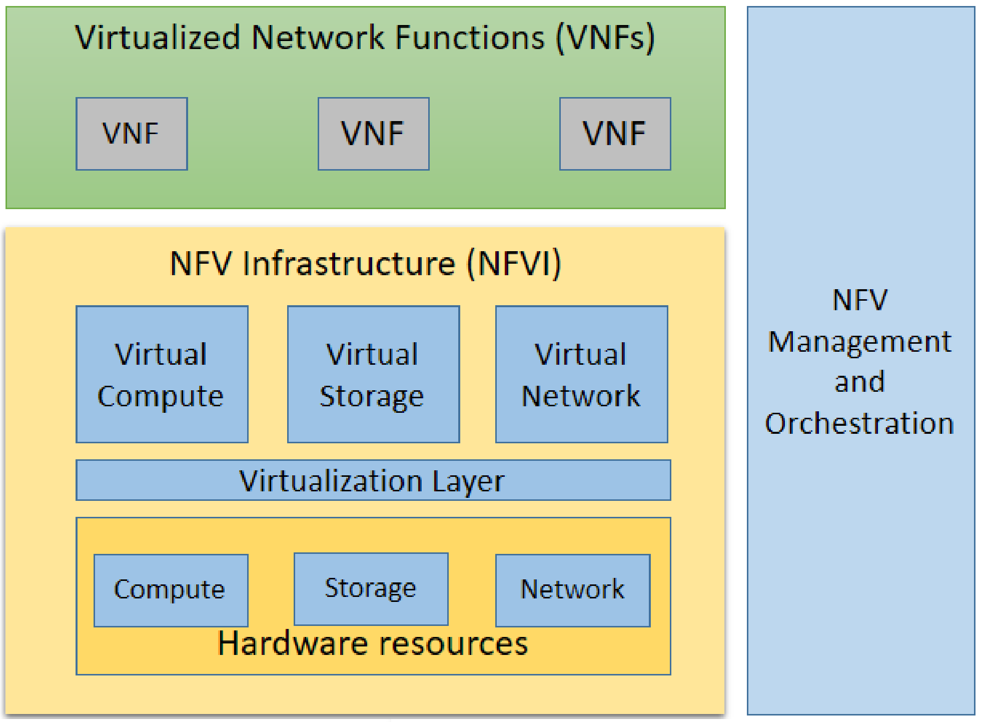


Figure 2.1: NFV Architecture [17].

the request service whether VNF are run on dedicated hardware or virtual machines [2].

ETSI has defined different use cases [18] in which the VNF concept is feasible to apply as Next Generation Firewall (NGFW) Acceleration, Virtual Base Station (VBS) L1 Acceleration, Virtual Acceleration Interface for VNFs, Deep Packet Inspection (DPI), among others.

2.2 Network Service Chaining in NFV

One of the most important issues around NFV is NSC. However, in much of the NFV literature, the NSC concept has not been standardized. Thus, the authors of Research directions in network service chaining [1] define NSC as a carrier-grade process for continuos delivery of services based on network functions associations. Continuos delivery is the ability of dynamic orchestration and automated deployment of NF to increase an operational improvement. Carrier grade refers to high availability and fast fault recovery [1].

NSC is to provide an order to the network elements in order to provide flexible network services to the users. It also describes the deployment of composite network services and allows policy-based approach to the provision of these services [19]. The NSC goal is to increase efficiency and flexibility for future operators networks [5] and allowing optimal services chaining without requiring the current high operating costs.

2.2.1 Fundamental concepts

NSC has associated concepts such as service function, service instance, and service node, which the authors of Software defined network service chaining [14] define, respectively, as follows: it is an abstract way to treat network packets and which has associated a service instance for the network deployment; it is a hardware or software operational derivation of the associated service function. Service instance is deployed in the network and it delivers specific handling of a service function to the network packets; and, it that provides a run-time environment and allows linking service instances on the network.

2.2.2 NSC types

Nowadays, there are three solutions types to address the NSC challenge which are: static, semi-automatic, and automatic. Static NSC refers to the NSC used in traditional computer networks (the most used today) and it essentially is the deployment of network services using specialized hardware and predetermined service chains. Therefore, it is necessary to acquire specialized hardware for the deployment of a new network service. In addition, network administrators must have extensive knowledge in computer networks, protocol management and installation of a wide variety of network devices (*e.g.* switch, router, firewall).

The semi-automatic NSC uses the advantages of virtualization, *i.e.*, it uses concepts related to SDN/NFV ecosystem. Moreover, it NSC type requires human intervention, network administrator, to a lesser extent than static. This undergraduate proposal argues the possibility of implementing semi-automatic NSC with the use of mashups technology. The mashups allow performing NSC through an unique Web application which decrement the use and the complexity level in the installation of new hardware. In addition, the user has more customization options in the selection of network services without requiring high levels of knowledge in computer networks.

The automatic NSC refers to the use of algorithms that perform a complete automation of the NSC. For it case, human intervention is not necessary, which shows the few personalization options. In addition, to make some changes in the NSC, it is necessary to modify the algorithm in use.

In figure 2.2, we can see the reduction of the travel of network traffic by using semi-automatic and automatic NSC. The continuos line indicates the route from Customer Premises Equipment (CPE) to a Data Center. It route is the one that performs network traffic passing through the different middleboxes in static NSC. The dotted black line shows the way to go using automatic NSC. Also, the solid green line indicates the path traveled by the network traffic using semi-automatic NSC, where it is emphasized that selecting the network services is a function of the network administrator.

2.3 Mashups

In the recent years there is an important trend focused on facilitating, to the end users, the use of any of the services available through the Web. Therefore, a technology know as mashups emerged, which are composite Web applications focused on the end users and created by the combination of different resources available through Internet [7]. Focused on the end users refers to that the mashups can be development by users who do not normally have advanced programming skills.

The fundamental basis is in the cooperation between the end users and the existing

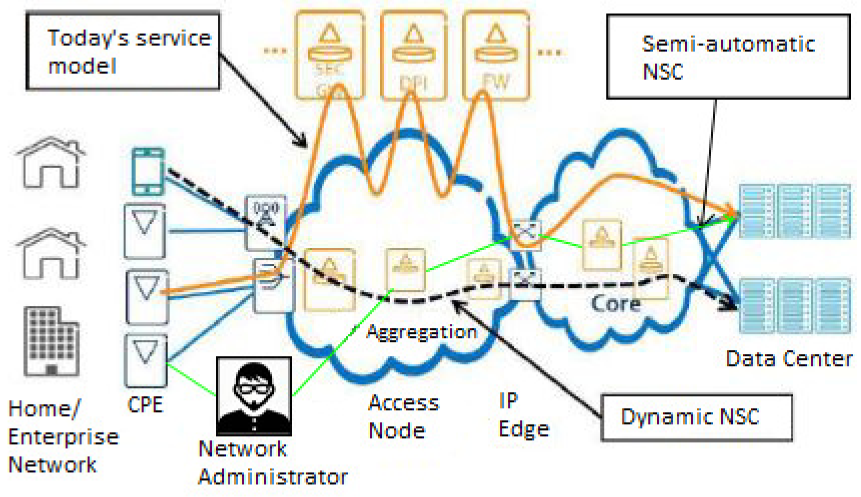


Figure 2.2: NSC types [1].

Web applications allowing users to create their own custom applications according to their needs. It technology is characterized mainly by a simple composition model, which allows easy and quick customization, in addition users can perform the direct execution of their own applications.

2.3.1 Fundamental concepts

A mashup is a web-based network resource that is composed of resources of the existing services, either content, data or functionality of an application. It has the purpose of creating or adapting robust applications from existing resources in business environments [9]. On the Internet there are many Web Services available with different QoS that provide the same solution to a specific task. Thus, an adequate selection of the services to be implemented is necessary and, therefore, the mashups-based services require an optimal set of services for the construction of a compound service and look the best QoS based on the requirements of the users and the needs of the resources.

Advocates of the use of mashups for the convergent services creation promote the mechanisms generation of services composition using friendly interfaces. It is done following the mashups Web philosophy based on that a large number of services are accessible through open interfaces on the Internet.

Approach	Automation	Required knowledge	Difficulty implementing new service
Static	None	High	Half
Semi-automatic	Half	Low	Low
Dinamyc	High	Low	Low

Table 2.1: Comparison of convergent service approaches [1].

2.3.2 Mashups in Networking

The mashups technology allows creating composite web applications by combining resources available in public or private clouds, in local repositories, and in general over the Internet [20]. The main feature of mashups is the re-use of preexisting applications and cooperation between end-users. The mashups allow end-users, who do not have advanced knowledge in programming, to create their own custom applications according to their needs. An important fact that has increased the possibility of using mashups is the quantity and variety of Application Programming Interfaces (APIs) available online and this allows the creation of more dynamic and flexible applications supporting the creation of mashups [8].

The mashup technology is supported by a simple composition model that allows fast and easy development and running of custom applications. The creation of mashups is carried out by a mashup system that allows their storing and execution. Mashup systems apply the resource abstraction [20] that is the ability to hide technical details of underlying resources for end-users; which is possible by using APIs that provides a complete description of the service interfaces including operation names, parameters and data types [21]. In addition, mashup systems support the re-use of existing composite applications to promote the creation of more sophisticated applications.

Nowadays, mashups have been used in different domains such as meteorology, telecommunications, and networking. Regarding networking there are different works related to the improvement of QoS [9], monitoring of virtual nodes [8], solution of situations of network management [22]. But, to the best of our knowledge, the semi-automatic NSC in NFV using mashups technology has not been addressed. Therefore, our work focuses on proposing a new type of mashups-based NSC. Our proposal defines a new NSC type, semi-automatic NSC, which is supported in a mashups-based architecture that allows the construction of vNSs. Mashups is the essence of our proposal because it allows encapsulating the complexity of the underlying network infrastructure, the access interfaces to VNFs and, furthermore, allows the vNSs composition regardless of the network topology and with a high flexibility. The semi-automatic NSC allows the network administrator to interact with the process of creating the vNSs. Furthermore, this work allows the vNSs association from different providers, *i.e.*, the heterogeneity of each vNS is not a constraint to create a service chain.

Chapter 3

Related Works

In this chapter we present the closest works that were found in the review of the state of the art regarding our approach, NSC and the use of mashups in similar environments. In the first section, we describes the related works to NSC. In the second section, we presents the related works to mashups.

3.1 Network Service Chaining

In the works consulted, there are different proposals that address the network service chaining, however, to the best of our knowledge, there are no proposals that address the semi-automatic NSC using mashups. Some of the most relevant proposals are described below:

The work "Enabling network function via service chain" [4] propose a service chain instantiation framework based on SDN/NFV. Their work introduces a new concept called Atomic Function (AF). An AF (*e.g.* WAN optimizers, Content Delivery Network and NAT-Protocol Translator) performs specific handling of network packets, and one or more AFs compose a service chain. AFs only expose the public characteristics (*e.g.* ID, name, type, action) of NFs. The authors implement a proof of concept, called MatchMaker, demonstrating that, with different service policies, the throughput of service chain is improved.

The work “Network service chaining with optimized network function embedding supporting service decompositions” [5] proposes two algorithms to map the NSs chains on the network infrastructure and enable the decomposition of NFs. The first algorithm is based on Integer Linear Programming and focuses on minimizing the cost of mapping VNFs over network infrastructure based on NS requirements and infrastructure capabilities. The authors explain that advantages of services decomposition enable the reduction of Capital Expenditures (CAPEX) and Operating Expenses (OPEX). The second algorithm is heuristic and resolves the scalability issue of the first one. A simulation is presented comparing heuristic-based approach and ILP-based algorithm, this simulation is measured in terms of acceptance ratio and cost/revenue value. As a result, more services can be mapped over time and the service acceptance ratio increases in the long run.

Piecing Together the NFV Provisioning Puzzle: Efficient Placement and Chaining of Virtual Network Functions [6] proposes an Integer Linear Programming model to solve the network functions placement and chaining problem that causes end-to-end delays because of misallocation of resources in traditional networks. The authors obtain a 25% reduction in end-to-end delays compared to chaining models on traditional networks. The model is evaluated by the processing time for three topological components: Line, Bifurcated path with different endpoints, and Bifurcated path with a single endpoint.

The proposal “Network service chaining with efficient network function mapping based on service decompositions” [3] defines an algorithm to help network operators to: *(i)* minimize CAPEX and OPEX by reducing the infrastructure resources used in the mapping of a network service request; and *(ii)* solve the Virtual Network Embedding Problem by decomposition of abstract NFs into several NFs interconnected. The proposed scheme increases the acceptance ratio significantly while decreasing the mapping cost in the long run.

“A QoS aware services mashup model for cloud computing applications” [9], defines a model to efficiently solve the problem of selecting service routes that do not satisfy QoS constraints. The authors propose the QoS Aware Service Mashup model that facilitates the service instances selection and data flow paths to satisfy QoS requirements. A heuristic algorithm is designed to find the service paths to route the data flow. A simulation of large-scale cloud computing system with 104 service providers demonstrates that QoS Aware Service Mashup meets the desired QoS requirements unlike fixed and random algorithms.

The proposal "Service function chaining simplified" [23] propose an optimization model to solve the throughput limitations of VNFs without worrying about the network infrastructure. Furthermore, such a model that solves the lacks related to VNFs such as placement and deployment. The model is implemented using Mixer Integer Programming in CPLEX. Furthermore, the authors propose a heuristic solution, called Kariz, that achieves competitive acceptance ratio of 80-100%.

The authors of "On orchestrating virtual network functions" [24] identify the VNF Orchestration Problem (VNFOP), demonstrating that dynamic VNF orchestration reduces OPEX. The problem is formulated as an Integer Linear Programming and implemented in CPLEX to find optimal solutions. Furthermore, the authors propose a heuristic algorithm to solve larger-scale problems of VNFOP. This algorithm provide solutions that are 30% better than the optimal solution.

Papers	**Proposal**			**Concepts used**		
	Dynamic NSC	**Networking**	**Semi-automatic NSC**	**SDN**	**NFV**	**Mashup**
[4]	$\surd$			$\surd$	$\surd$	
[5]	$\surd$			$\surd$	$\surd$	
[6]	$\surd$			$\surd$	$\surd$	
[3]	$\surd$			$\surd$	$\surd$	
[23]	$\surd$				$\surd$	
[24]	$\surd$				$\surd$	
[9]	$\surd$					$\surd$
This work		$\surd$	$\surd$	$\surd$	$\surd$	$\surd$

Table 3.1: Related work

Table 3.1 summarizes the cited articles approach regarding dynamic NSC, including the concepts used. Table 3.1 highlights some important aspects such as: (i) in the literature reviewed there are no papers focused on semi-automatic NSC, (ii) NSC is highlighted as an important concept in the implementation of the SDN / NFV ecosystem, and (iii) a related work uses the mashup approach to solving QoS issues in dynamic NSC in cloud environments.

3.2 Mashups for NSC

In previous works carried out by Telematics Engineering Group of the University of Cauca (GIT) and Computer Networks Group of the Federal University of Rio Grande do Sul (CNG) several solutions have been proposed for the network management using mashups. However, such solutions do not include concepts of NFV or NSC. Such solutions are briefly described below.

A mashup ecosystem for network management situations [7] proposes a new ecosystem of mashups that allows network administrators to perform all the activities and interactions necessary to provide a mashment from an ecosystem. A mashment is mashup focused on solving network management situations.

The authors of "Monitoring virtual nodes using Mashup" demonstrate that the mashups technology can be used to perform the integrated monitoring of heterogeneous virtual nodes. Such demonstration is performed with the implementation of a mashup-based architecture that corroborates low traffic generation and low response times.

The proposal "A Mashup-based Approach for Virtual SDN Management" [10] defines a novel mashups-based approach for network management. Such approach allows network administrators to create their own network management solutions based on the requirements of the different devices that make up an SDN. The authors present a prototype for the evaluation of the model confirming the low response time in the construction of mashups.

The authors of "Rich dynamic mashments: An approach for network management based on mashups and situation management" [11] introduce an architecture named Rich Dynamic Mashments in order to facilitate the daily work of network administrators when faced with unexpected, dynamic and heterogeneous situations. Such architecture makes use of the discipline Situation Management and technology mashups.

Papers	**Proposal**			**Concepts used**		
	Dynamic NSC	**Networking**	**Semi-automatic NSC**	**SDN**	**NFV**	**Mashup**
[7]		√		√		√
[8]		√				√
[10]		√		√		√
[11]		√		√		√
This work		√	√	√	√	√

Table 3.2: Related work

Table 3.2 summarizes the approach of the cited articles regarding Mashup. Table 3.2 highlights aspects such as: (i) the mashup technology to be implemented in the semi-automatic NSC supports its validity and use in the development of networking jobs, and (ii) related work using the mashup technology does not refer to concepts of NFV and NSC.

Chapter 4

An Architecture for Semi-Automatic Network Service Chaining by using Mashups Technology in NFV

The process of creating, configuring, and executing the chains is specified in an mashups-based architecture that follows a layered pattern. Such an architecture is selected because it allows defining the elements, components, and functionalities needed in each of the layers. In addition, the interfaces required for communication between the layers are described. In this way, our architecture is the fundamental and essential contribution in the description of the semi-automatic NSC.

On the other hand, our architecture is based on the architecture proposed by the works [8] [11]. Moreover, this work generates a new contribution to the topics developed in GIT and CNG because previous works use mashups technology but no one focus on the semi-automatic NSC.

In this chapter, initially, we describe motivating scenarios. Later, we present an overview of our work. Finally, we introduce our mashups-based architecture, its layers and elements.

4.1 Motivating Scenarios

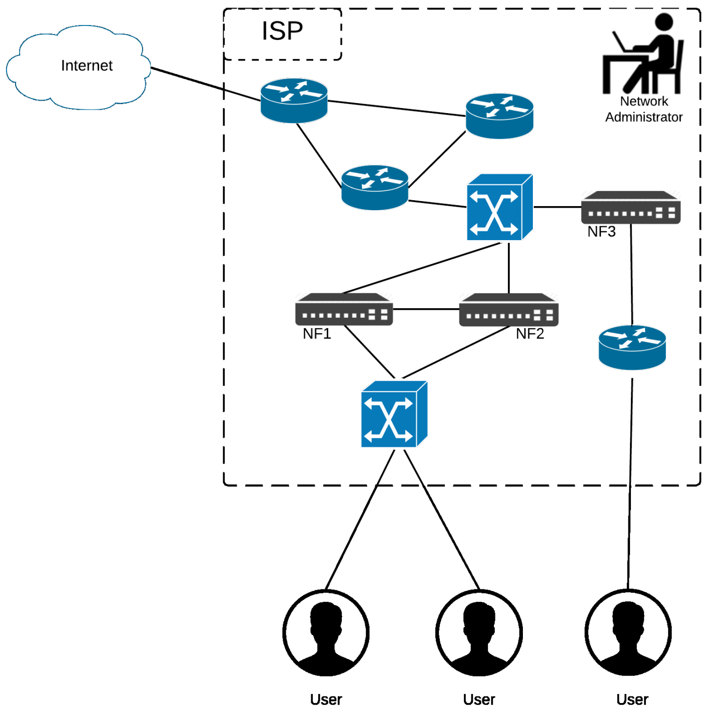


Figure 4.1: Scenario 1

First, we consider a network from an Internet Service Provider (see Figure 4.1) that provides NSs to different users. The network is composed of different intermediary network devices, network functions running on specialized hardware. Implementing an NS requires infrastructure changes such as installing new middleboxes, setting up new intermediary devices, and redirecting network flow. For example, to install a Firewall, a Network Administrator must connect the wiring of incoming and outgoing network traffic flow, which implies changes in the infrastructure. Furthermore, Service Level Agreements (SLA) established with users are changing according to the type of user. Therefore, the Network Administrator in charge of providing the different types of NSs to the users must establish what type of additional middleboxes are required to comply with the SLA and carry out a plan of implementation and deployment of the new NSs. In this way, CAPEX and OPEX are increased to the Internet Service Provider because the acquisition of new devices and the personnel required for implementation. Instead, our proposal facilitates the chains deployment without acquiring additional hardware, in a short time, and with greater flexibility.

We consider a second scenario where a Network Administrator must manage a network based on an SDN / NFV ecosystem and which is composed of a large number of devices. The Network Administrator uses different platforms for network management, and in the case of network services chaining are used automated algorithms. The algorithms are responsible for creating network services chains based on user requests, and taking into account algorithm constraints. For instance, to execute a chain, the algorithm is in charge of selecting the VNFs and correlating their inputs and outputs. In this way, NSC process is completely automatic and isolated from the decisions of the Network Administrator. Thus, to make a change in the service chain or in a particular network function it is necessary to access the algorithm and make the modifications. Therefore, the network administrator should have high knowledge in software development and knowledge of various programming languages. In this way, the customization of a service chain becomes a tedious task which increases the workload of the Network Administrator. Instead, our proposal increases the interaction of the Network Administrator with the NSC process and improves the customization options on the chains.

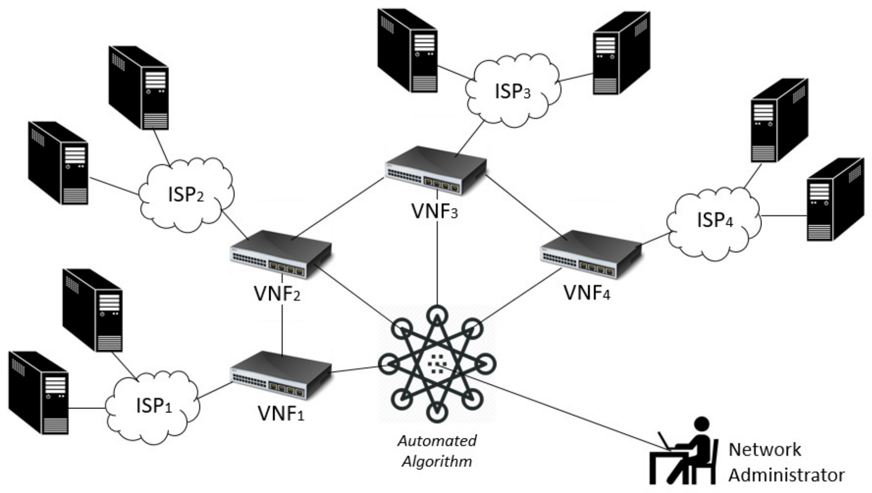


Figure 4.2: Scenario 2

Based on the previous scenarios, we have observed some shortcomings concerning the NSC in current NSC types. For instance, some shortcomings in a traditional environment (static NSC) are:

- Topological dependence of the network which reduces the flexibility in the new NSs deployment.
- Buying and configure hardware devices for the implementation of new NSs.
- Network Administrator must generate a plan for deploying and mounting the new hardware devices purchased. In addition, the network administrator must adapt to the proprietary interface that each network function has. This results in the increase of CAPEX and OPEX.

Some shortcomings in an automated environment (dynamic NSC) are:

- Network Administrator must possess advanced knowledge in networking, programming languages, and in the SDN / NFV ecosystem.
- Low customization of the chains.
- Low control of Network Administrator for advanced configuration.

To address the above shortcomings, our work uses the mashups technology and incorporates the semi-automatic NSC concept, offering:

- Creating NSs in a more efficient and less time-consuming way.
- More customization options applicable to the service chains or in a particular network function. In addition, it allows the visualization of the running service chains.
- Statistics on the performance of each network function, *e.g.* the rejected packet rate of a firewall.

4.2 Overview

Current NSC approaches do not provide Network Administrators with optimal customization options for configuring NSs. Such approaches tend to automate NSC process but limiting interaction with Network Administrators. Although, the mashups provide basis for creating composite web solutions, the mashups have not been still used for NSC. Hereinafter, we expose how semi-automatic NSC using mashups provides Network Administrators with a new chaining concept that allows meeting the end-user (*e.g.* Internet Service Provider, Application Service Provider) requirements considering the heterogeneity of network services and resources.

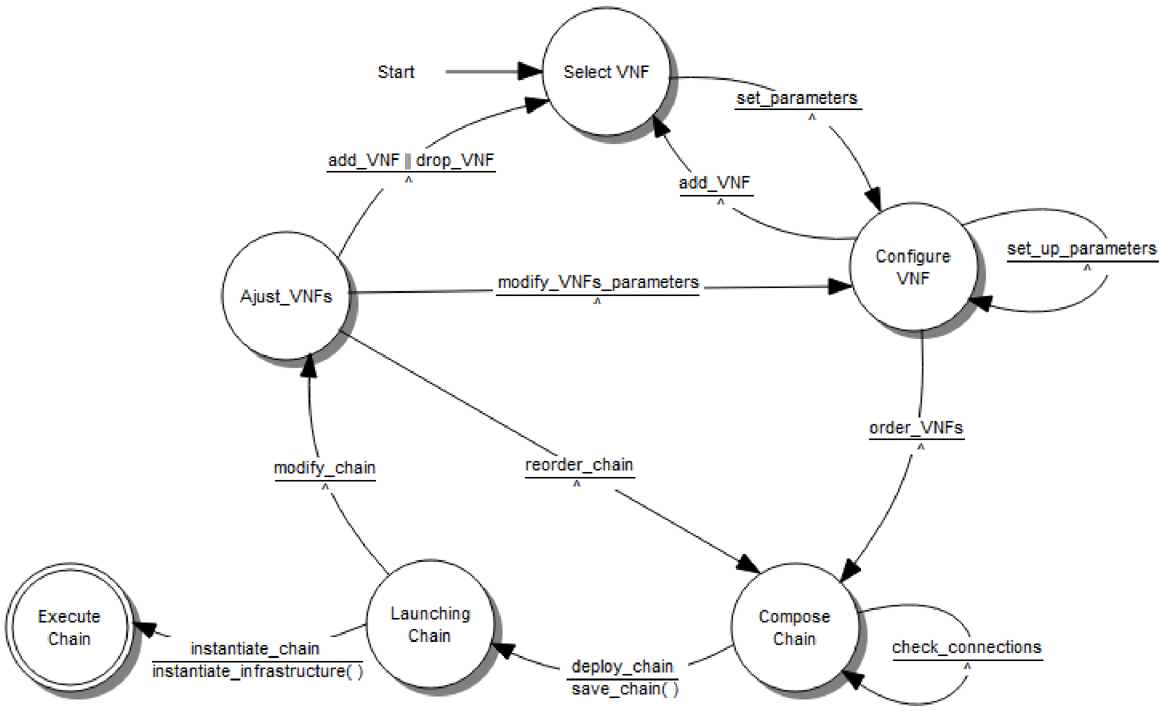


Figure 4.3: Overview

We use a Mealy Finite State Machine (FSM) to illustrate the semi-automatic NSC process. A Mealy FSM is a deterministic machine that produces outputs on their state transitions after receiving inputs [25]. Figure 4.3 presents FSM to illustrate each semi-automatic NSC process stage, the events that cause the change of state, and the actions performed at each transition. All events except `set up parameters` and `instantiate chain` are caused by the Network Administrator, and actions are performed by the system. Our FSM has six states and starts with the `Select VNF` state in which each VNF is selected to create the chain (the number of VNFs depends on end-user needs). This selection can include chains already created and ready to be used. The event causing the transition to the second state is named as `set parameters`, and no action is performed which is symbolized with ^.

The second state is `Configure VNF` in which the basic parameters are setted to selected VNFs for their proper functioning. Thus, each time a VNF is added, it is mandatory to set the basic parameters to change to the next state, and no action is performed.

In the third state, `Compose Chain`, the selected VNFs are chained between them based on end-user needs. This state allows correlating the inputs and outputs of VNFs that form the chain. The event causing the transition to the four state is `deploy Chain`, and the action executed is `save chain` which allow storing, in a local repository, the chain that is represented as a data which contains the information of the used VNFs and their setted parameters.

The fourth state is `Launching Chain`. This state is to request the execution of the chain that has already been stored. The event that caused the transition to the fifth state is `instantiate chain` and the executed action is `instantiate infrastructure` which allows creating VNFs instances used in the chain on network infrastructure.

The fifth state is `Execute Chain` in which the chain is executed on the network infrastructure, *i.e.*, VNFs instances used have been created. This state of the NSC process is hidden to Network Administrators, and in this state the chaining process ends. Even so, the sixth state is added, `Adjust VNFs`, to make additional configurations when the chain has already been created and stored or even when it is running. It state has three transitions which are `add VNF || drop VNF`, `modify VNFs parameters` and `reorder chain` corresponding to the first, second and third state already described. Thus, it is possible to make any modification to the chain or to a particular VNF and update the changes in the repository. Therefore, this state allows Network Administrators to customize the chains and deploy them efficiently and dynamically.

The FSM described facilitates the understanding of the semi-automatic NSC process and specifies the main actions required for such a process, supporting the feasibility to develop our proposal compared to current types of chaining. For example, static NSC requires infrastructure changes to install new network devices for which the Network Administrator must have extensive knowledge in infrastructure, networking and network device configuration. Furthermore, static NSC increases CAPEX and OPEX because of personnel displacement for the installation process. Instead, our new chaining concept supported by mashups-based architecture allows conforming chains with low impact on network traffic, low time-consuming in creation, configuration, and launch of the chains, and with greater flexibility and usability. Moreover, changes or modifications to the chain can be made at runtime. Dynamic NSC provides very few chain configuration or modification options, furthermore, Network Administrators does not directly interact with the chaining process and the algorithm in charge of the process is the one who makes the chain conformation decisions. Instead, our proposal provides a high degree of interaction between the Network Administrator and the NSC process with greater process control and better chain configuration and customization options. Furthermore, the chaining decisions are made by the Network Administrators.

4.3 Architectural Layers and Elements

Figure 4.4 introduces our mashups-based architecture that focuses on carrying out semi-automatic NSC considering the described FSM. The proposed architecture follows a layered architectural pattern [26] in which the lower layer provides services to the upper layer. Our architecture is formed by three layers: Presentation, Chains Composition, and Virtualization. In a broad sense, chains are displayed and launched in the Presentation Layer. The Chains Composition Layer supports the building of chains by combining resources of the Virtualization Layer which provides a physical computational resources abstraction generating a virtualization environment for chains execution. The actors involved in the semi-automatic NSC, the layers and their components are described in detail in the following subsections.

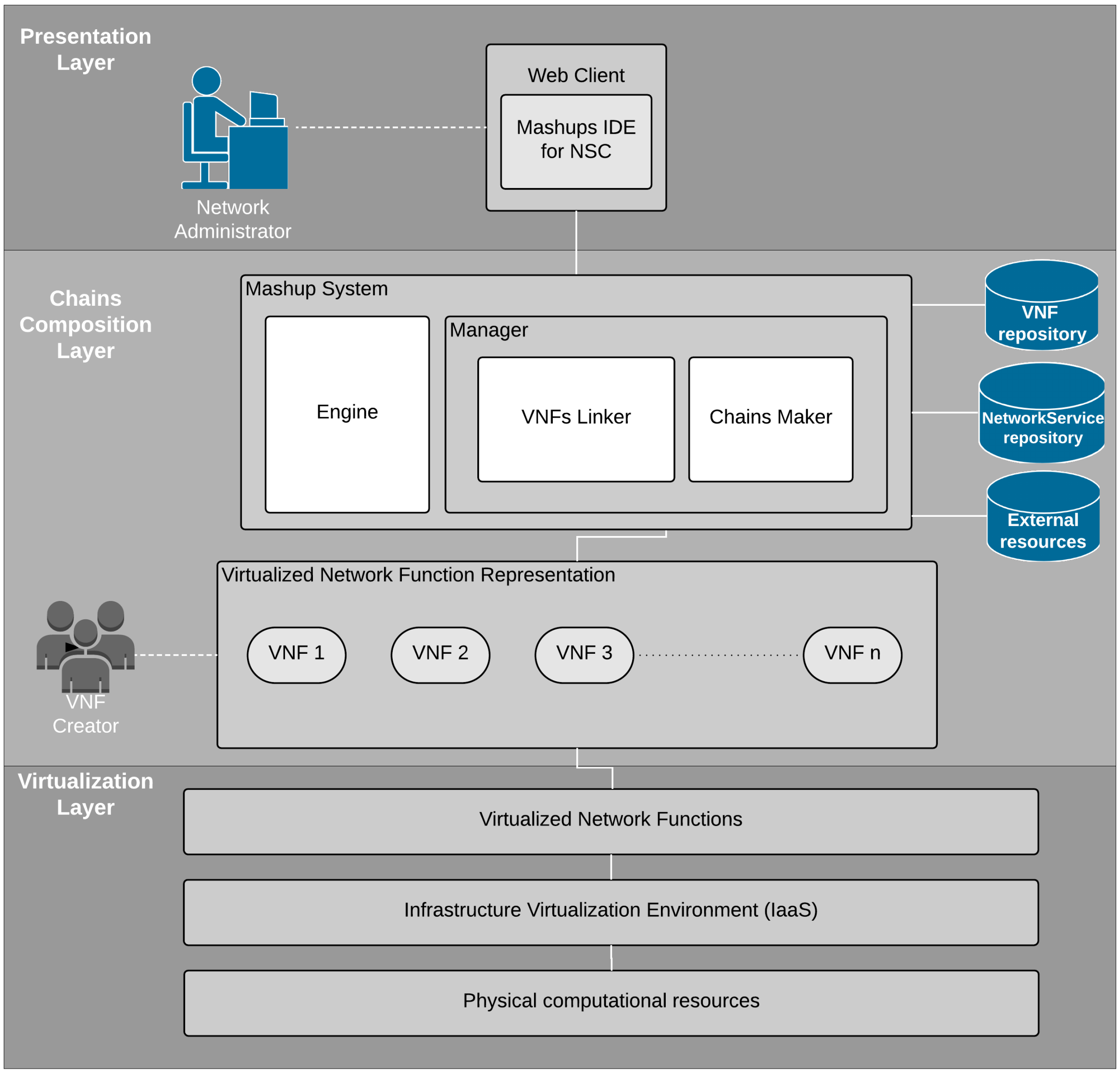


Figure 4.4: Mashups-based architecture for semi-automatic NSC

4.3.1 Actors

There are two actors involved in the semi-automatic NSC process: Network Administrator, and VNF Creator.

The *Network Administrator* is responsible for creating, configuring, and executing chains by combining resources such as: graphic elements, available VNFs, configuration elements, and stored chains. These resources may be external or internal. A resource is external if it is located in a third-party, for example, VNFs, the libraries used for the visual interface. An internal resource refers to VNFs stored in a local repository and that are available for creating chains. For the chains creation, Network Administrators do not need advanced skills in Web programming because the mashups operate in a high-abstraction level. This actor is in charge of monitoring the chains that are running and, if necessary, modifying chains at runtime. Chains must be monitored to verify their optimal functioning because it is necessary to check that the configuration applied to each VNF is working according to what is necessary.

The *VNF Creator* is expected to be an Information Technology (I.T) Professional with significant knowledge about software engineering, computer networks, and virtualization of NFs. This actor is in charge of creating and publishing VNFs, *i.e.*, he/she must abstract the NFs operation that are executed on middleboxes and perform the software development of NFs in the most appropriate programming language. Furthermore, the Creator must provide interfaces by which all features provided by any VNF are exposed. The created and published VNFs are stored in a local repository. Such VNFs are used in the chain composition process. It is important to highlight there may be external VNFs whose access Uniform Resource Locators (URLs) are stored in a local repository. Furthermore, this actor determines the hardware resources (*e.g.* processing cores, RAM, disk space) required for optimum VNFs operation.

4.3.2 Presentation Layer

Figure 4.5 depicts the Presentation Layer that allows building and launching chains by a Mashups Integrated Development Environment that runs over a Web Client. Our Mashup IDE is based on visual and drag-and-drop mechanisms with which the Network Administrator combines different interaction elements to select, create, configure, launch and execute chains. The proposed Mashup IDE encapsulates the entire semi-automatic NSC process, *i.e.*, it is the only way the Network Administrator interacts to create chains.

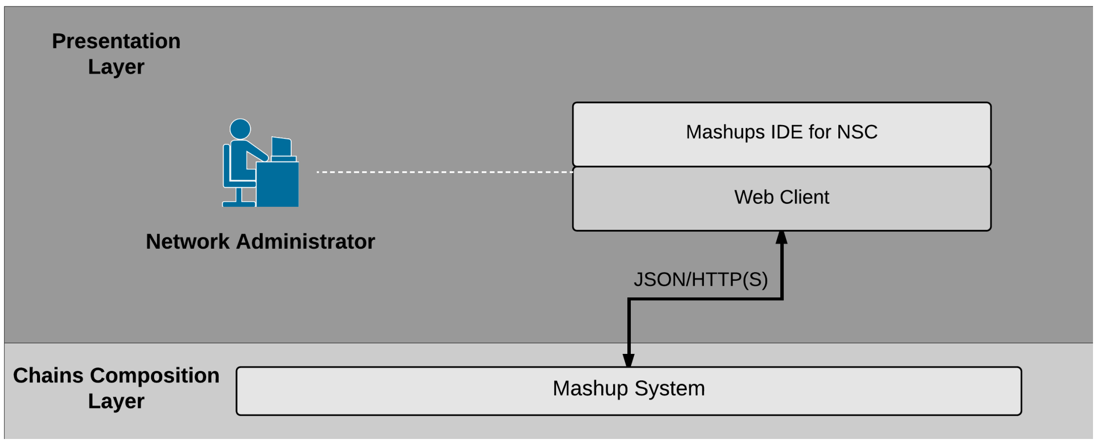


Figure 4.5: Presentation Layer

The Web Client is formed by a Mashups Runtime Environment represented by browsers, which are software entities in charge of presenting the Graphical User Interfaces (GUIs) of the Mashups IDE to Network Administrators, and allow launching chains. GUIs of Mashup IDE are: GUI for creating chains, and GUI for configuring VNFs that compose the chains. For example, GUI for creating chains refers to the section in which the VNFs are located to compose the chains. The other GUI has configuration elements according to each VNF, for example, the GUI for configuring a Firewall has options to allow network traffic by indicating the source and destination IP addresses. GUI for configuring a LoadBalancer has options to define the virtual IP address and select the servers to do the load balancing. These components provide the Network Administrators with the necessary resources and concepts to create, launch, and customize chains. It is important to highlight that the Presentation Layer communicates with the Chains Composition Layer via JSON / HTTP (S). In this communication, request / response messages are exchanged in which information is sent regarding VNFs used, their parameters, and the order of execution of VNFs. The information sent in request / response messages is described in the following subsection.

4.3.3 Chains Composition Layer

The Chains Composition Layer (see Figure 4.6) is composed of a Mashup System, a set of VNF representations, VNFs repository, a Network Service repository, and an External Resources repository. The Mashup System is formed by the Engine and the Manager which contains two components that are VNFs Linker, and Chains Maker. In a broad sense, the Manager coordinates the invocation of VNF representations used in the composed chain and, furthermore, it guarantees the correct order of VNFs execution and instantiation on the Virtualization Layer. Specifically, the Engine is the lifecycle manager of the Mashup System and the chain as a whole. Thus, when the Presentation Layer sends a request message to run a chain, the Engine invokes the Manager. The Manager instructs the Engine for using VNFs Linker to ensure that VNFs instantiation order is the requested in the Presentation Layer.

The Chains Maker communicates with the VNFs Linker to generate the VNFs instantiation on the Virtualization Layer, that instantiation must be in the order requested by the Presentation Layer. The Chains Maker uses each VNF representation to perform the instantiation over network infrastructure.

The VNF Repository contains metadata about the VNFs available to compose the chain. This metadata is structured by means of the JavaScript Object Notation (JSON) as follows [{*IDVNF : id,NAME : name,PAR : parameters*}].

Where, *IDVNF, NAME* and *PAR* represent the identifier, the name and the functioning parameters of a VNF. Capital and lowercase letters refer to names and values of JSON object properties, respectively. For example, a Firewall (see Figure 4.7) would be structured in JSON notation as [{*IDVNF : 1,NAME : Firewall,PAR : param*}].

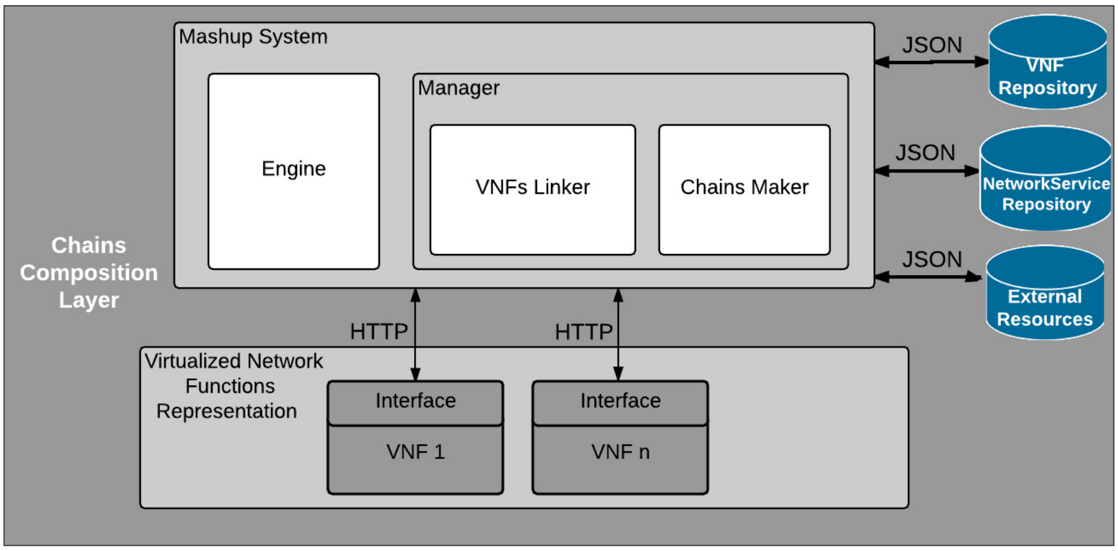
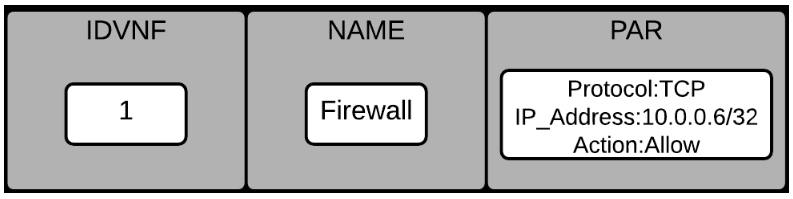


Figure 4.6: Chains Composition Layer



IDVNF	NAME	PAR
1	Firewall	Protocol:TCP IP_Address:10.0.0.6/32 Action:Allow

Figure 4.7: JSON example of a Firewall

The PAR parameter indicates the initial default values (*i.e.* in this case firewall rules) that the firewall contain, for example, network flow protocol, host IP-Address to which the rule applies, and the action, *i.e.*, if the network flow is accepted or rejected. In case of no default values the PAR field will be empty and will be modified when the firewall is added to a chain.

The Chains repository stores metadata that describes a chain that have been created, *i.e.*, VNFs that are used, execution order of VNFs, and the parameters of each VNF. This metadata is structured by means of JSON (see Figure 4.8) as follows $[\{IDSERVICE : id, NAME : name, DESCRIPTION : desc, CONTENT : \{\{IDVNF_1 : id_1, ORDER_1 : order_1\}, ..., \{IDVNF_n : id_n, ORDER_n : order_n\}\}, SIZE : size\}]$.

Where, *IDSERVICE, NAME, DESCRIPTION, CONTENT* and *SIZE* represent the identifier, the name, the network service description, the content (*i.e.* VNFs that compose a chain), and the chain size (*i.e.* the number of VNFs that compose a chain). The content parameter is composed by the IDs of VNFs that compose a chain and by the order or position in which they are executed.

The External Resource repository stores the URLs of third-party VNFs. Each VNF contains the fields previously described. Network Administrators evaluate the chain creation request to determine whether third-party VNFs or local VNFs are used in the chain creation. It is important to highlight that the use of local VNFs has priority because it generates less delay in the network flow.

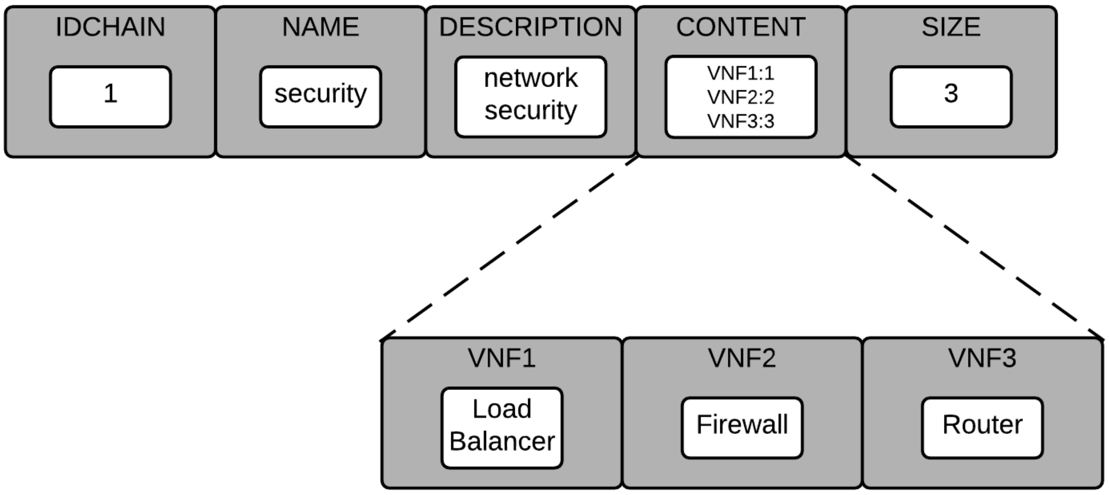


Figure 4.8: JSON example of a chain

As previously mentioned, the main function of the Chains Composition Layer is to interpret the actions performed in the Presentation Layer and execute the necessary actions to instantiate the chains and the parameters defined on each VNF on the Virtualization Layer. For instance, when a Network Administrator receives a request about creating a chain, the Network Administrator must do the following:

- Check if the requested chain has already been created which should be consulted in the Network Service Repository. If the chain has already been created then the Network Administrator only uses the stored chain in the Network Service Repository and launches it again from the Presentation Layer, *i.e.*, a new instance of the chain will be created on the Virtualization Layer, finishing the chain creation request. If the chain has not been created then the chain creation process continues.

- Check for the existence of the requested VNFs in the VNF Repository and External Resources Repository. If the required VNFs are not available, the chain can not be created.
- Network Administrator selects VNFs to compose the chain.
- The operating parameters are set in each VNF based on the request requirements.
- Network Administrator chains VNFs in the order needed.
- Network Administrator must launch the chain and configure additional parameters to the chain based on the operation of the chain in a production environment.

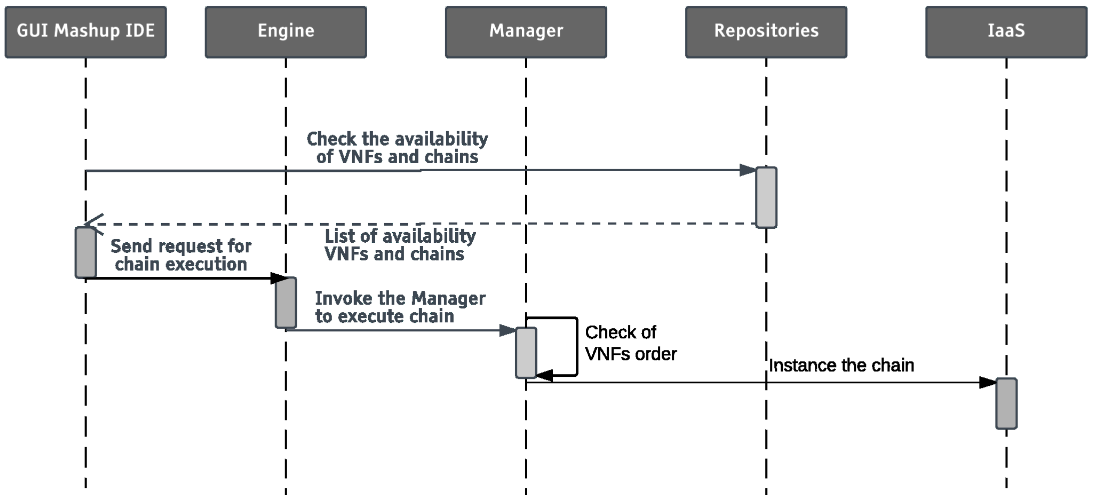

GUI Mashup IDE
Engine
Manager
Repositories
IaaS
Check the availability of VNFs and chains
List of availability VNFs and chains
Send request for chain execution
Invoke the Manager to execute chain
Check of VNFs order
Instance the chain

Figure 4.9: Sequence diagram

The sequence diagram in figure 4.9 complements the action of the Network Administrator to launch a chain. In a broad sense, Presentation Layer sends a request message to the Composition layer to execute a chain. Then, Chains Composition Layer requests the instantiation of the VNFs used in the chain to the Virtualization layer. Thus, the requested chain remains in a state of execution.

For instance, a chain named network-security (see Figure 4.8) formed by a load balancer, a firewall and a router is structured by JSON as $[\{IDCHAIN : 1, NAME : network-security, DESCRIPTION : manages-and-monitors-incoming-and-outgoing-connections, CONTENT : \{\{IDVNF_1 : 1, ORDER_1 : 1\}, \{IDVNF_2 : 2, ORDER_2 : 3\}, \{IDVNF_3 : 3, ORDER_3 : 2\}\}, SIZE : 3\}]$.

When a chain has been set and the operating parameters have been defined for each selected VNF, the JSON formed for the chain is stored in the Network Service Repository and each VNF and its parameters are stored in VNF Repository.

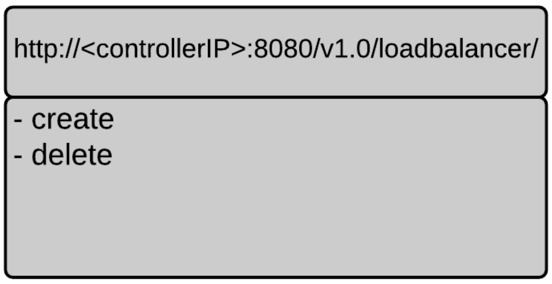


Figure 4.10: VNF representation

The VNFs are a software abstraction of NFs that run on middleboxes and they are deployed on servers in an Infrastructure-as-a-Service (IaaS) environment provided by the Virtualization Layer. In the Chains Composition layer are the VNF representations that are manipulated by the Mashup System. The VNFs are structured as services based on the Representational State Transfer (REST) Architectural Model that follows the HTTP request/response model. Every VNF (see figure 4.10) is represented by Uniform Resource Identifiers (URIs). These URIs are obtained through HTTP(S) request such as GET and POST which obtain HTTP(S) responses. For instance, the URI *http://<controllerIP>:8080/loadbalancer/* allows to perform a load balancing (create), or eliminate the existing load balancing (delete). One of the advantages of our proposal is to store chains in the Network Service Repository to which Network Administrators has access, this facilitates the deployment of a new instance of stored chain increasing the flexibility and decreasing the time-consuming.

To sum up, the Chains Composition layer is the core of the semi-automatic NSC process because it is the interpreter between the visual components (Presentation

Layer) and the instantiation and execution of the chains (Virtualization Layer). In other words, the Chains Composition Layer encapsulates and hides the technical details of the semi-automatic NSC process, which increases flexibility in the chains deployment.

4.3.4 Virtualization Layer

Figure 4.11 introduces the Virtualization Layer that provides the infrastructure to deploy the chains. Virtualization Layer is composed of VNFs, Infrastructure Virtualization Environment (IaaS), and hardware resources. Hardware resources refers to computers that act as servers to deploy the IaaS.

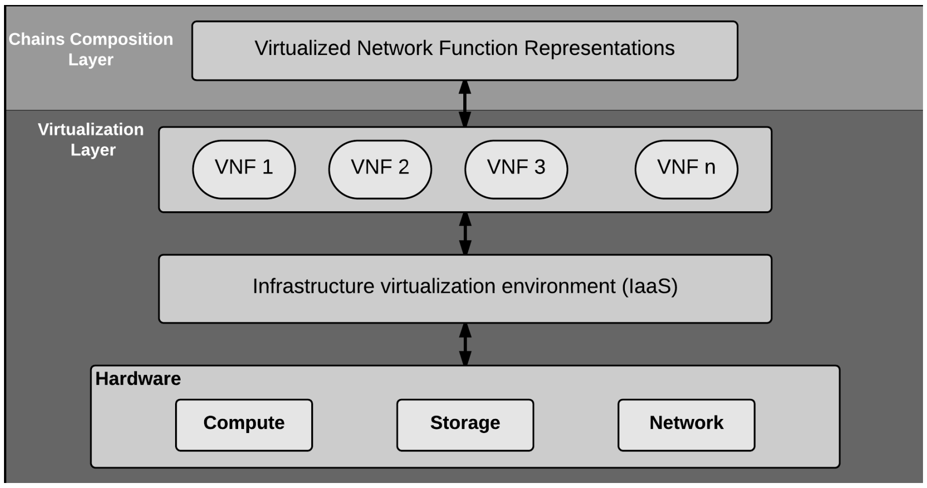


Figure 4.11: Virtualization Layer

The Infrastructure Virtualization Environment (IaaS) provides, through the concept of resource abstraction, the means for VNFs to be deployed. VNFs refers to the software abstraction of NFs that run on the current middleboxes. Therefore, our proposal focuses on the NFV concept that provides virtual resources. Such virtual resources are provided by a Virtualization Layer that separates virtual resources from physical resources. The main advantage of NFV is that allows using commodity hardware to executed VNFs reducing implementation costs. Currently, there are different Open Source and licensed tools that allow creating the Virtualization Layer. For example, OpenStack, Cloudstack, and Nebula are open source clouds to create private clouds. Amazon, Microsoft Azure, and VMware are public clouds that provide the abstraction of hardware resources.

Chapter 5

Proof-of-concept

To evaluate the effectiveness and feasibility of our proposed approach, we performed the following:

(1) We implemented a prototype called NEMASOF that is an instance of the architecture described in chapter 4.

(2) We built a test environment.

(3) We performed a quantitative and qualitative evaluation of NEMASOF. The quantitative evaluation was carried out in terms of time-consuming, time-response, and network traffic [11].

- The time-consuming is the time the Network Administrator needs to launch a chain.
- The time-response refers to the time the system takes to respond to a chain execution request.
- Network traffic refers to the amount of load that is generated in the network when using NEMASOF.

The qualitative evaluation was carried out in terms of flexibility, extensibility, and usability.

- The flexibility [8] refers to our approach allowing the Network Administrator to launch chains in a matter of seconds.
- The extensibility [8] refers to the ability to integrate own or third-party resources.
- Usability [27] refers to the facility offered by NEMASOF to display chains.

5.1 Prototype

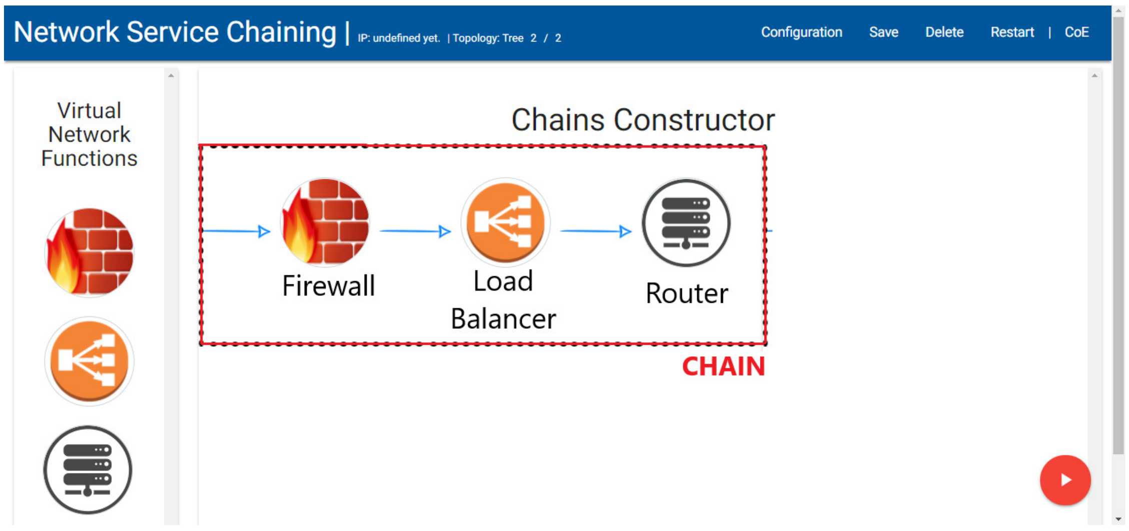

Network Service Chaining |
IP: undefined yet. | Topology: Tree 2 / 2
Configuration
Save
Delete
Restart
CoE
Virtual Network Functions
Chains Constructor
Firewall
Load Balancer
Router
CHAIN

Figure 5.1: NEMASOF

Figure 5.1 depicts the GUI of the semi-automatic Network Service Chaining Prototype (NEMASOF) that is formed by VNF representations, Buttons (*Configuration, Save, Delete, Restart, Chains on Execution (CoE),* and *Run (Play Symbol)*), and Chains Constructor. The Visual Resources, Buttons and Chains Constructor are Web elements implemented using Cascading Style Sheets (CSS) and JavaScript. Some VNF representations implemented are Firewall, Load Balancer, and Router, which are the visual representation (*i.e.* a high-level abstraction) of VNFs.

A drag-and-drop mechanism is used to compose chains, *i.e.*, VNFs and chains are dragged-and-dropped in the Chains Constructor section, where VNFs or chains are joined in the required order and it are automatically linked in that order to avoid unnecessary consumption of time and thus reduce to the maximum the time-consuming. Our prototype allows displaying statistical information of each executed VNF, for instance, in the Load Balancer case, the statistical information provided is about the packets transmitted to the servers.

NEMASOF was developed using JavaScript to implement the Mashup System, Django framework based on Python to implement the Mashups IDE, and VNFs based on REST. The necessary repositories for executing NEMASOF were implemented in MySQL database. This prototype is available on github.com/cngunicauca/NEMASOF.

5.2 Test scenario

We define a case study in which a Network Administrator must create, configure, and launch a VNFs chain consisting of a load balancer, a firewall, and a router (see figure 5.1). The chain is deployed on an emulated SDN with Mininet v2.2.1, and the Ryu controller v3.22 acting as the Manager. We created a SDN (see figure 5.2) with tree topology in Mininet defining a depth = 2 and fanout = 2. Depth refers to the number of levels that the tree topology has, and fanout is the number of outputs per switch. Therefore, emulated SDN is formed by three switches and four host. VNFs that compose the chain must have initial operating configurations: the Firewall is configured to run on all three topology switches. Three host are established as servers for load balancing with the virtual IP 10.0.0.10. The router is configured with a rule that enables the network flow between host two and three by means of a gateway with IP address 10.0.0.1.

Figure 5.3 describes the test environment for the case study. To make such an environment, first, we create a SDN. Second, we develop NEMASOF, the VNFs repository, the Network Service repository, and External Resources repository in a single MySQL database separated by tables. Third, we deploy NEMASOF and the necessary resources for the creation and release of the chain. When the Network

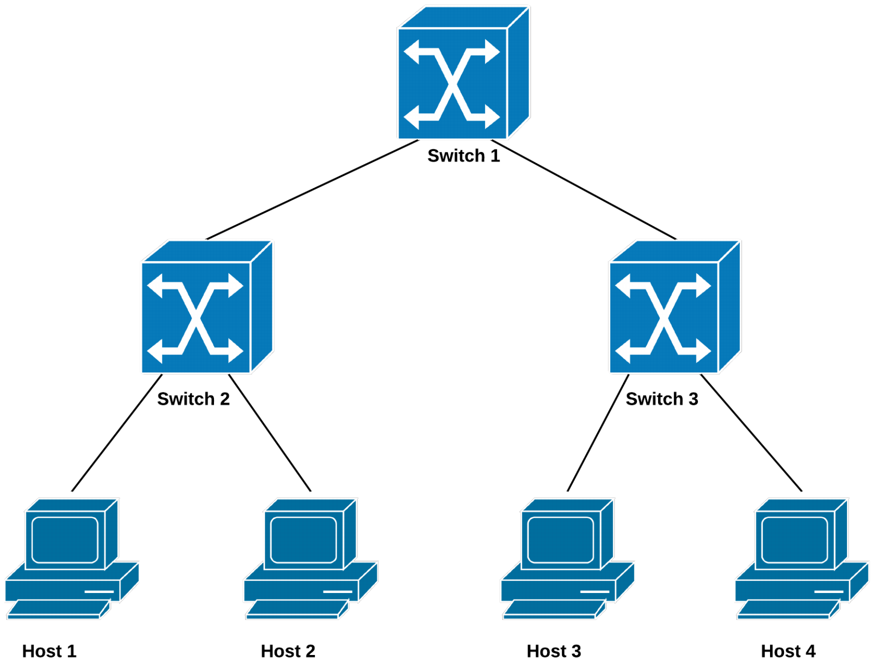


Figure 5.2: Tree topology

Administrator must deploy a chain, NEMASOF allows him to create, configure, and launch VNF chain.

Our prototype was deployed in a Cloud Computing environment. Cloud Computing is a model for enabling resources (*e.g.* network, servers, storage, applications, and services) on-demand. The model used in the NEMASOF deployment is IaaS, which refers to the provision of processing, storage, networks and other computational resources.

Microsoft Azure was used for the NEMASOF deployment and to create an SDN network. Two Virtual Machines (VM) were used, a VM to run the Ryu controller and Mininet, which is the SDN network emulation software, and a VM to deploy NEMASOF. In this sense, NEMASOF takes advantage of the features of flexibility of a Cloud Computing environment.

The SDN network was created by using Open vSwitch and a controller named Ryu. Ryu controller is based on Python programming language. The switches, hosts and controller were deployed on Mininet. Mininet was run on a VM with 4-core

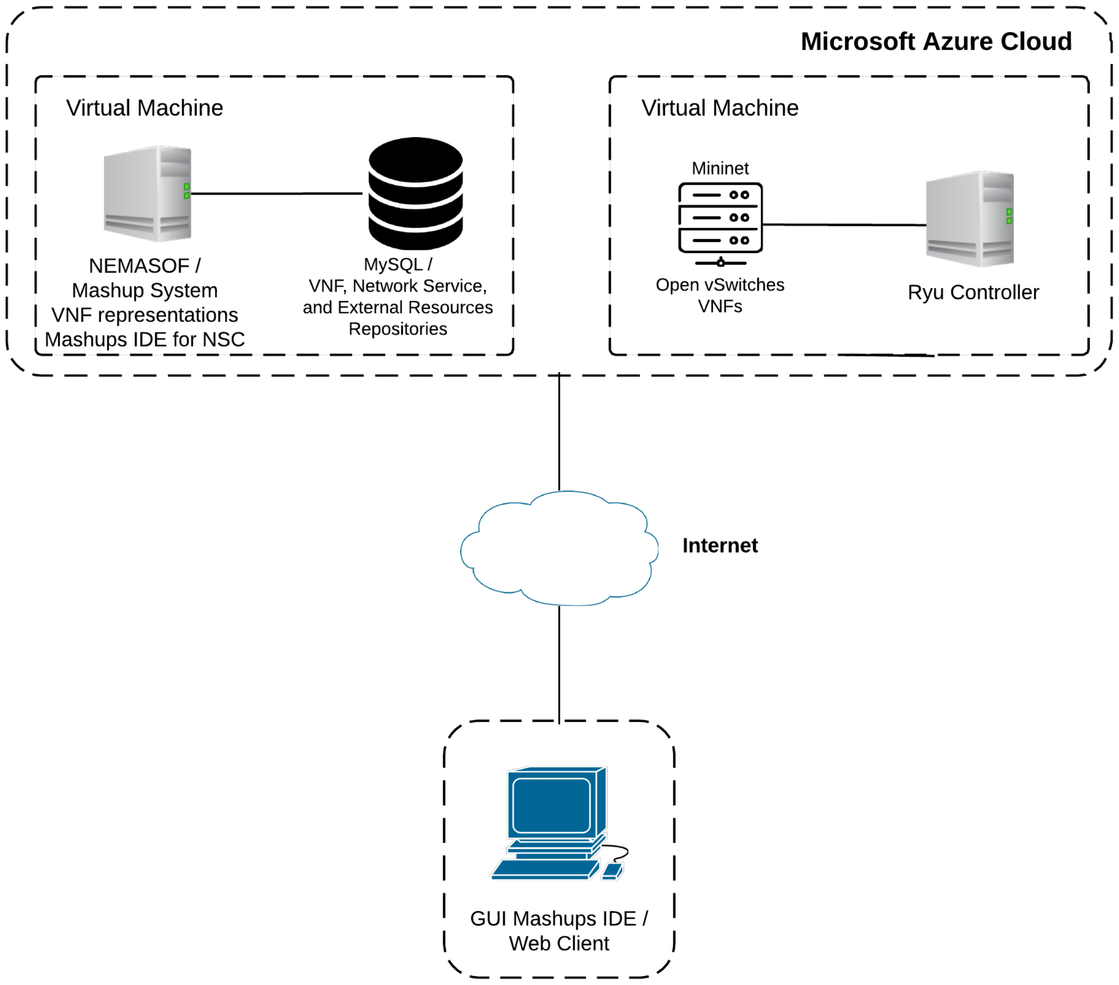

Microsoft Azure Cloud
Virtual Machine
NEMASOF / Mashup System VNF representations Mashups IDE for NSC
MySQL / VNF, Network Service, and External Resources Repositories
Virtual Machine
Mininet
Open vSwitches VNFs
Ryu Controller
Internet
GUI Mashups IDE / Web Client

Figure 5.3: Test environment

processing, 16 GBytes RAM, and 28 GBytes Solid State Disk (SSD) on the Microsoft Azure platform. NEMASOF was deployed in a VM with 2-core processing, 8 GBytes RAM, and 15 GBytes Solid State Disk (SSD) on the Microsoft Azure platform. The Test Client had i5-core processing, 8 GBytes RAM, 120 GBytes Hard Disk Drive (HDD).

The chain that the Network Administrator created in the test scenario (see Figure 5.1) is structured and stored in the Network Service repository, in a general way, as a JSON object as follows $[\{IDCHAIN : 1, NAME : network - security, DESCRIPTION : manages - and - monitors - incoming - and -$

$outgoing-connections, CONTENT : \{\{IDVNF_1 : 1, ORDER_1 : 1\}, \{IDVNF_2 : 2, ORDER_2 : 3\}, \{IDVNF_3 : 3, ORDER_3 : 2\}\}, SIZE : 3\}]$.

In turn, each VNF that forms the chain is structured as follows:

- Firewall is structured as $[\{IDVNF : 1, NAME : Firewall, PAR : \{\{SWITCH1 : on\}, \{SWITCH2 : on\}, \{SWITCH3 : on\}\}\}]$.
- Load Balancer is structured as $[\{IDVNF : 2, NAME : LoadBalancer, PAR : \{\{IP-VIRTUAL : 10.0.0.10\}, \{HOST1 : on\}, \{HOST2 : on\}, \{HOST3 : on\}\}\}]$.
- Router is structured as $[\{IDVNF : 3, NAME : Router, PAR : \{\{IP-SOURCE : 10.0.0.2\}, \{IP-DESTINATION : 10.0.0.3\}, \{IP-GATEWAY : 10.0.0.1\}\}\}]$.

The created chain was taken as the basis for the creation of chains with a large number of VNFs, in order to evaluate our proposal and determine its behavior over the network infrastructure.

5.3 Evaluation and analysis

We evaluate our proposal by the case study described in the test scenario. The evaluate steps are as follows.

(1) Theoretical and experimental time-consuming measurement to determine the time used in creating, configuring, and launching the chain.

(2) Measurement of the time-response to assess NEMASOF responsiveness to a request to execute the chain.

(3) Measurement of network traffic to assess the additional traffic generated by NEMASOF.

5.3.1 Time-consuming

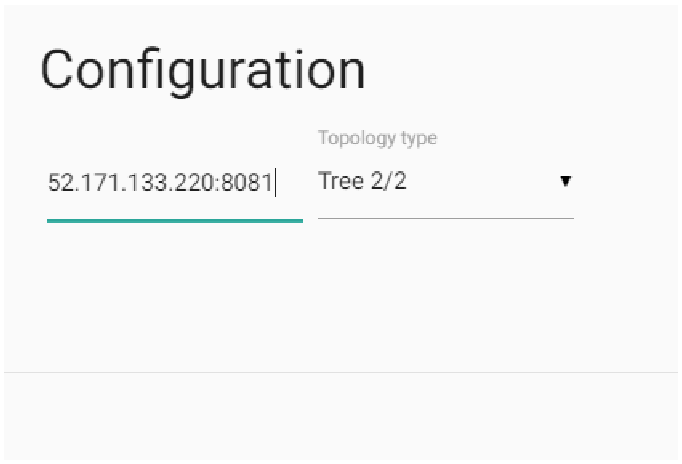
Measuring time-consuming allows demonstrating that NEMASOF is a feasible solution because of the short time spent in deploying a chain compared to other NSC types. To determine the time-consuming, we use the Keystroke Level Model (KLM) [28] because it allows to estimate the time that the Network Administrator takes to create, configure, and launch a chain using the computer mouse and keyboard. In KLM, each task is formed by a sequence of actions, and each action adds a certain time:

- Hold or release the mouse $\rightarrow B = 0.1s$.
- Press and release a key $\rightarrow K = 0.2s$.
- Type a string $\rightarrow nk * 0.2s$.
- Move the hand from mouse to keyboard or vice-versa $\rightarrow H = 0.4s$.
- Point the mouse $\rightarrow P = 1.1s$.
- Drag-and-drop a visual element $\rightarrow dnd = 1.3s$.

For practical purposes, sometimes we join the actions Point the mouse and Hold or release the mouse in a single action defined as Point, hold and release the mouse assigned the variable X equivalent to P + 2B.

The measurement of theoretical and experimental time-consuming consists in: (i) configure the IP address (see Figure 5.4) of the VM where Mininet, VNFs, and Ryu are hosted, (ii) select and configure each VNF that forms the chain, *i.e.*, create the chain; and (iii) launch the chain.

The sequence of actions for the IP-configuration is as follows: (i) Point, hold and release the mouse over Configuration button $\rightarrow X$, (ii) Point, hold an release the mouse over Controller IP address field $\rightarrow X$, (iii) Move the hand from mouse to keyboard $\rightarrow H$, (iv) Configure the IP address 52.171.133.220:8081 of the server where the controller and VNFs are hosted. Entering the IP corresponds to pressing


Configuration

52.171.133.220:8081

Topology type: Tree 2/2

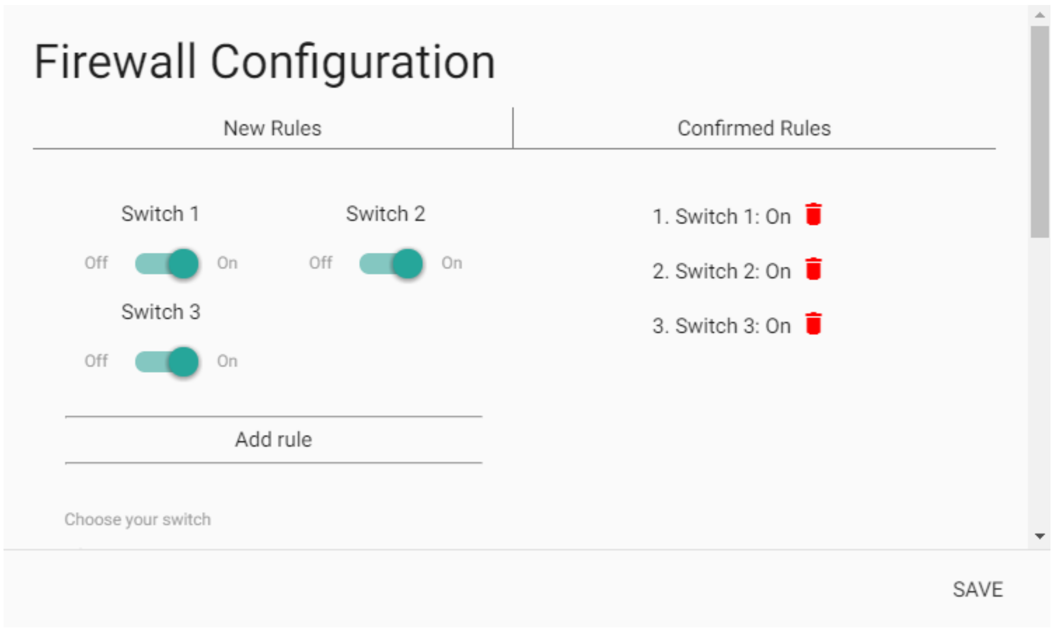
Figure 5.4: IP address Configuration

19 keys → $19K$, (v) Move the hand from keyboard to mouse → H, (vi) Point, hold and release the mouse to select topology → X, and (vii) Point, hold and release the mouse over save button → X.

The actions to create the chain consist of selecting and configuring three VNFs, Firewall (see Figure 5.5), Load Balancer (see Figure 5.6), and Router (see Figure 5.7), and such actions are described in the same order.

The actions to select and configure the Firewall are: (i) Point the mouse over Firewal VNF → P, (ii) Drag-and-drop the Firewall VNF → dnd, (iii) Hold and release the mouse over Firewall VNF → $2B$, (iv) Point, hold and release the mouse over switches buttons (three switch) → $3X$, and (v) Point, hold and release the mouse over Save button → X.

The actions to select and configure the Load Balancer are: (i) Point the mouse over Load-Balancer VNF → P, (ii) Drag-and-drop the Load-Balancer VNF → dnd, (iii) Hold and release the mouse over Load-Balancer VNF → $2B$, (iv) Point, hold and release the mouse over IP-Virtual field → X, (v) Move the hand from mouse to keyboard → H, (vi) Enter the IP-Virtual 10.0.0.10 → $9K$, (vii) Move the hand from keyboard to mouse → H, (viii) Point the mouse over choose your hosts submenu → P, and (ix) Point, hold and release the mouse over hosts button (three host) → $3X$, and (x) Point, hold and release the mouse over Agree button → X.


Firewall Configuration

New Rules | Confirmed Rules

Switch 1: Off / On
Switch 2: Off / On
Switch 3: Off / On

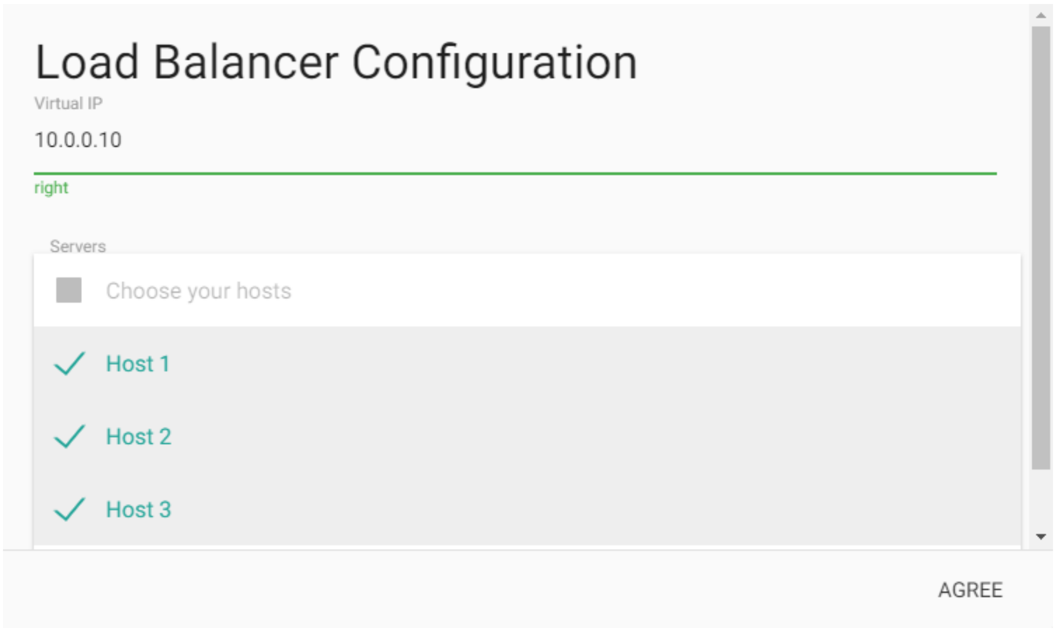
Add rule

Choose your switch

1. Switch 1: On
2. Switch 2: On
3. Switch 3: On

SAVE

Figure 5.5: Firewall configuration


Load Balancer Configuration

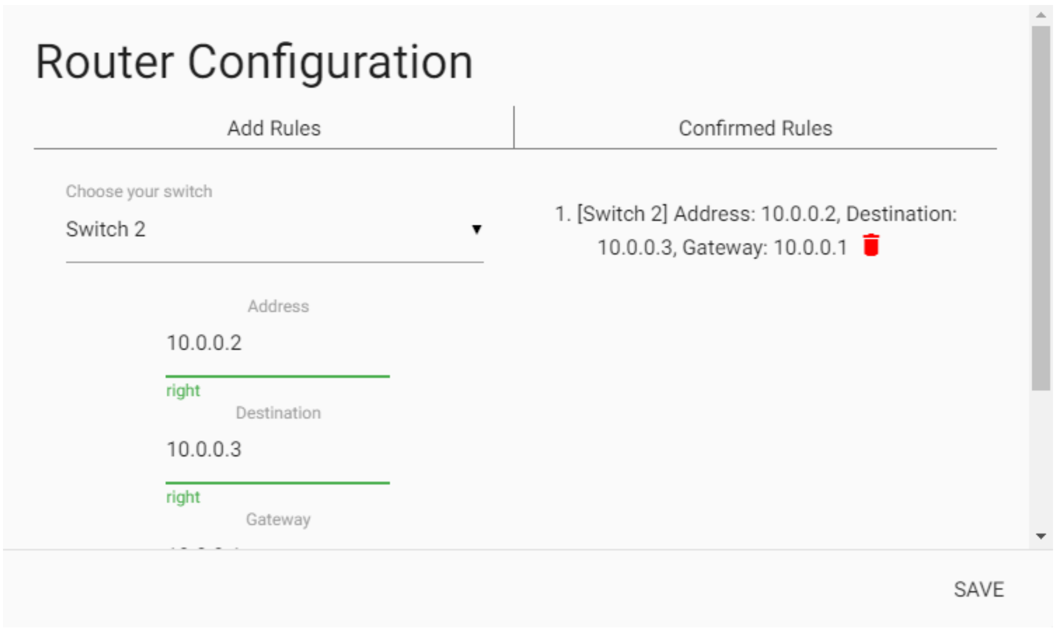
Virtual IP
10.0.0.10
right

Servers
Choose your hosts
✓ Host 1
✓ Host 2
✓ Host 3

AGREE

Figure 5.6: Load Balancer configuration

The actions to select and configure the Router are: (i) Point the mouse over Router VNF $\rightarrow P$, (ii) Drag-and-drop the Router VNF $\rightarrow dnd$, (iii) Hold and release the



10.0.0.2

10.0.0.3

1. [Switch 2] Address: 10.0.0.2, Destination: 10.0.0.3, Gateway: 10.0.0.1

Figure 5.7: Router configuration

mouse over Router VNF $\rightarrow 2B$, (iv) Point the mouse over the select switch submenu $\rightarrow P$, (v) Point, hold and release the mouse over the selected switch $\rightarrow X$, (vi) Point, hold and release the mouse over IP-Source field $\rightarrow X$, (vii) Move the hand from mouse to keyboard $\rightarrow H$, (viii) Enter IP-Source 10.0.0.2 $\rightarrow 8K$, (ix) Move the hand from keyboard to mouse $\rightarrow H$, (x) Point, hold and release the mouse over IP-Destination field $\rightarrow X$, (xi) Move the hand from mouse to keyboard $\rightarrow H$, (xii) Enter IP-Destination 10.0.0.3 $\rightarrow 8K$, (xiii) Move the hand from keyboard to mouse $\rightarrow H$, (xiv) Point, hold and release the mouse over IP-Gateway field $\rightarrow X$, (xv) Move the hand from mouse to keyboard $\rightarrow H$, (xvi) Enter IP-Gateway 10.0.0.1 $\rightarrow$ $8K$, (xvii) Move the hand from keyboard to mouse $\rightarrow H$, (xviii) Point, hold and release the mouse over Add-Rule button $\rightarrow X$, and (xix) Point, hold and release the mouse over Save button$\rightarrow X$.

Finally, the action to launch the chain is Point, hold and release the mouse over Run chain button $\rightarrow X$.

The time-consuming ($T_{con:chain}$) will depend on the number of VNFs that compose the chain and the configurations per VNF. The created chain as test meets the minimum execution and operation requirements in the emulated SDN. In general, the time spent in each stage of the process performed is defined mathematically as:

- IP-configuration: C = 4M + 2H + 19K
- Firewall: FW = P + dnd + 2B + 4M
- LoadBalancer: LB = 2P + dnd + 2B + 5M + 2H + 9K
- Router: R = 2P + dnd + 2B + 6M + 6H + 24K
- Lanzar: L = M

Then, the total time-consuming in general is:

- $T_{con:chain}$ = C + (n1*FW) + (n2*LB) + (n3*R) + L

Where, n1, n2, and n3 are variables that correspond to the number of instances used of each VNF. For the created chain an instance of each VNF was used, so the theoretical time-consuming is:

- $T_{con:chain}$ = C + FW + LB + R + L
- $T_{con:chain}$ = 9,8s + 7,8s + 12,8s + 18,7s + 1,3s = 50,4s

Then, it is expected that by using NEMASOF a Network Administrator late 50.4s in creating, configuring, and launching the chain consisting of three VNFs with initial operating parameters.

We also conducted an experimental study to measure time-consuming. The study involved 40 people whose age range is from 20 to 30 who often use web tools. Figure 5.8 illustrates the time measured in each part of the process mentioned in the theoretical measurement, and is compared with the time measured experimentally. The experimental average time result was 56.07s. This confirms the KLM predictions and demonstrates the feasibility of using NEMASOF in SDN.

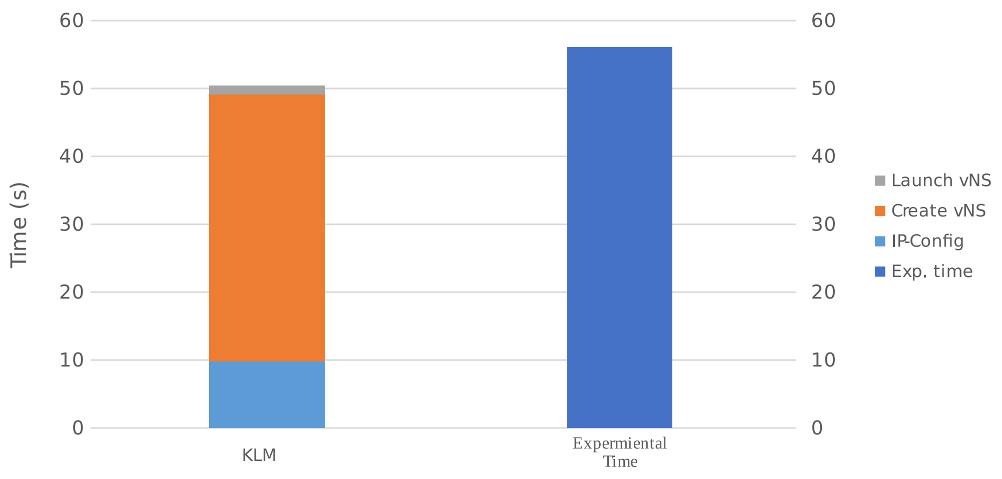


Figure 5.8: Time-consuming

5.3.2 Time-response

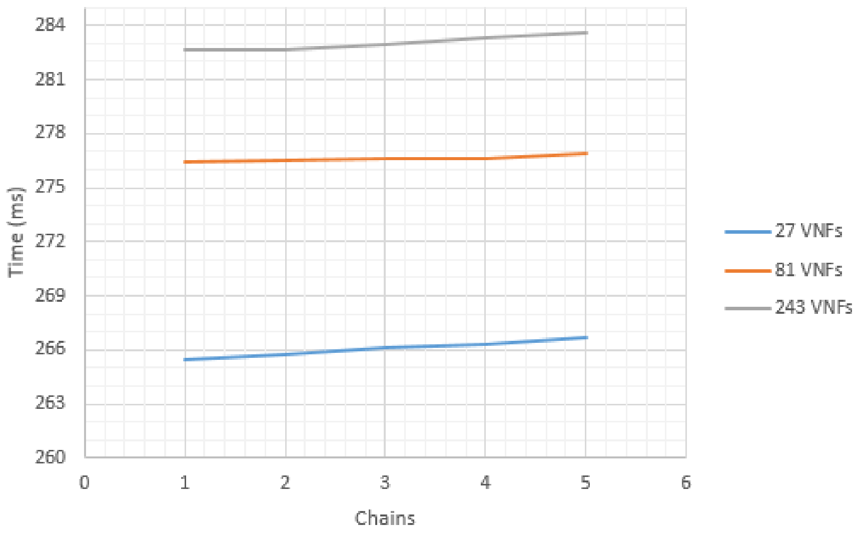


Figure 5.9: Time-response

Evaluating the time-response provides information about NEMASOF response capacity to determine that it is a good solution for NSC. It is appropriate to measure the time-response of NEMASOF when the chains are launched with different amount of VNFs. In this evaluation the time-response is measured in milliseconds (ms), 32 measurements were taken with a 95% confidence level.

In the time-response evaluation, the number of chains launched in the SDN was varied from 1 to 5. Thus, the total number of VNFs executed in each evaluation was 27, 81 and 243. The time-response (r in ms) of Web systems has ranges as optimal (r<100), good (100<r<10000), and deficient (r>10000), therefore, the time-response result expose: NEMASOF is in the good range because his time-response is between 265 ms and 330 ms in the tests performed (see Figure 5.9).

5.3.3 Network traffic

The main objective of measuring network traffic is to determine the impact of using NEMASOF to deploy a chain over an SDN-based network. We measure the network traffic in the SDN network controller. The network traffic measurement test was performed in a cloud environment.

When a Network Administrator has created a chain and configured each VNF, the Network Administrator launches the chain using NEMASOF. Launching a chain consists of sending messages to the controller with information about the operations to be performed on the SDN. The controller is in charge of executing VNFs that form the chain and, furthermore, sends OpenFlow messages to the switches of SDN about the execution of VNFs and their operating parameters.

In this sense, we carry out the measurement of the network traffic on the controller when it receives requests of execution of a chain from NEMASOF. We measure network traffic in bytes when the controller receives a chain execution requests that are compose of 27, 81 and 243 VNFs. Figure 5.10 presents the corresponding results in which the topology of the network is not taken into account because our approach is topologically independent. The results reveal: *(i)* the traffic generated by launching chains by NEMASOF increases as the number of VNFs to be executed increases, *(ii)* the traffic growth generated is slightly exponential, and (iii) a chain with the highest number of VNFs generates 2536 Bytes (2,4766 KB) which is a very low value based on the bandwidth used today.

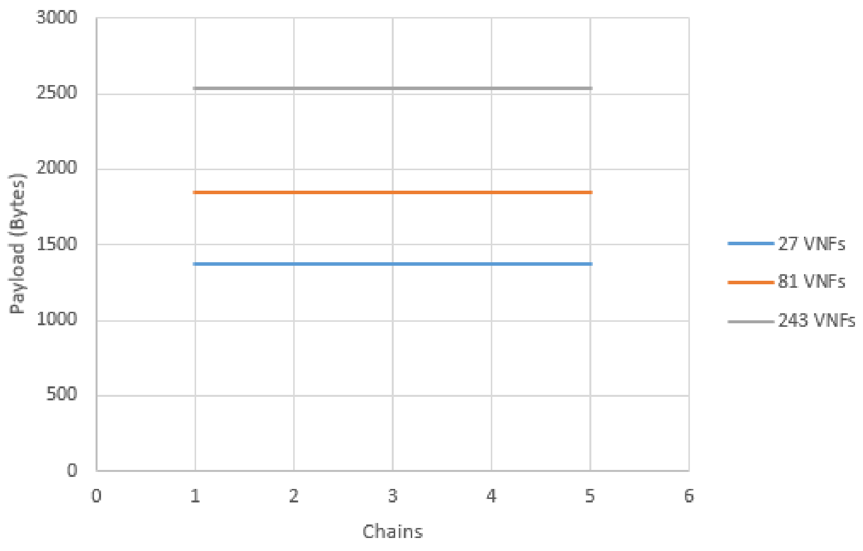


Figure 5.10: Network traffic

5.3.4 Qualitative Analysis

From the qualitative point of view, we briefly analyze the behavior of NEMASOF, which refers to analyze its main characteristics that are flexibility, extensibility, and usability as follows:

Flexibility refers to our semi-automatic NSC approach based on mashups allowing Network Administrators by themselves to create, customize and launch chains to decrease their daily workload. They do not need Web programming tools to deploy chains because our approach provides a high level of abstraction regarding the underlying network infrastructure and VNFs. VNFs are represented in visual form, and the customization options are graphically made. In addition, unlike other types of NSC, using mashups technology provides a flexible and easy way to compose chains (see Table 5.1).

Extensibility refers to Network Administrators being able to create, configure, and launch chains by a simple process using existing VNFs. It is possible because of the mashups-based approach that takes advantage of the composition, abstraction and reuse of own or third-party resources. In this way, it highlights the effectiveness and ease of launching chains unlike other NSC types. Table 5.2 presents a comparison

Approach	Flexibility
Static	Low
Dynamic	Half
NEMASOF	High

Table 5.1: Flexibility

regarding the extensibility between the different NSC types.

Approach	Extensibility
Static	Low
Dynamic	Half
NEMASOF	High

Table 5.2: Extensibility

Usability refers to the ease with which a product can be used to achieve defined objectives with efficiency, effectiveness and satisfaction [27]. NEMASOF provides the Network Administrator with a tool with the fundamental components to deploy chains in SDN. In addition, the use of NEMASOF is intuitive, which allows the Network Administrator to perform the chain deployment process efficiently (see Table 5.3).

Approach	Usability
Static	Low
Dynamic	Low
NEMASOF	High

Table 5.3: Usability

Chapter 6

Conclusions and future work

This chapter starts answering the research question. Then, we present the main conclusions of our work. Finally, we provide insights for future work.

6.1 Conclusions

This work presented the investigation carried out to answer the research question: **How to carry out NSC by using mashups?.** In response to this question, we have proposed an architecture that is composed of three layers (*i.e.,* Presentation, Chains Composition, and Virtualization) directed to the composition and customization of chains. The proposed architecture uses NFV and Mashups technology concepts for chains composition.

Mashups technology allows creating composite web applications by combining available resources (*e.g.* VNFs, graphic elements) in public or private clouds, local repositories, or the Internet. Furthermore, mashups allow encapsulating technical details of the underlying resources using a resource abstraction. In addition, mashups do not require advanced programming skills.

NFV allows decoupling NFs from the specialized hardware, which allows NFs to be executed in commodity hardware as software instances. Thus, NFV provides flexibility, scalability and ease of deployment of NFs which allows the chains composition.

SDN allows centralizing network control by separating the control plane from the data plane. In addition, SDN makes the network more flexible which facilitates the management of NFs and allows the underlying network infrastructure to be abstracted for NFs.

In this work, we have implemented a prototype of the proposed architecture as well as an extensive evaluation and analysis in a qualitative and quantitative way. The evaluation carried out is aimed at demonstrating the feasibility of our semi-automatic NSC approach based on mashups. The quantitative evaluation was addressed in terms of time-consuming, time-response, and network traffic:

- Time-consuming was evaluated experimentally and KLM, which allows us to demonstrate that the time required for a Network Administrator to launch a chain is a matter of seconds (estimated time = 50.4s, experimental time = 56.07s). But, the time required to launch a chain depends on the number of VNFs to be used and the configuration parameters of VNFs. The estimated time with KLM was corroborated with the experimental evaluation and, thus, the feasibility of using NEMASOF to launch chains. In addition, the short time needed to deploy a chain demonstrates that our proposal facilitates the daily tasks of Network Administrator.
- The time-response results corroborates that NEMASOF is an excellent alternative to deploy vNSs in a short time. NEMASOF is well-performing in terms of time-response. According to the ranking of websites, NEMASOF is ranked as good. Thus, the evaluation demonstrates that the prototype carried out is an adequate solution so that the Network Administrators can deploy chains.
- The network traffic results show that our proposed approach has a good behavior in terms of network traffic because the additional traffic generated over the network is low. The number of deployed VNFs does not increase considerably the traffic generated, which corroborates that our approach has an excellent

performance and is considered as an optimal solution for deploying chains in SDN using few resources.

From a qualitative point of view, the main characteristic provided by our proposal in this work are flexibility, extensibility, and usability. The flexibility refers to that NEMASOF allows deploying chains independently of the network topology and network resources. Moreover, our approach considers NFV concepts allowing NEMASOF to deploy a cloud environment. The mashup technology used in the development of our proposal facilitates the interaction of the Network Administrator with the process of creating, personalizing and deploying chains, which confirms that flexibility is inherent to the proposed approach.

Regarding the extensibility, it is important to highlight that it is an implicit feature of NEMASOF because it is capable of being adapted to use VNFs and network controllers from different providers. Thus, the possibility of deploying chains of different sizes and functions increases as well as the integration with varied network topologies.

Usability of our approach was demonstrated by the evaluation of the time-consuming because the little time spent to launch a chain corroborates that NEMASOF is intuitive.

6.2 Future work

During the carrying out of this undergraduate work, we observed interesting opportunities for improving this proposal and further researches. These opportunities are outlined below.

- We are very interested in being able to implement a system of recommendation of VNFs, *i.e.*, a system that allows identifying the most optimal VNF to be used according to the user requirements, the VNF location, the computational resources used by VNF, and network traffic to access VNF.

- We propose to implement a security system based on Artificial Intelligence (AI) that protects chains from unauthorized modifications and that determines the level of reliability of VNF to be used.

The above opportunities will be addressed by the GIT.

Bibliography

[1] W. John, K. Pentikousis, G. Agapiou, E. Jacob, M. Kind, A. Manzalini, F. Risso, D. Staessens, R. Steinert, and C. Meirosu, "Research directions in network service chaining," in *Future Networks and Services (SDN4FNS), 2013 IEEE SDN for*, 2013, pp. 1–7.

[2] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. D. Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.

[3] S. Sahhaf, W. Tavernier, D. Colle, and M. Pickavet, "Network service chaining with efficient network function mapping based on service decompositions," in *Network Softwarization (NetSoft), 2015 1st IEEE Conference on*, 2015, pp. 1–5.

[4] G. Cheng, H. Chen, H. Hu, Z. Wang, and J. Lan, "Enabling network function combination via service chain instantiation," *Computer Networks*, vol. 92, Part 2, pp. 396 – 407, 2015, software Defined Networks and Virtualization.

[5] S. Sahhaf, W. Tavernier, M. Rost, S. Schmid, D. Colle, M. Pickavet, and P. Demeester, "Network service chaining with optimized network function embedding supporting service decompositions," *Computer Networks*, vol. 93, Part 3, pp. 492 – 505, 2015, cloud Networking and Communications {II}.

[6] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, "Piecing together the nfv provisioning puzzle: Efficient placement and chaining of virtual network functions," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2015, pp. 98–106.

[7] O. M. C. Rendon, F. Estrada-Solano, and L. Z. Granville, “A mashup ecosystem for network management situations,” in *2013 IEEE Global Communications Conference (GLOBECOM)*, 2013, pp. 2249–2255.

[8] O. M. C. Rendon, C. R. P. dos Santos, A. S. Jacobs, and L. Z. Granville, “Monitoring virtual nodes using mashups,” *Computer Networks*, vol. 64, pp. 55 – 70, 2014.

[9] Y. M. Chen and Y. J. Peng, “A qos aware services mashup model for cloud computing applications,” *Journal of Industrial Engineering and Management*, vol. 5, no. 2, 2012.

[10] O. M. C. Rendon, F. Estrada-Solano, and L. Z. Granville, “A mashup-based approach for virtual sdn management,” in *Computer Software and Applications Conference (COMPSAC), 2013 IEEE 37th Annual*. IEEE, 2013, pp. 143–152.

[11] V. G. O. Caicedo, F. Estrada, L. Tarouco, and L. Granville, “Rich dynamic mashments: An approach for network management based on mashups and situation management,” *Computer Networks*, 2015.

[12] T. Devi and V. Reddy, “Work breakdown structure of the project,” *International Journal of Engineering Research and Applications*, vol. 2, no. 2, pp. 683–686, 2012.

[13] K. Schwaber and J. Sutherland, “The scrum guide,” July 2013.

[14] Y. Li and M. Chen, “Software-defined network function virtualization: A survey,” *IEEE Access*, vol. 3, pp. 2542–2553, 2015.

[15] N. F. V. N. E. I. S. G. (ISG), “Network functions virtualisation (nfv), terminology for main concepts in nfv,” *ETSI GS NFV 003 V1.2.1*, vol. 2, pp. 1–13, 2014.

[16] R. Mijumbi, J. Serrat, J. l. Gorricho, S. Latre, M. Charalambides, and D. Lopez, “Management and orchestration challenges in network functions virtualization,” *IEEE Communications Magazine*, vol. 54, no. 1, pp. 98–105, 2016.

[17] ETSI-NFV-ISG, “Nfv architectural framework,” *(GS NFV 002 V1.2.1)*, 2014-12.

[18] N. F. V. N. E. I. S. G. (ISG), “Network functions virtualisation (nfv), acceleration technologies, report on acceleration technologies & use cases,” *ETSI GS NFV-IFA 001 V1.1.1*, vol. 2, pp. 1–38, 2015.

[19] H. Jeon and B. Lee, “Network service chaining challenges for vnf outsourcing in network function virtualization,” in *Information and Communication Technology Convergence (ICTC), 2015 International Conference on*, 2015, pp. 819–821.

[20] J. Yu, B. Benatallah, F. Casati, and F. Daniel, “Understanding mashup development,” *IEEE Internet Computing*, vol. 12, no. 5, pp. 44–52, Sept 2008.

[21] E. M. Maximilien, A. Ranabahu, and S. Tai, “Swashup: Situational web applications mashups,” *Conference on Object-oriented Programming, Systems, Languages, and Applications*, p. 797–798, 2007.

[22] O. M. C. Rendon, F. Estrada-Solano, and L. Z. Granville, “An approach to overcome the complexity of network management situations by mashments,” in *2014 IEEE 28th International Conference on Advanced Information Networking and Applications*, 2014, pp. 875–883.

[23] M. Ghaznavi, N. Shahriar, R. Ahmed, and R. Boutaba, “Service function chaining simplified,” *arXiv preprint arXiv:1601.00751*, 2016.

[24] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, “On orchestrating virtual network functions,” in *Network and Service Management (CNSM), 2015 11th International Conference on.* IEEE, 2015, pp. 50–56.

[25] D. Lee and M. Yannakakis, “Principles and methods of testing finite state machines-a survey,” *Proceedings of the IEEE*, vol. 84, no. 8, pp. 1090–1123, 1996.

[26] S. Keshav, “An engineering approach to computer networking: Atm networks, the internet, and the telephone network,” *Reading MA*, vol. 11997, 1997.

[27] W. Iso, "9241-11. ergonomic requirements for office work with visual display terminals (vdts)," *The international organization for standardization*, vol. 45, 1998.

[28] S. K. Card, T. P. Moran, and A. Newell, "The keystroke-level model for user performance time with interactive systems," *Communications of the ACM*, vol. 23, no. 7, pp. 396–410, 1980.