UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Department of Signal Theory
and Communications

Resource Optimization in SDN-based Inter/Intra Data Centre Networks

PhD THESIS

Author:
Rafael Montero Herrera

Director:
Salvatore Spadaro

Presented in fulfilment of the degree:
Doctor of Philosophy in Signal Theory and Communications

Universitat Politècnica de Catalunya (UPC)
Department of Signal Theory and Communications (TSC)

Barcelona, June 2020

Abstract

Upon the demanding requirements brought by next-generation services, current network infrastructures are required to evolve in the way they manage different types of resources at the data layer. To this end, a set of novel technologies have risen to provide the functionalities needed for this evolution, thus representing also a change in the architectural paradigm for future network implementations.

Regarding this, the following PhD Thesis presents an analysis on these specific technologies, focusing on the particular case scenario of inter/intra-DCNs. In this way, the introduction of SDN, NFV, Network Slicing and optical technologies in these scenarios is investigated, with the goal of identifying current technical issues that can be solved by the design and application of new software-based techniques, as well as with the development and/or augmentation of architectural components.

With this purpose, this thesis defines a set of proposals, considering crucial aspects such as the required SDN-control of optical devices to allow the control of hybrid electric/optical networks, the necessity for a dynamic optical topology discovery mechanism capable of exposing accurate network information, and the analysis of the existing gaps towards the definition of a common architecture for supporting upcoming 5G and beyond communications.

In order to validate these proposals, a set of experimental validations through the use of specifically defined testbeds is also presented, to prove the enhanced control, orchestration, virtualization and management of resources in the seek for optimizing their utilization. The results exposed in this thesis, besides demonstrating the correct operation of the introduced methods and components, open the path towards new ways of adapting current network deployments to the challenges driven by the start of a new era in telecommunications.

Resumen

Considerando los altos requerimientos de los servicios de nueva generación, las infraestructuras de red actual se han visto obligadas a evolucionar en la forma de manejar los diferentes recursos de red y computación. Con este fin, nuevas tecnologías han surgido para soportar las funcionalidades necesarias para esta evolución, significando también un gran cambio de paradigma en el diseño de arquitecturas para la futura implementación de redes.

En este sentido, la siguiente tesis presenta un análisis sobre estas tecnologías, enfocado en el caso de redes inter/intra Data Centre. Por consiguiente, la introducción de tecnologías como SDN, NFV, Network Slicing y de soluciones basadas en redes ópticas ha sido investigada, con el fin de identificar problemas técnicos actuales que puedan llegar a ser solucionados mediante el diseño y aplicación de nuevas técnicas, asimismo como a través del desarrollo o la extensión de los componentes de arquitectura de red.

Con este propósito, se han definido una serie de propuestas relacionadas con aspectos cruciales, así como el control de dispositivos ópticos por SDN para habilitar el manejo de redes híbridas, la necesidad de definir un mecanismo de descubrimiento de topologías ópticas capaz de exponer información precisa, y el analizar las brechas existentes para la definición de una arquitectura común en fin de soportar las comunicaciones 5G.

Para validar estas propuestas, se han presentado una serie de validaciones experimentales por medio de escenarios de prueba específicos, demostrando los avances en control, orquestación, virtualización y manejo de recursos con el fin de optimizar su utilización. Los resultados expuestos, además de corroborar la correcta operación de los métodos y componentes propuestos, abre el camino hacia nuevas formas de adaptar los actuales despliegues de red respecto a los desafíos definidos en el inicio de una nueva era en las telecomunicaciones.

Acknowledgements

At First, I would like to thank my parents, Rafael and Julia, for all their support and encouragement throughout these last years, without them all this adventure abroad would not have been possible. My most sincere thanks to them and to my brother Alan for all the family time they sacrificed, so I was able to pursue this PhD degree.

I want to express my deep gratitude to my thesis advisor Prof. Dr. Salvatore Spadaro, who gave me the opportunity to join the GCO group at UPC and provided his guidance so I could develop my research on this particular field. Also, I would like to acknowledge the collaboration of Dr. Fernando Agraz and Dr. Albert Pagès, their valuable support during these years was crucial to make this work possible.

Special thanks to Elsa, my partner in life, for her continuous support and for sharing with me both the happy and difficult moments I went through in my pursuit to finish the PhD. I feel very grateful to life for having her on my side and sharing our present together, besides our many hopes and dreams for an exciting future.

I would like to mention as well the important persons who kept me motivated to fulfil my goals. I thank Ana for her support and friendship during these years, it really meant a lot to me. I want to thank as well my good friends from the master, David, Luis and Sergio. The colleagues and friends I met during this journey, Juan, Jeison, Alba, Rubén, Samael, Luis and Saeed. My family in Barcelona, my aunt Lilian, my uncle Josep Maria, Blanca and Pau. As well as my friends here, Melissa, Shaggy, Marianne, Alai, Tina, Valerie, Francy, Camila, Loreto, and many more. Finally I would also like to thank my friends from Bolivia who also gave me a lot of support and were eager to share very good moments with me when it was possible. To the ones I miss mentioning here, thanks to all, you all played a big part on this.

At last, I also want to thank the AGAUR and the Generalitat de Catalunya for providing the financial support for the accomplishment of this work, through the PhD scholarship 2017FI_B_00083.

INDEX

INDEX OF FIGURES

INDEX OF TABLES

INDEX OF CODE

Acronyms

5G	Fifth Generation Mobile Technology
5G-PPP	5G Infrastructure Public Private Partnership
A-CPI	Application-controller Plane Interface
ACK	Acknowledgement
API	Application Programmable Interface
BER	Bit Error Rate
BSS	Business Support System
CAM	Content Addressable Memory
CAPEX	Capital Expenditures
CLI	Command Line Interface
CM	Configuration Management
CPU	Central Processing Unit
CSMF	Communication Service Management Function
DB	Database
DC	Data Centre
DCN	Data Centre Network
D-CPI	Data-controller Plane Interfaces
DevOps	Development and Operations
DS	Datastore
E2E	End to End
ECA	Event-Condition-Action
eMBB	Enhanced Mobile Broadband
ES	Inter-cluster Optical Switches
ETSI	European Telecommunications Standards Institute
FM	Fault Management
GMPLS	General Multiprotocol Label Switching
IaaS	Infrastructure as a Service
ICT	Information and Communication Technologies
IM	Interface Manager (NFV-C)
I/O	Input/Output
IoT	Internet of Things
IS	Intra-cluster Optical Switches

ITU	International Telecommunications Union
KPI	Key Performance Indicator
LLDP	Link Layer Discovery Protocol
LM	Lifecycle Management
LS	Latency Sensor
MANO	Management and Orchestration
MD-SAL	Model-Driven Service Abstraction Layer
ME	Monitoring Engine
MEC	Multi-access Edge Computing
MF	Management Function
MM	Monitoring Manager
MM	Monitoring Manager (NFV-C)
mMTC	Massive Machine Type Communications
NBI	Northbound Interface
NE	Network Element
NETCONF	Network Configuration Protocol
NF	Network Function
NFMF	Network Function Management Function
NFV	Network Function Virtualization
NFV-C	Network Function Virtualization Coordinator
NFVI	Network Function Virtualization Infrastructure
NFVO	Network Function Virtualization Orchestrator
NGMN	Next Generation Mobile Networks
NRM	Network Resource Model
NS	Network Service
NSD	Network Service Descriptor
NSI	Network Slice Instance
NSMF	Network Slice Management Function
NSSI	Network Slice Sub-Network Instance
NSSMF	Network Slice Subnet Management Function
OCS	Optical Circuit Switch
ODL	OpenDaylight
ODTN	Open Disaggregated Transport Network
O/E/O	Electro-Optical/Opto-Electrical
OF	OpenFlow

OFDP	OpenFlow Discovery Protocol
OLD	Optical Link Discovery
ONAP	Open Network Automation Platform
ONF	Open Networking Foundation
ONOS	Open Source Networking Operating System
OPEX	Operational Expenditures
OPM	Optical Provisioning Manager
OPNFV	Open Platform for Network Function Virtualization
ORVM	Open Resource Virtualization Manager
OSM	Open Source MANO
OSS	Operations Support System
OTN	Optical Transport Network
OVS	Open Virtual Switch
OVSDB	Open Virtual Switch Database Management Protocol
PCE	Path Computation Engine
PLR	Packet Loss Ratio
PM	Performance Management
PM	Policy Manager (NFV-C)
PNF	Physical Network Function
PoP	Point of Presence
QoE	Quality of Experience
QoS	Quality of Service
RAM	Random Access Memory
REST	Representational State Transfer
RPC	Remote Procedure Call
RTT	Round-Trip-Time
SDN	Software Defined Networking
SLA	Service Level Agreement
SNMP	Simple Network Management Protocol
SM	Slice Manager (NFV-C)
TCAM	Ternary Content Addressable Memory
TCP	Transmission Control Protocol
TF	Tungsten Fabric
TM	Topology Manager
TM	Tunnel Manager (NFV-C)

ToR	Top of Rack
TSM	Test Signal Mechanism
URLLC	Ultra-Reliable and Low Latency Communications
VDC	Virtual Data Centre
VIM	Virtual Infrastructure Manager
VLD	Virtual Link Descriptor
VM	Virtual Machine
VNF	Virtual Network Function
VNFD	Virtual Network Function Descriptor
VNFFGD	Virtual Network Function Forwarding Graph Descriptor
VNFM	Virtual Network Function Manager
VTN	Virtual Tenant Network
WAN	Wide Area Network

PART I

INTRODUCTION

Chapter 1. Introduction

The first chapter encompasses the motivation and scope of the PhD Thesis, taking into account the role of optical networks in inter/intra data centre networks (DCNs) and the need for their adoption, along other novel technologies, in future network scenarios such as in the next-generation mobile networks technology (5G).

1.1 Motivation

The constant growth of data flowing in and through current data centres networks (DCNs) potentiated by the high bandwidth utilization of emerging applications and paradigms (e.g. Cloud-based services, Internet of Things (IoT), Big Data, etc.), besides the demanding requirements set for the evolution towards 5G networks [1] (e.g., high throughput, low latency, ultra-high reliability), has driven data centres (DCs) to incorporate optical technologies and next-generation solutions in order to handle such traffic loads with diverse requirements. However, the introduction of these changes still demands efforts to address not only the planning and design of future DCs, but also the operation, control and support aspects required to improve the usage efficiency of their resources. To this end, technical solutions already used in electronic-based scenarios can be adapted to optical-enabled DCNs.

In this regard, the fifth generation of mobile technologies (5G) represents the base network scenario for the upcoming new era of communications, where the application of a set of technologies and novel solutions is given, to handle the demanded performance across all 5G network segments [2] (i.e., Radio Access Network, Edge Network, Transport Network, Core Network). As for the specific case of DCNs, covered in this thesis, DCs in the context of 5G are considered at the Edge and Core segments according to their provided functionalities, taking also in consideration the required DCN connectivity at the involved network segments.

One of the aforementioned novel solutions in 5G scenarios is Software Defined Networking (SDN), capable of providing network managers and business applications with an abstract view of the underlying topology and resources. In this

matter, efforts nowadays show mainly support for electronic-based technologies, where certain SDN use cases and related industry products are already reaching a mature state. The application of SDN in DCNs then, particularly facilitates the integration of the control plane with orchestration tools to allow the automated deployment of services at the DC, while guaranteeing performance requirements related to quality of service (QoS) and quality of experience (QoE) demands.

Following this, the provisioning of these services over virtually allocated resources, enabled by the Network Slicing concept in 5G, has the potential of providing elastic infrastructures to customers and of optimizing the use of physical resources by accommodating different services over the same underlying network. In this given case, dedicated virtual infrastructures based on specific service requirements of computing, storage and networking are assigned to particular network services (NS), where the functions composing each NS can be also virtualized, following the Network Function Virtualization (NFV) model. The use of optical technologies in these scenarios, increases bandwidth and lowers the latency, improving the overall communication between intra-data centre servers, as well as the communication between different DCN segments. The introduction of SDN on the other hand, enables this efficient use of network resources by exposing their capabilities and providing granular control.

Optical related SDN implementations continue to be at the early stage, but have already risen the interest of the research community. Still, the usage of SDN architecture and protocols with optical resources still requires further research. In this regard, this thesis aims to contribute to the adoption of SDN-based solutions in optical DCNs taking as a reference the 5G framework and focusing on the efficient usage of resources in these types of scenarios.

1.2 Network Innovation in 5G Optical Data Centre Networks

The future of networks is tied to the need for supporting the increasing bandwidth and different set of requirements of new applications/services. In particular, for the presented thesis, the progression towards such high-capacity connectivity given by the introduction of optical technologies, and the application of novel solutions to

address efficient resource management at inter/intra DCNs is set for investigation. With this in mind, significant changes are considered at all related network segments and layers to improve flexibility, configurability, speed, scalability and transparency, while allowing to manage the automation of actions during service deployment and operation phases, as defined for 5G communications. In this way, and by means of network innovation, the aim of future network implementations is also set on reducing Capital Expenditures (CAPEX) and Operational Expenditures (OPEX), looking forward to a sustainable migration from the existing infrastructure to an evolved one powered with new technologies [3].

Fig. 1-1 presents the scenario under investigation, where the study centres on the DCN scenarios support for 5G and future services/applications, describing related network segments (i.e., edge, transport, core.) and architectural layers (i.e., control, orchestration, application) for the management of DCN resources. With this purpose, the control of optical resources, the discovery of optical topology, as well as the provisioning and maintenance of 5G services is considered in order to design, test and deploy resource management solutions at given optical-enabled DCN scenarios. In view of all the required implementations set for enabling such optimizations, the proposal of a novel management architecture for DCN scenarios is also essential for next-generation deployments, considering its design shaped by the introduction of 5G technologies such as SDN, NFV, Network Slicing, seeking to accommodate current deployments to the requirements of the 5G framework.

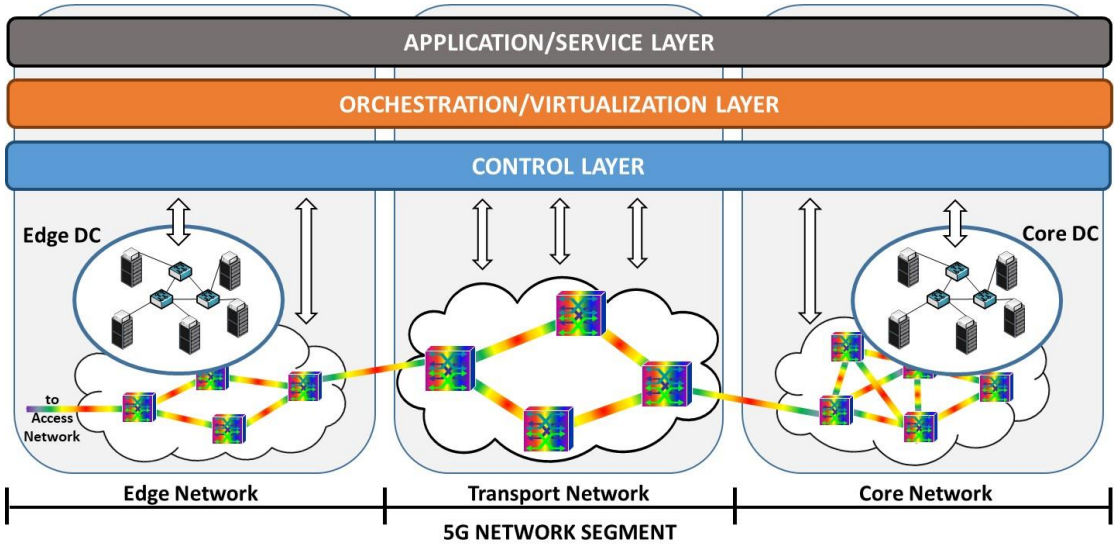


Fig. 1-1. Main Thesis Scenario

1.3 Scope of the PhD Thesis

After analysing the aspects that defined the motivation for this PhD Thesis and giving a view on how the scenario under study is foreseen for next-generation networks, it becomes crucial to outline the scope of the presented thesis. In this sense, the focus of this thesis is set on optimizing the use of network resources in DCNs. Considering in particular, intra-DC and inter-DC communications in the 5G context, and thus studying and proposing resource management techniques at the DC and in segments providing DCN interconnectivity (i.e., edge, transport, core). Moreover, the use of previously discussed technologies and solutions (i.e., Optical, SDN, NFV, Network Slicing) is also seen as crucial and will serve as the base for defining innovation in the presented resource usage optimization solutions.

For a better organization of the manuscript, the document has been divided in four parts. The outline of each part is described next:

- **PART I - Thesis Introduction**

The introductory part of the document seeks to describe the basis of this thesis, exposing its motivation, scope and set of defined objectives, as well as introducing the state of the art of technologies related to the study.

- **PART II - Optimization in Intra Data Centre Networks**

The second part comprises the proposed resource-usage optimization methods considered within the DC segment. Seeking for the optimization of the intra-DCN functionality, by means of optical device control and optical topology discovery.

- **PART III - Optimization in Inter Data Centre Networks**

The third part covers networking solutions under the scope of several segments of the 5G framework, related to the studied DCN scenarios. Focusing on improving resource utilization in inter-DCNs through automation and orchestration.

- **PART IV - Results Dissemination & Conclusions**

Finally, the last part summarizes the conclusions and future work in respect of the achievements of this thesis. Besides describing all the related published work and project collaborations.

Chapter 2. State of the Art

This section presents the current state of the art regarding technologies involved in the development of the thesis considering the defined scenario under study. In case of the associated technologies, an introduction of SDN, NFV, Management and Orchestration (MANO), Network Slicing, 5G telecommunications and their optical related appliances and extensions is given.

2.1 Intra and Inter Data Centre Networks

Data Centre Networks (DCNs) have seen many changes throughout the time; most recently, the introduction of optical devices in data centres has been potentiated mainly due to the high bandwidth requirements of modern networks [4]. Thousands of servers in legacy DCNs have to be interconnected with high bandwidth; unfortunately, this is currently achieved by high power consuming electronic packet switches. In this context, data centre administrators are beginning to consider a migration to the use of optical interconnections to take profit of the high throughput, low latency and less power consumption offered by optical technologies [5].

The bandwidth requirements of current networks have increased in the past years mainly because of the emergence of modern trends such as Cloud Computing, Big Data, the Internet of Things (IoT) and IT Consumerization, where features and services brought by these trends have created the need for Data Centres (DCs) with higher performance in terms of speed, computation, storage, automation, IT/network resource usage, among others. Another aspect that has influenced is the change of traffic and networking patterns, considering nowadays more machine-to-machine communications and users accessing content for any device at any place and time, following an “always available content” basissis.

The introduction of optical technologies to DCs has also brought the use of technologies such as SDN to further improve the use of network resources [6], positioning hybrid SDN-enabled DCNs with both electronic and optical capable devices to adapt networks to higher requirements in a more dynamic way [7]. In

this regard, Fig. 2-1 describes the scenario given for the development of this thesis, focusing in this case on the control and management of inter/intra DCNs. In particular, this thesis looks forward in solving existing resource utilization issues in DCNs through the implementation of novel network architectures and resource optimization methods capable of controlling and orchestrating advanced network technologies, searching for the dynamic provisioning of services to guarantee the optimal utilization of networks and resources. The selected scenario will be used as a base to identify potential issues and set goals to optimize resource utilization.

In some cases, as the one presented in Fig. 2-1, coordination between multiple SDN controllers and other higher-level components such as orchestrators of multi-segment management entities could be required, where depending on the network segment, a set of particular components could be in charge of specific resource control. As an example, each DC may be managed by its own controller and orchestrator (i.e., intra-DCN management), while other components may be in charge of controlling optical networks between DCs (i.e., inter-DCN management). Therefore, communication between peer SDN controllers or between these and higher-level management components must be considered. In principle, these components may also belong to different operators or network providers.

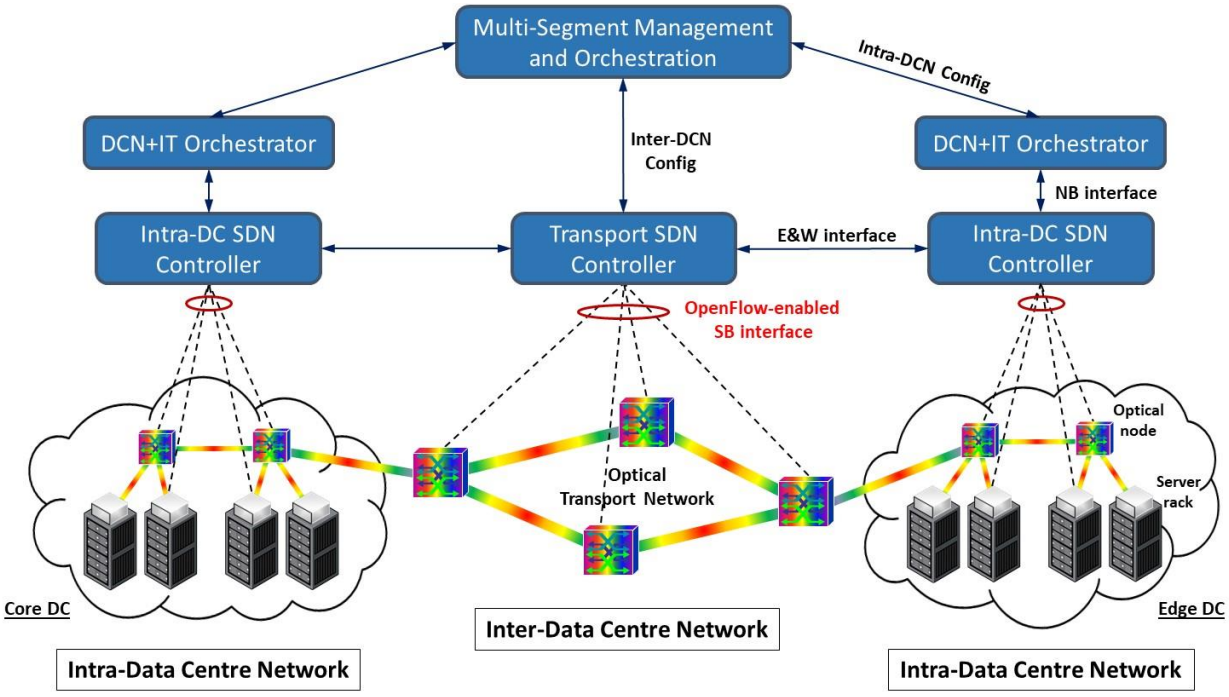


Fig. 2-1. Intra and Inter Data Centre Network Scenarios

2.1.1 Intra-Data Centre Network Scenario

One of the scenarios identified in the figure above relates to networks implemented at the DC segment, providing the required intra-DCN connectivity. These networks powered by SDN look forward to the rapid configuration of network elements, both optical and electronic based, to accommodate to the requirements of deploying applications, services and the use of orchestration and virtualization tools.

One of these services, as discussed in the introduction chapter, would be the provisioning of Network Services (NS), which can be identified as a key use case to test resource optimization techniques. The use of optical devices in DCNs is being considered to bring higher throughput between endpoints inside the DC, where efforts regarding the application of SDN technology over such hybrid DCNs are still needed to further optimize the use and control of optical resources.

2.1.2 Inter-Data Centre Network Scenario

Another scenario present in the previously depicted figure is the one allowing connection between DCs or more usually known as inter-DCN connectivity. In this case, the application of SDN would allow, among other applications, to dynamically configure optical resources by using path selection algorithms based on the current state of the network. Using such approach, the network would provide a faster response to network changes and failures, improving the availability of connections and establishing highly programmable networks between DCs [8].

The previously mentioned coordination between several controllers at this level would also be considered a key topic for this particular scenario, considering multiple segments of the network to be involved. Regarding this, software components at higher layers of the architecture (i.e., orchestration, application) would be necessary to provide of such network coordination. In this way, the capabilities and information of each segment would need to be exposed to upper layers, so the intelligence to make changes over the multi-segment scenario comes from having a complete view of all the underlying infrastructure, enabling making decisions end-to end and delegating required actions to per-segment components.

2.2 Software Defined Networking

The current network software-ization trend has boosted deployments of network architectural solutions with a decoupling of the control plane from data forwarding functions, by means of employing software components at a logically centralized control layer capable of controlling such underlying forwarding resources. This approach, has been identified as Software Defined Networking (SDN), and gives network managers the opportunity to manage, configure, automate and optimize network resources via software applications, thus avoiding inefficient per-device configurations and improving the overall network performance. Opening with the application of this concept, a new path in networking evolution [9].

2.2.1 Fundamentals of SDN

The proposal that SDN brings to networking is based on a set of fundamental characteristics, which were established in order to take a step forward to current network limitations due mostly to high bandwidth utilization, network complexity, vendor dependence and scalability issues [10]. The aforementioned characteristics are the following:

- **Separation of Planes**

It consists basically on the separation of control and data planes. In this case forwarding functionality such as tables and logic remain in the data plane while the intelligence is kept in the control plane. Hence, fundamental actions like forward, drop, consume or replicate packets are the ones performed at the data plane by determining the correct action after looking at an address table. On the other hand, the protocols, logic and algorithms that are used to program the data plane stay at the control plane, since they may require a global knowledge of the network in order to operate. In SDN, the control plane is migrated from the switching node to a logically centralized controller so they no longer co-reside in the same device.

- **Centralized Control**

Following the previous characteristic, the idea of moving the control software from devices to a centralized controller allows simplifying the management of the network. Using this approach, a software-based centralized controller is able to

manage the network by having a global view of it and using higher-level policies. It is also important to understand that centralized control does not mean physical control centralization, which would lead to a single point of failure in the network and horizontal scalability limitations. Instead, SDN proposes a logically centralized control that would be able to provide management of the network without compromising security or scalability.

- **Openness**

Another characteristic of the SDN paradigm is that interfaces should be exposed using well-documented open standards. This openness will allow the community to research into innovative methods of network operation and will potentiate the customization of SDN platforms to specific environments. However, the most important goal of openness is vendor interoperability. To achieve this, it will be necessary to secure both southbound and northbound as open interfaces. In this way, a competitive environment will be able to arise, which will consequently result in the cost reduction of network equipment.

- **Simplification of Devices**

As control functionalities would be taken out of switching devices, these can become simpler, allowing the reduction of overall equipment costs and letting the central control handle devices only as resources, ordering them on how to behave. Although this may seem an appropriate solution, it may take time to be applied, especially in commercial equipment where a stage of adaptation has to be considered. This stage may include the use of hybrid equipment that could operate, depending on the use case scenario, by using the control plane inside the switching device or not, in order to support the transition from legacy networks towards SDN.

- **Network Automation & Virtualization**

The abstraction of distributed forwarding and configuration states, plays an important role in the programmability of SDN networks; it allows setting up complex actions through programming abstractions, specifying forwarding behaviours without the need of knowing vendor-specific hardware and applying actions in the network without the need of considering how the data plane will implement them.

Southbound interfaces allow the control plane to communicate with data resources while northbound interfaces provide access to applications, which can be programmed to manage specific network resources in certain ways, allowing the automation of operations without the need of knowing individual device characteristics. This abstraction also considers an approach to virtualization, where devices connected through southbound interfaces can either be virtual or physical, and applications managing these resources will be unaware.

2.2.2 The SDN Architecture

The Open Networking Foundation (ONF) [11], is an organization dedicated to the development and standardization of SDN, and provides a graphical representation of the defined SDN architecture [12], depicted below in Fig. 2-2. This architecture is based, as previously introduced, on the separation of planes and the interaction between them via northbound and southbound open interfaces. According to the ONF, the proposed SDN architecture has the following characteristics that meet fundamentals reviewed in the previous sub-section:

- **Direct Programmability:** As the control plane is decoupled from the data plane forwarding functions, it is possible to directly program network control.

- **Agility:** Network-wide traffic flow can be dynamically managed in order to be adjusted to changing needs.

- **Central Management:** A global view of the entire network is logically centralized at the SDN controller and can be used by applications or policy engines running on top of the control plane.

- **Programmatic Configuration:** Dynamic and automated SDN programs, allow network managers to configure, optimize, manage and secure network resources without the need of proprietary software.

- **Open Standards Based:** Open standard implementations avoid the need of vendor-specific devices and protocols while simplifying the overall network design and operation.

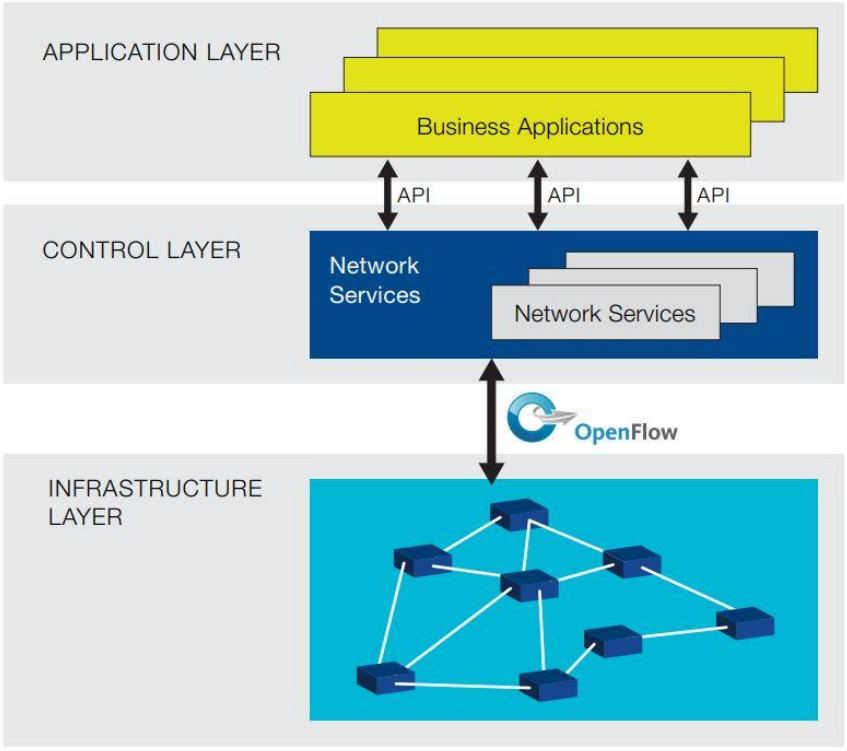


Fig. 2-2. SDN Architecture
[Adapted from *ONF, "OpenFlow-enabled SDN and NFV", 2014*]

The architecture defines three separate layers, where the infrastructure layer, also known as data plane, comprises network devices that expose their capabilities to the control layer via a southbound interface; one common example of a southbound interface implementation is the OpenFlow protocol [13]. Then, by knowing data plane capabilities, the control layer is able to optimize performance, provide granular control over network resources and deliver relevant information and services up to end-user business applications running on the application layer. These ones in turn, connect to the control layer via Application Programmable Interfaces (API), to let the controller know about their network requirements.

2.2.3 SDN Components

To better understand how SDN works, it is necessary to characterize the role of each of its components inside the previously analysed architecture model and how these components are related to each other. Fig. 2-3 follows up the architecture approach previously introduced, with a focus now on the specific SDN components throughout the different network layers [14].

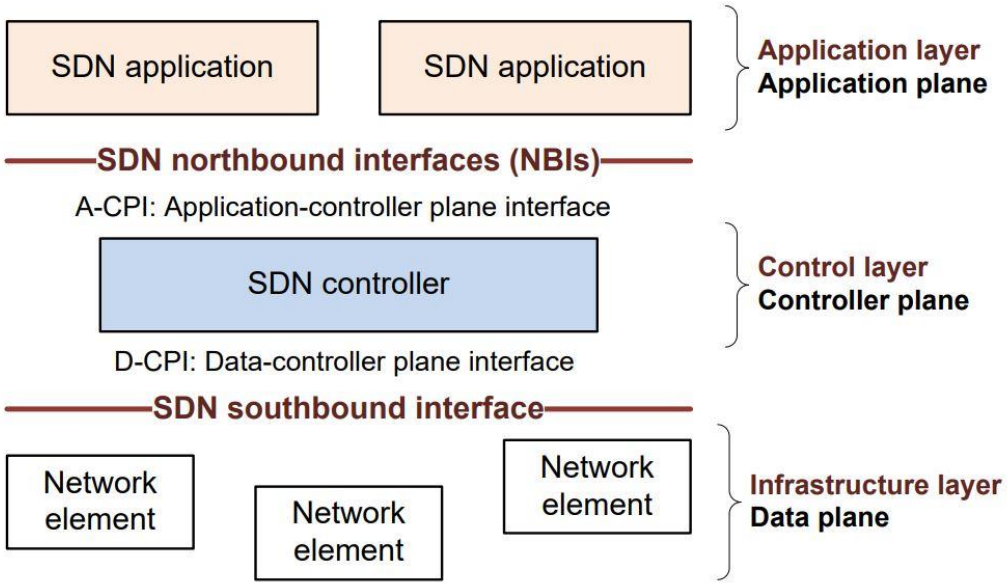


Fig. 2-3. Basic SDN Components
[Adapted from *ONF, "SDN Architecture 1.0", 2014*]

Among the set of different components, and looking from a top-to-bottom point of view, SDN applications reside at the top of the architecture, connected to the SDN controller via Application-controller Plane Interfaces (A-CPI), better known as SDN northbound interfaces (NBIs). Besides, the controller is connected to SDN-enabled devices or network elements (NE) via the Data-controller Plane Interface (D-CPI), more commonly known as SDN southbound interface. NEs in turn, may also require a set of SDN plugins or agents to support this connection. A more in-depth look on the functionalities of each particular component follows.

2.2.3.1 SDN Application

An SDN application can be defined as a program running on top of the controller which is connected to it through a northbound API. An application could retrieve or consume the abstracted view of the network from the controller, so it can make internal decisions based on this information. The application is usually conformed of a logic base and a northbound agent/driver that enables the communication to the controller. An SDN application may represent some level of abstraction in network control, providing flexibility and programmability. Types of SDN applications include programs for network virtualization, network monitoring, intrusion detection (IDS) and flow balancing, among many other possibilities.

2.2.3.2 SDN Northbound Interfaces

Northbound interfaces work between the SDN applications and the SDN controller, providing direct communication of the network requirements/abstraction through actions such as events or methods as is depicted in Fig. 2-4.

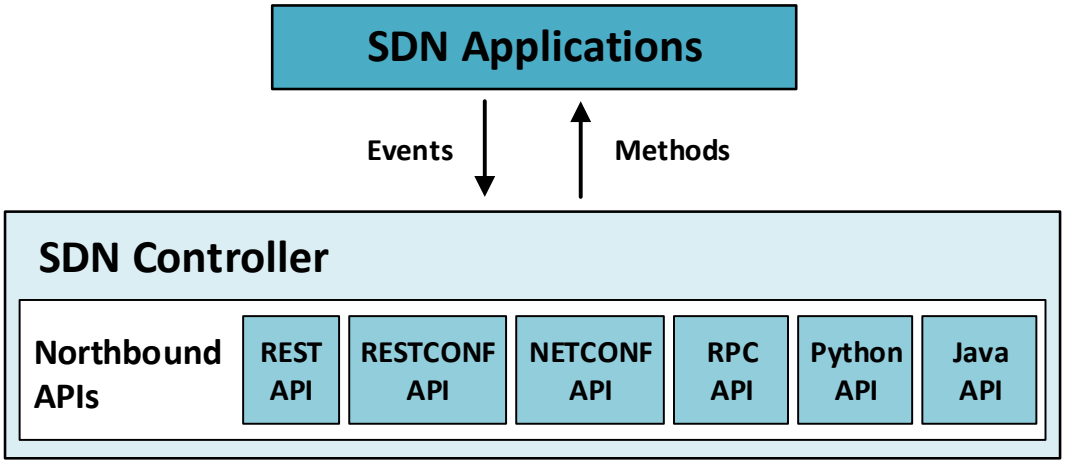


Fig. 2-4. SDN Northbound Interface Connection

Besides providing connection to SDN applications, NBIs are also used to integrate the controller with Development and Operations (DevOps) automation tools [15], as well as with orchestration platforms such as OpenStack [16]. The NBI supports different APIs for inter-layer communication, where Representational State Transfer (REST) and Remote Procedure Call (RPC) APIs are commonly used, as well as APIs based on Java or Python programming languages, besides many other options. Communication at this level is today a crucial topic in seek for SDN standardization, since a standardized open interface is required to guarantee vendor-neutrality and interoperation between applications and controller.

2.2.3.3 SDN Controller

The SDN controller is basically a logically centralized unit that is in charge of:

- Translating requirements from SDN applications down to the SDN-enabled network elements at the data plane.
- Providing an abstract view of the network to applications, considering also network events and statistics.

A controller, whose internal architecture is depicted in Fig. 2-5, may be composed of several northbound API agents to offer accessibility in different ways, through REST/RPC interfaces, or through Java/Python APIs as an example. The controller logic resides below these interfaces, where several modules can offer functionalities like network and topology discovery, device management and specific content related databases. Southbound API plugins in turn, reside at the bottom of the controller, providing connection to the network elements, also called SDN devices. In this case multiple possibilities for southbound protocol plugins are shown, supporting protocols such as OpenFlow (OF), OF-Config, Simple Network Management Protocol (SNMP), Network Configuration Protocol (NETCONF), Open vSwitch Database Management Protocol (OVSDB), among others.

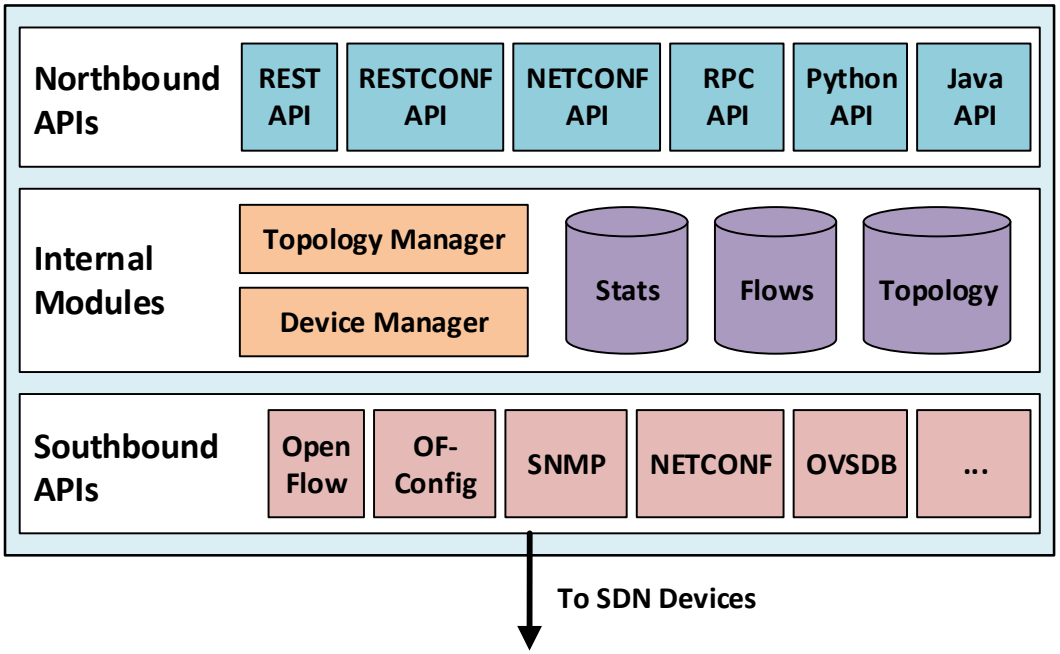


Fig. 2-5. SDN Controller Architecture

Even though the controller architecture seems to show a “north-south” design, it also allows setups like federation of multiple controllers, hierarchical connections, communication interfaces between a set of different controllers or virtualization/slicing of network resources. Depending on the controller, internal components can vary. Southbound interfaces may offer support to one to many protocols, controller modules can provide specific functionalities and finally northbound interfaces can allow a variety of northbound APIs for SDN applications interconnection or, in some cases, just a single one.

2.2.3.4 SDN Southbound Interface

The southbound interface interconnects the controller and the network elements, providing functions such as:

- Programmatic control of forwarding operations.
- Capabilities advertisement.
- Statistics reporting.
- Event notification.

The interface is able to support many different protocols. However, it is important to take special consideration on the OpenFlow (OF) protocol [13], an open source protocol designed to provide open connections in the southbound and that is seeking for standardization along SDN since the very beginning.

In particular, this thesis focuses on the use of the OF protocol to design, implement and test, different functionalities all over the presented optimization mechanisms in optical networks. The reason of using this specific protocol lays on its wide support from many different SDN related platforms and controllers, since the initial deployments of SDN, considering as well its extended support for optical technologies. Nowadays other protocol options have arisen for controller-to-device communication, such as the P4 protocol [17], but still require time to acquire maturity. Then, for the development of this thesis OF has been selected as main protocol reference. Further analysis on OF is provided in *Section 2.2.4*.

2.2.3.5 SDN-enabled Network Elements

An SDN device is a networking element, logical or physical, capable of exposing visibility and control over its forwarding and data processing capabilities to the upper control layer. Depending on the type of element, its composition can vary:

- **SDN Logical Network Element**

A logical, virtual or software NE, is composed of a SDN plugin/agent that allows its connection towards the controller, besides containing in its core, a well-defined abstraction layer with flow tables to allow performing actions on incoming traffic.

Such tables, allow matching traffic so software-based packet processing functions can handle packet forwarding. Alternatively, in case no traffic to flow matches are found, traffic is redirected it to the controller for further processing.

- **SDN Physical Network Element**

A physical NE, as depicted in Fig. 2-6, shows a similar composition to a logical one, where instead of allocating packet processing software, all packet processing and forwarding functions are embedded in the hardware, through the use of Content Addressable Memory (CAMs) and Ternary Content Addressable Memory (TCAM) tables as an example for layer 2-3 forwarding.

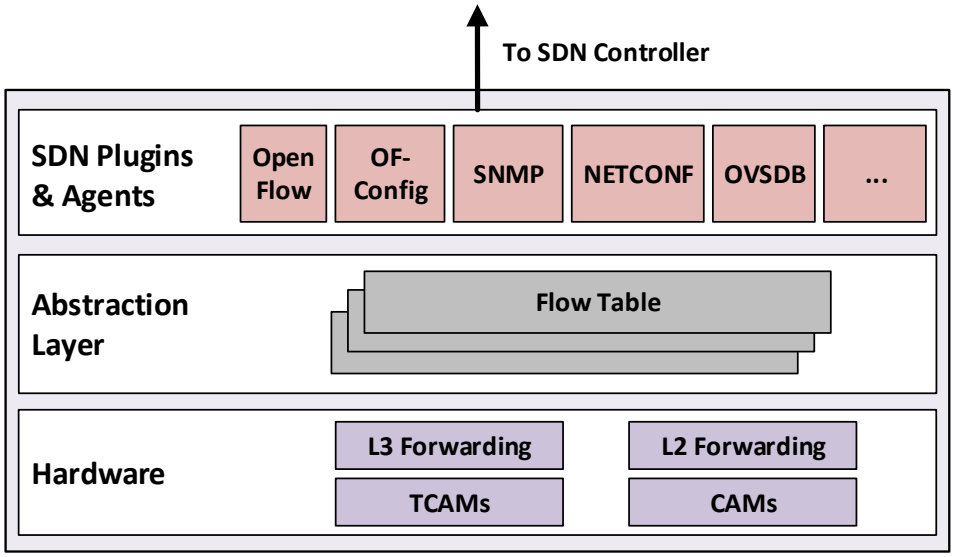


Fig. 2-6. SDN Physical Network Element Architecture

Besides physical and logical NEs, there exists another type considered as a hybrid approach, which allows supporting both legacy an SDN networks.

2.2.3.6 SDN Plugins & Agents

Interface plugins and agents are software modules implemented on NEs, SDN applications and SDN controllers, to provide inter-layer connectivity through the “north” or “south” interface, by establishing a connection with a neighbouring plugin/agent on the other layer. Although it is expected that these modules come already integrated in the components, or at least being supported by them, there exist cases where there is no agent at the network component, so the use of an external driver/agent is required to enable its connection to the SDN framework.

2.2.4 OpenFlow Protocol

OpenFlow (OF) is a standard protocol used to implement the southbound interface, interconnecting control and data plane in the SDN architecture [18]. It provides direct access to forwarding functionalities of network elements, logical or physical, allowing moving control out of the devices towards the logically centralized control software (i.e., SDN controller). The protocol specifies a basic set of instructions that can be used by applications at the controller level to directly program the forwarding functions of underlying network elements. These “instructions” define specific rules according to match/action tasks.

As it can be seen in Fig. 2-7, OF is implemented on both sides of the southbound interface by means of plugins/agents. A secure channel is established to set up the connection of the network element to the controller.

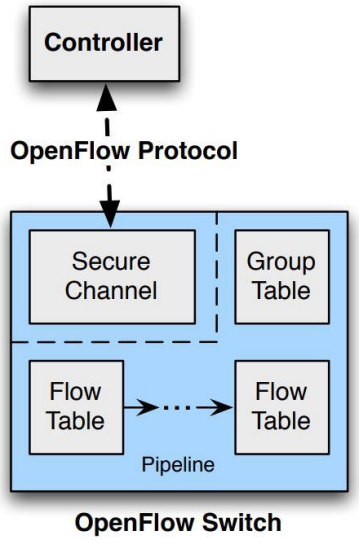


Fig. 2-7. OpenFlow Implementation
[Adapted from *ONF, “OpenFlow Switch Specification v1.1.0”, 2011*]

Inside the OF-enabled device, there exists a flow table and a group table, which allows packet lookups and forwarding. OF uses the concept of flows to identify network traffic based on a set of match/actions rules, the previously mentioned “instructions”, which can be programmed and managed directly by the controller by means of OF messages. This per-flow basis programming provides extremely granular control, enabling components at higher levels (e.g., application, user, session) to trigger a fast response to real-time changes of the network.

2.2.4.1 Optical Extensions

There exist different version specifications for the OF protocol. However, most of them are focused on the control of electronic-based devices. Then, to enable the control of optical devices and thus open SDN-based control to optically-enabled network segments, a set of protocol requirements must be addressed, as identified by the ONF in [19]. In this matter, extensions to the OF protocol have been proposed, taking profit of the OF concept basis and SDN technology, to allow supporting the control of both electric and optical devices with the same protocol. Some of the most relevant extension proposals are presented next:

- **Addendum to OpenFlow Protocol Specification (v1.0) - Circuit Switching Addendum v0.3** [20]

This proposal presents extensions to support optical circuit switching in the v1.0 specification of the OF protocol. Extensions consider the components and basic functionality of circuit switching optical devices based on switching time-slots, wavelengths and fibres. As it describes the addendum, the extended version of the protocol would also allow the support of hybrid devices with both optical and electrical switching capabilities.

The proposal in particular, intends to establish the basic OpenFlow messages and data models to support the communication of optical related information between the controller and the devices, where the addition of the *OFPT_CFLOW_MOD* and *OFPT_CPORT_STATUS* messages and the definition of the ofp_phy_cport data model are the most relevant introductions among others. The addendum looks forward to become standard and be included in future regular versions of the OF protocol, but as for now, it is still handled only as an extension and requires of a manual implementation of the added/extended optical support, both at the device OF agent and at the OF library and plugin at the controller.

- **Optical Transport Protocol Extension v1.0 - ONF** [21]

The more recent ONF proposal on the other hand, proposes extensions to provide control of optical transport networks and equipment considering the v1.3 and v1.4 specifications of the OF protocol. One of the most important characteristics of this

implementation is the use of the experimental extension mechanism supported in OF to define message and attribute extensions to the protocol specification. In this means, extensions are identified using the experimenter ID assigned to the Optical Transport Working Group in search of the future standardization of the extensions and their integration in new versions of the specification.

At the proposal, extensions related to match/action functions, port attributes and adjacency discovery are defined along with identified future work areas. All in support of Optical Transport Networks (OTN) optical and electrical connections. It is important to mention that at the moment these extensions also require of a manual configuration of the new attributes at the protocol agents/plugins.

- **Optical Circuit Switch OF Protocol Extensions - CALIENT** [22]

Another effort, is the one presented by CALIENT Technologies. In this proposal, extensions covering Optical Circuit Switches (OCS) support for the v1.3 and v1.4 of the OF protocol specification are presented. The effort bases on the limited OCS support at the previously discussed Optical Transport Protocol Extensions given by the ONF, and focuses on enhancing such support to cover all the requirements for OCSs. Between these, the ability of the controller to discover and report adjacent NEs through OF, and the support for OCSs with packet technologies, including port/circuit attributes and port/flow statistics, are addressed. Likewise, as previously presented extensions, these ones also require of implementation.

2.2.5 Open Source SDN Controllers

The present-day SDN market allocates a variety of SDN solutions, where both vendor specific and open source SDN controllers can be found. The later ones can be considered more suitable for research projects such as the one presented in this thesis document due to their openness, adaptability and community driven support. Therefore, an analysis on the most relevant open source controllers used nowadays is considered crucial in a way to discuss aspects such as their architecture approach, scalability, modularity, fault tolerance, interfaces, programming languages and orchestration platforms and community support [23].

Following this, Table 2-1 describes three of the most well-known open source SDN controllers, introducing a brief description of each one.

Table 2-1 Open Source SDN Controllers

SDN Controller	Description
Opendaylight[1]	Industry-led collaborative open source SDN platform hosted by the Linux Foundation and designed for customizing and automating networks at any size/scale.
ONOS[2]	Carrier-grade SDN network operating system designed for high availability, performance, scale-out and well-defined northbound and southbound abstractions and interfaces. Driven by the ONF and Linux Foundation.
Tungsten Fabric[3]	Linux Foundation driven project, previously known as OpenContrail. It provides networking, analytics and security, as well as integration with many cloud technology stacks.

In terms of architecture and modularity, ONOS and Opendaylight (ODL) present centralized architectures which provide low latency reachability between internal modules through the use of built-in communication mechanisms. Besides, their Java-based nature and use of OSGi containers [24] to load software bundles that define modules operation, allows offering a very flexible way to extend/add the controller functionality. Regarding interfaces support, both ONOS and ODL are compatible with plenty of protocols for southbound control (e.g., OF, P4, OVSDB, OF-Config, etc.) as well as with multiple northbound APIs (e.g., RPC, REST, RESTCONF, etc.). In other aspects, both support the use of clusters, to provide fault tolerance in case of master controller failure. Finally, it is important to say that both controllers are backed up by large developer and user communities and are supported by today's most popular orchestration tools.

The case of large scale deployments with multi-segment scenarios, may require ODL and ONOS of following a more distributed approach to handle communication between a set of segment controllers, or to depend on higher-level entities to maintain a centralized view of the complete network. This particular subject is better clarified and discussed in upcoming sections of this chapter.

[1] OpenDaylight <https://www.opendaylight.org/>
[2] ONOS <https://onosproject.org/>
[3] Tungsten Fabric <https://tungsten.io/>

Tungsten Fabric (TF), an evolution of the OpenContrail SDN controller, is an initiative oriented for cloud-grade networks support. It offers a highly-modular platform with micro services based on the use of Docker containers [25], endorsing failure resiliency and availability. It supports different programming languages (e.g., C++, Python, Go, Node.js), southbound protocols (e.g., XMPP, BGP) and northbound interfaces (e.g., Web GUI, REST, plugins). Moreover, it provides cluster scalability and support for controlling different SDN islands/segments.

Within the development process of this thesis. The ODL controller has been selected as the reference SDN controller for the presented research on resource optimization mechanisms. Reasons for this selection, lay on its wide support for communication interfaces, the flexibility to modify/extend controller functionalities, which opens the path to enhance the controller for optical technologies support, and the existence of numerous use cases given by the research community. Besides, the previous experience with the use of this controller and its adoption by Spanish and European project collaborations performed throughout the thesis have also been considered.

2.2.6 Optical Networks Control

After having analysed the extensions proposed for the OF specification to support optical device control, it is important to provide a higher-level view from the SDN architecture sight to consider the control of end-to-end optical/electrical networks, usually partitioned different network segments or administrative domains. To accomplish such end-to-end control, and taking into account the different types of technologies present, a hierarchical organization of the controllers is required [8]. In this sense, a set of child SDN controllers would be in charge of managing each network segment or domain and of exposing their particular resource infrastructure capabilities, either physical or virtual, to a higher-level parent SDN controller. This component in turn, also recognized as a Network Orchestrator, would have a complete view of the whole underlying network infrastructure, and would be able to orchestrate the required orders/actions to each child controller to configure network elements when needed, based on its analysis of the current state of the network. This particular architecture scenario is depicted in Fig. 2-8.

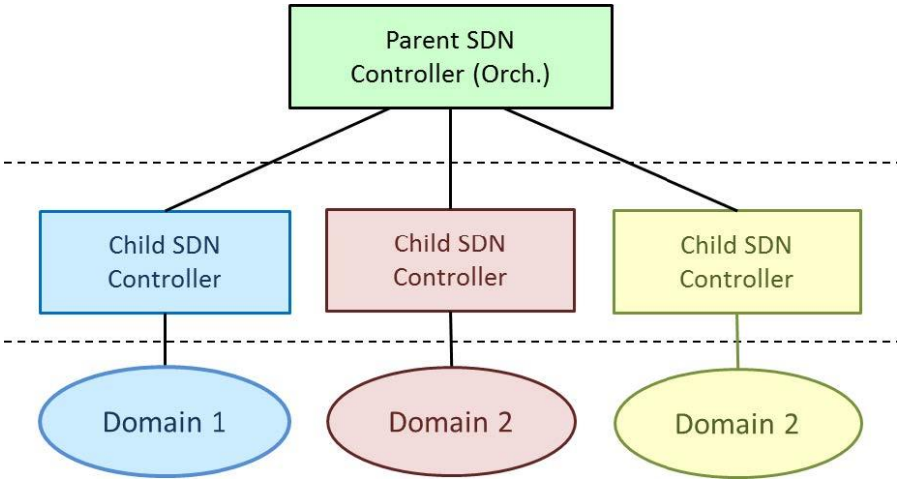


Fig. 2-8. End-to-end Hierarchical SDN Architecture
[Adapted from *ONF, "SDN Architecture for Transport Networks", 2016*]

The introduction of SDN is then considered to be crucial in order to support the control of optical networks [26], while the rise of other modern paradigms such as NFV, MANO, Network Slicing and 5G telecommunications is set to furtherly enhance the orchestration and management of these networks. More detail on how these technologies deliver solutions on this subject is presented on next sections.

2.2.7 Open Issues in Optical SDN

Regarding optical-enabled networks scenarios, there could be noticed some specific issues that still require efforts from the community to head towards optical resource optimization. These are discussed next:

- **Southbound Optical Device Control**

Besides the different set of optical extensions proposed for southbound protocols, there exist a necessity to standardize this communication. Currently, only a set of specific optical network elements and SDN controllers support the control and exposure of optical capabilities, and in most of the cases, this support has to be added manually. Becoming a major issue in the SDN adoption for optical networks.

- **Optical Topology/Link Discovery**

Related to the device control, is the ability to detect all the underlying optical topology, considering all the optical interconnections (i.e., optical links) and the optical network elements. This subject in particular requires further investigation.

- **Virtualization**

Although virtualization has been seeing an increase of deployments in the past few years, scenarios enabled with optical network elements are still having a lack of support when it comes to virtualizing devices to better utilize optical resources. In this regard, the development of applications at the controller level and a more granular control through standardized southbound protocols can be seen as the most critical requirements to accomplish virtualization of optical network elements.

- **Orchestration**

Orchestration is another subject that has grown fast in the last decade, where deployments show Data Centres scenarios as the preferred use cases. In this sense, the introduction of optical technologies at the DCs powered by the SDN technology, could be used to further optimize the usability of the network and the orchestration of resources on top of it.

- **Service Provisioning & Maintenance**

The provisioning of services in inter/intra Data Centre Networks could also be potentiated by means of the SDN technology. Taking profit of a granular control of both optical and electronic based devices and the network configurability given by the controller, to provision more services over the same resources and take less time to establish connections between endpoints at the data plane. Besides, the maintenance of these services could also be enhanced by the analysis of optical related statistics and the ability of performing rapid network reconfigurations.

2.3 Network Function Virtualization

The Network Function Virtualization (NFV) concept, brought by the European Telecommunications Standards Institute (ETSI) and with the support of world leading telecom operators [27]. Introduces the virtualization of network functions, considering specific functionalities being deployed purely in software to bring more flexibility to networking. In this way, a Network Service (NS) can be composed of one to many virtual network functions (VNFs), which could be deployed all over the network infrastructure across different segments/domains, working together to enable the required service functionality.

In this sense, NFV becomes an initiative that intends helping overcome the limitations of current telecom networks in terms of operational challenges and high management costs, aiming to the reduction of Capital Expenditures (CAPEX) and Operational Expenditures (OPEX) [12]. Considering with the virtualization of network functions, a way to facilitate service deployment and maintenance that used to depend on dedicated hardware implementations, hence promoting the use and adoption of cloud technologies.

As in respect of the relation of NFV and SDN. Both technologies can be complementary to each other, but not necessarily dependable. NFV could be implemented without considering an underlying SDN-controlled network. Similarly, SDN could work with different higher-layer applications that do not require fitting in the NFV concept. Despite this fact, the combination of both SDN/NFV can bring substantial benefits to network operators. NFV in turn, could provide and manage the infrastructure, allowing the SDN software to run within and coordinating the deployment of compute and storage resources, while SDN could be focused on the control of network resources. Such combination of technologies, would provide greater agility in network automation and virtualization.

2.3.1 ETSI NFV Architectural Framework

As introduced, NFV encompasses the softwarization of Network Functions (NFs), so these can be implemented and deployed over a NFV Infrastructure (NFVI). With the purpose of accomplishing this, an architectural framework was proposed by the ETSI, as illustrated in Fig. 2-9, considering different requirements to enable constructing and managing NSs/VNFs and to provide them with the required inter communication [28]. These architectural requirements are the following:

- Supporting VNFs operation across different hypervisors and computing resources, in a way to provide access to shared storage, computation and physical/virtual networking.
- Enabling the construction of VNF Forwarding Graphs.
- Providing an interface from NFV management and orchestration software towards other management systems.

- Supporting network services with different requirements to take advantage of virtualization techniques.
- Ensuring security aspects and performance related issues related to virtualization to be addressed.
- Leveraging legacy DC technology.

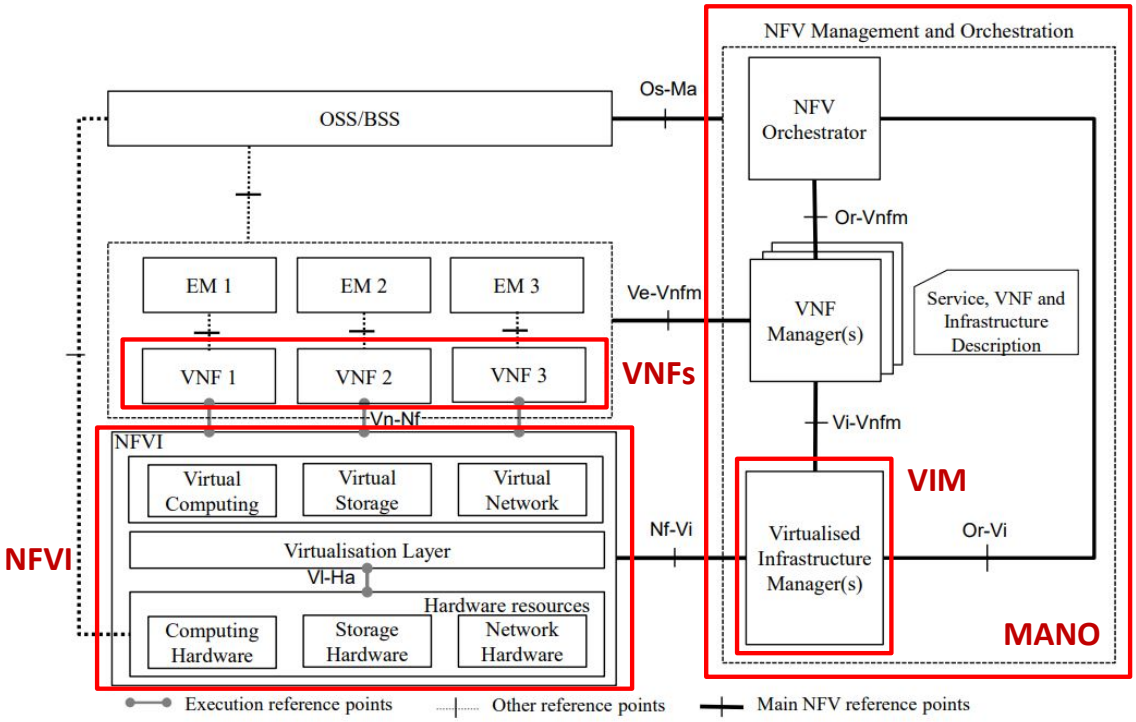


Fig. 2-9. ETSI NFV Architectural Framework

[Adapted from ETSI, *"NFV, Architectural Framework v1.2.1", 2014*]

The ETSI NFV architecture shown above, depicts the functional software modules and the reference points of the proposed framework. Looking from a high-level view, there are three main working domains to be identified:

- **Virtual Network Function (VNF):** Refers to the implementation in software of a network function, which runs over the NFVI.

- **NFV Infrastructure (NFVI):** Consists on the underlying infrastructure filled by a diversity of physical resources, and its correspondent virtualization.

- **Management and Orchestration (MANO):** Handles the orchestration and lifecycle management of NSs/VNFs and resources supporting virtualization. Manages all virtualization-specific management tasks.

Besides these main components, there exist Element Managers (EM) providing direct management to one to many VNFs. As well as management systems directly connected to EMs, VNFs, NFVI and NFV MANO, such as Operation Support Systems (OSS) and Business Support Systems (BSS).

2.3.2 Virtual Network Function

A VNF consists on the virtualization of a Network Function (NF) in a non-virtualized network environment, where a VNF can comprise many internal components that require of instantiation. In this regard, many Virtual Machines (VMs) can compose a single VNF, defining between them its required behaviour. On the other hand, a VNF can also be deployed on a single VM. Regardless the case, the VNF functional behaviour and external interfaces are expected to be similar as in Physical Network Functions (PNF), making no difference on the NF being virtualized or not.

There exist many different use cases for VNFs implementation, considering the virtualization of network elements (e.g., switches, routers, etc.), elements in home networks (e.g., gateway, media server, etc.), conventional network functions (e.g., servers, firewalls, etc.), among many others [29].

2.3.3 NFV Infrastructure

The NFVI comprehends all the hardware and software components that serve to build the infrastructure where VNFs are deployed and executed, representing as well all the space to be managed from the MANO component. In addition, this infrastructure could be implemented over several different locations, known as NFVI Points of Presence (PoPs), where the network providing the connectivity between them is also considered part of the NFV infrastructure.

In regard of hardware components, these mainly comprise computing, storage and network resources that are capable of providing all the required functionality for VNFs implementation, where such process is coordinated through a virtualization layer (e.g., hypervisor). This layer in turn, is in charge of abstracting all the hardware resources, considering their logical partitioning or virtualization, so

VNFs can be deployed over such virtual resources and have their own independent lifecycle decoupled from the underlying hardware. In this way, a VNF composed by a set of VMs could be instantiated over different physical resources.

In the specific case of networking resources, mostly comprised by switching functions, the virtualization of these could be enabled by the SDN technology, where the hypervisor may have a direct communication towards the controller to demand for the configuration of networking requirements. Besides this, many other virtualization mechanisms could also be employed. At the NFVI, two different types of networks can be identified: the network providing connectivity within a PoP (e.g., intra DC network) and the one interconnecting NFVI-PoPs (e.g., transport network), which could belong to the same or different operators.

2.3.4 Management and Orchestration

The MANO software component, part of the NFV framework, provides all the required management operations for the provisioning and maintenance of NSs and VNFs [30]. To achieve this, it is divided in a set of functional modules:

- **Virtual Infrastructure Manager (VIM):** Provides control and management of the NFVI resources (i.e., compute, storage, network).
- **VNF Manager (VNFM):** Handles the lifecycle management of VNFs.
- **NFV Orchestrator (NFVO):** Orchestrates resources across multiple VIMs. Handles the lifecycle management of NSs.

In addition to these modules, the MANO component includes a set of data repositories, such as the NS and VNF catalogues, which allocate all the on-boarded NS deployment templates and VNF packages. NS templates in turn, contain the Network Service Descriptors (NSD), Virtual Link Descriptors (VLD) and VNF Forwarding Graph Descriptors (VNFFGD) that define the composition of each NS. Respectively, VNF packages contain the VNF Descriptors (VNFD), software images, manifest files, etc., that define the instantiation requirements and operational behaviour of each VNF.

Along the catalogues, other repositories exist to hold information about all the deployed NS and VNF instances, as well as for maintaining the current state of the resources at the registered NFVIs (i.e., available, reserved, allocated). Finally, a set of reference points (i.e., Os-Ma, Ve-Vnfm, Nf-Vi) are also present to communicate MANO with the EMs, VNFs, NFVIs and OSSs/BSSs, besides other internal ones for intra MANO communication (i.e., Or-Vnfm, Vi-Vnfm, Or-Vi).

2.3.4.1 Virtualization Infrastructure Manager

A VIM is related to the control and management of NFVI resources, usually at one administrative domain (e.g., all resources in a NFVI-PoP, across several NFVI-PoPs) or within the boundaries of a network segment (e.g., DC segment). Besides, it may support only some specific types of resources (e.g., compute-only, storage-only, networking-only) or manage multiple types of them depending on the case.

Regarding its inter-module communication. Northbound interfaces provided by the VIM (i.e., Or-Vi), expose the NFVI functionality and resources so these can then be configured and deployed through the VIM southbound interfaces (i.e., Nf-Vi, Vi-Vnfm). Considering at times, the use of SDN controllers to delegate network resources configuration. Following this, main VIM functions are summarized next:

- Orchestrating operations related to NFVI resources (e.g., allocation, modification, release, reclamation, etc.).
- Managing the association of virtual to physical resources deployment and keeping an inventory on this allocation relation.
- Supporting VNF Forwarding Graphs, by creating and maintaining links, networks, sub-nets, ports, etc.
- Managing security group policies for network/traffic access control.
- Keeping an information inventory of the NFVI hardware resources.
- Managing the virtualization resource capacity and usage, as well as the validation of software images before its addition to image repositories.
- Collecting performance and fault information of hardware, software and virtualized resources. Forwarding this data when required by other entities.
- Managing catalogues of consumed NFVI virtual resource configurations.

Among the different VIMs solutions available in the open source market, the OpenStack [16] system, appears as one of the most popular software for NFVI management. In particular, it offers a mature cloud computing platform for resource orchestration backed up by the support of a large community of developers and industry members. In this sense, and for the scope of this thesis, OpenStack has been selected as the reference VIM to be used across the different set of presented resource optimization mechanisms.

2.3.4.2 VNF Manager

It is the component responsible for the VFN lifecycle management, where multiple VNFMs can be deployed in a NFV framework environment, but each VNF may only depend on a single VNFM. On the other hand, one VNFM may control several VNFs respectively. Most of the functions performed by the VNFM are related to the instantiation, software update, modification, scaling and termination of VNF instances. Besides other tasks, like the collection of performance measurements and fault management information such as events/faults.

The information describing the operational behaviour and the deployment requirements for a VNF is captured from its VNFD that is stored in the form of a VNF package, and consists of both the computational characteristics defined for the proper VNF instantiation and operation, and the configuration tasks to be performed either at provisioning time or during runtime (i.e., on-demand tasks).

2.3.4.3 NFV Orchestrator

The NFVO, as the name describes, is in charge of orchestrating all the NFVI resources across all registered VIMs. This responsibility corresponds to the resource orchestration functionalities of the NFVO, which are summarized next:

- Handling NFVI resource requests coming from VNFMs.
- Managing NFVI resources across different domains, segments, VIMs.
- Supporting the use of repositories and the information coming from VIMs to maintain awareness of the relationship between VNFs and NFVI resources.

- Managing and enforcing policies for NS and VNF instances.
- Collecting usage information of NFVI resources through the VIMs/VNFMs.

In addition to these functionalities, the NFVO is also in charge of the NS lifecycle management, considering a set of network service orchestration functions:

- Managing NS templates and VNF packages with correspondent validation.
- Handling NS functions during the NS lifecycle such as instantiation, update, modification, scaling, statistics and even collection, termination.
- Managing the instantiation of VNFMs.
- Managing the instantiation of VNFs through VNFMs.
- Assuring the integrity and visibility of NSs during their lifecycle.
- Managing the automation of NS instances.
- Managing and evaluating policies for NS and VNF instances.

To accomplish the previously analysed functionalities, the NFVO connects via southbound interfaces (i.e., Or-Vnfm, Or-Vi) to the VNFMs and VIMs. On the upper side, its northbound interface (i.e., Os-Ma) connects the NFVO with management systems to accept requests for NS/VNF lifecycle management, exchange policies, data analytics, NFVI resource usage information, state information, inventory, etc.

2.3.5 Open Source MANO Controllers

Regarding available MANO solutions, a set of community driven approaches exist, which have acquired popularity over the last years. Between them, it is possible to highlight the Open Source MANO (OSM) project [31], the Open Network Automation Platform (ONAP) [32], the Open Baton framework [33] and the Open Platform for NFV (OPNFV) [34]. All of them currently go through a different stage of development and are competing for becoming the reference component at this level. In this regard, these particular solutions cover principally the functions of the NFVO and VNFMs, and offer support for most well-known VIMs (e.g., OpenStack). Besides, each of them present other different characteristics and internal software modules to enable more functionalities, making it crucial to analyse the best option. Table 2-2, presents a description of each aforementioned NFV-MANO platform.

Table 2-2 Open Source NFV-MANO Software Distributions

SDN Controller	Description
OSM[4]	ETSI-hosted open source platform delivering a production-quality MANO stack for NFV, capable of consuming openly published information models, open source available, suitable for all VNFs, operationally significant and VIM-independent.
ONAP[5]	Linux Foundation driven project, provides a platform for real-time, policy-driven orchestration and automation of physical and virtual NFs, enabling software, network, IT and cloud providers and developers to rapidly automate new services and support complete lifecycle management.
Open Baton[6]	Extensible and customizable NFV MANO-compliant framework, capable of orchestrating NSs across heterogeneous NFVIs, integrating on a plug-and-play basis with different VIMs and NFVMs.
OPNFV[7]	Project and community that facilitates a common NFVI, continuous integration (CI) with upstream projects, stand-alone testing tool sets, and a compliance and verification program for industry-wide testing and integration to accelerate the transformation of enterprise and service provider networks.

Among these solutions, OSM and ONAP in particular, have seen lately plenty of use case implementations and have been considered as the preferred options for NFV-MANO deployments [35]. Nevertheless, they both present very different characteristics and requirements. OSM in this regard, offers a very light deployment stack capable of running on a single PC. It has a very user-friendly web interface and supports a powerful Command Line Interface (CLI) and northbound REST API. It also handles multiple VIM support and has built-in performance monitoring that can be open/adapted to external clients. It represents a very useful tool especially for research and development scenarios.

ONAP in turn, has higher computational requirements, needing a set of servers for its deployment. Moreover, it is not so user-friendly and has a learning curve higher than OSM. Besides these aspects, it offers a very powerful platform for performance monitoring, considering also its support for Kubernetes container orchestration system [36] and for managing a multi-user supported environment.

[4] Open Source MANO project <https://osm.etsi.org/>
[5] Open Network Automation Platform <https://www.onap.org/>
[6] Open Baton <https://openbaton.github.io/>
[7] Open Platform for NFV <https://www.opnfv.org/>

Open Baton and OPNFV are also solutions that offer very flexible platforms and other set of functionalities for NFV-MANO. However, taking into consideration the added support for Network Slicing on new OSM releases and the consolidation of its plugin models for inter/intra DCN scenarios [37], OSM has been selected as the reference MANO platform for the development of resource optimization mechanisms throughout this thesis. More detail on the Network Slicing concept and its relation to NFV-MANO is given in the next section.

2.4 Network Slicing

Network Slicing is a 5G enabling technology, introduced by the Next Generation Mobile Networks (NGMN) Alliance, to leverage legacy network deployments towards the upcoming requirements of next-generation networks [38]. In this regard, Network Slicing entails that a physical network can be partitioned (i.e., sliced), either physically or virtually, so services with different requirements can be deployed and run over the same underlying physical infrastructure. A network slice then, can be understood as a collection of resources (i.e., computing, storage, network) and a set of configured network (virtual) functions/applications, that work together to meet the requirements set by service clients for a specific use case.

A single network slice, can allocate multiple services for a particular tenant, where each tenant would be deploying these services over an infrastructure shared with other tenants. To guarantee the normal service operation of all these services, and secure the client-agreed performance levels (e.g., Service Level Agreements (SLA), Key Performance Indicators (KPI)), maintaining a level of coexistence and/or isolation between slices is required. To accomplish this, and considering that end-to-end network slicing may cover many different network segments or domains, management entities at a higher level are also needed to handle the lifecycle of all running services and slices, and the relationships between them.

2.4.1 The Network Slicing Concept

The original concept of Network Slicing, introduced by the NGMN in [39], considers the definition of three layers; the service instance layer, the network slice instance

layer and the resource layer. Fig. 2-10 then, depicts such concept along with the different components that operate at each layer.

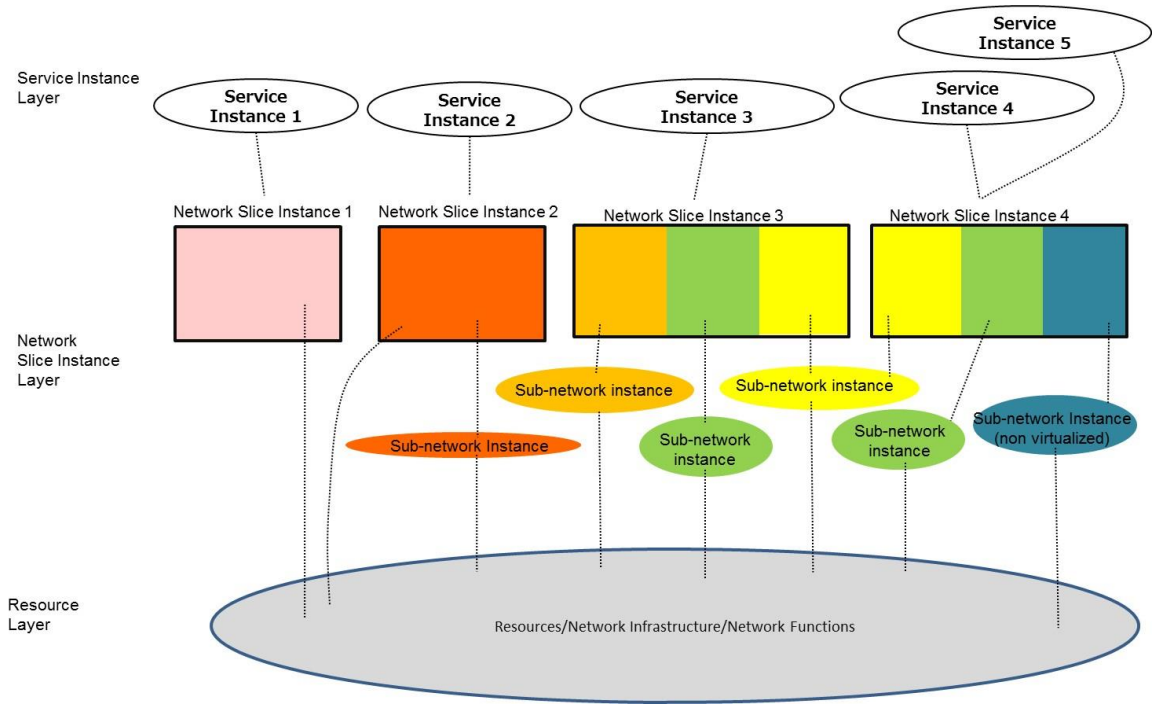


Fig. 2-10. Network Slicing Concept

[Adapted from *NGMN, "Description of the Network Slicing Concept", 2016*]

At the service layer, different services are to be supported, represented in a set of Service Instances. Each service in turn, runs over resources provided by a Network Slice Instance (NSI) at the middle layer. To set up a NSI, a Network Slice Blueprint is also required, where multiple services may be allocated to a single NSI. Besides, lower-level Network Slice Sub-Network Instances (NSSI) may exist, and could be shared among different NSIs. Each NSSI respectively, may contain a set of network functions and resources. More detail on each component is given next:

- **Service Instance:** End-user service instance or business service instance that runs over a single NSI, which could also be shared with other services.

- **Network Slice Instance (NSI):** Set of network functions and resources allocated to fulfil the requirements set by the service instances.

- **Network Slice Blueprint:** Description of the NSI instantiation requirements.

- **Network Slice Sub-Network Instance (NSSI):** Similar to an NSI, but at a lower level, comprises a set of network functions and resources.
- **Network Slice Sub-Network Blueprint:** Description of NSSI requirements.
- **Network Function (NF):** Network processing functions, physical or virtual.
- **Physical/Logical Resources:** Network, computing or storage assets, physical or partitioned. Dedicated to a NF or shared among different NFs.

2.4.2 Network Slicing Management Functions

As introduced previously, the analysed components require to be managed from higher level entities to support its deployment and operation. These entities are named Management Functions (MF), and are used by business clients, service providers or network operators to manage the Network Slicing components [38]. Fig. 2-11 depicts the principal MFs, considering the Communication Service Management Function (CSMF), the Network Slice Management Function (NSMF) and the Network Slice Subnet Management Function (NSSMF). Besides these, a Network Function Management Function (NFMF) also exists at the lowest level. Given detail on the provided functionalities by each MF follows.

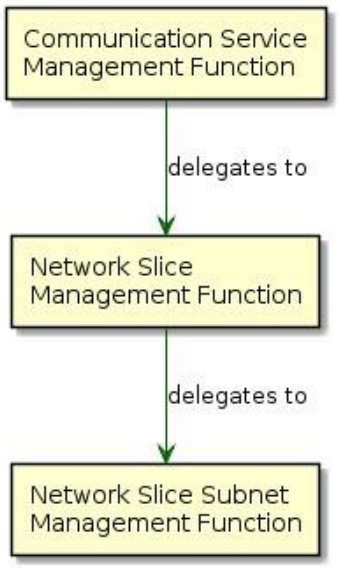


Fig. 2-11. Network Slicing Management Functions

[Adapted from *3GPP, "Study of Management and Orchestration of Network Slicing for Next Generation Network", 2018*]

- **CSMF:** Manages communication services. Receives service requirements coming from service customers and translates them into network slice requirements (e.g., network type, network capacity, QoS). Delegates the instantiation and management of NSIs to the NSMF.

- **NSMF:** Manages and orchestrates NSIs. Derives NSSI requirements from the NSI requirements. Delegates the management of NSSIs to the NSSMF.

- **NSSMF:** Manages and orchestrates NSSIs. Derives NFs requirements from NSSI requirements. Delegates the management of NFs to the NFMF.

- **NFMF:** Manages NFs. These could either be Physical Network Functions (PNF) or Virtual Network Functions (VNF).

Following this hierarchy, each MF interacts with the subsequent higher or lower level MF, where besides the delegation of components orchestration, such interaction considers as well the communication of management information related to Configuration Management (CM), Fault Management (FM), Performance Management (PM), and Lifecycle Management (LCM) of the resources.

2.4.3 Information Model Correlation to NFV-MANO

Looking towards the upcoming implementation of 5G networks, it is important to clarify the relation of Network Slicing and NFV MANO components/resources. Regarding this, in 3GPP Network Resource Model (NRM), the use of NSIs, NSSIs, and NFs is defined for Network Slicing [40], while in ETSI Information Model, the use of Network Services (NS) and VNFs is set for enabling NFV [41]. Considering this, and the fact that both technologies should be aligned to enable 5G, it becomes necessary to make a direct correlation between both information models.

ETSI provides an analysis on this particular subject [42], represented in Fig. 2-12. The required correlation in this case, is set between models at the NSSI to NS level showing a direct mapping of these resources. Besides, a NF to VNF correlation is also presented. Following this approach, the starting request for NSI

provisioning, after being decomposed into a set of NSSIs, can be in turn mapped to the NFV-MANO model, in the form of NSs. Furthermore, the NFs derived from each NSSI, could also be mapped to correspondent VNFs.

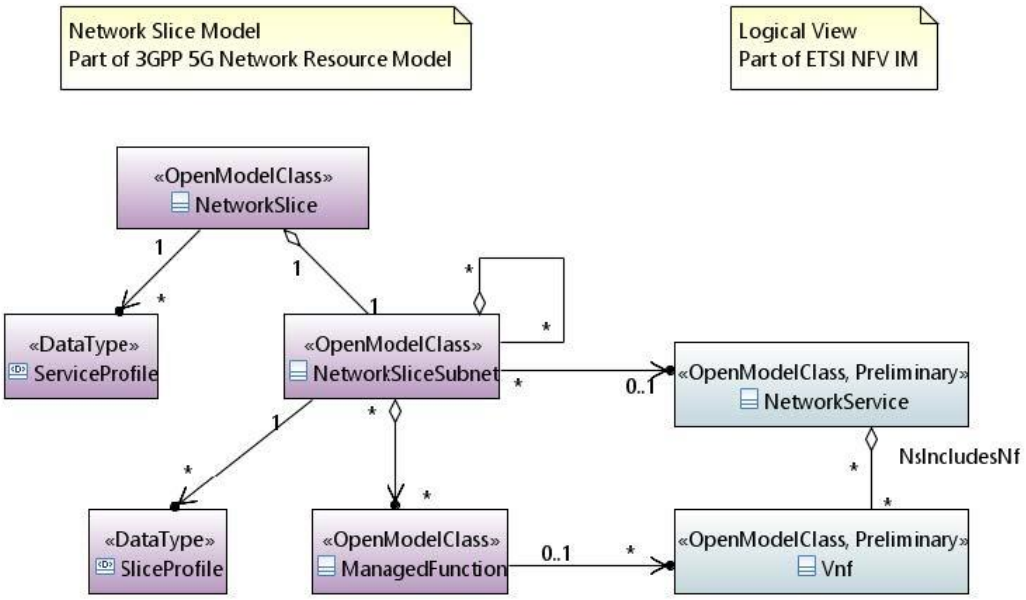


Fig. 2-12. 3GPP NRM to ETSI NFV Information Model Correlation

[Adapted from *ETSI, "Information Modelling; Report on External Touchpoints related to NFV Information Model", 2019*]

2.4.4 Interfaces to NFV-MANO

Taking as a base the model correlation previously described, the MFs and the components of NFV-MANO should also follow the necessary coordination in order to support both technological concepts. To interconnect these components then, and thus provide a communication channel between 3GPP and ETSI frameworks, the use of well-defined management interfaces is required.

A conjunctive work from these organizations, presents a way to use the existent interfaces proposed for NFV-MANO to connect its components to the MFs defined for Network Slicing [43]. With this approach, the NSSMF would be able to consume the LCM services of NSs and VNFs from the NFVO, connecting to the NFVO northbound interface (i.e., Os-Ma). Moreover, the NFMF would be capable of handling application level management, including CM, FM and PM of deployed VNFs in coordination with the VNFM, via their common interface (i.e., Ve-Vnfm).

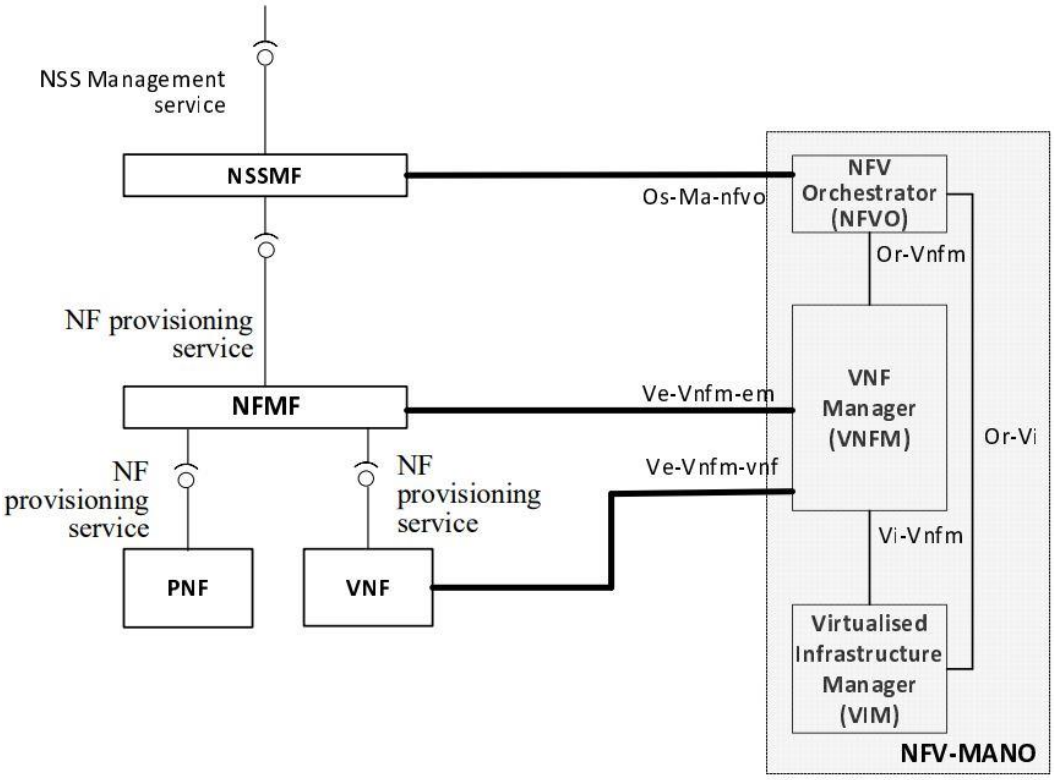


Fig. 2-13. Network Slicing MFs with Interface to NFV-MANO
[Adapted from *3GPP/ETSI, "5G; Management and orchestration; Architecture framework", 2018*]

Fig. 2-13 depicted above, shows the discussed interface relationship. In this way, after the NSMF receives the request for NSI instantiation coming from the CSMF. It derives the NSSI requests and delegates their deployment to the NSSMF. This one in turn, now directly interfaced with the NFVO, is capable of coordinating with NFV-MANO the provisioning and management of NSSIs/NSs. At a lower level, the interaction between NFMF and VNFM, also enables the common management of deployed VNFs, considering their required configuration and maintenance.

2.4.5 Slice Composition

The Slice Composition concept (i.e., "slice-cum-slice"), was introduced by the 5G Infrastructure Public Private Partnership (5G-PPP) to enable composing a slice out of individual slices [44]. In this way, an end-to-end NSI can be composed by the combination of a set of segment-specific or domain-specific NSSIs. By following this approach, flexibility to the NSI operation and provisioning is added, where all required configurations and actions at each NSSI are set to run independently from

the ones at other NSSIs, thus facilitating not only the provisioning of the required NSI resources, but also the modification of functionalities and the triggering of particular maintenance actions at specific points of the NSI. Moreover, the chance to accurately identify or isolate slice performance threatening failures or malfunctions is also highly improved.

Fig. 2-14 describes then, an example for NSI provisioning through slice composition. NSI-1 in this case, is composed by two NFs, more specifically VNF1 and VNF2, which in turn are interconnected via Data and Management networks. As the availability of resource functionalities may require both VNFs to be deployed on different network segments/domains, NSI-1 can be furtherly de-composed into NSSI-1 and NSSI-2. Another reason could be the requirement from the service to allocate some of its functions closer to the user (e.g., at Edge DC), and others on other parts of the network (e.g., at Core DC). The definition of NSSI-1 in turn, would include VNF1 and its interfaces towards the external networks, while NSSI-2 would define similar characteristics for VNF2. The combination of NSSI-1 and NSSI-2 plus the network configuration required to interconnect them, usually done via a Wide Area Network Infrastructure Manager (WIM), would enable composing the resources and overall functionality required for the provisioning of NSI-1.

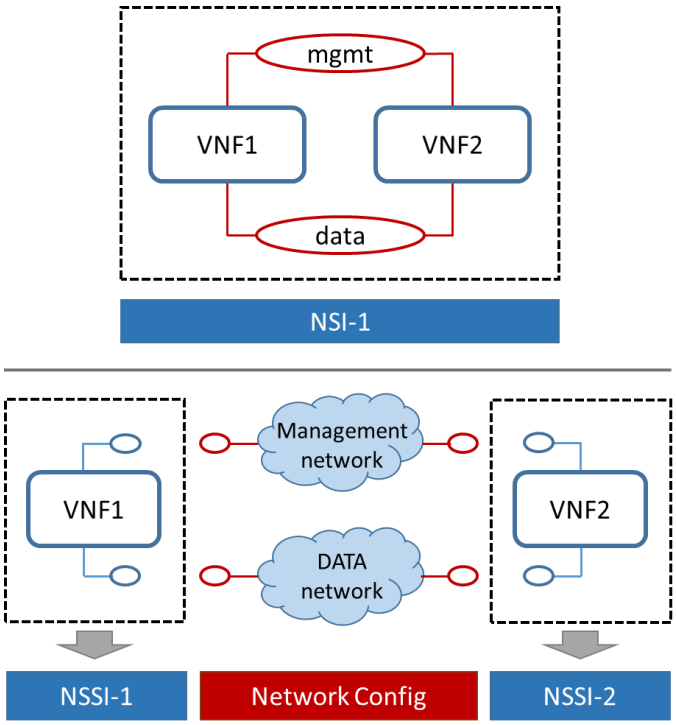


Fig. 2-14. NSI to NSSI provisioning through Slice Composition

Another different concept, is the one related to recursive slicing (i.e., “slice-in-slice”), in which the already partitioned resources corresponding to a slice can be furtherly sliced as required, thus creating slices within slices. In principle, this approach only enables sub-slices to obtain resources from parent slices, but not functionalities, then keeping sub-slices completely isolated. As the concept further evolves, it may include in the future a consideration for the interaction between sub-slices and parent slices, enabling the possibility to exposing functionalities.

2.5 5G Paradigm

Upon the rise of a new era in telecommunications, the fifth generation of mobile technology (5G) is introduced to address the new challenges and requirements brought by a currently changing network paradigm. These changes in turn, driven by the development towards a fully mobile and connected society, relate to different aspects, such as the significant growth in connectivity density, the variety of introduced 5G use cases, the expectation for new business models, the increase of network traffic thanks the rise of machine-to-machine and machine-to-human applications, the definition of new traffic patterns, among many others [1].

To cope with the highly changing network environment and guarantee the adoption of 5G enabling technologies, a set of concepts have to be acknowledged and standardized by existing collaboration forums and Standards Development Organizations (SDO). In this regard, a 5G compliant architecture considering all required technologies, besides the definition of use cases, types of services and requirements given for 5G communications must be defined.

2.5.1 5G Architecture

The 5G-PPP, driven by the European Commission and European industry members on Information and Communication Technologies (ICT), delivered a first view on the architecture required for 5G [2]. Its definition, bases on the need for a highly flexible and programmable end-to-end (E2E) infrastructure, that requires to provide awareness not only at the application and service level, but also in location and context. In this regard, some specific aspects are considered:

- Implementing Network Slicing in a cost efficient manner.
- Addressing end user and operational services.
- Supporting native softwarization.
- Integrating computation and communication.
- Integrating heterogeneous technologies (e.g., fixed and wired)

To accomplish this, the adoption of new technologies and the introduction of novel mechanisms is required at every network domain and layer. Furthermore, the requirement for an approach to support the orchestration and management of 5G services, in both provisioning and operation stages, is also considered crucial.

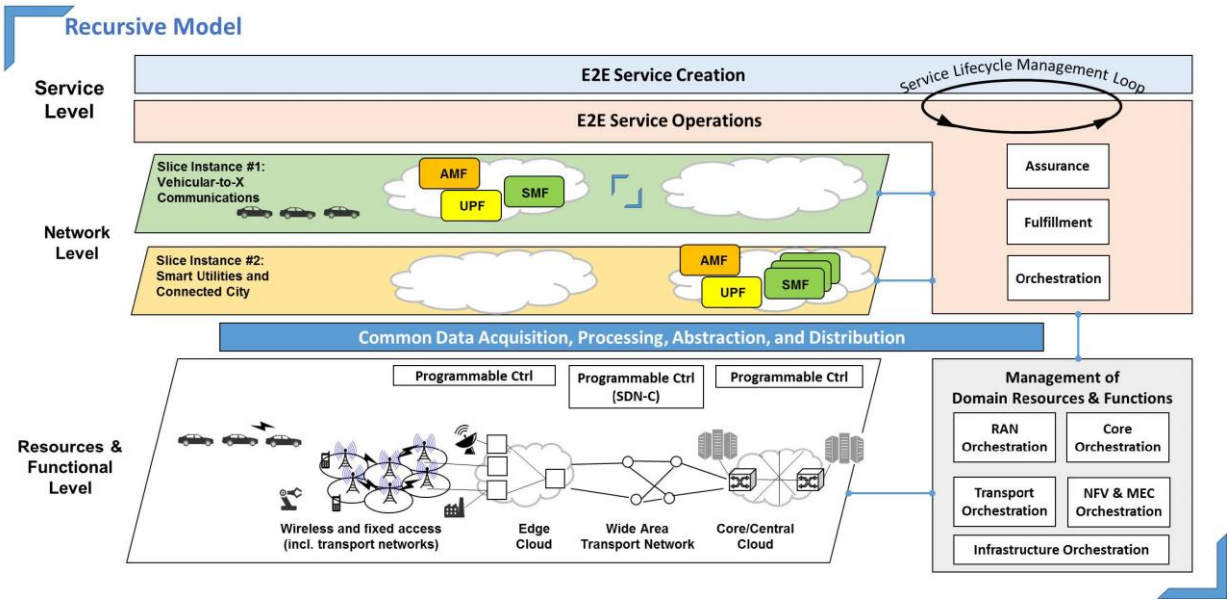


Fig. 2-15. 5G Overall Architecture
[Adapted from *5G-PPP, "5G Architecture White Paper", 2019*]

The architectural proposal that follows the previous requirements analysis consists of an architecture comprised of three levels [45], as depicted in Fig. 2-15. At the Service Level, the lifecycle management of E2E services is defined, as well as the orchestration of all service related operations. The introduction of NFV and SDN technologies is fundamental in this sense, to allow the virtualization of network functions and to open network resources programmability. In addition, the adoption of the Network Slicing concept in 5G, further optimizes the use of all underlying resources, by partitioning the Network Level on demand.

The E2E operations managed from the Service Level, are coordinated with elements at the Management of Domain Resources and Functions, which provide specific control over particular network domains or segments. The components providing these functionalities, may also allow executing specific policies and rules on the Resource and Functional Level. Finally, a platform for managing data is also defined in the architecture, with the purpose of providing a common access point for all layers to the related subscribers, services, slices and resources data.

2.5.2 Role of Optical Networks in 5G

The road to 5G, includes the integration of different technologies (i.e., mobile, fixed, satellite and optical) to support the different service types defined for 5G verticals. Between them, the introduction of optical technologies in access networks, DCNs and the use of high-capacity optical transport networks to connect massively distributed cloud computing and storage centres, such as Core DCs and Edge DCs, becomes fundamental to support the requirements and capabilities analysed previously. Optical technologies in turn, have been identified as the leading solution for the high-speed, low-latency connectivity requirements in 5G, presenting an evolution in all their hardware/software related areas to accommodate optical-enabled scenarios to the 5G paradigm and the use of SDN-control [26, 46].

In this thesis, special focus is set on the distributed and strategic allocation of functions on specific parts of the network according to service needs, considering its relation to the different set of characteristics provided by optically enabled and inter-connected DCs, such as the high computational capabilities and long-term response times of Core DCs and the lower capabilities but faster response times given on Edge DCs. In this way, services composed by a different set of functions are most likely to be provisioned over multiple DC segments, then requiring of an end-to-end management and orchestration platform capable of controlling both optical and electrical technologies at intra/inter DCNs, to provide support for service provisioning and maintenance during their whole lifecycle, and taking also into account the required partitioning of the underlying infrastructure to guarantee service functionality.

2.5.3 5G Service Types

Considering the variety of services given for existent market verticals and industry sectors, the definition of service types in the context of 5G has been proposed [44, 47]. Based on the support for different Key Performance Indicators (KPI) related to specific service requirements, three types have been defined:

- **Enhanced Mobile Broadband (eMBB):** Also known as Extreme Mobile Broadband. Addresses human-centric use cases for access to multi-media content, services and data, considering faster and more uniform data rates, lower latency and a reduced per-bit cost in order to cope with high density, high traffic capacity, seamless coverage and high mobility requirements.

- **Ultra-Reliable and Low Latency Communications (URLLC):** Comprises new services enabled by high throughput, ultra-reliable and ultra-available low latency links. Such services include; remote medical procedures, control of critical infrastructures, autonomous vehicles, etc.

- **Massive Machine Type Communications (mMTC):** Refers to a massive device connectivity density required in networks comprised by large number of low-cost, long-battery-life connected devices, usually transmitting a low volume of non-delay-sensitive data. Demanding to scale down data rates, mobility and power to support low-cost solutions for these cases.

Besides these defined types, additional ones could be added in the future as new use cases are yet to emerge. This ability to flexibly adapt to changes in the service requirements spectrum is a necessary feature in next-generation 5G systems.

2.5.4 5G Verticals

In the path towards next generation networks implementation, 5G technologies intend to provide a flexible platform capable of integrating all the different business cases and models, denominating them as industry verticals or 5G verticals. The importance given to verticals comes from the fact that these will be responsible for most of the traffic and data flowing through 5G networks, thanks to the advantages

and capabilities introduced in 5G, besides other features based on the 5G cloud-native principles, that empower verticals to deliver quality services in short time. In this sense, Network Slicing will also become crucial to allow the 5G infrastructure sharing between all the industry verticals, letting them coexist in a common network environment while keeping them active in the market competition. Moreover, each vertical presents a variety of specific requirements related to the use of particular 5G service types, as introduced in previous section, then becoming necessary to analyse each vertical independently [48]. More detail on verticals is given next:

- **eHealth Vertical Sector:** This sector identifies a set of use cases, such as; Assets and intervention management in Hospitals, Robotics (i.e., remote surgery, assisted living), Remote monitoring of health or wellness data, and Smarter medication. Requirements for this vertical relate to all three defined service types, with special focus on URLLC.

- **Factories-of-the-future Vertical Sector:** Use cases identified for this sector are; Time-critical process optimization and control, Non time-critical factory automation, Remote maintenance and control of digital factories, Seamless intra/inter enterprise communication and Connected goods. The vertical requirements in this case relate mostly to mMTC and URLLC.

- **Energy Vertical Sector:** Main use cases for the energy sector are; Grid access, Grid backhaul and Grid backbone. The role of 5G in this vertical is crucial to enable the two-way energy flow of the “smart grid”, implicating needs for service requirements related to eMBB and URLLC.

- **Automotive Vertical Sector:** Automotive industry also presents a set of use cases; Automated driving, Road safety and traffic efficiency services, Digitalization of transport and logistics, Intelligent navigation and Information society on the road. KPIs at this specific vertical relate to all the defined service types, especially to URLLC and mMTC.

- **Media & Entertainment Vertical Sector:** 5G should enable a set of use cases related to this vertical, such as; Ultra high fidelity media, On-site live event experience, User generated content and machine generated content,

immersive and integrated media, Cooperative media production and Collaborative gaming. All related to requirements of URLLC and eMBB.

The goal in the 5G ecosystem, is set to allow multi-tenancy and multi-service support, so all the use cases presented for industry verticals can be provisioned and operated over a 5G-powered underlying network and computing infrastructure, comprising multiple network technologies and providing end-to-end partitioning capabilities to address the demands of each particular vertical or use case.

2.5.5 5G Capabilities

The introduction of the 5G Vision by the 5G-PPP, came along with a set of targeted capabilities for its future deployment, focused on the advantages of 5G systems in respect of previous generations [49]. The defined 5G targets look forward to:

- 1000 x in mobile data volume per geographical area.
- 1000 x in number of connected devices (density ≥ 1 million terminals/km^2).
- 100 x in user data rate (terminal data rate ≥ 1 Gb/s for cloud apps).
- 1/10 x in energy consumption.
- 1/5 x in end-to-end latency (delays ≤ 5 milliseconds).
- 1/5 x in network management OPEX.
- 1/1000 x in service deployment time (deployment ≤ 90 minutes).
- Guaranteed user data rate ≥ 50 Mb/s.
- Support of IoT terminals ≥ 1 trillion.
- Service reliability ≥ 99.999% in specific mission critical services.
- Mobility support at speed ≥ 500 km/h for ground transportation.
- Accuracy at outdoor terminal location ≤ 1 meter.

2.5.6 5G Requirements

Considering the targeted capabilities and the specific service types, use cases and verticals defined for 5G, the requirements specifications that need to be addressed in order to support them, represent a great evolution in network technologies, in comparison to 4G specifications [48]. More detail on these requirements follows:

- **Data Rate:** Refers to the high bit rate required for apps to function correctly, with values in the order of Gb/s as in the Media & Entertainment vertical.

- **Mobility:** Maximum speed under which certain reliability must be achieved, with values in the order of 500 km/h as in Automotive and eHealth verticals.

- **E2E Latency:** Maximum acceptable time from packet generation at source to packet reception at destination. Most demanding use cases correspond to Factories vertical with values of 100 μs to 10 ms.

- **Density:** Maximum total number of connected devices per unit area with 5G capabilities, in Factories vertical up to 100/m^2.

- **Reliability:** Maximum tolerable packet loss rate considering maximum end-to-end latency for apps, eHealth vertical with values up to 99.99999%.

- **Position Accuracy:** Maximum positioning error accepted by apps, with values as in Automotive vertical in the order of 0.3 m.

- **Coverage:** Area within apps should properly function while their specified requirements are reached. Different for all verticals according to capabilities.

- **Autonomy:** Component operational life without supplied power, related to battery life, battery load capacity and energy efficiency.

- **Security:** Protection of resources considering security characteristics such as authentication, data confidentiality, data integrity, access-control, etc.

- **Data Volume:** Amount of transferred data, in both downlink and uplink, in an interval of time at a specific area. 5G targets 10 Tb/s/km^2.

- **Service Deployment Time:** Time required to set up end-to-end network slices across the infrastructure, considering the fast deployment of services (targeted around 90 minutes), and network level requirements guarantees.

- **Identity:** Capability to identify sources and recognize entities in the system.

Chapter 3. Thesis Objectives

This section, presents the objectives defined for the development of the thesis, considering, for their definition, the formerly discussed state of the art technologies and the introduction given on 5G optical inter/intra Data Centre Networks (DCNs).

The main objective of this thesis, as specified in the introduction chapters, focuses on the resource usage optimization at optically enabled DCNs, following the context of the 5G ecosystem, by means of implementing techniques through novel management and orchestration architectures powered by SDN. In this regard, specific objectives considering the fundamental parts of this research are defined next, to make clear the process of achieving the main thesis objective.

3.1 SDN-based Control of Optical Devices

The correct control of optical devices and the exposure of their features to the SDN controller is considered a fundamental objective, as such control would allow the proper use of resource optimization techniques in optical networks. Then, by controlling optical devices from the SDN controller, the exposed information of the devices would be made available to any application either inside or outside the controller, allowing them to trigger operations over the underlying network. Such operations, in the concern of this thesis, would seek for resource management in particular optical network scenarios (i.e., intra DCN, inter DCN).

3.1.1 Southbound Protocol Extensions for Optical Support

The use of the OpenFlow protocol in electronic-based networks will be taken as a reference for the southbound interface, with the goal set to achieving this particular level of control also in optical networks by using the same concepts. The protocol in this case, would require extensions or modifications, such as the ones presented in *Section 2.2.4.1*, in order to support the different attributes and operations seen in optical devices. Such changes would affect mostly the design of the protocol plugin/library at the controller level and the protocol agent at the devices.

3.1.2 Design and extension of SDN Applications

Besides the modification of protocol libraries/plugins, the development of new SDN applications at the controller level, related to the control and operation of optical devices, would be also required. Moreover, other existent applications may also require modification or extensions to support controlling both electric and optical technologies. These applications will enable SDN controllers to provide granular control and expose the capabilities of hybrid or fully optical DCNs.

3.1.3 Assessment of Device Control in Optical Network Scenarios

In order to guarantee the proper control of optical devices, all these optical protocol extensions and introduced/extended SDN applications must be demonstrated and tested experimentally. For this thesis, the use of emulated DCN testbed scenarios is considered, in view of the flexibility given by components emulated in software (e.g., SDN device agents, virtual switches) to test different device configurations and to allow the extraction of specifically case-adapted device information.

3.2 Dynamic SDN-based Optical Topology Discovery

This objective is focused on the correct mapping at the controller level of the underlying optical topology, as it is identified to be decisive in order to guarantee the functionality of control-layer applications and the proper exposure of data plane resources towards upper layers (i.e. application, orchestration). On this regard, miscorrelations between the topology known by the SDN controller and the actual physical topology could generate not only an incorrect global visualization of how the resources are allocated and connected between each other, but also become the root source of errors in SDN applications when deploying solutions over the network using such incorrect topology data.

3.2.1 Design of New Methods for Topology Discovery

To enable the SDN controller-based gathering of accurate topology information, either from hybrid or fully optical networks, it would be necessary first to understand the existent techniques for optical resources discovery. As nowadays, this process

usually depends on manual or static configurations, which tend to result in possible misconfigurations and tedious work in large scale networks.

Considering this, the design of a discovery mechanisms capable of precisely recognizing optical nodes and links must be proposed. In the particular case of this thesis, the optical topology is intended to be managed dynamically by means of specific OpenFlow messages, extended for optical discovery. By these means, a well-designed SDN-based discovery method would pave the way to the correct disposition of operations over the controlled optical networks.

3.2.2 Implementation

To allow the designed discovery method to operate, both extensions to the OF protocol and the design/extension of SDN applications must be considered. Its implementation then, will directly depend on the ability of the controller to handle and retrieve such specific optical information, and of making it available to clients and applications. Furthermore, at the optical physical level, some technique may also be required to properly recognize the configured optical links. In order to test this, it would also become necessary to emulate specific optical device behaviour in software, to then prove the complete topology discovery workflow.

3.2.3 Experimental Assessment

The emulated DCN testbed scenario considered in the previously analysed optical devices control objective, will also represent a valid way to validate the proposed optical topology discovery method. In this way, different network topologies can be configured and proved by correctly recognizing their characteristics (i.e., optical links and nodes) from the controller, by means of the implemented mechanisms.

In case several techniques are provided, these emulated scenarios could also help with the benchmarking of the different discovery methods, allowing to compare the advantages/disadvantages between them. In this way, the validation in software of the most accurate method to discover optical resources, would open the path for its future implementation on real case scenarios for further testing.

3.3 Optimized 5G Service Provisioning and Maintenance in Optical Data Centre Networks

The use of SDN control in optical-enabled DCNs and the introduction of NFV and Network Slicing technologies enables the definition of new ways for handling 5G service provisioning and maintenance. Taking in consideration the use of these technologies, this objective sets as a goal to define and prove a multi-segment 5G architecture capable of optimizing the management and orchestration tasks related to service deployment and the operations required to guarantee their expected performance during the service lifecycle.

3.3.1 Study on Community Driven Approaches for 5G Architecture

Before jumping into the architecture proposal, it would be necessary to analyse the already existent approaches towards the definition of the 5G architecture. In this sense, all the integration steps between the different technologies must be studied, considering in particular their specific definition of management components. Besides, the proposals of other community related projects must also be reviewed to understand the still existing gaps towards the consensus for a common solution.

3.3.2 Service Maintenance

Optimizations would also require looking forward into the use of specific techniques to secure service performance levels, in case these become affected by potential service degradations during runtime. In this matter, sensors and actuators are the denominated components in 5G to help accomplish such guarantees, by allowing the gathering of KPI-related statistics data from running services to then execute the necessary corrective operations. An analysis on the optimization possibilities given by the use of specific sensors and actuators is then required.

3.3.3 Design of Required Software Additions

Considering all the technologies, data models, architectural layers and components available in 5G networks, the introduction of additional software may be required to provide a seamless integration between all these aspects. In this

sense, and considering the Network Slicing to NFV model and interface correlation, components or extensions added to the proposal should be able to further enhance the end-to-end coordination process for 5G service provisioning and maintenance.

3.3.4 Architecture Proposal and Definition

The 5G scenario, is comprehended over a set of network segments that provide the necessary resources for network service deployment, where each segment in particular may be related to specific management and orchestration functions. In this thesis, besides intra-DCN optimization, the focus is also set on optimizing the management of resources providing the connection between DCNs (i.e., inter-DCN networking). Therefore, the definition of a multi-segment architecture, in the context of 5G, is required to setup all the designed enhancing functionalities and integration modules at these scenarios, considering all the specific characteristics of each segment involved (e.g., DC segment, Core segment, Metro segment).

The architectural proposal is set to be defined in two specific versions according to the particular service stage (i.e., provisioning, maintenance).

3.3.5 Experimental Assessment in Multi-Segment Testbed Scenario

The proposed architecture, along all its introduced functionalities and software additions, will require of assessment at the experimental level. In this matter, a testbed considering all the parts of the 5G architecture related to inter-DCN optical network scenarios would be required to allow proving implementations related to both service provisioning and maintenance stages. To enable the configuration of the testbed, multiple components are considered, such as network emulation tools, software interfaces, open source SDN controllers, open source orchestrators, open source MANO components, added software modules, among others.

PART II

OPTIMIZATION
IN INTRA-DC NETWORKS

Chapter 4. SDN-Control at intra-DCNs

This chapter describes the efforts added at the intra-DCN scenario during the development of the thesis, associated directly to the design and implementation of techniques at the SDN control level for resource usage optimization.

The provisioning of Virtual Data Centres (VDC) in this case, also seen as DC network slicing, is the key use case selected for the characterization of a novel DCN architecture based on the introduced software additions and extensions.

4.1 Virtual Data Centres

The VDC service in turn, is a result of the introduction of the Infrastructure as a service (IaaS) paradigm [50], where the main concept of IaaS relies on the offering of physical infrastructures as a service so these can be used by third parties. By these means, clients could be provided with customized infrastructures (i.e., VDC) according to the network, storage and computational resources they require. In this matter, virtualization is employed to enable the offering of the VDC service [51], which in turn allows slicing the underlying physical infrastructure of the DC segment into virtual resources that are used to create DCN sub-slices to be offered to clients.

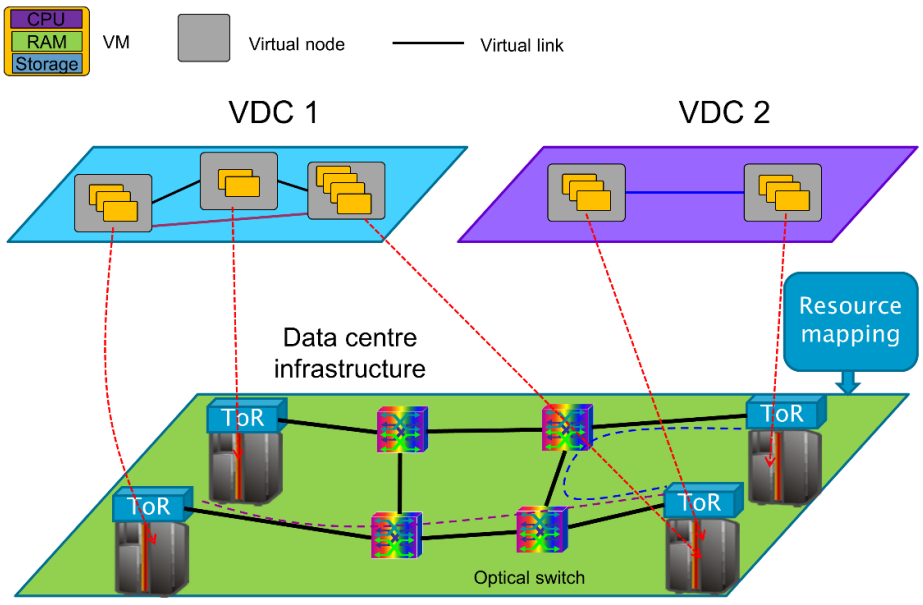


Fig. 4-1. VDC Service Scenario

As exposed in Fig. 4-1, multiple tenants by means of the VDC service can rent portions of the provider's physical infrastructure, which at the same time can be leased in further portions to other clients. These portions are seen as network sub-slices, composed by a set of DC virtual resources (i.e., virtual machines, virtual nodes, virtual links), provisioned at the time the service is deployed over the shared infrastructure. In this specific case, a more granular control of the optical devices at the data plane would further boost the implementation of resource optimization techniques considering the increased usage of optical technologies in DCNs.

To accomplish an enhanced use of sub-slices in DCNs, the integration of SDN and optical technologies is fundamentally required. Then, a novel architecture capable of controlling both optical and electric based technologies, as well as managing the required higher level coordination to trigger VDC configurations has to be provided. In this regard, the role of the software components added in the scope of this thesis, relates specifically to the control level functions that coordinate the deployment of each sub-slice, with more data to be given on next section.

4.2 SDN-Based intra-DCN Architecture

The presented intra-DC architecture, based on the proposal of the FP7 COSIGN European project [52], considers the usage of enhanced software and hardware components across all architectural planes (i.e., data, control, orchestration). Fig. 4-2 provided in this section further describes elements at each plane along the most important control modules given for the management of the VDC service.

Starting at the data plane, novel Top of the Rack (ToR) switches [53] and optical fibre switches [54] are introduced to provide fast inter-rack communication. ToRs at each rack then, provide electrical to optical conversion to manage incoming server packets by aggregating them into an optical circuit, thus allowing the transfer of data between source and destination racks. Optical circuits in turn, are arranged in an inter-rack flat fibre network composed by optical fibre switches. By following this approach, a high performance data plane solution to intra-DCNs is provided, taking into account the benefits of the introduced optical technologies, such as high throughput and low latency.

To enable the dynamic configuration of the electric/optical data plane and support the control of ToRs and fibre switches, the SDN technology is introduced. Then, at the control plane, an enhanced SDN controller based on the Opendaylight (ODL) open source controller [55] and the use of an extended OpenFlow (OF) protocol is provided. ODL in this case, presents extended software modules to provide enhanced functionalities for supporting optical technologies and the VDC service: Topology Manager (TM) maintains topology data considering both optical and electric switches/links; Virtual Tenant Network Manager (VTN) configures the overlay network (i.e., virtual network) to provide the connectivity between VMs at physical servers by configuring as well the Open Virtual Switch (OVS); OF Plugin and libraries support optical extensions [20] to trigger circuit flow configurations and retrieve optical related data from switches; ODL DLUX web graphic interface allows representing the complete optical DCN.

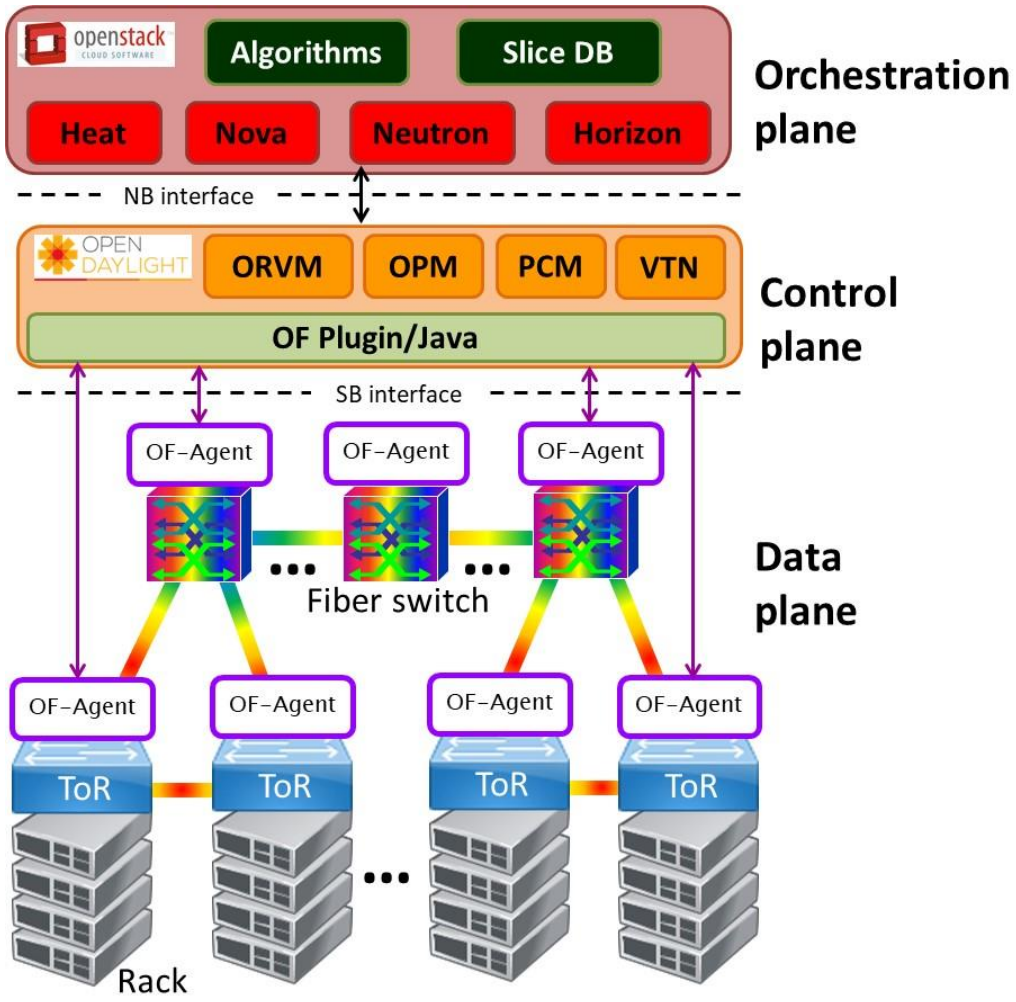


Fig. 4-2. Intra-DCN Architecture

Besides these extensions, other modules were developed specifically for this architecture: Optical Resource Virtualization Manager (ORVM) supports the management of virtual optical nodes/links and coordinates the configuration of the optical segment; Optical Provisioning Manager (OPM) triggers the creation or termination of optical cross connections at the data plane; Path Computation Manager (PCM) calculates the best route for optical connections by using simple routing algorithms or by requesting the orchestrator for a higher level decision.

Finally, the orchestration layer, based on the use of the OpenStack open source orchestrator, takes charge of coordinating the deployment of sub-slices resources, managing in a direct manner the allocation of computing resources and triggering the request for required network configuration towards ODL. Moreover, the platform is also extended with the introduction of a novel Algorithms module capable of optimizing the placement of resources on the DC infrastructure by considering the current network and computing state, thanks to the interaction between ODL and OpenStack and the use of a Slice Database (DB).

The work realized on this thesis comprehends the design and development of the ORVM module at the SDN-based control plane and all the related extensions required for its functionality, more in-depth detail follows.

4.2.1 Optical Resource Virtualization Manager

The ORVM module is an SDN application running at ODL that was designed for handling the management and creation of optical virtual instances, which are directly associated to the underlying optical resources at the data plane. Besides, it is also in charge of coordinating the optical segment configuration for each sub-slice request coming from the orchestrator. In this sense, ORVM keeps information regarding the virtual topology considering the created optical nodes and links instances and manages them according to sub-slices requirements.

A virtual optical node in turn, considers a subset of optical ports at the physical fibre switch, while a virtual optical link refers to the optical connection between two virtual nodes. Then, in order to configure a new connection across

the optical domain, ORVM must interact with other SDN modules, which in turn provide specific network information and operations required for its configuration. In particular, these interactions relate to inventory data considering physical optical and electrical switches, requests for sub-slice creation, overlay virtual links at the electric domain, network topology data, configuration of optical connections and the verification of the computed optical paths.

Additionally, the ORVM is configured to provide host tracker functionalities, that serve to identify the servers or hosts currently connected in a sub-slice and create/store their information on the corresponding topology datastore (DS), to then make it available to other applications for its visualization and management.

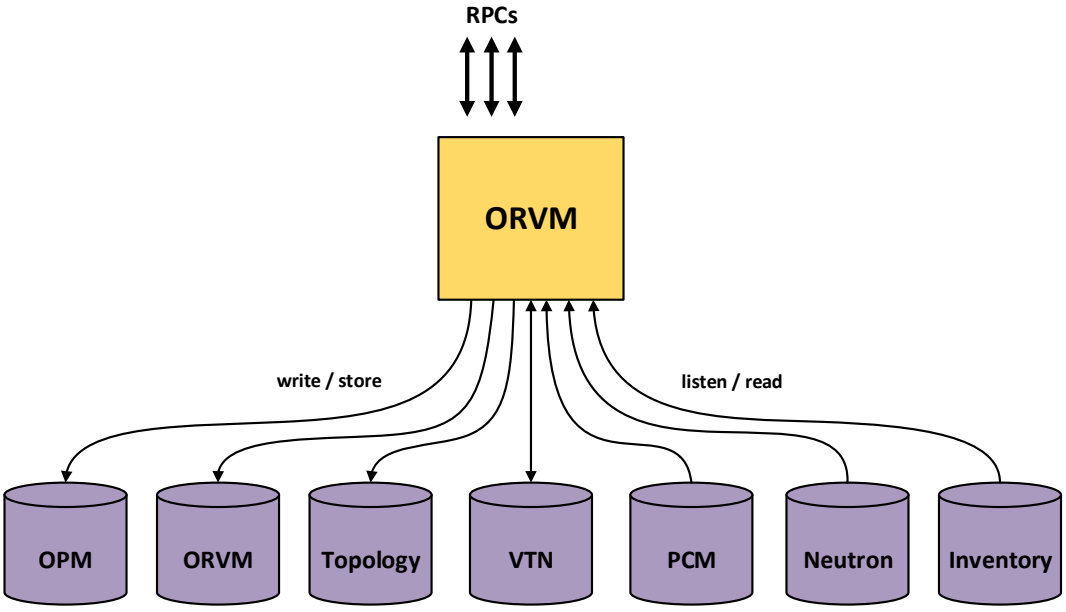


Fig. 4-3. ORVM Communication Interfaces

To enable all these functionalities, and following the communication options given by the ODL platform, ORVM is interfaced to other internal SDN modules by means of datastores, where interaction is realized through operations such as reading/listening to information changes and writing/storing new data. This method offers a flexible way of exchanging data without generating dependencies between modules. In case of external communication towards other layers or components, a set of Remote Procedure Call (RPC) functions are also defined. In this matter, Fig. 4-3 represents all these given communication interfaces, where each one is configured with a specific purpose as exposed next:

- **Interface to Inventory DS:** Serves to verify the existence of physical nodes. The ORVM subscribes to this DB as a listener to be aware of newly added optical-enabled nodes (i.e., fibre switch).

- **Interface to Neutron DS:** Enables triggering the configuration of a VTN-link covering the optical segment from source to destination ToR, upon the notification of a new network request coming from the orchestrator and handled by Neutron (at VDC deployment).

- **Interface to PCM DS:** Allows reading/listening to the computed optical path information regarding the connections to be configured on the data plane. With this data, ORVM is capable of creating the correspondent optical virtual topology at the logical level.

- **Interface to VTN DS:** Serves to write an overlay VTN-link so VTN bypasses the optical connection by configuring only OpenStack OVSs and delegating optical device configuration to ORVM. Besides, it allows listening to VTN dataflows, which appear once the first packet passes through an OVS, then triggering the configuration of the optical path.

- **Interface to Topology DS:** ORVM uses the VTN dataflow information to identify existent servers/hosts at the sub-slice. Then, these are stored at Topology DS to make them available to other modules/applications.

- **Interface to ORVM Datastore:** Allows storing all the data related to the optical virtual topology, considering virtual optical nodes and links.

- **Interface to OPM Datastore:** Serves to request optical connectivity between two nodes by writing OPM connections, using the source and destination nodes data gathered from the VTN dataflow.

- **Remote Procedure Calls (RPC):** Allows providing functions to directly create virtual optical resources from external clients.

The implementation of ORVM functionalities and required interactions is described in Java-based code, where each particular aspect of the module behaviour is

programmed. The implementation files are based on the use of Java classes and controller-level datastores, which are generated in the ODL platform thanks to the definition of specific YANG models [56]. In this regard, the main YANG model describing ORVM virtual nodes, links and topology structure and the defined RPCs has been included at the end of the document in *Appendix I*. Moreover, the YANG model related to the host-tracker capabilities has been also added in *Appendix II*.

4.2.2 Additional Software Extensions

Besides the development of the ORVM module, extensions to other modules at the SND-control level were also introduced, to cope with the required functionalities related to the sub-slices allocation and the control of optical-enabled nodes/links. In this matter, the following modules were modified/extended:

- **VTN:** The VTN Manager module was extended to bypass the recognition of switches with optical capabilities when building the VTN internal inventory, which is constructed by listening to changes related to added/removed nodes at the Inventory DS. In this way, the module deploys the overlay virtual network between endpoints (e.g., servers, hosts) by triggering only the configurations required on packet-based switches (e.g., OpenStack OVSs). Meaning that only this type of switches will be considered for the network setup, and thanks to the VTN-link stored by the ORVM that covers all the optical segment of the network. The coordination of the optical path configuration then, will be indirectly delegated to the ORVM, when this consumes the notification for a new VTN dataflow at the VTN DS.

- **TM:** Extensions made at the TM module, allow to distinguish between all the different types of connected switches (i.e., OVS, ToR, fibre switch), at the time nodes are added to the Inventory DS after the establishment of their OpenFlow session with the controller. To this end, the TM uses the capabilities exported by each switch to classify and store them in the common Topology DS according to their specific type. In the case of optical capabilities, these are available thanks to the use of the extended OF protocol which allows switches to expose such information.

4.3 Network Operational Workflows

Considering the analysed additions and extensions given for the deployment of DCN sub-slices and more specifically for the provisioning of the required network resources/connectivity, it is possible to identify operational workflows from different points of view. In the first case, focus is put on the inter-module communication required at the SDN-control level, where all the steps for network configuration are followed to clarify the role of the ORVM module and the other extensions. The second workflow, considers the point of view of the data layer, to realize how each component is configured and who is the responsible for such configuration.

To define the proposed workflows, it is also important to consider the two stages of configuration of sub-slices. These are described next:

- **Stage 1:** Relates to the operations performed upon the request of a new sub-slice (i.e., VDC), considering the allocation of computing resources by the orchestrator and the demand for network requirements towards the SDN-controller. At this stage, the overlay virtual network is created but no specific flow configuration is yet pushed to network devices.

- **Stage 2:** Following the SDN principles [6], the second stage begins when the allocated servers/hosts become operative and send the first packet to the nearest switch or network element. At this point, the SDN-controller is able to process the packet and use this information to configure all the devices on the path (i.e., by generating a VTN dataflow), so the connectivity between source and destination endpoints is enabled.

4.3.1 SDN Control-level Workflow

The control-level workflow, as introduced before, centres on the process realized by the developed/extended software modules at the SDN-controller to setup the required network configuration. In particular, the process is described following a step-by-step approach for better understanding of each module functionality. Fig. 4-4 in turn, depicts all these steps considering as a starting point, the network provisioning request coming from the orchestrator towards the controller (step 0).

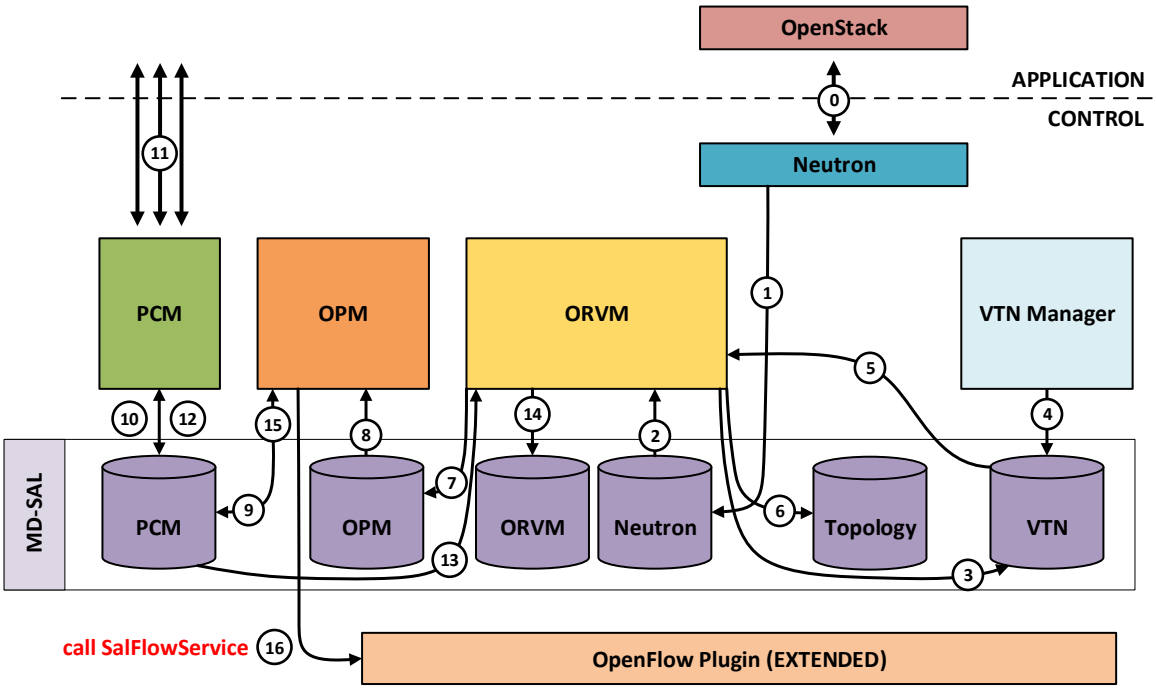


Fig. 4-4. Control Plane-level Workflow

As Neutron module receives the request, it stores on the Neutron DS a new network request instance containing data of the source and destination endpoints that require network connectivity (step 1). ORVM then, listens to this information as uses it to identify the corresponding OVS to each endpoint (step 2) and creates an overlay VTN-link between them on the VTN DS, covering the optical segment (step 3). At this point, stage 1 of sub-slice configuration is fulfilled.

After computational resources (i.e., server, host, VM) are deployed and become operative, the first packet arrives at the OVS next to the source endpoint. At this point, stage 2 of VDC provisioning begins as the packet is forwarded to the controller, where VTN uses its source/destination data to trigger required network configuration. In this case, as VTN only recognizes non-optical network elements, it pushes a new VTN dataflow instance to the VTN DS considering a direct connection between source endpoint, OVSs and destination endpoint (step 4), which is used by other modules for configuring the required flows at the OVSs.

The ORVM in turn, listens to this new VTN-dataflow and uses its information to perform two operations. First, it identifies the location of the endpoints and creates new host instances that are stored at the Topology DS, considering as well

the links between hosts and next-hop switches (step 6). The second operation relates to the construction of a new connection request instance, that is stored at the OPM DS to demand for the required optical connectivity setup between the specified OVSs (step 7). The OPM then, reads this connection request (step 8) and uses it to create an optical path request, which demands the PCM for providing the optimal path for enabling connectivity across the optical segment. The path request instance is stored at the PCM DS.

Once the PCM reads the request (step 10), it is able to provide the best optical path in two ways, depending on its internal configuration. The first option considers using a simple calculation method based on shortest path algorithms, while the second entails requesting/delegating path calculation to components at a higher-level, that would consider the current state of both computing and network resources across all deployed sub-slices, in a way to provide a more accurate computation of the ideal path (step 11). The PCM then, stores at the PCM DS the new computed path instance with the results (step 12).

The ORVM reads the information contained in the computed path (step 13) and uses all the data related to the selected optical nodes and links, to create the optical virtual elements in correspondence. In this sense, it stores on the ORVM DS, all the virtual optical nodes and links considering their direct reference to the physical elements at the infrastructure, and representing them as well in a virtual optical topology for each configured sub-slice (step 14).

At the same time, the OPM reads the response for the previously demanded path computation (step 15) and triggers the configuration of the required optical flows by calling the *SalFlowService* exposed by the OpenFlow Plugin module, which considers all the required OF extensions for circuit switching technologies and enables configuring the optical devices in the computed path, thus setting up the complete end-to-end connectivity between hosts (step 16).

It is important to mention that the recognition tasks executed by the ORVM and the extended TM module, considering their interaction with the Inventory DS, were not considered on this specific workflow, as these take place during the OF

session establishment stage coordinated between optical devices and OF Plugin at the time nodes are initially connected to the controller. Finally, it is also worth exposing that the structure of datastores required for performing all read/write operations for inter-module communication is defined thanks to the use of the ODL Model-Driven Service Abstraction Layer (MD-SAL).

4.3.2 Data Plane-level Workflow

Looking from the infrastructure point of view, the workflow is focused mainly on the stage 2 of the sub-slice provisioning process. In this sense, and considering that ORVM has already configured an overlay VTN-link between OVSs (step 0), the workflow starts after hosts/servers become active and start exchanging packets.

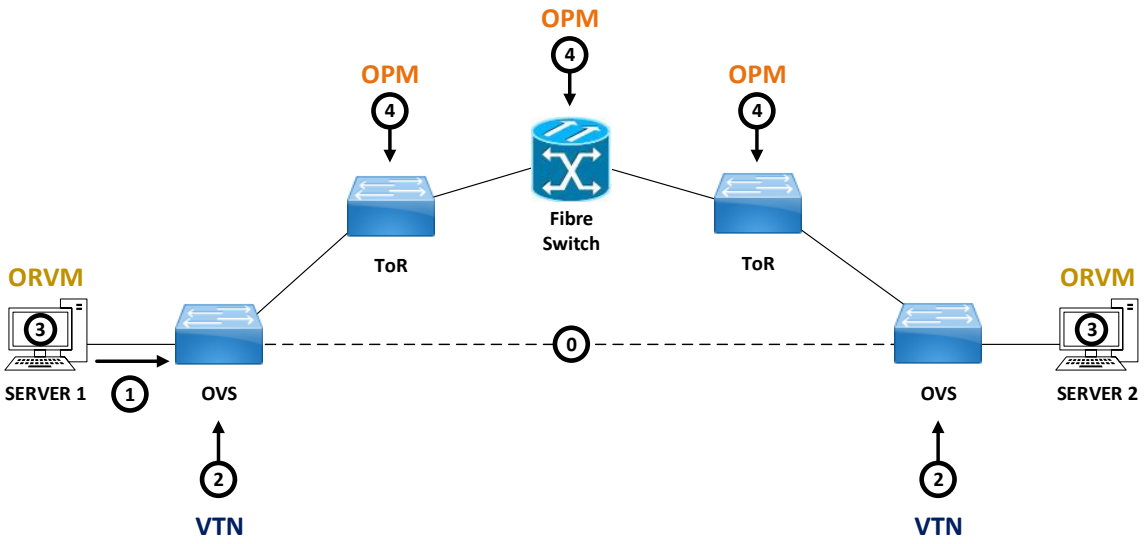


Fig. 4-5. Data Plane-level Workflow

Following the same step-by-step approach as previously, Fig. 4-5 depicts how device configuration is triggered. Then, after Server 1 sends the first packet destined to Server 2 (step 1), the SDN-controlled OVS next to the server forwards the packet to the controller, triggering a response from VTN Manager that configures both OVSs at the edge of the link (step 2). As VTN also generates the VTN dataflow, it serves ORVM with the information to configure both endpoint hosts at the common Topology DS (step 3). Moreover, ORVM also coordinates the optical path setup which is triggered, after path computation calculations, by the OPM (step 4) to configure the specified optical devices (i.e., ToRs, Fibre Switch).

4.4 Virtual Network Mapping for VDC Deployment

To clarify how the virtual network is mapped over the underlying physical resources upon the requirement for VDC network configuration, an example considering the provisioning requests for VDC1 and VDC2 is described in Fig. 4-6.

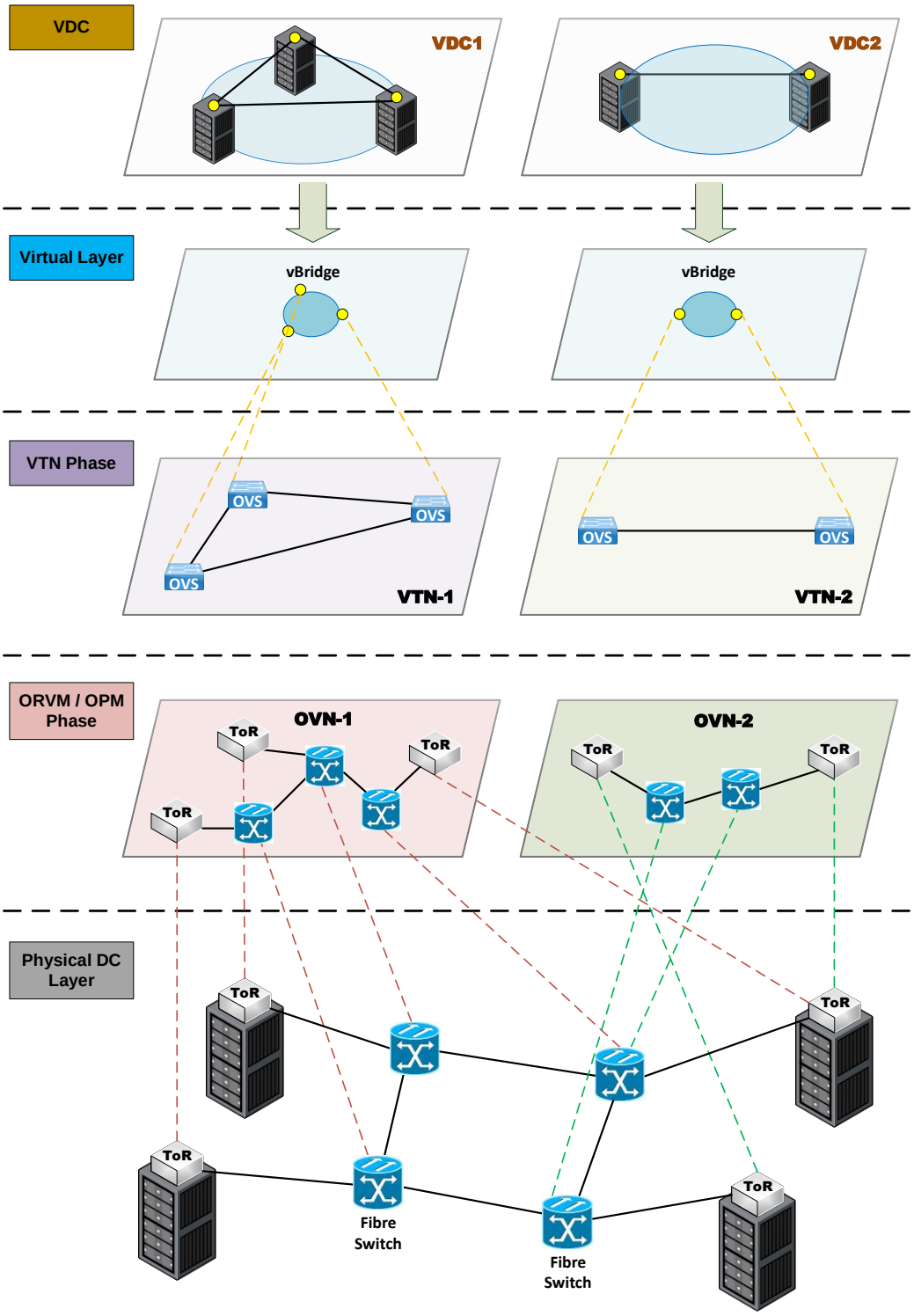


Fig. 4-6. Virtual Network Mapping

In this particular case, network requirements set for VDC1 entail providing connectivity between three endpoints, each located at a specific DC server rack and representing the allocated computing resources (i.e., host, server, VM). At the virtual layer then, a virtual bridge (vBridge) is created to construct an overlay virtual network. Following this, the ingress physical ports given for all endpoints at edge switches are also mapped to corresponding vBridge logical interfaces, identifying the specific Switch ID and Port ID for each of them. The VTN Manager, takes charge of this process, and uses the mapping of the virtual network to configure flows at switches and enable network connectivity between endpoints. In this case, for VTN-1, the OVSs of three racks are configured, matching at the inserted flows the data of the mapped physical ports to enable packet transmission.

Considering that the VTN configuration phase bypasses all optical-enabled devices, the ORVM module uses virtual network data to setup the required optical network connectivity. As the configured VTN-1 consist of three specific connections to interconnect endpoints, ORVM must coordinate with OPM the computation and configuration of three optical paths. Once the paths are computed, the ORVM is able to construct an Optical Virtual Network (OVN), with all the specified virtual optical nodes and links allocated for VDC1. In turn, OPM is also able to push all required optical flows to the physical devices at the infrastructure layer.

As for VDC2 deployment, its configuration requires a simpler process. In this sense, only two endpoints require connectivity as VTN-2 is constructed. Hence, only one connection is given between endpoints and only one optical path must be calculated for OVN-2. After all underlying devices get configured, it is possible to see how both VDC1 and VDC2, despite being logically isolated in VTNs and OVNs, are mapped physically over the same underlying infrastructure, sharing in some cases the same physical switches or links, thus optimizing resource utilization.

4.5 VDC Provisioning over Optical intra-DCNs

After analysing all the workflows, software modules, virtual mappings and operative functionality of the presented architecture, it becomes necessary to perform some experimental testing to validate VDC provisioning on optical intra-DCN scenarios.

4.5.1 Experimental Testbed Scenario

With this purpose, the use of an experimental testbed considering the control of an emulated DC scenario is introduced. The testbed in turn, consists of three physical servers allocating necessary software control platforms and computing resources:

- **Physical Server 1:** Allocates an instance of the OpenStack platform, from where the orchestrator control node manages all computing resources.
- **Physical Server 2:** Allocates an instance of the ODL controller, considering the extensions and modules for electric/optical network resources control.
- **Physical Server 3:** Allocates an instance of the orchestrator compute node, that represents computing resources of another rack at the DC.

At the infrastructure level, physical optical switches allow connectivity between control and compute OpenStack nodes, including three ToRs and one fibre switch. Besides, a management network interconnects all servers as depicted in Fig. 4-7.

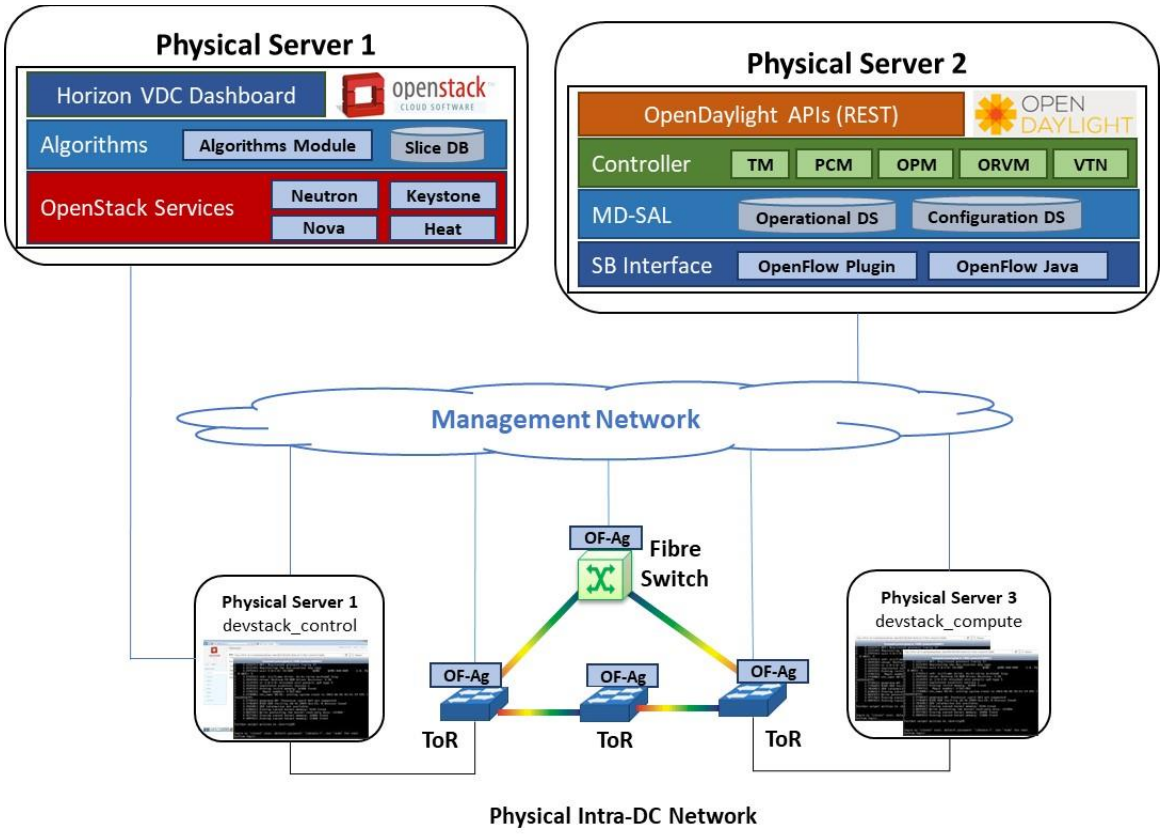


Fig. 4-7. Experimental Intra-DCN Testbed

4.5.2 Testing

After setting up the experimental testbed, the VDC provisioning service can be tested. In this regard the OpenStack Horizon dashboard is enhanced so clients can configure the instantiation of a VDC according to their specific requirements. In this case, a VDC consisting of two interconnected virtual servers is considered.

As the client triggers the request, OpenStack takes charge of deploying the virtual servers (i.e., VMs) considering the available computational resources. In this way, first server is deployed at the orchestrator control node and the second at the compute node respectively. While VMs are being deployed, the orchestrator delegates the configuration of the inter-VM network connectivity to ODL.

The network configuration at the controller refers to the workflow analysed in *Section 4.3.1* where the necessary operations are performed to select the ideal connection path and configure the required OVSs, ToRs and/or Fibre Switch. In this case, the calculated optimal route between the orchestrator computing nodes passes through the connection given by the fibre switch, bypassing the three ToR alternative route and pushing configuration flows only to the specified switches.

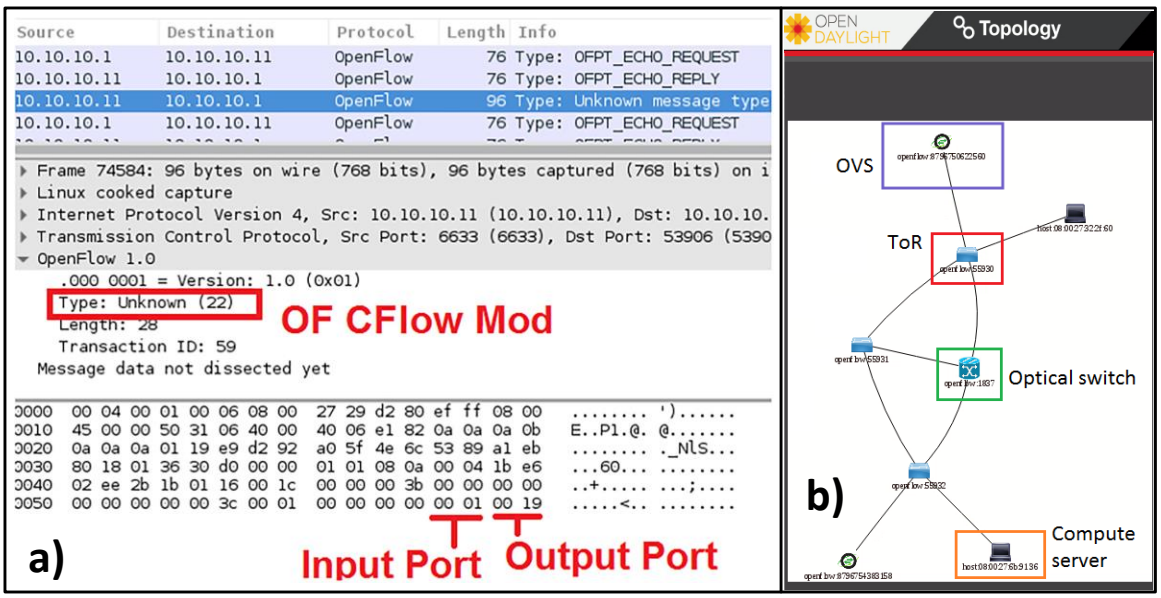


Fig. 4-8. Wireshark capture of optical flow configuration (a) Network topology as seen in ODL DLUX graphical interface (b)

Fig. 4-8 then, depicts the results of the operational coordination at ODL. In this sense, the extended configuration flow pushed towards the fibre switch (i.e., OF_CFLOW_MOD) is shown in (a), where matching is performed at the circuit switching port-level to enable the required connectivity between ToRs. Then, the graphical visualization of the topology DS (b), shows the recognized optical nodes and links that represent the same network configuration physically defined for the testbed. Besides, the server host nodes allocating the deployed virtual resources and the OpenStack OVSs are also displayed.

4.6 Conclusions

The slicing of the intra-DCN infrastructure, appears as a promising solution for future cloud-based DCs with high demanding requirements, considering the adoption of SDN and optical technologies as a fundamental part of the network evolution in these scenarios. In the work related to this chapter, a fully functional architecture has been proposed, to enable the dynamic configuration of a hybrid electric/optical DCN, allowing for the deployment of computational and network resources by means of an extended SDN-Controller and Network Orchestrator.

The presented VDC use case, has proven to optimize physical resource utilization in intra-DCN scenarios by enabling the allocation of multiple VDCs (i.e., sub-slices) for tenants over a common shared infrastructure. The improved SDN control in this sense, driven by the management of virtual optical and non-optical resources and the ability to coordinate the configuration of different data layer technologies, has opened the path towards further innovation in this subject.

Finally, the sub-slice provisioning tests performed with the experimental testbed, have served to validate the correct operation of the introduced ORVM and other related controller-level modules. Demonstrating the functionality of the analysed control and data layer operational workflows and the proper mapping of the created virtual networks to the underlying physical resources.

4.7 Publications

The work realized in the scope of this thesis chapter has been disseminated in the following scientific journal and conference collaborations:

1. S. Spadaro, A. Pagès, F. Agraz, **R. Montero** and J. Perelló, "Resource orchestration in SDN-based future optical data centres," 2016 International Conference on Optical Network Design and Modeling (ONDM), Cartagena, 2016, pp. 1-6.
   <https://doi.org/10.1109/ONDM.2016.7494057>

2. A. Pagès, F. Agraz, **R. Montero**, G. Landi, R. Monno, J.I. Aznar, A. Vinez, C. Jackson, D. Simeonidou and S. Spadaro, "Experimental Assessment of VDC Provisioning in SDN/OpenStack-based DC Infrastructures with Optical DCN," ECOC 2016; 42nd European Conference on Optical Communication, Dusseldorf, Germany, 2016, pp. 1-3.
   <https://ieeexplore.ieee.org/document/7767632>

3. S. Spadaro, A. Pagès, F. Agraz, **R. Montero** and J. Perelló, "Orchestrated SDN-based VDC provisioning over multi-technology optical data centre networks," 2017 19th International Conference on Transparent Optical Networks (ICTON), Girona, 2017, pp. 1-4.
   <https://doi.org/10.1109/ICTON.2017.8025181>

4. A. Pagès, F. Agraz, **R. Montero**, G. Landi, M. Capitani, D. Gallico, M. Biancani, R. Nejabati, D. Simeonidou, S. Spadaro, “Orchestrating virtual slices in data centre infrastructures with optical DCN,” Optical Fiber Technology, vol. 50, 2019, pp 36-49.
   <https://doi.org/10.1016/j.yofte.2019.02.011>

Chapter 5. Optical Link/Topology Discovery

The work presented in this chapter, introduces topology discovery methods that help to secure the correct optimization of network resources usage in SDN-based networks by providing an accurate mapping of the optical data plane configuration.

5.1 Introduction and Related Work

The process of network topology discovery in specific, provides control and management components with the logical representation of the physical topology. In this way, miscorrelations at the mapped topology data, considering differences between the identified topology and the actual physical topology, would represent an incorrect visualization of the data plane-level distribution and interconnection of network resources, thus becoming a source of errors for applications that depend on this information. Consequently, it becomes necessary to guarantee the correct recognition of the physical network topology, with the focus set on the dynamic link/topology discovery in optical network scenarios.

In this respect, static optical topology-data configurations still performed in practice, are no longer efficient and could lead to errors and tedious configuration tasks. Especially considering modern network scenarios that look forward to establishing dynamic environments, to support constantly changing and highly complex network architectures. For this reason, providing a technique capable of dynamically discovering the optical network topology becomes decisive. Besides, the management of optical discovery events would also provide applications with up-to-date topological data and improve their effectiveness in front of network events, thus facilitating the proper operation of resource optimization mechanisms.

Regarding topology/link discovery efforts in optical networks, methods have been studied much before the introduction of the SDN technology as a control solution for this type of networks. That being said, a proposal was presented by the International Telecommunication Union (ITU) describing two different methods for automatic link discovery [57], based on previously analysed requirements [58].

The first method considered the use of the client layer payload to exchange optical discovery related data, while the second required the in-band exchange of discovery messages embedded in the trail trace identifier header. Both methods enabled automatic link discovery but also came with some specific concerns. The first method allocates discovery messages inside client payload, which may impact client traffic at the link connection. In other cases, it may also consider the use of a dedicated control channel to exchange discovery data, as more commonly seen in general multiprotocol label switching (GMPLS)-based implementations [59]. The second method in turn, implicates the in-service trail termination of the transport overhead so the discovery related data could be retrieved or added by the optical switch. Such process does not affect the client signal, but requires terminating the overhead at each link endpoint, which entails further processing.

Following the approach presented by the ITU on exchanging discovery data through the use of in-band overhead, the ONF presented extensions to the OF protocol to allow the switch-controller-switch communication of discovery data [21]. At the proposal, the introduced extensions are used to configure and retrieve node adjacency data to/from OF agents, considering the coordination of the whole process by an SDN controller in order to allow the confirmation of reachability between adjacent nodes at both link ends. This approach enables accomplishing the trail trace identifier-based discovery of optical links. Nevertheless, the proposed extensions still require further proof of concept validation from the research community. Another important aspect is that the extensions are built based on version 1.4 of the OF protocol, which is not fully supported by most SDN controllers in the market. In spite of this, if a wide adoption of these extensions is achieved in the future, these could be considered for the southbound management of the methods presented in this thesis.

As just stated, this thesis work presents two different mechanisms for optical link/topology discovery considering all previously discussed facts. The methods fit in the preservice category of layer 1 adjacency discovery use cases as defined by the ITU [58], where the discovery process is set before link connections become operative, then not affecting any traffic, in contrast to the previously analysed in-service method that could potentially represent problems in the network.

5.2 Link Discovery in SDN-based Optical Networks

The topology discovery process is considered an essential matter in SDN-based scenarios, as it entails the representation of the underlying network resources in terms of host, node, and link elements. In this respect, the study presented in [60] discussed the importance of this process for a network environment to behave as expected. In general, this work recognises the accurate discovery of the network topology to be critical, so tasks related to network monitoring, resource optimization management, scheduling, diagnosing, among others, are able to operate properly. In turn, most services at both application and control layers depend on using this topological data. From the applications perspective, importance is set in the access to a correct abstraction of the network. Then, the methods used by the controller to build the topology representation are not particularly relevant to consumers.

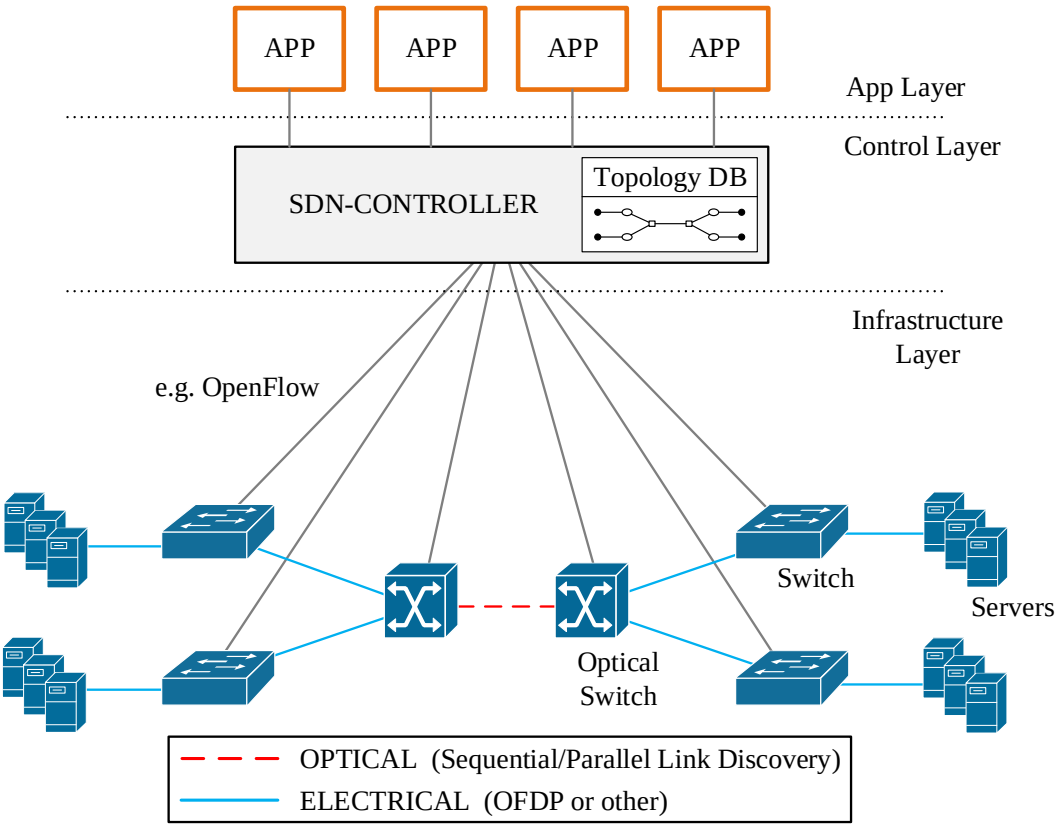


Fig. 5-1. SDN-based Link Discovery in Hybrid Opto-electrical Networks

In this matter, Fig. 5-1 shows a representative hybrid opto-electrical network scenario where the SDN-controller enables discovering both electrical and optical links, besides hosts and nodes, so that applications can consume the abstracted

network data. The complete topology representation in this case is usually stored at the topology database at the controller. Applications, in turn, use the northbound communication interface to access and retrieve the data. As for the link discovery process, it is coordinated by the use of the OF Discovery Protocol (OFDP) [61] (or other method) to map the electrical interconnections and the optical link discovery methods presented in this work, considering given sequential and parallel versions. Assuming that scenarios in practice may vary, this example mainly serves to highlight the significance of having a dynamic discovery process for both optical- and electronic-based technologies in modern SDN-controlled networks.

As discussed before, the SDN-controller should provide the mapping of network elements from the underlying data plane, where such elements can have different capabilities, either electrical or optical as for the analysed case. Therefore, the controller is required to coordinate different procedures for host, node, and link discovery according to the type of network resources under its control.

The discovery of the interconnection between adjacent network nodes in this regard, more commonly known as link discovery, presents a challenge, as optical circuit-switching network elements do not handle packet insertion and extraction from the data flow, which is a key part of typical discovery methods in electronic-based networks. Therefore, the recognition of links usually requires the controller to use a discovery protocol or mechanism, capable of fulfilling the correct mapping of optical links.

In the case of packet-based networks, OFDP is the common option for discovering adjacencies between network nodes, where the discovery process bases on exchanging Link Layer Discovery Protocol (LLDP) packets containing chassis-ID and port-ID identifiers in the packet payload, so the controller makes use of this information, along with other metadata, to map the underlying links at the control level. In this matter, various analyses [60, 62, 63, 64] have remarked the vulnerabilities of OFDP, proposing enhancements to the current protocol and other methods for achieving link discovery in these types of networks. Specifically, the work of Pakzad [62] presents an enhanced version of the protocol with less LLDP message exchanges (OFDPv2). Then, the work of Azzouni [63] introduces

a secure OF topology discovery method by enhancing protocol security introducing minimal changes to the OF switch design. Finally, the work reported by Alharbi [64] proposes the use of hash-based message authentication codes to improve the authentication of LLDP packets.

In optical networks, the scenario changes as no insertion or retrieval of packets is possible between adjacent nodes. Hence, other type of information exchange or technique to identify optical links must be used, to allow the controller having a complete visibility of the SDN domain. To this end, this work proposes the use of methods capable of providing an accurate and efficient discovery of optical links, to then allow the use of valid topology data by applications. Additionally, the proposed link mapping considers full dynamicity, so changes in the topology do not affect the performance of network services and higher-layer consumers.

Next sections present the basics for the introduced sequential and parallel link discovery methods along with the required software/hardware implementations for their application and the message exchange workflow of the discovery process.

5.3 Sequential Discovery

The first proposal presents a novel cost-effective method based on the automatic discovery of optical links on a one-by-one basis (i.e., sequential). To accomplish this, specific implementations and extensions are required at both data and control layers to allow the correct gathering and exposure of topological data.

At the data plane, the proposal considers optical switches to be controlled thanks to the use of an extended version of the OF protocol [20]. Then, OF agents running on top are required to gather the adjacent node information (i.e., node and port identifiers) in a dynamic manner. To this end, the method bases on the use of a Test Signal Mechanism (TSM), which entails emitting light from the active fibre ports an optical switch and letting adjacent nodes to detect it, using the same technique in the opposite way. The detection and emission of light signals would trigger the mapping of optical links between adjacent nodes by exchanging specific

messages between OF agents and the SDN-controller. In this way, the support by switch agents for the TSM and the extended OF protocol becomes essential.

At the controller level in turn, the intelligence to manage the exchange of the controller-switch messages, and the ability to retrieve peer node and port data from them is required, considering as well the creation of the correspondent optical links at the topology database once this data is correlated. Therefore, the development of an SDN application able to provide these functionalities is also fundamental.

5.3.1 Test-Signal Mechanism

The proposed TSM makes use of orderly exchanged test signals between optical ports, that are triggered at the moment the physical connection is established and before client traffic starts flowing through. In this way, the specific use of separate associated channels for link discovery can be avoided, thanks to the coordinated use of TSM across optical nodes/links by means of the SDN-controller.

As previously introduced, test signals used in TSM entail the transmission of light by an active emitter fibre port, its detection at the receiver port, and a final validation of its reception by another test signal sent through the same fibre link back to the emitter, requiring then at least one transmitter per-node. The use of this method allows to communicate, via specific controller-switch messages, the peer node and port data at both link ends, between which the TSM has been effectively performed. As no link correlation mechanism is available at the data plane level due to its optical transparent nature, the mapping of optical links is performed at the controller level by correlating the received peer node and port data. Moreover, to guarantee the correct recognition of links, the controller employs a queue where newly recognized active ports are stored. This way, the TSM is triggered for each port at the queue sequentially, so the controller is able to configure each link based on the successful confirmation of the tested TSM by the interconnected nodes.

The exchange of test signals without affecting link traffic is also an important matter for the mechanism. In this sense, and as stated before, the use of TSM will not represent an issue since the exchange of signals is triggered at the moment

the controller identifies a new active fibre port. Actually, the required per-port adjacency discovery time should be considered negligible compared to the setup time each port takes to begin transferring data. Hence, the performance of the TSM would be directly affected in scenarios where optical switches with already active fibre ports, considering the current transmission of data, were connected to the controller. In this situation, the TSM would not be able to properly operate due to the already operational links. Nevertheless, such restriction could be overcome by configuring TSM to avoid the discovery in ports with an already active transmission.

As a final concern, links removal at the control level would be handled by consuming OF agent notifications regarding changes on the fibre ports status to a down state, thus allowing link discovery and deletion process to be fully dynamic.

5.3.2 Discovery Message Workflow

This section defines the OF message workflow necessary for the execution of the discovery process. The workflow is depicted in Fig. 5-2, and begins when the OF agent connects to the controller by means of a HELLO message exchange. Once the optical switch is detected, the controller sends a request demanding for its capabilities, by sending a FEATURES_REQUEST message. Upon reception of this message, the agent exposes the information of the optical switch physical ports in a FEATURES_REPLY message. At that point, the controller identifies all active ports and sequentially triggers the TSM-based discovery process on each of them.

To begin, the controller sends a PACKET_OUT message to the OF agent (step 1), specifying the active port identifier and a SET_DL_SRC output action, so the agent knows on which port the TSM should be executed to test adjacencies. Then, the agent instructs the specified port to send a test signal (step 2). At the time the adjacent node detects the reception of light at one of its ports, its agent becomes aware and activates the transmission of a response test signal back through the same port, to confirm a successful reception (step 3). Then, it sends a PACKET_IN message with NO_MATCH value in the reason field to the controller, containing as well the reference of the port where the signal was detected (step 5). At the same time, when the first agent receives the confirmation test signal, at the

same port where TSM started, it sends an additional PACKET_IN message to the controller (step 4), this time with an ACTION reason and the reference to the port where the initial test signal was transmitted and the test reply was detected.

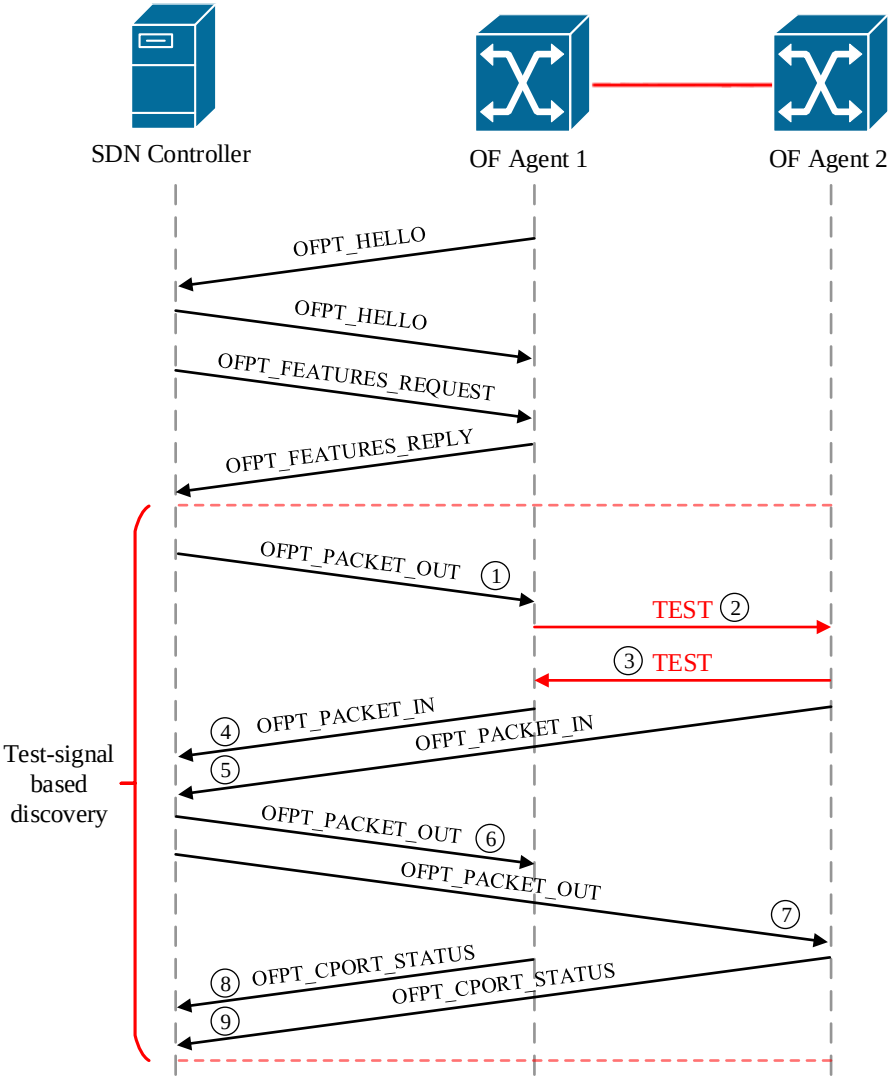


Fig. 5-2. OF-based Message Workflow for Sequential Optical Link Discovery

Once both PACKET_IN messages are received at the controller, they are processed using the developed discovery module, where port data exposed in both messages is associated to determine which node acts as the emitter and which as the receiver. As the correlation is accomplished, the controller recognizes the existence of a link between both nodes at the data plane level. The controller, at this point, would be able to create a link logical instance to store it in the topology datastore. However, following the suggestions of OF standards [18, 20], that entail data plane devices to be responsible for sending peer data to the controller, the

discovery process is completed as follows. The controller sends PACKET_OUT messages towards both switch agents related to the discovered link (step 6-7), using a SET_DL_DST action value and attaching the specific peer data for the link (i.e., peer node and fibre port identifiers). At the time agents receive this data, they become aware of their node adjacencies and expose peer data by sending CPORT_STATUS messages to the controller (step 8-9). These last messages are then consumed by the extended topology module that ends the discovery process by adding the new optical link to the topology datastore.

5.3.3 Extended SDN Architecture

The sequential discovery mechanism takes advantage of the SDN architecture to enable the discovery process. In this sense, the centralization of control functions at a single logical controller, simplifies the coordination of the discovery messages, allowing to process link data by implementing control-level functionalities. Fig. 5-3 next, depicts the extended SDN-based architecture for this particular scenario.

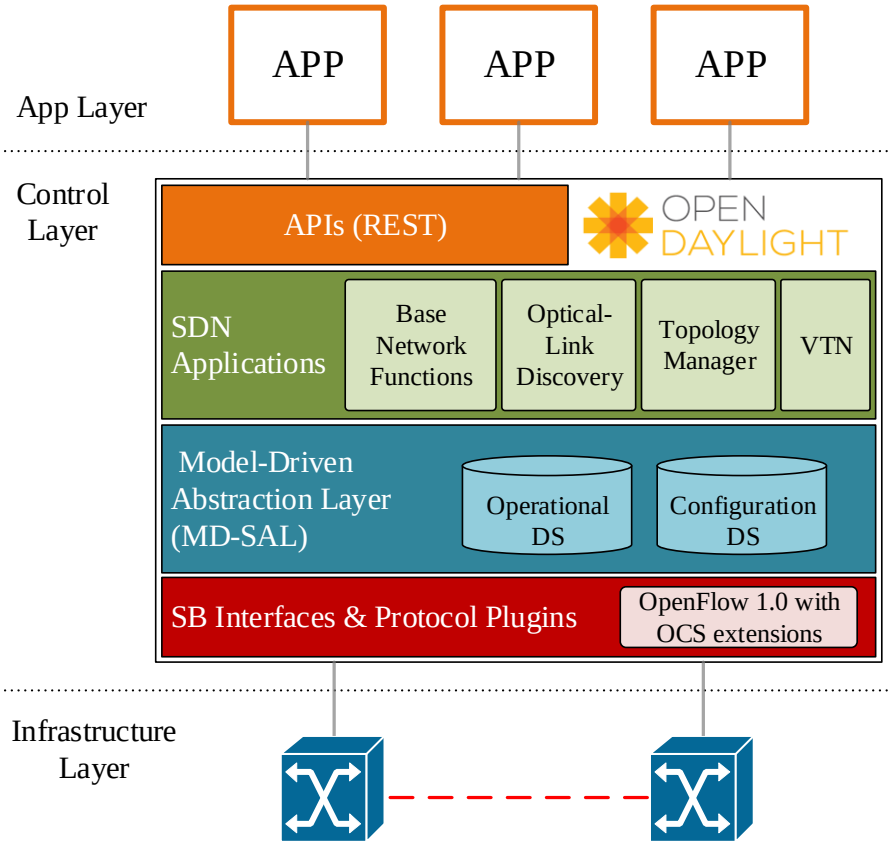


Fig. 5-3. SDN Architecture for Optical Link Discovery

The architecture in this case, focuses on the SDN control layer, which bases on the use of the OpenDaylight (ODL) platform, to identify the software additions and extensions introduced in this thesis. But, considering as well the optical devices and OF agents at the data layer and the other external applications that could be running on top of the controller at the application layer.

At the lower level of the SDN controller, the OF Plugin and OF Java modules implement the extensions required to exchange optical related data with OF agents [20], providing a direct communication point to optical devices via the southbound interface. The MD-SAL then, allocates specific model-defined data structures that serve as a source of self-describing data between internal modules, becoming the core communication element between them. In addition, external applications are also able to retrieve data required for their operation, via the northbound interface (NBI). Considering the internal SDN applications at the controller, some of them have been extended while others have been specifically developed in this thesis.

In this case, the Optical Link Discovery (OLD) module has been introduced, to address the requirements of the discovery mechanism. The module in particular, implements a port queue to organize all recognized active fibre ports and trigger the discovery process over them in a sequential way, instructing agents to use the TSM to test adjacency reachability. Additionally, it allows processing incoming confirmation messages from agents to correlate link related information, to then provide them with their correspondent peer data. In this manner, the OLD allocates the necessary functionalities to coordinate the discovery process. It is important to mention here, that the agents must also support the extended version of the OF protocol and be able to handle the discovery related messages, to execute the TSM upon controller request and to send back confirmation/peer data messages.

Regarding extensions to existent controller modules, the Virtual Tenant Network (VTN) Manager, responsible of providing L2 functionalities, was extended to identify optical discovery related messages (i.e., PACKET_IN, PACKET_OUT) and forward them to OLD for their processing. The Topology Manager (TM) in turn, was enhanced to handle peer data received in the CPORT_STATUS message, and use it to create or delete optical links at the topology database.

The implemented functionality at the controller level, enables the topology discovery to be fully dynamic, by allowing the mechanism to be sensitive to network changes such as the addition or removal of optical nodes/links. In this way, the controller is able to react to such particular events by triggering the discovery of links upon the recognition of a new node or active port, or by deleting all related data in case of removed hardware or a port down status notification. Thanks to this behaviour, an up-to-date representation of the topology can be constructed, and be then exposed to consumers which depend on this critical information.

5.3.4 Experimental Assessment

To assess the proposed discovery mechanism, a SDN-based testbed has been employed considering two emulated 9-node and 14-node optical network scenarios with different topology configurations, as illustrated in Fig 5-4. To accomplish the TSM emulation, considering test signal transmission and detection, software-based emulators of optical nodes (i.e., multi degree reconfigurable optical add-drop multiplexers [65]) and their correspondent OF agents have been used.

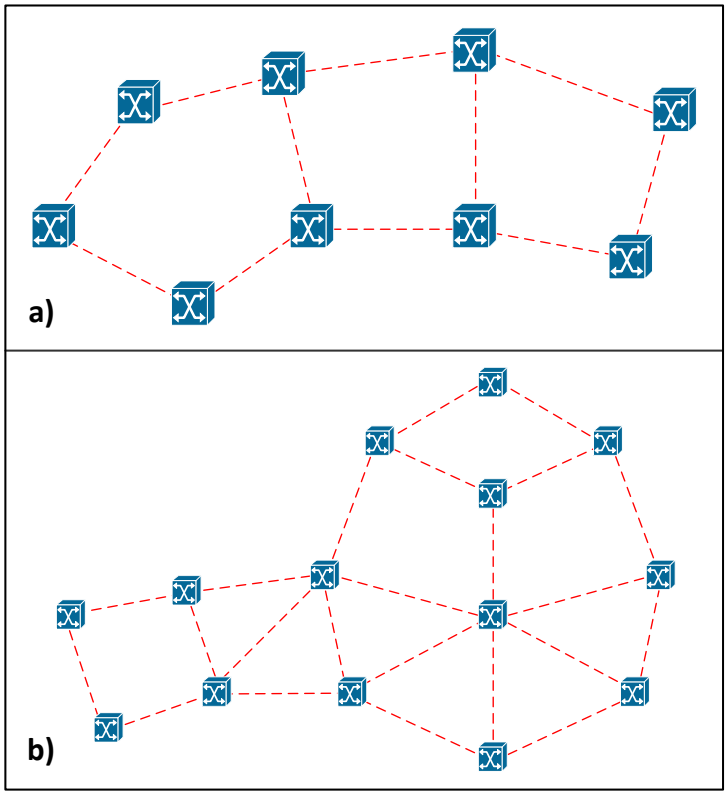


Fig. 5-4. Optical Network Topology Configurations: 9-node (a), 14-node (b)

Source	Destination	Protocol	Length	Info
192.168.101.21	192.168.3.1	OpenFlow	76	Type: OFPT_HELLO
192.168.3.1	192.168.101.21	OpenFlow	84	Type: OFPT_FEATURES_REQUEST
192.168.101.21	192.168.3.1	OpenFlow	76	Type: OFPT_HELLO
192.168.101.21	192.168.3.1	OpenFlow	340	Type: OFPT_FEATURES_REPLY
192.168.3.1	192.168.101.21	OpenFlow	80	Type: OFPT_STATS_REQUEST
192.168.101.21	192.168.3.1	OpenFlow	1136	Type: OFPT_STATS_REPLY
192.168.3.1	192.168.101.21	OpenFlow	176	Type: OFPT_PACKET_OUT
192.168.3.1	192.168.101.10	OpenFlow	88	Type: OFPT_STATS_REQUEST
192.168.101.10	192.168.3.1	OpenFlow	80	Type: OFPT_STATS_REPLY
192.168.101.10	192.168.3.1	OpenFlow	87	Type: OFPT_PACKET_IN
192.168.3.1	192.168.101.21	OpenFlow	88	Type: OFPT_STATS_REQUEST
192.168.101.21	192.168.3.1	OpenFlow	80	Type: OFPT_STATS_REPLY
192.168.101.21	192.168.3.1	OpenFlow	87	Type: OFPT_PACKET_IN
192.168.3.1	192.168.101.10	OpenFlow	100	Type: OFPT_PACKET_OUT
192.168.3.1	192.168.101.21	OpenFlow	100	Type: OFPT_PACKET_OUT
192.168.101.10	192.168.3.1	OpenFlow	164	Type: OFPT_PORT_STATUS
192.168.101.21	192.168.3.1	OpenFlow	164	Type: OFPT_PORT_STATUS

TSM

LINK

Fig. 5-5. Wireshark capture describing the message workflow for the sequential link discovery process between two network nodes

To begin, Fig 5-5 above depicts the message sequence exchange between two nodes and the controller for the adjacency discovery. The figure in turn, starts showing a node connecting to the controller via exchanged HELLO messages. Then, a FEATURES_REPLY message is used to expose its features, so the controller can trigger the discovery process by sending a PACKET_OUT message to the node instructing the use of the TSM. After its execution, between the node recently connected and its adjacent node, PACKET_IN messages are sent to the controller from both nodes to advertise the fibre ports where the signals have been emitted and detected respectively. Lastly, after the controller has successfully correlated the adjacency data, it sends PACKET_OUT messages to both nodes so they can expose their peer data via CPORT_STATUS messages back towards the controller, which is then used to configure the new link at the topology database.

Fig. 5-6 and 5-7 then, show the average topology discovery time (left Y-axis) of the proposed method considering the 9-node and 14-node scenarios, where data points at the graphs has been found after averaging 10 executions of the discovery process. In both cases, tests have been executed for a different number of fibre ports per-node adjacency, as in multi-fibre network scenarios. In particular, configurations of 1, 3, 6 and 10 bidirectional fibre ports have been set between adjacent nodes. With this approach, the impact of the number of network nodes and links on the topology discovery time can be analysed. Moreover, the average time of per-fibre port discovery (right Y-axis) has also been taken into account to

evaluate if the number of links to be discovered and the weight of handling more nodes at the controller could affect link discovery time depending on the scenario.

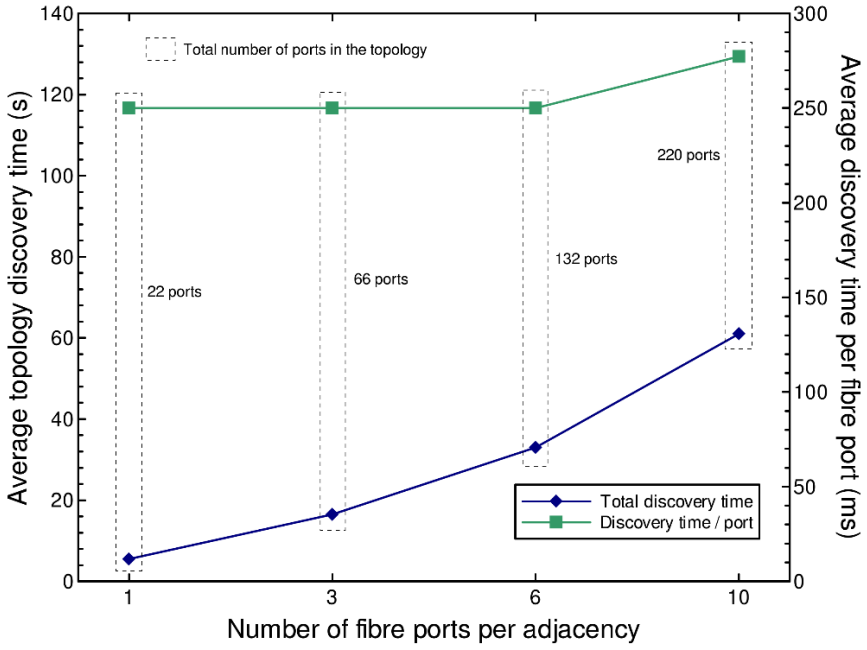


Fig. 5-6. Average topology discovery time (left Y-axis) and per-fibre port discovery time (right Y-axis) in the 9-node network scenario.

By analysing the results, the number of network nodes and links can be identified to have a significant impact on the topology discovery time. In turn, for the 9-node scenario shown in Fig. 5-6, the average topology discovery time was around 10 seconds when 22 links had to be discovered. In the following tests, this time started to increase progressively as the scenario introduced more links. For instance, with 66 links the resultant time was around 16 seconds, reaching up to a total of 61 seconds in the scenario with 220 links. Such increase in the topology discovery time, although mainly dependable on the number of links to discover, still implies a relative low total time for the discovery process, taking into account that for these tests, all nodes were connected to the controller at a single time, which is not usually the common case in practice. As for the 14-node scenario shown in Fig. 5-7, comparable time results were also observed. In this setup, the process takes around 11 seconds with 46 links, 34 seconds with 138 links, 70 seconds with 276 links and 129 seconds with 460 links, for discovering the network topology.

In contrast, the average per-fibre port discovery time (right Y-axis of the graphs) also analysed in the figures, shows results that consider a minimal impact

in relation to the number of nodes/links in the topology. In this sense, comparing at both scenarios the cases with less number of links to the ones with 10 times more, the results vary only around 30 to 40 ms. In view of this results and the fact that the average time for this variable on cases with fewer links is around 250 ms, such variance can be seen as negligible and not directly linked to the discovery process. Actually, it can be associated to data processing capacity required by the controller for managing scenarios with a higher number of nodes and links.

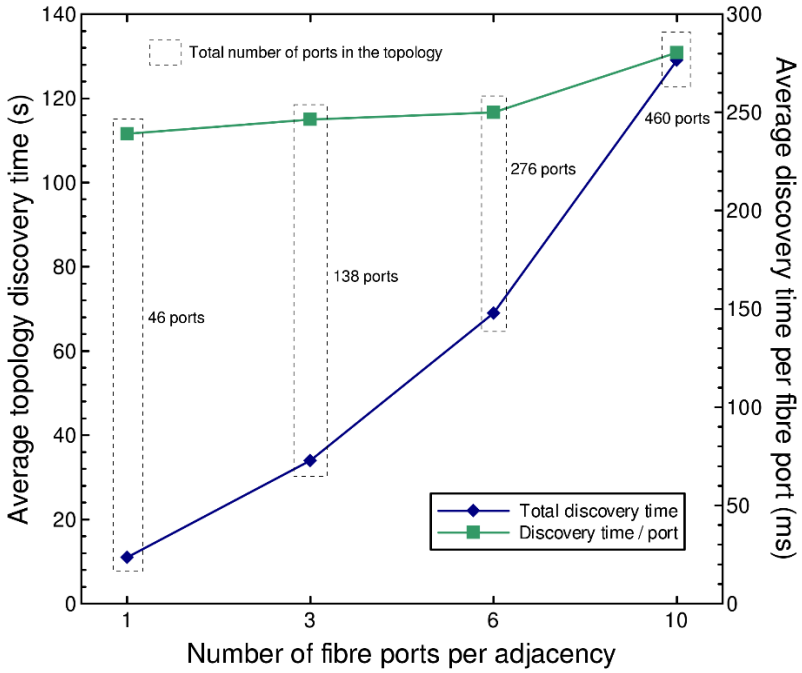


Fig. 5-7. Average topology discovery time (left Y-axis) and per-fibre port discovery time (right Y-axis) in the 14-node network scenario.

As a result, the introduced TSM and OF-based mechanism allows achieving link discovery considering a low-variant per-fibre port discovery time in respect of the number of discovered links. Besides, it provides a considerable limited total time of average topology discovery for different network configurations, then proving the scalability aspects of the proposed sequential discovery mechanism.

5.4 Parallel Discovery

The parallel discovery of links introduces a new variant in adjacency discovery techniques for optical networks. This method in particular, enhances the presented work at the sequential mechanism by proposing the discovery of links in a parallel

manner, through the use of specific link-binding data. To this end, a preservice exchange of test-signals on specific wavelengths is considered for all active fibre ports in the topology, so adjacent nodes can advise the controller on a successful exchange attaching wavelength related data, that can be then used at the controller for correlating links and building the logical topology. The contribution of this work resides then, on the ability of the controller to handle multiple discovery processes at the same time, thus improving the total time of topology discovery and helping overcome the one-by-one discovery limitations of the sequential approach.

Table 5-1 next, presents a high-level comparison of the features introduced by each mechanism. The following sub-sections then, clarify the specific features of the parallel link discovery along with the required set of techniques, extensions and additions that enable its operation. This is to address topics related to the controller-level coordination of the discovery process, the OF messages involved, and the data used to correlate the creation of links, among other details.

Table 5-1 Feature comparison of parallel vs sequential methods.

Features	Parallel Discovery	Sequential Discovery
Preservice mechanism behaviour	X	X
Use of OF v1.0 optical extensions	X	X
Awareness of port supported λ's	X	
Per-process use of available λ	X	
λ tag embedding in OF message	X	
Frequency grid based λ mapping	X	
Simultaneous discovery processes handling	X	

5.4.1 Wavelength-specific Test Signal Mechanism

As a first concern, the design of the parallel mechanism considered the restriction of not being able to exchange link data between adjacent nodes, as it will require terminating connections at every link endpoint to insert or extract discovery related data. In light of this, and as previously analysed in the sequential method, the use

of the TSM at both link ends was selected to be an effective way for confirming reachability between optical ports, as it only represents sending a signalling tone port-to-port without carrying any discovery message or related data. Nevertheless, some weaknesses could be identified on this method, as the controller is only able to recognize links on a one-by-one basis, since confirmations sent back from OF agents do not carry any link-binding data that could make the discovery of each link independent to any other discovery related confirmations. To overcome this, the sequential method was constrained to the use of a time window for the discovery of each link, which finally represented penalties not only in the total topology discovery time but also on the mechanism scalability aspects.

In the presented mechanism, all these aspects were considered, to enhance the TSM to allow mapping links in parallel, without the restrictions of the sequential per-fibre port discovery. To accomplish this, it was mandatory to have some sort of link-binding data that would serve to correlate incoming confirmation messages, considering the one sent by an adjacent node and its correspondent pair sent by the node which started the discovery process. Besides, the controller required a faster triggering of the discovery process after detecting active optical ports, and the ability of identifying links and storing them in the topology database by using the confirmations sent by agents in despite of their arrival time or order of arrival. To this end, the mechanism bases on the use of a wavelengths/frequencies pool, so test signals involved in a specific fibre link discovery are sent over the same wavelength. The controller in this case, requires keeping the knowledge of which wavelengths are being used by active discovery processes, so an available one can be selected to avoid link discovery inconsistencies. In turn, after the discovery process is finished, the designated wavelength should be made again available at the control-level database. The wavelength data in specific, is mapped as a code for its exchange between agent and controller, serving as the link-binding data used by the controller to recognize the correlation between arrival confirmations. In this way, the controller is able to handle simultaneous link discovery processes.

The code mapping for wavelengths data is also implemented at the node-level in the form of a common frequency grid, containing all set of frequencies for all nodes, but which will only be used to map the specific wavelengths supported

by each of them. Therefore, the controller must appropriately assign a wavelength supported by the optical port, or the test signal exchange would not be executed. For this reason, this data must be gathered at the time node features are retrieved to then be considered when triggering the discovery process. As link discovery is executed at the network provisioning phase, before the network becomes operative (i.e., prior to service allocations), the optical transceivers equipped at optical nodes are used to send/receive test signals linked to the different discovery processes, which are triggered at specified fibre ports and through specified wavelengths.

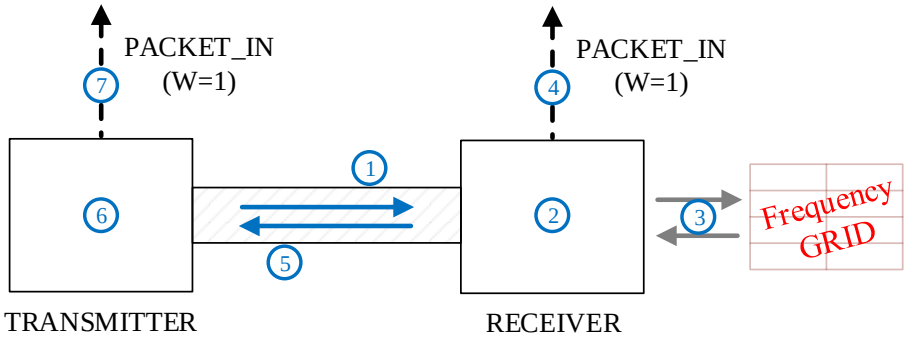


Fig. 5-8. Wavelength-specific Test Signal Mechanism.

Fig. 5-8 above depicts the exchange of test signals for the enhanced mechanism, providing an example for the usage of the assigned wavelength. The specific steps for the process execution are explained as follows:

- **Step 1 - Transmit Initial Tone**: As soon as the controller assigns an available wavelength from the database, the transceiver receives detailed instructions to begin the discovery process, which starts by sending the first test signal at the specified fibre port and over the specified wavelength.

- **Step 2 - Process Received Tone**: Once the receiver senses the test signal at one of its ports, it processes the signal digitally to identify the wavelength.

- **Step 3 - Code Mapping**: Additionally, it uses the common frequency grid to map the identified wavelength to a corresponding code.

- **Step 4 - Notify Controller (Receiver)**: The receiver then builds a specific OF message including the code (W=1) and the flags required to identify it as the receiver. It forwards this message to the controller as a test signal confirmation.

- **Step 5 - Transmit Response Tone**: At the same time, the receiver sends another test signal back through the port where it sensed the original signal and through the same identified wavelength.

- **Step 6 - Process Response Tone and Code Mapping**: At the transmitter, the response test signal is received and the procedure of identifying the wavelength and getting the code is repeated.

- **Step 7 - Notify Controller (Transmitter)**: Finally, the transmitter also builds a confirmation message with the code and the transmitter identifier flag. Then, it forwards the message to the SDN controller.

As a general rule, the controller should execute this process before the link starts transferring data (i.e., at the preservice stage), to guarantee the correct exchange and recognition of test signals. Another important aspect is related to the deletion of links, which should be deleted from the topology database by the controller upon the reception of port-down status notifications from OF agents. This would allow assuring the correct mapping of network resources, so the controller can always maintain an up-to-date representation of the topology.

5.4.2 Discovery Message Flow

The next aspect in the link discovery process was to address the communication between control and data layers, more specifically between SDN controller and the agents, considering then a set of OF messages exchanged between them using independent control TCP/IP connections established by means of a management network. As standard OF versions (i.e., OF v1.0, OF v1.3), do not support optical related data, the same extensions implemented for the sequential method were used to provide this functionality [20], so the capabilities of optical nodes and all their fibre ports could be identified by the controller. In this way, the controller would not only be able to retrieve data describing which fibre ports are in an active port-up state, but also allow it to trigger the discovery processes while supporting the exchange of the specific link-binding data. As the presented mechanism considers more complexity due to its parallel approach, Fig. 5-9 depicts the complete message workflow for the simultaneous discovery of two optical links.

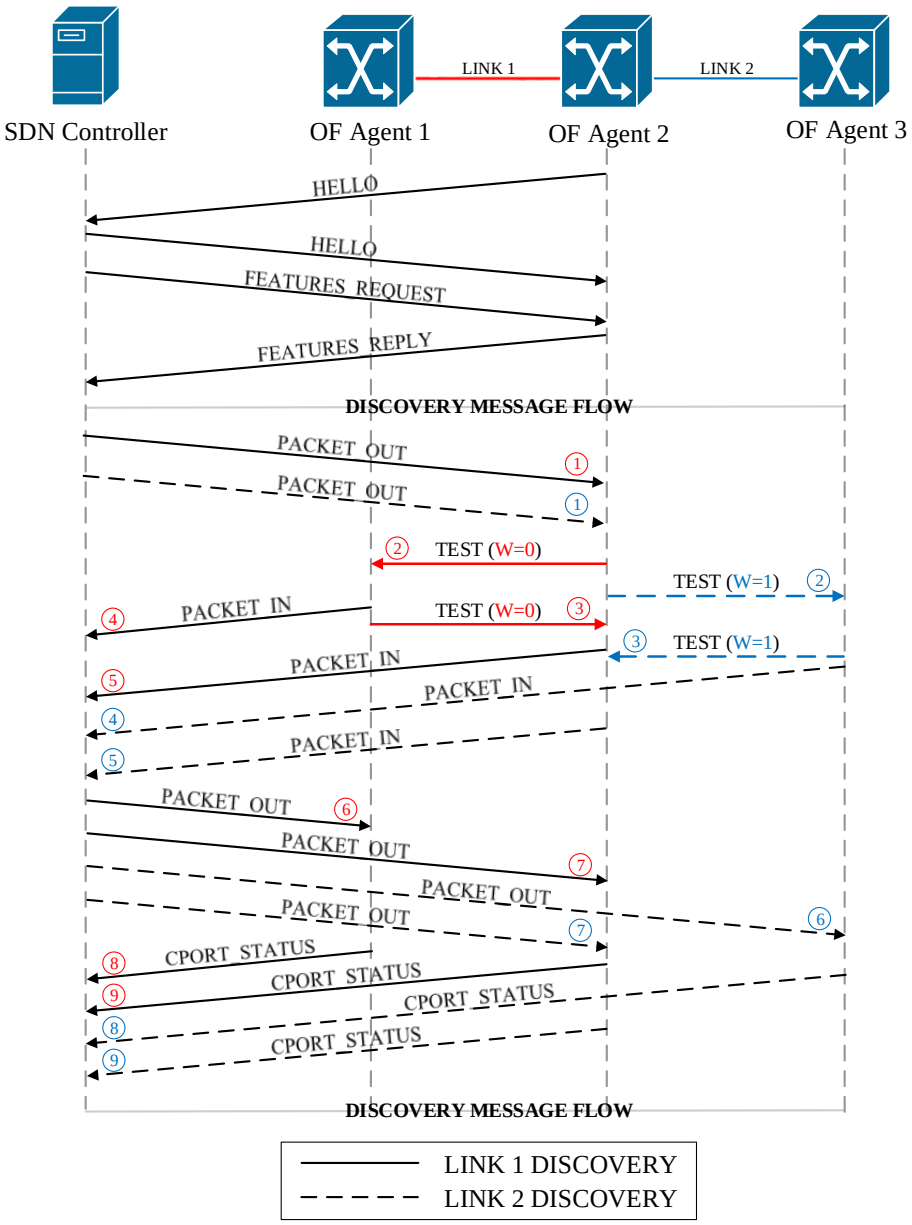


Fig. 5-9. OF-based Message Workflow for Parallel Optical Link Discovery

Before the discovery starts, an optical node connects to the controller through its agent (OF agent 2), establishing the connection by exchanging HELLO messages. Then, the controller sends a FEATURES_REQUEST message asking for its capabilities which are sent back by the agent through a FEATURES_REPLY message. At this stage, the controller has identified which fibre ports are active, so it begins sending OF messages to trigger one to many link discovery processes.

To this end, the controller sends PACKET_OUT messages (step 1) towards OF Agent 2, using the SET_DL_SRC action flag to distinguish them as discovery

triggering messages and providing orders to test adjacency reachability through the specified ports and via specifically assigned wavelengths. Following this, the agent sends test signals (step 2) to both adjacent nodes (OF Agents 1 and 3). Once peers detect and process these signals, both send back a response signal (step 3) through the same port/wavelength and construct confirmation PACKET_IN messages to let the controller know about the recent events (step 4). The agent which started the process receives then both response signals from peers and sends other confirmation PACKET_IN messages to the controller (step 5), one for each successful test signal exchange. To differentiate PACKET_IN messages arriving to the controller, OF Agent 2 uses an ACTION reason flag to identify itself as the transmitter node, while OF Agents 1 and 3 use a NO_MATCH reason flag to identify themselves as receivers or peers. The controller uses these reason flags along with other link-discovery related data (i.e., port numbers and wavelength codes) to correlate all the received data. By using this link-binding data, the number of discovery processes executed at a specific point in time is not relevant for the controller, as it is able to recognize which messages belong to which process. In turn, the only limitation now comes to be the number of supported wavelengths at fibre ports, as only a single wavelength can be assigned at the same time for testing at a specific port. After the process ends, the controller releases the wavelength so it becomes available for assigning it to a new discovery process.

Once the controller correlates data received from agents at both link ends, it sends PACKET_OUT messages notifying them about their connected peers (step 6-7). In contrast to the ones used at the first step, an SET_DL_DST action flag is set at these messages so that the agents can identify them as the ones confirming peer data. To conclude, after adjacent nodes learn peer information, they expose it again to the controller in the form of CPORT_STATUS messages (step 8-9), allowing the Topology Manager (TM) module at the controller level to process this data and construct new optical links at the topological database.

In this way, the described message workflow enables the mapping of links in a parallel way, using standard OF messages (i.e., PACKET_IN, PACKET_OUT), and extended ones such as CPORT_STATUS. As a result, the use of the extended protocol is considered crucial for the optical link discovery process to work.

5.4.3 Software Requirements

Concerning the software requirements for enabling the mechanism functionalities, the parallel discovery process bases on the use of the extended SDN architecture analysed in *Section 5.3.3*. Considering the already implemented functions in this scenario, the enhancements done to support this version of the mechanism can be identified according to their specific layer of application, as detailed next:

- **Infrastructure Layer:** At this level, OF agents must be extended to support the mapping of wavelengths to their correspondent code. To this end, agents have to digitally process incoming test signals to detect the used wavelength and get the code by checking the common frequency grid. Considering at this grid only the supported frequencies at the node fibre ports. Then, agents should be able to attach this link-binding data into specific OF messages sent to the controller.

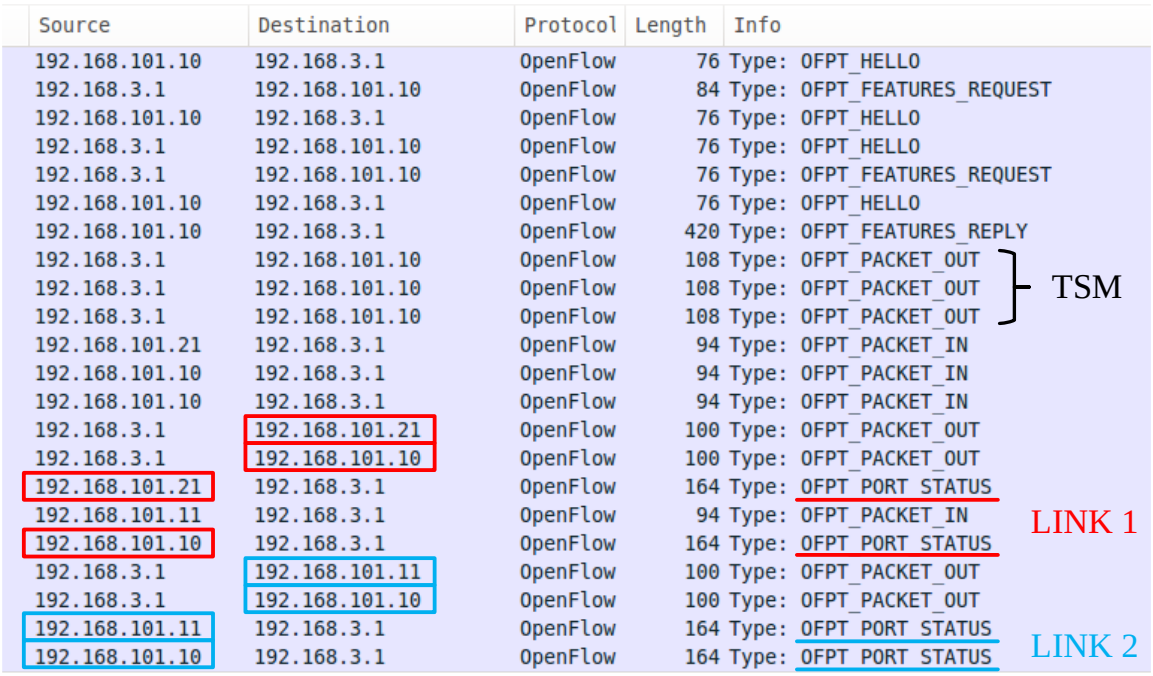
- **Control Layer:** Configurations at this layer require extending the Optical Link Discovery (OLD) module to handle multiple discovery processes at once, where the maximum number of simultaneous processes is directly associated to the number of supported wavelengths. Besides, the OLD should be able to use the retrieved link-binding data to correlate the incoming confirmation messages by identifying to which link and to which process they are related.

5.4.4 Experimental Assessment

The experimental assessment of this mechanism is based on the same setup as seen in *Section 5.3.4*, considering the 9-node and 14-node scenarios but enabling in this case, the emulated support of 10 wavelengths per-link (i.e., at the optical fibre port). Table 5-2 next, presents a summary of the key topology parameters.

Table 5-2 Optical Network Topologies Key Parameters.

	No. of Nodes	No. of Links	No. of Supported Wavelengths
Topology 1	9	22	10
Topology 2	14	46	10



Source	Destination	Protocol	Length	Info	
192.168.101.10	192.168.3.1	OpenFlow	76	Type: OFPT_HELLO	
192.168.3.1	192.168.101.10	OpenFlow	84	Type: OFPT_FEATURES_REQUEST	
192.168.101.10	192.168.3.1	OpenFlow	76	Type: OFPT_HELLO	
192.168.3.1	192.168.101.10	OpenFlow	76	Type: OFPT_HELLO	
192.168.3.1	192.168.101.10	OpenFlow	76	Type: OFPT_FEATURES_REQUEST	
192.168.101.10	192.168.3.1	OpenFlow	76	Type: OFPT_HELLO	
192.168.101.10	192.168.3.1	OpenFlow	420	Type: OFPT_FEATURES_REPLY	
192.168.3.1	192.168.101.10	OpenFlow	108	Type: OFPT_PACKET_OUT	TSM
192.168.3.1	192.168.101.10	OpenFlow	108	Type: OFPT_PACKET_OUT	
192.168.3.1	192.168.101.10	OpenFlow	108	Type: OFPT_PACKET_OUT	
192.168.101.21	192.168.3.1	OpenFlow	94	Type: OFPT_PACKET_IN	
192.168.101.10	192.168.3.1	OpenFlow	94	Type: OFPT_PACKET_IN	
192.168.101.10	192.168.3.1	OpenFlow	94	Type: OFPT_PACKET_IN	
192.168.3.1	192.168.101.21	OpenFlow	100	Type: OFPT_PACKET_OUT	
192.168.3.1	192.168.101.10	OpenFlow	100	Type: OFPT_PACKET_OUT	
192.168.101.21	192.168.3.1	OpenFlow	164	Type: OFPT_PORT_STATUS	
192.168.101.11	192.168.3.1	OpenFlow	94	Type: OFPT_PACKET_IN	LINK 1
192.168.101.10	192.168.3.1	OpenFlow	164	Type: OFPT_PORT_STATUS	
192.168.3.1	192.168.101.11	OpenFlow	100	Type: OFPT_PACKET_OUT	
192.168.3.1	192.168.101.10	OpenFlow	100	Type: OFPT_PACKET_OUT	
192.168.101.11	192.168.3.1	OpenFlow	164	Type: OFPT_PORT_STATUS	
192.168.101.10	192.168.3.1	OpenFlow	164	Type: OFPT_PORT_STATUS	LINK 2

Fig. 5-10. Wireshark capture describing the message workflow for the parallel link discovery process between three network nodes.

First, to illustrate the mechanism operation, Fig. 5-10 depicts the message exchange between the controller, a recently connected node (IP: 192.168.101.10) and its two adjacent nodes. In this case, two different discovery processes are triggered, one for the LINK1 connection towards the first adjacent node (IP: 192.168.101.21) and another for LINK2, towards the second (IP: 192:168:101:11).

The message flow in turn, follows the analysed workflow in *Section 5.4.2*, considering the exchange of HELLO messages and the exposure of node features through the FEATURES_REPLY message. As shown in the figure, the controller triggers then the TSM on all the recognized active fibre ports, so the OF agent can instruct sending the test signals at the specified ports/wavelengths. At the time adjacent node agents sense the signals and send back responses, they send confirmation of this events to the controller by means of PACKET_IN messages.

Once the node which started the test signals receives the responses, it also forwards confirmation messages. By processing these messages at the controller, OLD correlates the specific process/link data and sends peer information to the correspondent agents via PACKET-OUT messages. As a last step, agents send

the final port update using the CPORT_STATUS messages, which as seen in the figure, represent a pair of messages for each link (i.e., LINK1, LINK2). To finish, TM retrieves this data and creates the links at the topology database, thus making optical links visible to consumers feeding on this sensitive information.

5.4.4.1 Comparison between Sequential and Parallel Mechanisms

Going back to the full topology configurations, the main objective for the tests was checking the topology discovery times on both 9-node and 14-node scenarios to compare them with the ones achieved with the sequential mechanism. Note that the benchmarking performed between the two methods does not include the per-fibre port discovery time as its considered similar for both. Then, as the focus is set on analysing the average discovery for the complete topology, it is worth noticing the fact that the parallel method is able to handle several discovery processes simultaneously, which would allow expectations for a significant reduction on the average topology discovery when comparing the two different mechanisms.

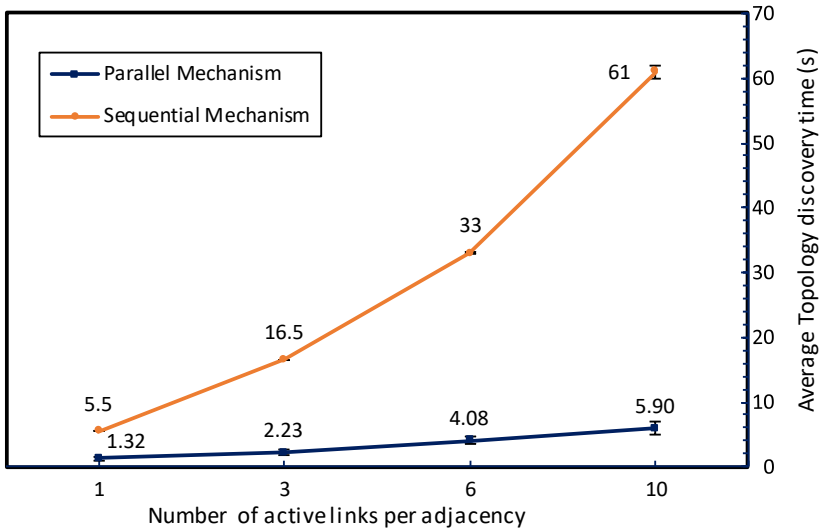


Fig. 5-11. Average topology discovery time comparison with parallel/sequential mechanisms in 9-node network scenario.

Fig. 5-11 and Fig. 5-12 in this sense, expose the comparison between both methods including the 95% confidence interval for the average topology discovery time. The results shown, are the calculated averages after executing the discovery process 10 times on each different network configuration and for each method, following the same methodology as in *Section 5.3.4*. To make this calculations, the

controller log times were used to identify the moment where the discovery process is triggered, to then compare it to the time link information is made available at the topology database. As depicted in the figures, similar comparisons can be found in spite of the tested scenario, which in turn seem to be directly related to the existent gap in link processing capacity between the one-by-one link discovery basis of the sequential method versus the simultaneous link discovery basis of the parallel one.

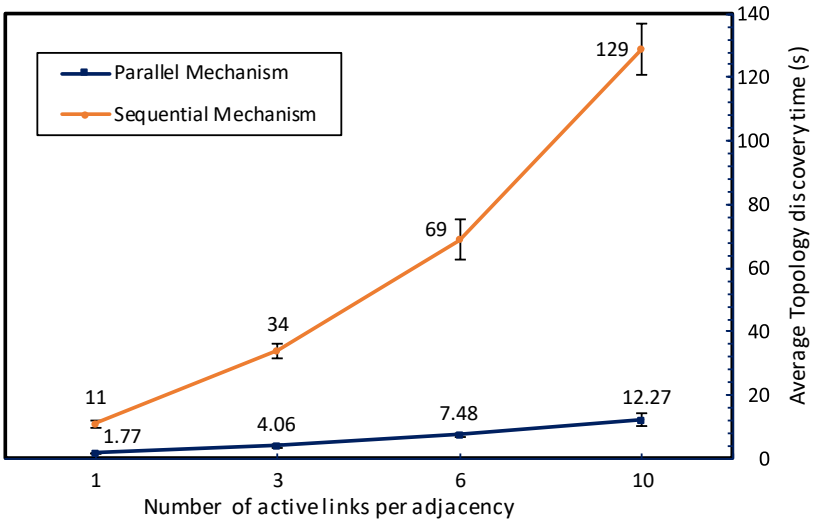


Fig. 5-12. Average topology discovery time comparison with parallel/sequential mechanisms in 14-node network scenario.

In Fig. 5-11 more specifically, the execution time results obtained with the proposed parallel discovery mechanism, already present important reductions in all types of configurations compared to the ones of the sequential discovery, even considering the fact that ports were arranged only with 10 supported wavelengths. In the 9-node scenario for instance, the topology discovery time reduces from 5.5 seconds in sequential mode to only 1.3 seconds in parallel discovery, when only one fibre link is set between adjacencies (22 links). Then, the gap starts increasing as the number of links to be discovered continues to grow. For example, with three link per adjacency (66 links), time reduces from 16.5 to 2.23 seconds. Moreover, in cases with six links (132 links) and ten links (220 links), the topology discovery time reduces from 33 to 4 seconds and from 61 to 5.9 seconds, respectively.

As for the 14-node scenario results, given in Fig. 5-12, the differences in time between both mechanisms are even greater. In this case, an analysis can be done on the time reduction impact considered at the configuration with more links.

Then, with ten links per adjacency (460 links), the average topology discovery time goes from taking 129 seconds with the sequential approach to only 12.2 seconds with the parallel one. Although previous cases show, at least 80% of improvement in the average time, the analysed case shows a reduction of around 90% compared to the time it used to take before, thus showing the scalability potential of the parallel method when applied to scenarios with an even higher number of links. In this matter, the mechanism behaviour of only acting in front of a new port event (i.e., port-up, port-down) also becomes decisive in front of the periodical discovery basis found in regular discovery protocols, as it avoids stressing the controller with constant message processing, that could consequently affect not only the topology discovery process but also other applications.

Fig. 5-13 then shows the relation between the number of configured links per-adjacent node and the total number of links to be discovered, used to achieve the discussed test results at both considered scenarios. As for the parallel process, is important to mention the ability of the controller on handling a number of ongoing discovery processes equal to the number of wavelengths supported by fibre ports (i.e., 10 in this case), which proves the major improvements in discovery times.

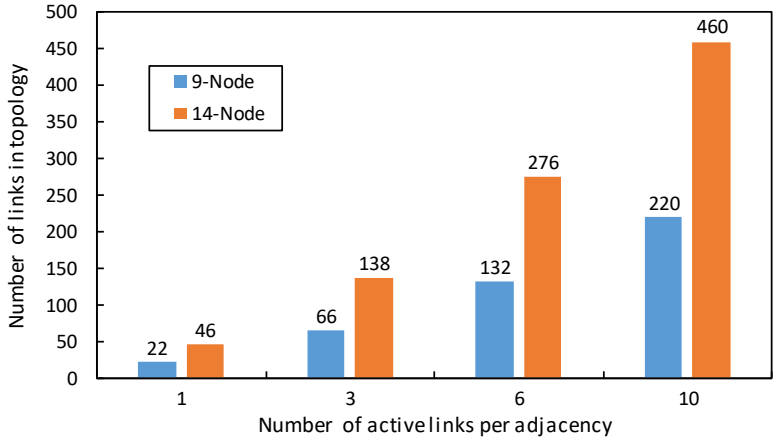


Fig. 5-13. Links to be discovered according to number of links per-adjacency.

5.5 Conclusions

Considering the more common use of optical solutions for different network scenarios, powered by the logically centralized control of the SDN technology, the importance of mapping a correct representation of the network topology becomes

crucial, so that consumers of this information can operate as expected. Regarding this, the need for dynamic optical link discovery mechanisms has been identified in this work to overcome the limitations of currently used techniques. In this way, two novel proposals have been introduced to address such necessity by means of coordinated sequential and parallel OF-based discovery processes, providing new methods for discovering optical topologies, thanks to their preservice nature.

The work presented in this chapter, has furtherly proven the functionality of both methods by making use of emulated network scenarios, where the times for per-fibre port discovery and average topology discovery, besides a given analysis on the message exchange workflow, have been presented to make a comparison between the sequential and the parallel approach. As a result, the parallel method has been recognized as the most scalable method, considering its ability to manage multiple discovery processes at once while still guaranteeing the correct construction of the network optical topology at the controller level.

5.6 Publications

The work in the scope of this thesis chapter has generated the following scientific journal and conference publications:

1. **R. Montero**, F. Agraz, A. Pagès, J. Perelló and S. Spadaro, "Dynamic topology discovery in SDN-enabled Transparent Optical Networks," 2017 International Conference on Optical Network Design and Modeling (ONDM), Budapest, 2017, pp. 1-6. <https://doi.org/10.23919/ONDM.2017.7958525>

2. **R. Montero**, F. Agraz, A. Pagès, J. Perelló and S. Spadaro, “SDN-based parallel link discovery in optical transport networks,” Transactions on Emerging Telecommunications Technologies, volume 30, e3512, 2019. <https://doi.org/10.1002/ett.3512>

Chapter 6. Control and Orchestration at Next-generation DC Architectures

The next chapter presents the efforts done to enable network slice deployment in next-generation DCN architectures, considering the monitoring and reconfiguration tasks for guaranteeing their quality of service (QoS). To this end, a SDN-based architecture to support the control and orchestration of network resources at the DC segment has been designed and experimentally assessed.

6.1 Next-generation DCN Architectures

The orchestration and control of DCN sub-slices, is focused in this case on its application over novel DCNs infrastructures, following the trend towards the use of all-optical networks at future DCNs deployments [4], which represents changes in their architectural design. In this sense, the OPSquare flat DCN architecture [66] scenario has been selected, to serve as the proof-of-concept use case to address this particular topic. This novel architecture is based on the use of fast optical switching technologies, to overcome the high latency, high power consuming Electro-Optical/Opto-Electrical conversions (O/E/O) and input/output limitations (I/O) of multi-tier electrical switches architectures. In this way, the introduced high data rate/format signal switching allows reaching high aggregation bandwidths.

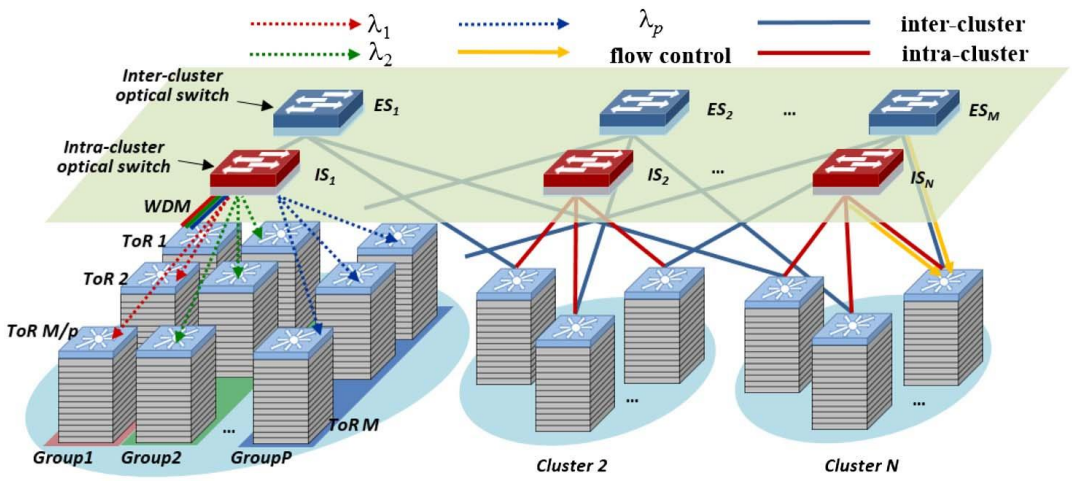


Fig. 6-1. Novel Flat DCN Architecture

[Adapted from *F. Yan, "OPSquare: A flat DCN architecture based on flow-controlled optical packet switches", 2017*]

Fig. 6-1 depicts the novel DCN architecture use case, where specific intra-cluster optical switches (IS), inter-cluster optical switches (ES) and top-of-rack (ToR) switches are used to allow different types of connectivity (i.e. intra-rack, intra-cluster or inter-cluster) between DC racks organized in groups and clusters.

In this sense, ESes allow inter-cluster communication between the ToRs of N clusters, while the ISes handle intra-cluster connectivity for M ToRs, considering the use of a specific wavelength per-group of racks. Finally, each ToR manages the corresponding intra-rack server communication. This configuration enables at least two different paths for the connectivity between any pair of racks, thus making the architectural solution fault tolerant and very scalable for large DCN scenarios.

6.2 Orchestrated SDN-based DC Architecture

As for the DC sub-slices, their deployment considers the allocation of computing resources, usually in the form of VNFs, which are provisioned according to the available compute resources and interconnected through the DCN, making use of the available connection paths at the given use case (i.e., via IS/ES/ToR switches). As a result, the deployment task entails managing compute and network resources not only at the provisioning stage but also during the complete sub-slice lifecycle.

In this thesis, the selected DCN architecture use case is enhanced with control and orchestration capabilities to handle the efficient management of DC sub-slices, considering the full control of the resource provisioning task and the maintenance of the sub-slice QoS by reacting appropriately in case of potential network degradations (e.g., bandwidth saturation, optical packet collisions, etc.). Fig. 6-2 describes the SDN-controlled and orchestrated architecture, consisting of the orchestration (top), control (middle) and the infrastructure layers (bottom).

At the lowest level, the novel DCN configuration based on the use of ES, IS and ToR switches is represented, where each switch connects towards the SDN controller via an integrated OF agent in order to expose their switching capabilities and enable the direct configuration of network connections through flow control. Besides, the agents are set to provide the controller with Port and Flow statistics,

which in turn will be aggregated per-network slice to perform monitoring tasks and allow acting in response to violations of the pre-defined QoS thresholds. In this case, southbound communication is enabled by means of a specific version of the OF protocol, extended for supporting fast optical switching related data [67].

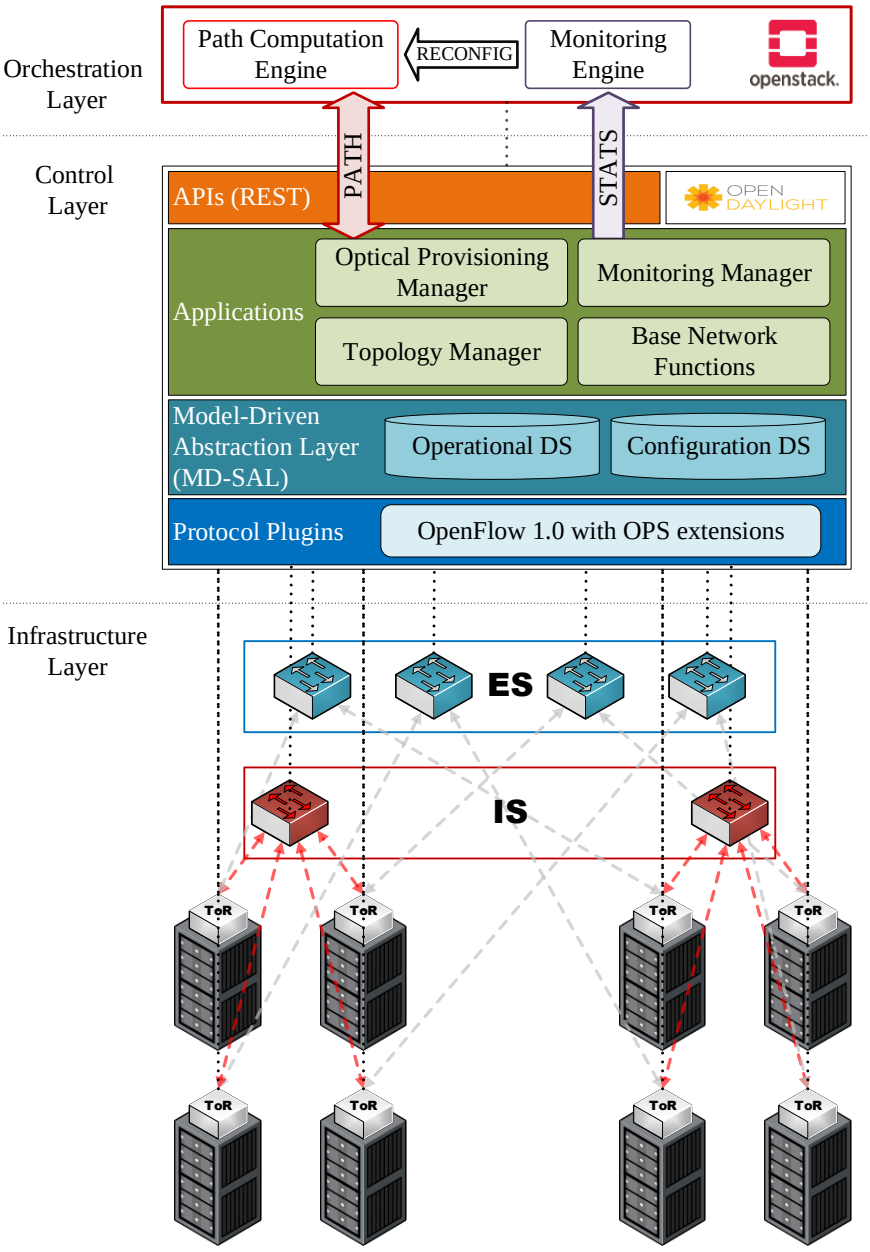


Fig. 6-2. Orchestrated SDN-based DCN Architecture.

The control layer then, is implemented considering the architecture of the OpenDaylight (ODL) SDN controller, where new modules are added while others are extended, to adapt its operational behaviour to the requirements of the given scenario. Regarding extensions to existent modules, the OF Plugin/Java modules

allocate the mentioned protocol extensions that enable attributes collection, flow configuration and statistics gathering between agents and the controller. Moreover, the Topology Manager (TM) module was also extended to identify and classify data planes devices, defining a new fast optical switch type for this particular case.

The Optical Provisioning Manager (OPM) module was introduced in turn, to handle the configuration of infrastructure-level switches (i.e., ESes, ISes, ToRs), required to set up connectivity requirements for network slice deployment. To accomplish this, the OPM sends flow configuration (i.e., FLOW_MOD) messages to the switches OF agents to configure their look-up-table accordingly. Besides, OPM is also able to coordinate with the orchestrator, the configuration of the best interconnection path between VNFs. Another addition considers the development of a new Monitoring Manager (MM) module, which enables the collection of port and flow statistics from data plane devices, to then expose this information towards the orchestration layer via REST APIs.

At last, the orchestration layer is defined by the use of a network orchestrator to manage all the computation and network resources coordination. In this case, it is based on the use of the OpenStack platform, where new architectural modules are introduced to enhance its functionalities. The Path Computation Engine (PCE) in this way, relies on the abstracted topological data retrieved from the controller database to provide a ToR-to-ToR path computation service, which can be used upon a Path Request received from the OPM (i.e., at slice provisioning), or at the time path reconfigurations are required (i.e., at slice maintenance). The Monitoring Engine (ME) in turn, subscribes to the monitoring data exposed by the MM and aggregates it to the network slice level, so it can identify when the QoS of a specific slice gets compromised to then trigger the reconfiguration of inter and intra cluster network connections in response, via the PCE.

6.3 Operational Workflows

To better understand the process of network slice provisioning and maintenance, operational workflows for both stages are discussed next, in a way to also clarify the role of the added control and orchestration software components.

6.3.1 Slice Provisioning Stage

The provisioning stage starts after defining the network slice composition in terms of computational resources (e.g., VNFs) and network resources requirements (i.e., required connectivity), considering as well the definition of the expected slice QoS. The orchestrator then, analyses the availability of compute resources at the DC (step 1) and allocates the slice VNFs accordingly (step 2). Additionally, it requests the SDN controller to provide the required network connectivity for the deployed VNFs (step 3). The computation of the path is then coordinated between the OPM and the PCE (step 4), so flow configuration gets triggered to the specified switches (step 5). Once all flows are pushed to the devices, end-to-end connectivity between VNFs is enabled and the slice functionality becomes operative (step 6). Finally, the orchestrator sets preventive thresholds at the ME considering the QoS agreements of the slice (step 7). Fig. 6-3 next, describes the complete provisioning workflow.

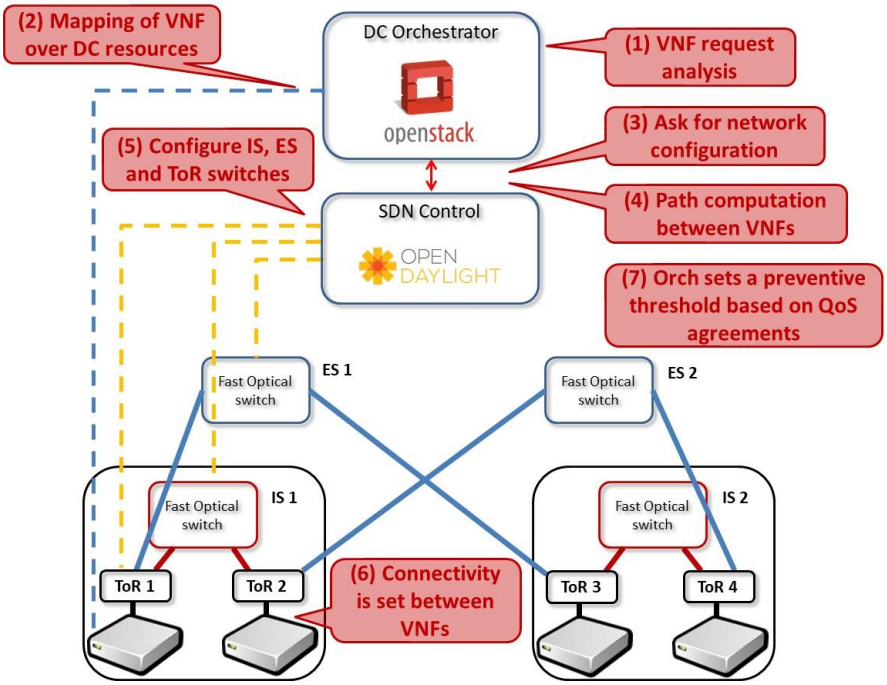


Fig. 6-3. Slice Provisioning Operational Workflow.

6.3.2 Slice Maintenance Stage

The maintenance stage in turn, begins after the slice becomes operative, through the collection of monitoring Flow/Port statistics data at the MM (step 1), which is performed continuously so it can be exposed via REST APIs (step 2). The ME then,

consumes this information and aggregates it to the network slice level (step 3). In case pre-defined thresholds are surpassed, the ME triggers the reconfiguration of the affected slice connections by sending a request to the PCE (step 4). Once an alternative path is computed, a request is sent to the controller for its configuration (step 5). At last, the SDN controller pushes the new flows to all the required devices (step 6) so the connectivity through the alternative path is set (step 7), allowing monitoring tasks to begin again the statistics processing and thus guaranteeing the agreed slice QoS. The complete workflow is depicted in Fig. 6-4.

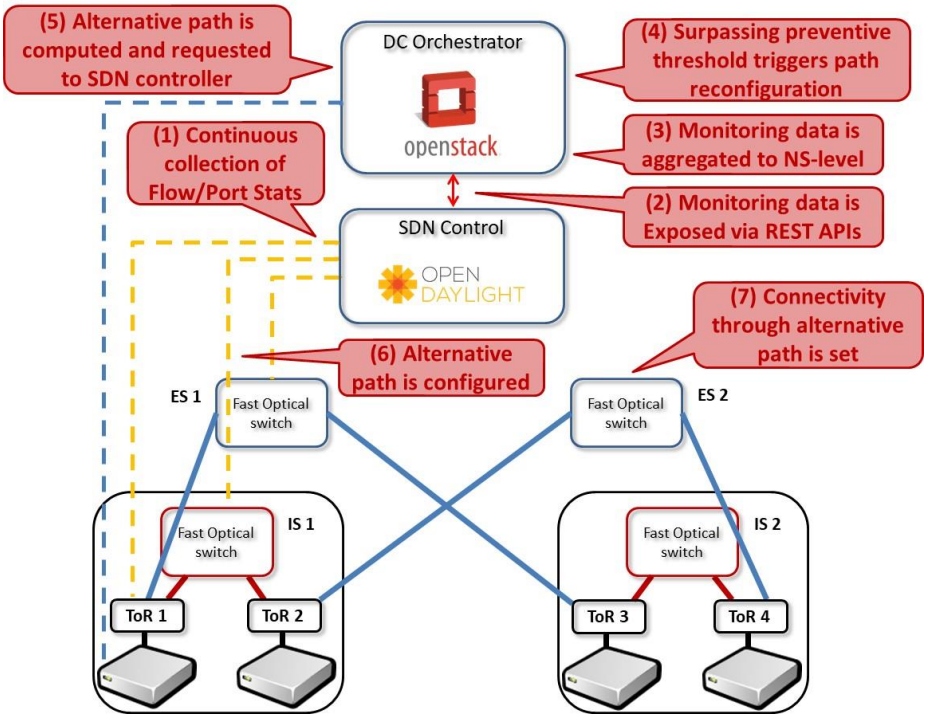


Fig. 6-4. Slice Maintenance Operational Workflow.

6.4 Slice Provisioning and Reconfiguration

An example of network slice provisioning and reconfiguration is given on Fig. 6-5, where the deployment of NS1 is described. From the virtual layer point of view, the definition of the slice comprises interconnected VNF1 and VNF2. At provisioning then, and following the set of given requirements, the orchestrator allocates VNFs in racks 1 and 4 respectively according to the available computing resources. The controller on other hand, coordinates with the PCE the configuration of the best path between ToR 1 and ToR4 to enable the required connectivity. In this case, as seen in the figure, the configured physical path includes IS1<->ToR2<->ES2.

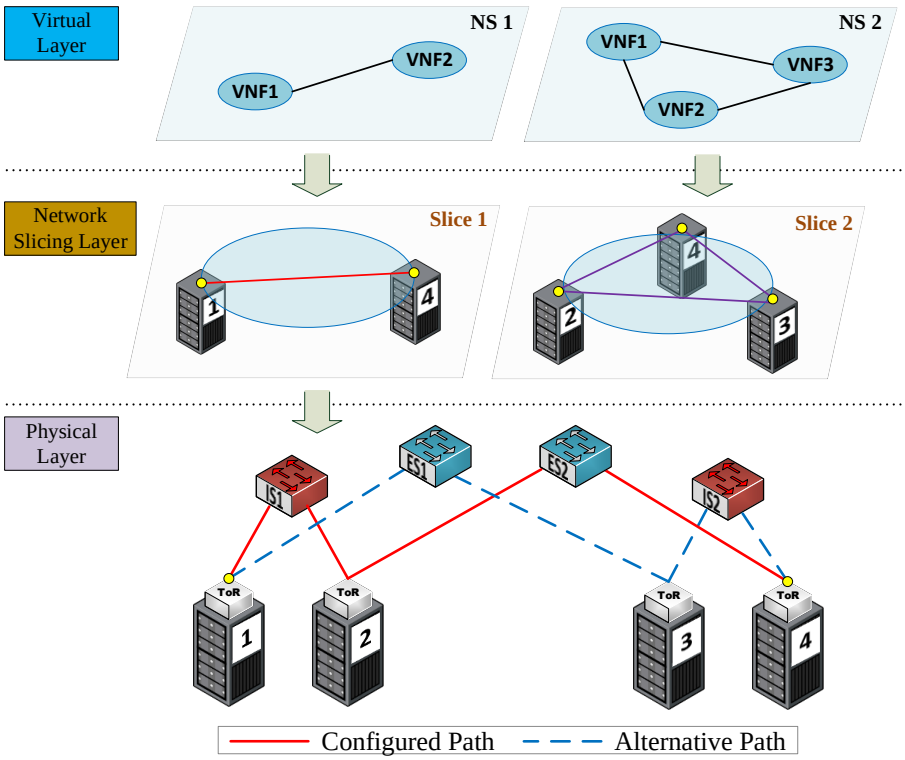


Fig. 6-5. Slice Provisioning and Reconfiguration.

Once the provisioning stage is done, the controller starts collecting statistics to monitor the number of packet losses due to collisions at fast optical switches. In this way, if any pre-defined packet-loss guard threshold is violated, the ME detects it and coordinates with the PCE the reconfiguration of VNF1 to VNF2 connection through an alternative path (e.g., ES1<->ToR3<->IS2), acting preventively to help secure QoS levels and without affecting optical packets transmission.

The request for a second slice with multiple interconnected VNFs (NS2) is also illustrated in the figure. In this regard, the novel DCN architecture enables the configuration of different paths between racks, using fast optical switching labels to identify connections belonging to each particular slice. In this way, multiple slices can be allocated and maintained over the shared infrastructure.

6.5 Experimental Results

To validate the correct provisioning and reconfiguration of network slices, ES, IS and ToR switches emulators where used, defining two set of testbed configurations for small and medium scale scenarios. Table 6-1 next, details the used parameters.

Table 6-1 Testbed Scenarios Configuration.

Size	ToR	IS	ES	λ	p	q	N	M
Small	8	2	4	2	2	1	2	4
Medium	32	4	8	4	4	1	4	8

The small size testbed scenario is configured with 14 emulated network devices (including 8 ToRs, 2 ISes and 4 ESes), where switches are organized in *N* clusters (*N* = 2), considering *M* ToRs each (*M* = 4). For the intra-cluster and inter-cluster connectivity then, 2 wavelengths are available to allow reaching *F* ToRs, where *F* = *M/p* (*p* = intra cluster groups) or *F* = *N/q* (*q* = inter-cluster groups) depending on the case. At each rack, the ToR interconnects *k* servers. If 40 servers are considered per rack, this scenario is able to interconnect 320 servers.

At the medium size scenario, 44 devices are configured (32 ToRs, 4 ISes and 8 ESes). The number of wavelengths, is doubled to 4 in this case, while the number of clusters (*N* = 4) and ToRs per cluster (*M* = 8) is also doubled. Using the same *k* value as before, the scenario allows interconnecting up to 1280 servers.

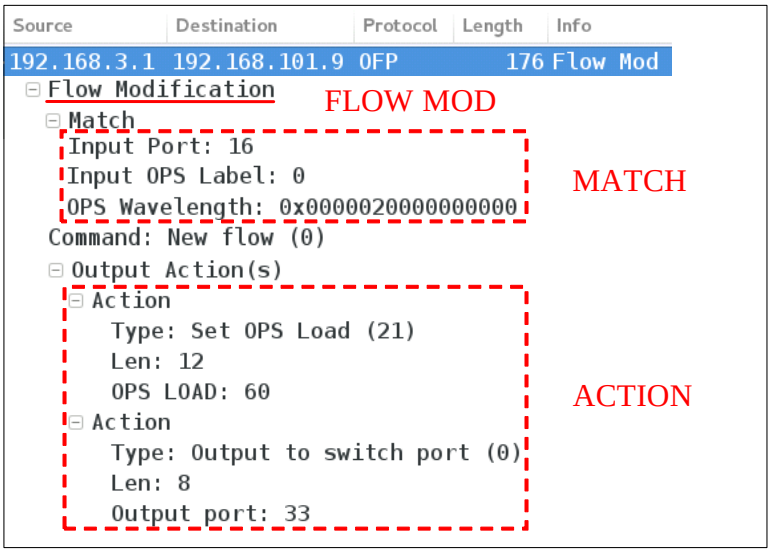


Fig. 6-6. OF Flow Modification Message Capture.

The configured device emulators, consider OF agents to connect to the SDN controller, using the extended protocol for fast optical switching. An example of the extended OF FLOW_MOD message then, used to configure optical connections

at both testbed scenarios, is given in Fig 6-6. As it is exposed, the matching values defined at the extended flows (i.e., Input_Port, OPS_Label, OPS_Wavelength) together with the set of given action values (i.e., OPS_Load, Output_Port), enable the forwarding of optical packets required to set up slices network connectivity.

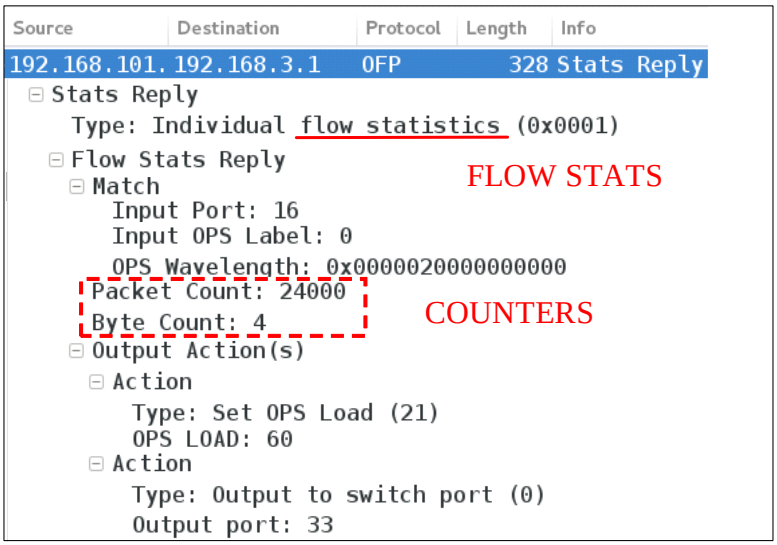


Fig. 6-7. OF Flow Statistics Message Capture.

Fig. 6-7 in turn, shows the per-flow gathering of statistics counters (i.e., packet count and number of collisions) from the connected optical devices via the OF FLOW_STATS message. This information is retrieved and exposed by the MM, so the ME can further aggregate it to the slice level. In this case, the collisions counter is mapped over the byte count field of the depicted message to avoid implementing additional protocol extensions.

As previously analysed, the QoS agreements defined at the network slice request, are translated during the provisioning stage into a set of preventive guard thresholds. The maintenance of the slice in this way, resides in the fact that control and orchestration systems are able to act before QoS agreements are violated. In turn, for the experimental tests, a threshold specifically related to optical packet losses is defined. In this manner, the surpassing of this threshold would trigger the preventive path reconfiguration process, as discussed in *Section 6.3.2*.

Following this approach, tests were performed in both emulated scenarios, to calculate the time the system takes to set an alternative connection path upon

the detection of significant packet-losses (i.e., identifying a reached threshold) at a specific optical connection. In this way, the reconfiguration time on the small-scale scenario was measured at 124 milliseconds (ms), while a time of 146 milliseconds (ms) was achieved at the medium size scenario. These results in particular, show a relative low impact in the total time of reconfiguration in respect of the number of controlled devices, thus proving the process scalability aspects.

6.6 Conclusions

The introduction of novel architecture solutions for DCNs, represents taking also considerations on the required control and orchestration systems that enable the optimization and configuration of underlying network resources. In this particular work, a fully orchestrated SDN-control architecture has been proposed to support fast switching technologies used in next-generation DC infrastructures.

The design and operation of the architecture, has been clarified by means of operational workflows and configuration examples. As well, an emulated testbed has been employed to test the provisioning of flows and the gathering of specific monitoring information. Finally, QoS guaranteeing has been proven by enabling taking preventive slice reconfiguration actions, demonstrating also its scalability.

6.7 Publications

The achievements reached on this thesis chapter have been characterized in the subsequent conference publication:

1. **R. Montero**, N. Calabretta, F. Agraz, A. Pagès and S. Spadaro, "SDN-controlled and Orchestrated OPSquare DC Architecture enabling Network Slice Deployment with QoS guarantees," 2018 Photonics in Switching and Computing (PSC), Limassol, Cyprus, 2018, pp. 1-3. <https://doi.org/10.1109/PS.2018.8751416>

PART III

OPTIMIZATION IN INTER-DC NETWORKS

Chapter 7. 5G Service Provisioning through Network Slicing

The following chapter opens a new part of the thesis development considering the resource usage optimization on inter-DCN scenarios across multiple segments of the network. This one in particular introduces the proposal of a 5G-enabling architecture capable of handling service provisioning thanks to the coordinated use of Network Slicing, SDN, NFV and orchestration technologies.

7.1 5G End-to-end Orchestration

As introduced in *Section 2.5*, the road to 5G telecommunications comprises the support for a wide variety of services, which include specific characteristics to address solutions for a set of given 5G verticals (i.e., automotive, industry, e-health, energy, etc.), considering between them the support of different KPIs related to the requirements set for defined 5G service types (i.e., eMBB, URLLC, mMTC).

To leverage the performance of current networks to the 5G requirements, a set of technologies were introduced (i.e., Network Slicing, SDN, NFV). Besides, the use of software components across architectural layers was also recognized as fundamental, to manage the use of network, compute and storage resources. In this regard, the architecture layout given in *Section 2.5.1*, based on the analysed proposals [30, 45, 68], defined the role of such components.

The challenge in this sense, comes as 5G services must be deployed over a set of network segments (e.g., Data Centre, Core, Metro, Access) while sharing the resources of a common infrastructure. This means that a coordination entity is required between components and segments to enable allocating and configuring resources for the provisioning of services, as well as for being able to maintain their expected performance during the service lifecycle. This coordination in turn, has received attention from the community [69, 70, 71, 72, 73, 74, 75], where proposals have been made to better define how to orchestrate the required control and orchestration level tasks, to guarantee the end-to-end management of 5G services.

7.2 5G Meta Architecture

Regarding the standards defined for the 5G architecture, both ETSI and 3GPP driven views have been analysed, to elaborate a consensus “meta-architecture” summarizing all the presented efforts [45]. This common architecture, despite not being essentially precise or definitive, intends to provide a base structure for the components in charge of handling the management and operation of a 5G system. Fig. 7-1 next, illustrates the single-domain version of the 5G meta architecture.

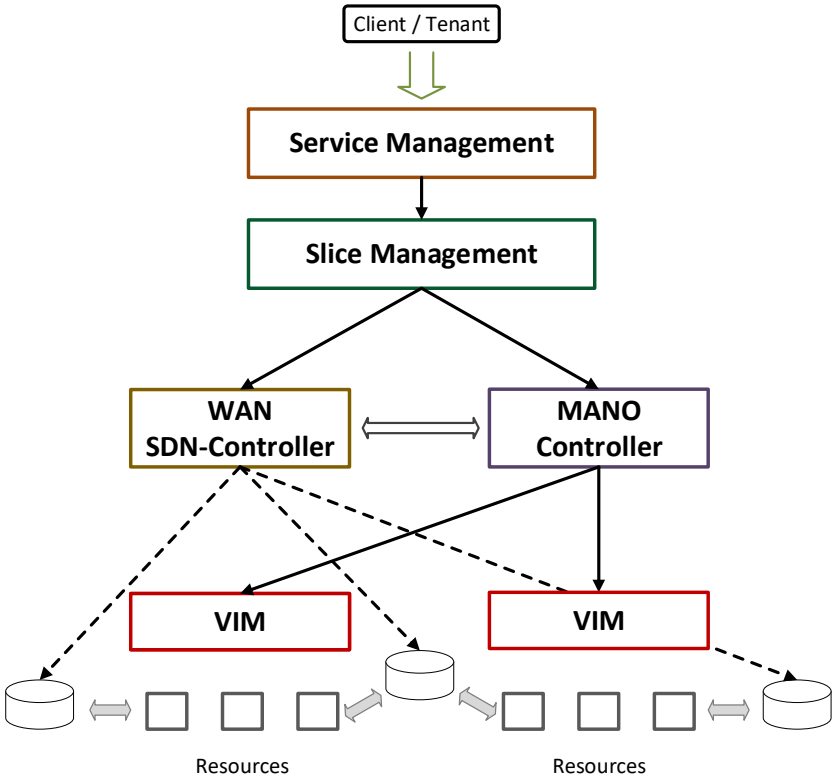


Fig. 7-1. Single-domain 5G Meta Architecture.

The structure of the architecture, considers 5G services to be provisioned over a set of specifically allocated resources (i.e., network slices) by means of control and management related components. To this end, each slice has to be configured and maintained by coordinating tasks at the slice management level through components such as Management and Orchestration (MANO) controllers and Wide Area Network (WAN) SDN controllers. In this manner, the required computing and storage resources can be allocated at a particular segment via its Virtual Infrastructure Manager (VIM), while network resources at the WAN can be directly configured to enable the necessary inter-segment connectivity.

Taking as a reference the analysed common architecture proposed by the 5G-PPP and the efforts of the community to prove the applicability of the standards, this work introduces a network slicing-based framework proposal for 5G service provisioning and maintenance focused on single-domain multi-segment scenarios. For its definition, the point of view of the Network Slice Instance (NSI) lifecycle has been considered to set up two versions of the framework. At this chapter then, the one defined for the provisioning stage is presented, based on the use of a novel coordinator component able to manage the required per-segment configurations. The introduced component, considers the structure of the common architecture in Fig. 7-1, and takes charge of the slice management level functions to coordinate operations with given MANO and WAN SDN controllers.

7.3 Network Slice Provisioning

As previously introduced, network slicing enables the provisioning of 5G services by allocating and isolating (i.e., logically or physically) the required resources over a common infrastructure. To this end, a NSI is defined for each service instance upon request from clients/tenants, where all functionalities and resources needed to support the service are contained. Moreover, NSIs are particularly arranged and configured to fulfil specific network characteristics, compliant with required KPIs. In this way, the NSI lifecycle is completely independent to the service instance.

In particular, the NSI lifecycle begins with a preparation stage, where the network environment is arranged and the slice templates are defined. Then, after all these preparation tasks have been set up, the provisioning stage follows, being the focus of this particular chapter. At this stage, all shared or dedicated resources belonging to a specific service are created and configured. A VNF then, can be instantiated through a VIM at a particular network segment, while the network path to provide its connection to another VNF in another segment can be configured via a WAN SDN-controller. The provisioning stage ends when the NSI becomes active, considering that all required tasks have been done to enable service operation.

From the point of view of given information models related to network slicing, as described in *Section 2.4.3*, a NSI can furtherly contain or be composed of lower-

level Network Slice Subnet Instances (NSSI). In addition, a NSSI can define a set of resources and functionalities, known as Network Functions (NF), taking also into account their interconnection information. Regarding this, the partitioning of an end-to-end NSI into segment specific NSSIs, and the possibility of mapping them at the MANO controller in the form of Network Services (NS), enables the triggering of specific slice configurations at particular network segments through NSSI/NS provisioning, following the principles of the Slice Composition technique as given in *Section 2.4.5*. By using this approach, the overall orchestration process for NSI provisioning can be enhanced. Then, in order to support management operations at this level, the use of a cross-segment coordinating entity is proposed.

7.4 NFV-Coordinator Component

Following the given analysis, the orchestration of per-segment configurations can be considered a challenge for the definition of a 5G-enabling architecture. Hence, in this thesis, the proposal of a software component capable to manage 5G service provisioning and maintenance across multiple network segments, besides serving as the correlation entity between network slicing and NFV information models, is considered. In this way, the defined component, denominated NFV Coordinator (NFV-C), looks forward to serve as a middle-point between service clients/tenants requesting service provisioning via Operation Support Systems (OSS)/Business Support Systems (BSS), and the components of the 5G architecture.

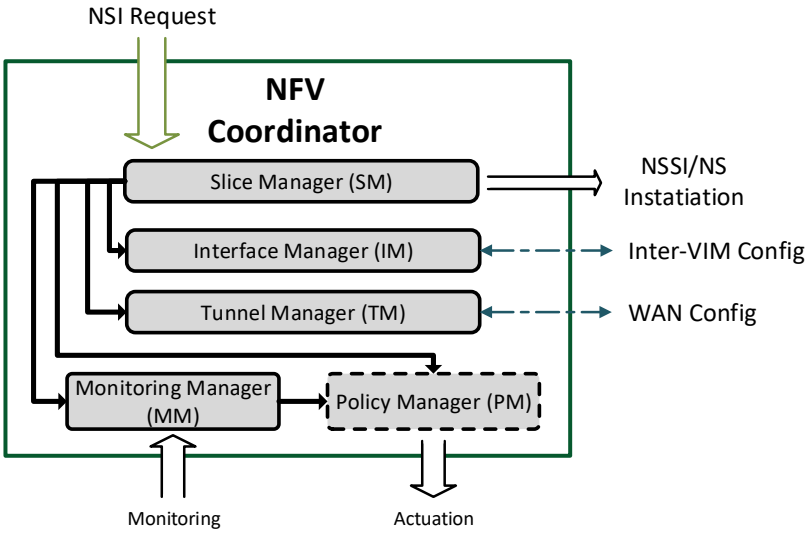


Fig. 7-2. NFV-Coordinator Architecture.

Fig. 7-2 in this respect, introduces in more detail the NFV-C architecture, depicting their internal module configuration. These modules in turn, provide the required interaction with other components (e.g., OSS/BSS, MANO controllers, WAN SDN-controllers) and the functionalities necessary to support managing the NSI and NSSI lifecycles. The description of each module follows:

- **Slice Manager (SM):** Considers the main coordination point between the client service request and the network. Handles the NSI request coming from the OSS/BSS and decomposes it into per-segment NSSIs, providing their required mapping to NSs for its instantiation at different Points of Presence (PoP) via the MANO controller. Coordinates the network connectivity set up between NSSIs through the IM and TM. Performs the initial configuration of the monitoring and actuation tasks via the MM and PM.

- **Interface Manager (IM):** Gathers VIMs interface related information. Using this data, configures routing/chaining connections between VNFs across different NSSIs. Interacts with the Inter-VIM modules at the SDN controller level.

- **Tunnel Manager (TM):** Configures the overlay tunnel across the WAN, by triggering the required network configurations to enable connectivity between NSSIs. Interacts with the WAN SDN-controller(s).

- **Monitoring Manager (MM):** Manages the instantiation and configuration of network sensors. Gathers specific parameter data related to the KPIs of each deployed service, retrieved from different layers of the architecture according to the monitoring configuration set by the SM during the NSI provisioning.

- **Policy Manager (PM):** Analyses retrieved parameter data by comparing it to the pre-defined guard thresholds set by the SM, which are based on a policy system (i.e., event-condition-action model). Triggers actuations/actions at the required network segments and through specific components to guarantee the overall QoS of the NSI.

As described, the IM and TM modules become crucial in the configuration of the NSI/NSSI network connectivity at the provisioning stage, while the MM and

PM modules are the ones responsible of guaranteeing the NSI performance at the maintenance stage. The SM in turn, considers a major role in both, managing the coordination of the complete NSI and NSSI lifecycles.

7.5 Mapping to the 3GPP / ETSI NFV Framework

After analysing the architecture and main core functionalities of the NFV-C, it is important to understand how it is mapped in respect to the existing standards. In this matter, and following the discussion given on the definition of a common meta architecture, Fig. 7-3 describes the relation of the NFV-C towards the 3GPP and ETSI NFV framework, considering their correspondent architectural components.

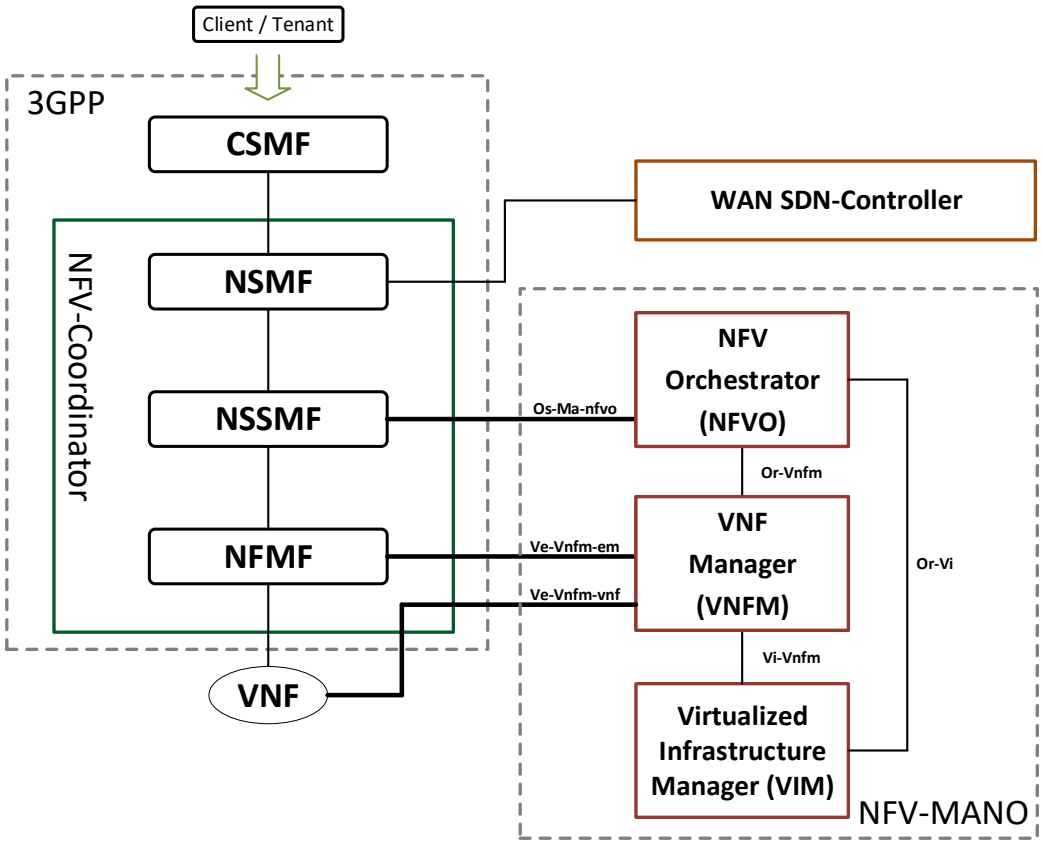


Fig. 7-3. NFV-C Mapping to the 3GPP/ETSI NFV Framework.

The figure in turn, bases on the joint study of the 3GPP and ETSI [43] that presents a first approach for the relationship of both frameworks, considering all the required communication interfaces as analysed in *Section 2.4.4*. In this way, the NFV-C can be mapped according to its provided functionalities.

Starting with the service request coming from the service client/tenant, the Communication Service Management Function (CSMF) component takes charge of analysing service requirements and of requesting the deployment of a new NSI, accordingly. The CSMF in particular, could be represented as a OSS/BSS module, taking responsibility for the Service Management level tasks, and the sending of a NSI request to the Network Slice Management Function (NSMF) component. This one in turn, manages the NSI lifecycle, by decomposing the request (i.e., NSI partitioning) into smaller segment-specific NSSIs, which are then provisioned via the Network Slice Subnet Management Function (NSSMF) component.

The NSSMF, is responsible of requesting the deployment and configuration of NSSIs via the Os-Ma-nfvo interface, which enables the connection towards the NFV Orchestrator (NFVO) following the interface specification [76]. To this end, the NSSI to NS mapping must be performed first at the NSSMF, so the NFVO is able to process NS requests and coordinate their deployment with the VFN Manager (VNFM) and with the underlying VIMs. Finally, the Network Functions Management Function (NSMF) component, would also be able to provide required configurations for VNFs, and to gather monitoring data for fault/performance management tasks.

Considering the analysed role of the NFV-C, its operational functionality can be defined at the Slice Management level of the meta architecture layout given in Fig. 7-1. Moreover, by analysing each of its internal modules, the SM would take responsibility for the NSMF and NSSMF functionalities, allowing the coordination of the NSI/NSSI provisioning and maintenance, while the IM and TM would be the ones triggering the connectivity setup through the WAN SDN-controller(s). The MM and PM in turn, would take part of the NSMF functions related to the monitoring of VNFs, in order to enable policy-based slice maintenance. In this way, the NSI/NSSI lifecycles will be managed at the NFV-C, while the NS/VNF lifecycles will be handled at the MANO controller, which considers NFVO and VNFM functionalities.

7.6 Slice Provisioning Architecture

Following the definition of the coordination entity (i.e., the NFV-C) and its mapping to the current standards, the framework proposed in this thesis is presented next,

tailored in this case for the NSI provisioning stage. The framework in general, is layered in different architectural levels, where components are placed according to their specific role in the provisioning process, taking as a base the common architecture of a 5G single-domain multi-segment scenario.

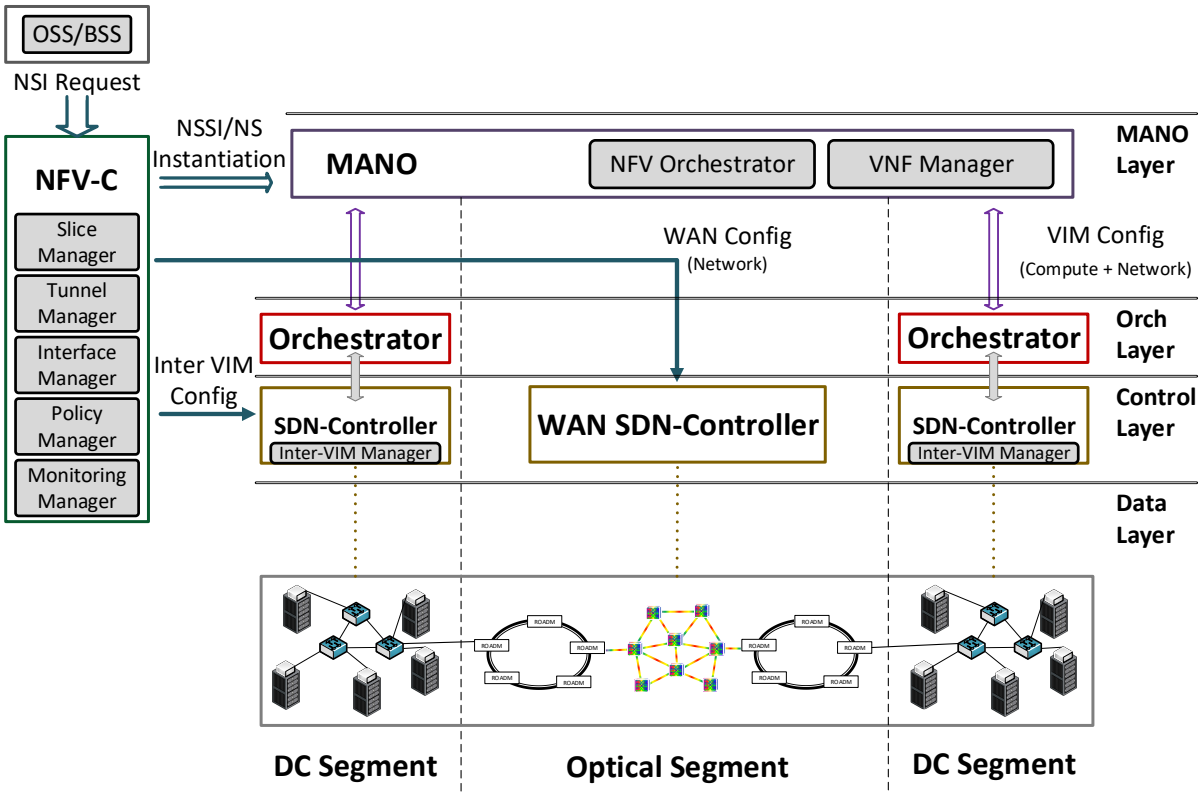


Fig. 7-4. Architecture for Multi-Segment Slice Provisioning.

The proposed architecture is shown in Fig. 7-4, where the main actions for NSI provisioning are detailed. Looking from a bottom-up perspective the layers are:

- **Data Layer:** Comprises all the computational and network resources at the physical level, distributed across network segments and considering different technologies (e.g., optical, electrical). Network resources in turn, expose their capabilities via southbound protocols, which open a communication path so configuration actions can be triggered over them, besides allowing the retrieval of real-time monitoring data. In this particular work, special focus is set on inter-DCN scenarios, which define an underlying multi-segment infrastructure.

- **Control Layer:** At the control level, reside software components capable of providing the necessary intelligence to configure network resources. The SDN-

controller in turn, contains plugins to enable southbound communication with the network resources. Moreover, it defines the APIs for exposing the network topology and all the controller related functionalities towards applications at the northbound. At last, it allocates specific software modules designed to enable network configurability, besides providing databases and data buses to allow data storage and the required inter-module communication.

- **Orchestration Layer:** The orchestrator is the component responsible for the allocation of computational resources, allowing the instantiation of network functions in the form of VNFs, VMs, containers, etc. Besides, it is in charge of requesting the corresponding segment SDN-controller to provide the required network connectivity between resources (i.e., intra-segment connectivity). This type of component is commonly present in segments allocating both network and computational resources (e.g., Data Centres).

- **MANO Layer:** A MANO entity in turn, is responsible of orchestrating NSSI level resources, in the form of NS requests. To this end, it must coordinate with the registered VIMs the instantiation of NFs, besides providing the necessary given configurations for their right operation, either at provisioning or runtime. In this manner, it becomes in charge of the NS/VNF lifecycles, in addition to providing the templates for their definition, given as Network Service Descriptors (NSD) and VNF descriptors (VNFD) [38]. In this regard, pre-defined charms or cloud-init files are kept at VNFDs, to define the required configurations to be executed on VNFs [77]. At this level, developments are pushing forward to add more functionalities, such as managing the triggering of WAN connectivity set up between NSSIs, supporting hybrid network services (i.e., VNFs and PNFs), enhancing service monitoring tasks, guaranteeing service performance by applying policy-based approaches, among others [78].

Starting with the event of a new NSI request from the OSS/BSS, the SM at the NFV-C analyses the requirements of the received request and takes charge of the NSI instantiation. In this sense, it decomposes it into segment NSSIs, according to the demanded resources, and triggers their provisioning in the form of NSs through the MANO controller. This one then, deploys each NS via the requested

VIM, so VNF instantiation and configuration can be set up, considering as well the required connectivity in this specific segments (i.e., Intra-VIM Config). As the VNFs get instantiated, the IM collects their interface data and configures the necessary routing/chaining between functions at different VIMs (i.e., Inter-VIM Config). At this same time, the TM sets up the overlay tunnel and network configurations required at the WAN. After VNFs have completed their boot up and their correspondent connectivity is provided, the end-to-end NSI functionality given by the combination of NSSIs becomes operative. The NSI provisioning stage ends with the SM passing the QoS monitoring specifics to the MM and PM, regarding the service KPI-related parameters for monitoring and the defined guard thresholds for the NSI.

7.7 Architecture Comparison

Given the presented architecture, and after having analysed its functionality and the scope of operation of its components, it becomes crucial to make a comparison with the efforts done by the community as in [69, 70, 71, 72, 73, 74, 75].

Table 7-1 Classification of community-driven approaches to the 5G architecture.

	Domain of Operation	Highest level of Operation	3GPP 5G Framework Compliant	ETSI NFV Framework Compliant	3GPP to ETSI Information Model Support
Proposed in thesis	Single-domain	Slice Management	Yes	Yes	Full mapping support
5G-Network Slice Broker	Single-domain	5G-Service Management	Yes	No	Not mentioned (slices only)
5G-Crosshaul	Multi-domain	Slice Management	No	Yes	Not mentioned
5G-ICN	Multi-domain	5G-Service Management	Yes	No	Not mentioned (slices only)
SDN/NFV MANO Architecture for Dynamic VNF Services	Multi-domain	MANO	No	Yes	Not mentioned (net. services only)
CogNet	Single-domain	MANO	No	Yes	Not mentioned (net. services only)
SELFNET	Single-domain	MANO	No	Yes	Not mentioned (net. services only)

Table 7-1 in this regard, characterizes the main characteristics of some of these approaches, with the focus set on comparing the drawbacks/advantages between them and the proposed architectural work. In turn, all these works present architectural solutions set to prove the applicability of the standards [30, 45, 68] towards the definition of the 5G architecture.

The given comparison focuses on specific features of these proposals, such as their domain(s) and highest level of operation considering the defined standards. Moreover, special attention is set on analysing their compliance with both ETSI and 3GPP frameworks and the data mapping support between them given the reference information models [40, 41]. Taking this into consideration, the architecture presented in this work, even though its range of operation is set to a single-domain on first hand, it accomplishes the coordination of operations from the Slice Management level and provides the data mapping to correlate 3GPP and ETSI NFV models. In this manner, it supports handling both NSI/NSSI and NS/VNF resources, introducing then a valuable contribution to the state of the art.

7.8 Conclusions

The definition of 5G standards towards the support of new types of services in next generation communication networks, brings a set of architectural requirements to cope with the functionalities of 5G enabling technologies, such as Network Slicing, SDN, NFV, among others. In the presented work, an analysis is presented on the required aspects to define a common architecture able to support the provisioning of 5G services over a shared single-domain multi-segment infrastructure.

In this regard, the relation of the proposal to the 5G architecture standards has been examined, taking into account the role of its components in respect to the 3GPP and ETSI NFV frameworks. In this way, its support for the correlation between reference information models has been proved. Additionally, the proposal features have been characterized to compare its novelty with community-driven approaches, where it has been recognized as a valuable addition to the state of the art in 5G deployments. The slice maintenance aspects and practical validation of the presented architecture are still to be analysed in following chapters.

7.9 Publications

The work realized throughout this thesis chapter has been disseminated in the following scientific journal and conference publications:

1. **R. Montero**, F. Agraz, A. Pagès and S. Spadaro, “Enabling Multi-segment 5G Service Provisioning and Maintenance through Network Slicing,” Journal of Network and Systems Management (2020). <https://doi.org/10.1007/s10922-019-09509-9>

2. **R. Montero**, F. Agraz, A. Pagès and S. Spadaro, "End-to-End 5G Service Deployment and Orchestration in Optical Networks with QoE Guarantees," 2018 20th International Conference on Transparent Optical Networks (ICTON), Bucharest, 2018, pp. 1-4. <https://doi.org/10.1109/ICTON.2018.8473996>

Chapter 8. 5G Service Maintenance through Network Slicing

This chapter follows up the architecture presented in the previous chapter, setting focus in this case on the NSI maintenance stage. Addressing all the aspects related to the collection of service-specific monitoring data and the corresponding actions to be executed in order to guarantee the expected service performance levels.

8.1 Network Slice Maintenance

Continuing the lifecycle of the NSI, once all resources have been allocated and configured and the overall service functionality has been enabled, it begins the slice runtime or maintenance stage. Through this stage, the NSI should be capable of guaranteeing all the requirements set for the proper operation of the service running on top. To this end, this stage entails the monitoring of specific parameters related to the service KPIs, so preventive actions/actuations (e.g., reconfigurations, NSI modifications, upgrades) can be triggered in case performance levels reach an undesired point. By these means, service operation in line with the established Service Level Agreements (SLA) can be guaranteed.

This chapter focuses on the runtime stage of the presented single-domain multi-segment architecture, analysing the monitoring operations and the policy-based actuation system at the NFV-C, that enable the maintenance of 5G services Quality of Service (QoS) and Quality of Experience (QoE) levels.

8.2 Monitoring and Sensors

Regarding the monitoring task, network sensors appear as an option to gather real-time data from network segments so this can be furtherly aggregated to the NSI level and processed. A sensor in particular, can be identified as any component capable of retrieving data related to the current state of the service, usually given by specific parameters (e.g., latency, throughput, etc.) that allow measuring the operational status of computational and network resources.

In practice, sensors can be found in many forms (e.g., VNF, VM, container, application) and at any level of the architecture (i.e., infrastructure layer, control layer, orchestration layer). Depending on the case, the sensor can be instantiated or configured at the NSI provisioning stage or during its maintenance.

This section then, analyses the gathering of specific parameters by the MM. In this sense, detail is given on how data is exposed to the NFV-C and whether its retrieval requires the triggering of specific configurations, or as in particular cases, the deployment of new components to make it available.

8.2.1 Latency Awareness

Latency has been identified in 5G as one of the most crucial requirements, considering service demands for end-to-end reachability below 1ms [48]. As a result, awareness on the real-time service latency becomes crucial for network slices operation and maintenance, especially when service KPIs/SLAs define a maximum latency value for the correct operation of the service.

In turn, to address the retrieval of latency data, the design of an innovative mechanism based on the use of a VNF sensor is presented. This approach, allows the measuring of latency on a currently deployed and running service by allocating a sensor strategically in-between functions so traffic going through the network is not affected. Further detail on the sensing method and sensor allocation follows.

8.2.1.1 Sensing Mechanism

As introduced, with the goal of measuring latency, a sensor is placed in between VNFs associated to a particular service (i.e., at the service chain). The sensor, in particular, is configured to analyse all the work traffic flowing through, by capturing the TCP packets exchanged between VNFs. The mechanism bases on the sensor ability to use the time delay of packets, given by the difference between the time value set at the time stamp of each packet and the one at its corresponding ACK (acknowledgment) packet. With this time difference, the round-trip time (RTT) between VNFs and the sensor can be calculated, and thus the latency.

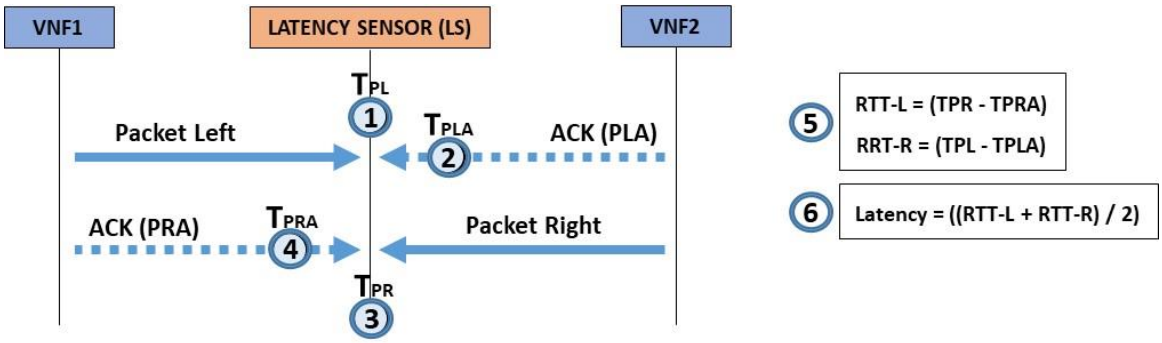


Fig. 8-1. Latency Sensing Mechanism.

Fig. 8-1 in this regard, presents the sensing functionality of the mechanism based on a number of defined steps, which are described next:

- **Step 1**: The sensor uses the time stamp value of the first arriving packet from VNF1 to get the Packet-LEFT arrival time (T_{PL}).

- **Step 2**: At the time the matching ACK packet is received from VNF2, the sensor uses its time stamp value to get the Packet-LEFT-ACK arrival time (T_{PLA}).

- **Step 3**: On the other side the same process occurs, with the sensor using the first packet coming from VNF2 to get the Packet-RIGHT arrival time (T_{PR}).

- **Step 4**: A matching ACK is also received in response from VNF1, then getting the Packet-RIGHT-ACK arrival time (T_{PRA}).

- **Step 5**: The sensor uses T_{PL} and T_{PLA} to get the RTT between the sensor and VNF2 (*RTT-R*), then it calculates *RTT-L* on the other side using T_{PR} and T_{PRA}.

- **Step 6**: At this point, the side-to-side RTT between VNF1 and VNF2 can be calculated by the sensor, to then finally get the latency between them.

By measuring the latency of packets flowing through the path set between VNFs of a specific service, the sensor is capable of providing real-time latency information of the current network state of the NSI to higher layer elements (e.g., controllers, orchestrators, MANO, NFV-C). To this end, and throughout the whole lifecycle of the slice, the sensor will continuously dump the retrieved data to a local database so it gets available to clients via an external management network.

8.2.1.2 Sensor Allocation

Following the given description on the sensing mechanism, the sensor is set to be strategically allocated between the VNFs of a particular service. To this end, the code and configurations required to support the mechanism are implemented in a VNF, which constitutes the sensor, so it can be deployed on a given NSI.

In this way, and considering that each NSI is furtherly partitioned into smaller per-segment NSSIs, the sensor would be provisioned at any of the de-composed NSSIs to make it fit in the VNF service chain. Such allocation is usually performed at the slice provisioning stage so the sensor gets deployed and interconnected to the other service related VNFs. However, there could be cases when its addition could be also done at the slice maintenance stage, considering the modification of one of the NSSIs at runtime.

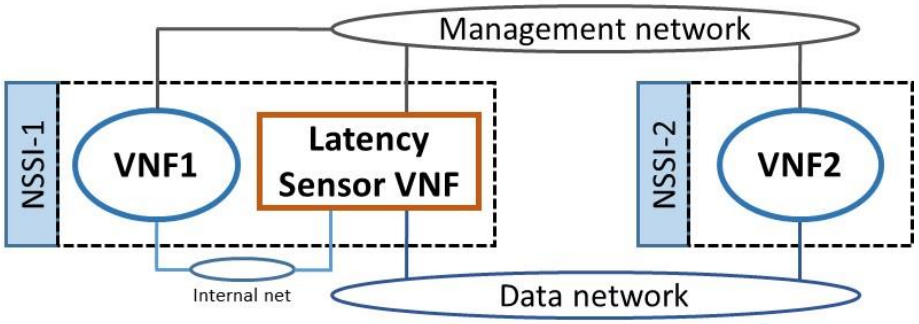


Fig. 8-2. Latency Sensor Allocation.

The latency sensor allocation is represented in Fig. 8-2. In this case, the NSI corresponding to a particular service comprehends two VNFs, to be provisioned on NSSI-1 and NSSI-2 respectively. As illustrated, the latency sensor is added to NSSI-1 so it is placed in between VNF1 and VNF2. It is important to consider that VNFs are deployed on different network segments, due to resource availability or to location constrains related to a particular service functionality. The sensor then, connects to VNF1 through an internal network and to VNF2 via the data network. In this manner, it can analyse all traffic flowing between them. Moreover, both VNFs and the sensor are also connected to a management network for external access and in case of the sensor, to enable exposing the retrieved latency data.

8.2.2 Throughput Sensing

By sensing the throughput between NFs deployed at different segments of the network, a significant value to test the current state of the service can be provided. In order to retrieve this data, the MM at the NFV-C must contact components at different layers of the architecture.

As a first approach, the collection of function metrics from the orchestrator layer can be triggered. By analysing the retrieved information, it would be possible to get the amount of bytes/packets transmitted through the interfaces of running functions (e.g., VM, VNF, container) involved in the operation of a specific service. Then, by knowing the amount of data transferred at a given period of time, the end-to-end throughput between functions can be calculated.

In case metrics collection is triggered at the controller level, the retrieved data would consist on the statistics of network devices given interfaces (e.g., switches, routers, access-points). This would also allow getting real-time statistics on the amount of data flowing through specific links or segments of the network. In this way, the hop-by-hop throughput can be measured, allowing to identify potential bottlenecks and network failures that could compromise normal service operation.

8.2.3 CPU / RAM Gathering

Gathering statistics related to the current state of deployed compute resources (e.g., VM, VNF, container) would also be relevant in order to analyse the status of the service. In turn, the component responsible of managing these resources (i.e., the orchestrator), would be the one collecting this information and exposing it to other components via particular interfaces as in [79].

To this end, the MM at the NFV-C must request VIM segment orchestrators, managing the resources of a specific service, to begin gathering and exposing the CPU/RAM metrics of such resources. The MM then, would be able to use this data to identify when potential service degradations are due to insufficient computational resources or if these could be related to network problems. In case a malfunction

is found at any deployed function, then preventive actions (e.g., VNF migration, VNF scaling, VM resizing) can be performed to guarantee service performance.

8.2.4 BER Monitoring

Setting the focus more on the network connectivity aspect, the monitoring of the Bit Error Rate (BER) on optical network resources can be considered an important variable to analyse potential service degradations related to any malfunction or traffic saturation at the WAN, or at optically enabled intra-DCNs.

The components allowing the gathering of this particular metric would be the SDN-controllers handling the control of optical resources at specific segments of the network (e.g., Metro/Access, Core, DC). Each controller then, must collect this data from devices and store it at a local database, so it becomes available to other components such as the MM. This one in turn, should be able to use this data to trigger preventive actions in case BER levels get higher than the established guard thresholds. In this sense, the rise of the BER could significate a potential physical problem at a specific network device or it may also be related to saturation due to the over-usage of physical network paths by active services. In this matter, actions executed to overcome these events are usually linked to path reconfigurations.

8.3 Actuations over Monitored Parameters

An actuation in the context of a 5G ecosystem can be considered as any action executed in response to any detected problem that could compromise service QoS/QoE, offering through this action a preventive or predictive solution so service performance is not affected. Components triggering such actions are denominated actuators, and hold the responsibility of accommodating the slice to the current state of the network. Like sensors, actuations consider either provisioning new components (i.e., VNF, sensor, application) or executing configurations at different layers of the 5G architecture. This section takes into account sensors and sensing mechanisms described previously, and analyses the policy-based approach used to handle all the retrieved information as well as defining a set of actuation cases for each type of metric. Finally, the use of higher-level metrics is also explained.

8.3.1 Policy Manager Architecture

The maintenance framework presented in this thesis, bases on a policy-based system to execute actions when required. To accomplish this, an ECA model [80] (i.e., event-condition-action) is applied, to allow defining the necessary behaviour to support monitoring and actuation tasks over a particular slice. More specifically, the model defines an event (e.g., high latency), a condition to be met (e.g., greater than 1ms) and an actuation to be executed (e.g., reconfigure service path). The PM component ant the NFV-C is this regard, is the one responsible for coordinating all these tasks, by providing a connection point between service requirements, the monitoring data passed from the MM, and the rest of the architecture components.

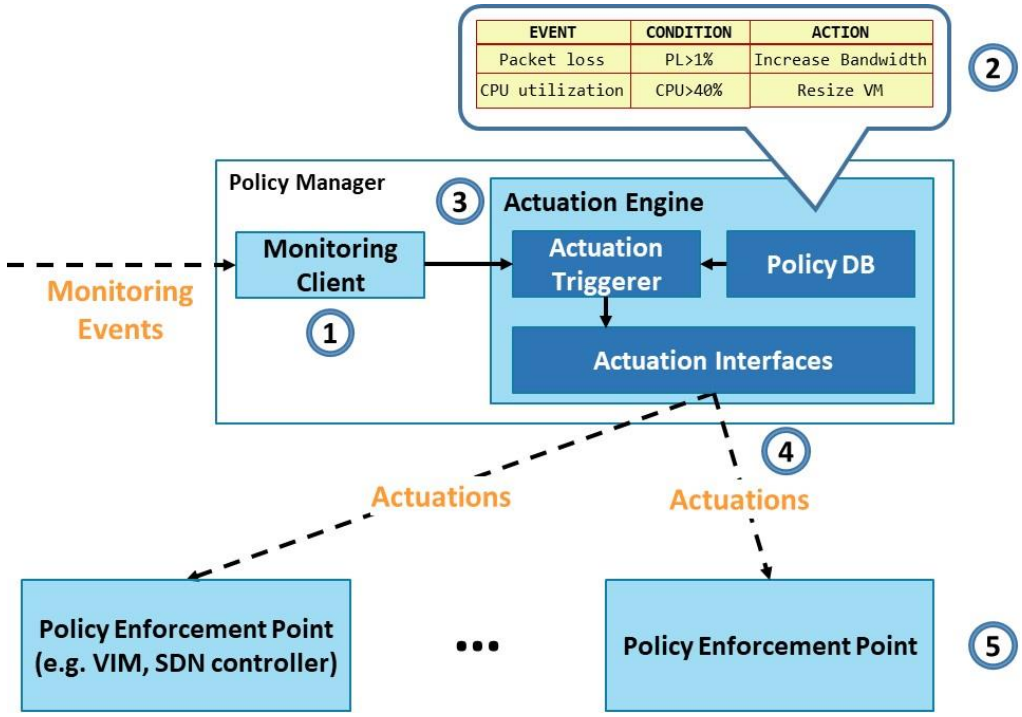


Fig. 8-3. Policy Manager Architecture.

The internal architecture of the PM is shown in Fig. 8-3, where its behaviour during the runtime stage, following the policies set at NSI provisioning, is described on five steps of operation:

- **Step 1 - Receive Monitoring events**: The Monitoring Client at the PM receives all the monitoring data passed by the MM, which considers only data relevant to the defined service KPIs, so it can analyse these specific parameters.

- **Step 2 - Available policies for actuation (ECA model)**: The policy database, initialized during the slice provisioning, exposes all the required specifications for service maintenance at each slice.

- **Step 3 - Determine policy to be applied**: The received monitoring data is then compared to the defined conditions at the DB. In case any guard threshold is violated, and following the model instructions, preventive actions are selected.

- **Step 4 - Distribute actuations across enforcement points**: The resulting actions are then triggered via specific actuation interfaces that coordinate their execution through one to many components of the architecture.

- **Step 5 - Apply desired (re)configuration**: After components have received the given instructions from the PM, required configurations are pushed towards network or compute resources to modify specific slice characteristics, in a way to guarantee its correct operation.

As an important consideration, policies set for slice maintenance would also be able to change during runtime, due generally to requests coming from service clients or to identified changes on the network performance. Moreover, besides applying the policy-based approach, the given actuations could also be defined by other data analysis techniques such as Machine Learning.

8.3.2 Actuation Cases over Latency Awareness

In the case of high latency events, surpassing a given condition/threshold, two particular actuation cases are analysed in this thesis, where each of them requires a different set of tasks to be executed as described next:

- **Path Reconfiguration**: In case of identifying a network failure or congestion affecting the latency of a specific service, the PM would check the MM database to analyse the latency levels of all alternative data paths. If one with a lower latency is found, or operative in case of a network failure, the PM would trigger the path reconfiguration task via the TM at the NFV-C. The TM then, would be the one responsible of contacting the SDN controller(s) of particular network

segments to send the required instructions. After the configurations are pushed to the corresponding network devices, the new data path would become active and the monitoring process would start again to check whether the measured latency has gone back to safe performance levels.

- **Slice Reallocation**: Another case, relates to identifying a network congestion event, that could take place due to the high amount of traffic flowing to multiple VNFs at a particular section of the network. In this situation, the PM would check via the SM if there are available locations for considering the migration of VNFs, so the current traffic bottleneck is lowered. After an affirmative check, the PM would request the MM for the best data paths in terms of latency to reconfigure connections towards migrated VNFs. Then, once the PM has all the requisites, it can trigger Slice Reallocation by sending a request to the SM, TM and IM, demanding for VFN migration, path reconfiguration and routing reconfiguration respectively. In this way, the SM would send slice modification instructions to the MANO controller which will take charge of pushing them via the involved VIM segment orchestrators. The TM then, will instruct the WAN-controller(s) to configure the new data path. Additionally, the IM would push the new routing configuration to the VIM SDN-controllers to complete the migration process. At this point, slice reallocation will be achieved and the traffic congestion event is expected to be mitigated. To validate this, the MM would start gathering again the slice latency data and continue checking policies compliance at the PM.

8.3.3 Actuation Cases over Throughput Sensing

Taking into account the measurement of throughput between given functions, the rise of packet losses would trigger response actions to overcome such situation. The actuations defined in this thesis are the following:

- **Throughput Shaping**: If the given Packet Loss Ratio (PLR) on one active data path surpasses the established guard limits, the PM would coordinate with the involved SDN-controller(s) the possibility of enhancing the network bandwidth at such specific connection. Then, and depending on how the bandwidth limits were configured, the controllers would reconfigure either network devices or

VNF interfaces to increment the assigned bandwidth for the connection. In this way, the PLR would continue to be monitored, expecting the sensing of lower packet loss values thanks to the increased network bandwidth.

- **Path Reconfiguration**: The path reconfiguration would also be considered as a response action. In this sense, if PLR levels turn high, the configuration of an alternative path with higher throughput or less congestion can be triggered, following the same steps as in the formerly analysed latency case.

8.3.4 Actuation Cases over CPU/RAM Gathering

As for the gathering of CPU/RAM consumption levels at deployed functions, the analysis of a specific actuation is considered in case they reach an undesired point, the given response action analysis follows:

- **VM Resizing**: Considering the event of a rise in CPU or RAM consumption at a particular VM-based function, the PM would first identify it, and then request the correspondent orchestrator to re-provision (i.e., resize) the affected VM to enhance its assigned computational resources. This would mean augmenting the allocated RAM memory and/or the number of virtual CPUs for the VM, so the normal operation of the function running over it is guaranteed. The ongoing monitoring of these metrics will allow then to confirm if the enhancements done at the VM have properly lowered the percentages of CPU/RAM utilization.

8.3.5 QoE Guaranteeing through Actuation-Decision Table

The QoE of a particular service represents the use of high level metrics to measure the expected service performance closer to the client perspective. In this regard, an Actuation-Decision table, considers the addition of specific intelligence to the PM to process collected metrics from several network segments so a higher-level response action decision can be triggered.

Following this approach, at slice provisioning, a set of parameters related to agreed service KPIs will be set for monitoring by the MM, besides the definition of their correspondent guard thresholds at the PM. Then, the PM would be configured

to address particular circumstances by considering the joint analysis of all gathered metric levels and considering the section of the network from where they are exposed. Enabling this, would allow the PM to act not only in response to a particular threshold violation event, but be able to identify the exact location of the problem and the available single or group actions that could be performed so any required modification to the slice is more accurately applied.

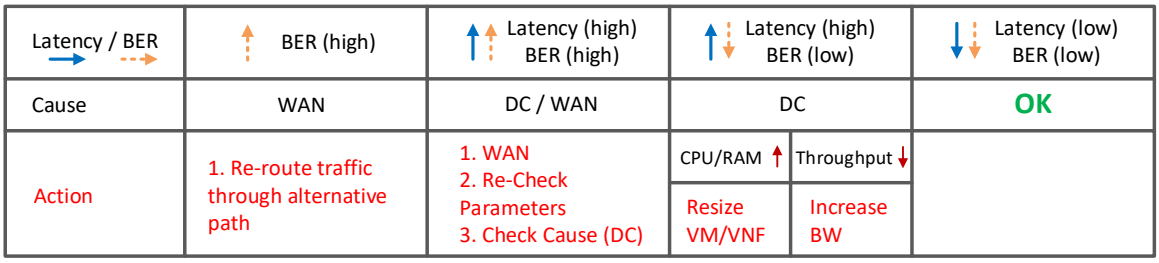


Fig. 8-4. Actuation-Decision Table Representation.

Fig. 8-4 in this regard, depicts an example of an actuation-decision table that represents the configured intelligence at the PM. In this case, the monitored parameters at a given slice include the gathering of E2E latency, BER, throughput and CPU/RAM. The slice in turn, is composed of two allocated VNFs, VNF1 at DC1 and VFN2 at DC2, considering resource provisioning and configurations across all the required network segments. In this sense, the example case entails identifying first, if a possible degradation of the slice performance is related to problems at the optical WAN or at any DC segment. To this end, latency and BER gathered metrics are used as a reference, so the PM can begin isolating the problem.

Using this approach, four cases can be considered as shown in the figure, depending on whether latency and BER values expose high or low levels (i.e., high-level linked to guard threshold violation). As for the case with both parameters on low levels, no action should be executed considering the overall slice state of being on good quality standards. The rest of the cases are analysed next:

- **BER (high)**: In the event BER levels rise, reaching a compromised state for service performance, the PM would identify the segment related to the root cause of the problem by checking the origin of the analysed metrics, in this case the WAN. The response action would then be set for path reconfiguration at the

WAN segment, rerouting traffic for example through a less congested path. The PM would trigger this action in coordination with the WAN SDN-controller.

- **Latency (high)**: If the PM advertises high latency levels, then it will identify the DC segments as the ones involved in the potential service degradation cause. As the latency is given E2E by the deployed sensors, the PM must perform one extra step. In this way, it will further analyse the CPU/RAM consumption of the running VNFs and the throughput passing on their interfaces, so it can detect the specific root cause. As a result, if any VNF presents a high value on CPU or RAM consumption, the PM would trigger its resizing to enhance the allocated computational resources. On the other hand, if the measured throughput is low, the PM would trigger the increase of network bandwidth at DC segments for the required data path (i.e., throughput shaping).

- **Latency & BER (high)**: Another possibility is that both BER and latency levels present high values. In this event, the PM would consider first acting over the WAN, by triggering path reconfiguration. After this action is executed, it must re-check both metrics to see if they have lowered back to safe levels. Then, in the case these are correct, the process will finish meaning that the root cause was found at the WAN. In the case latency metrics continue to show high levels then the PM should also be required to act over the DC segments, following the DC case through an analysis of VNFS CPU/RAM consumption and throughput.

As introduced before, the presented actuation-decision table represents the configured policy-based engine at the PM, to address maintenance tasks specific to each slice. In this sense, and with the purpose of guaranteeing the expected service QoE, any given slice would require a different analysis of its high-level metrics considering both the defined service KPIs and the available metrics/actions at the network segments where its resources are deployed.

8.4 Slice Maintenance Architecture

Based on the definition of the framework presented for slice provisioning in *Section 7.6*, an approach tailored for the slice maintenance stage can be defined, focusing on the PM/MM driven tasks required to guarantee the normal operation of the slice.

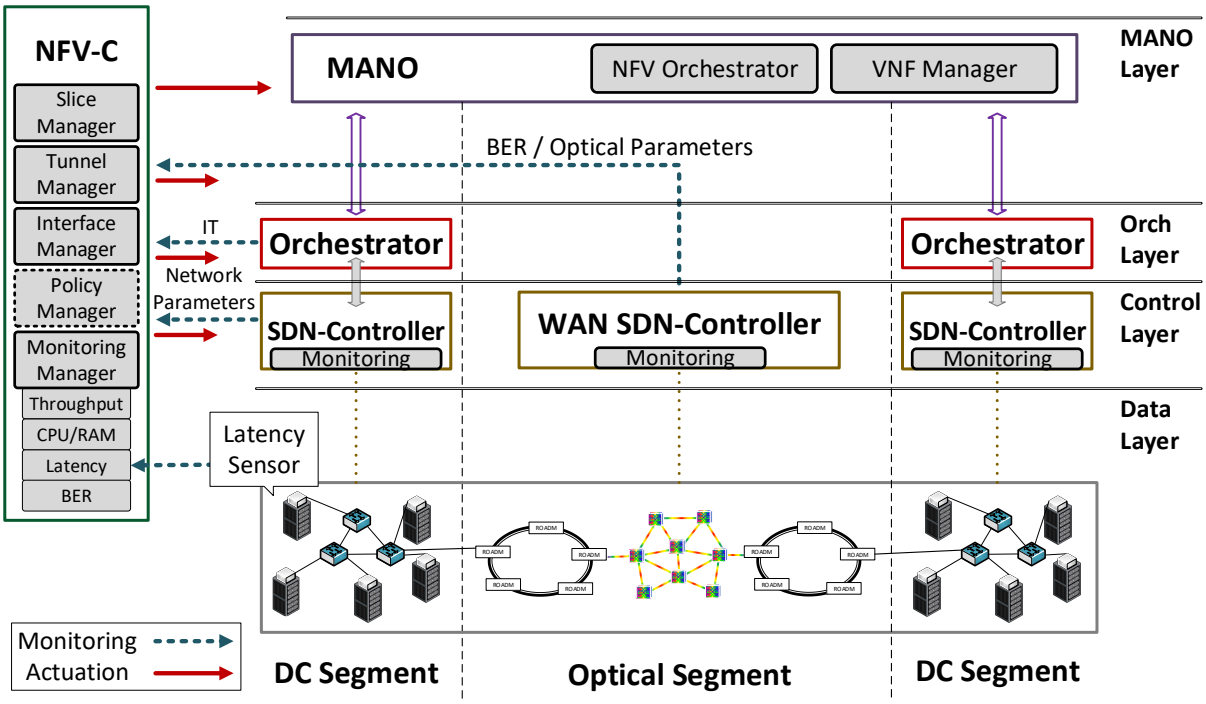


Fig. 8-5. Architecture for Multi-Segment Slice Maintenance.

The framework for slice maintenance is presented in Fig. 8-5, where the role of MM and PM interactions across architectural layers is illustrated. The MM in this sense, would enable the gathering of the previously analysed parameters from different components at different levels, while the PM would also have contact with many elements to set up the required configurations as policy-based actions are triggered. In addition to the coordination done from the NFV-C, other components are also involved in the maintenance process considering other specific tasks.

In order to give a better description of the maintenance architecture, sensing and actuations tasks have been classified taking into account the layer associated to their operation, as detailed next:

- **Data Layer:** Sensing tasks at this level require the use of a deployed function (i.e., sensor) that would collect and expose the required information from the data layer. The designed latency sensor in turn, presents a valid way to gather metrics data of a running service straight from the NFV-C without the need of contacting other components. In case of actuations, the reconfiguration of a sensor could also be done directly, but for a better practice, other components should be taking charge of these actions (e.g., orchestrator, controller).

- **Control Layer:** At this level, the SDN controller represents the main sensing point for network resources metrics, allowing to retrieve statistics data from both optical and electrical domains. The collected information, would be stored in local databases at the controllers to be then exposed via REST interfaces. As these components have a broad control of the network infrastructure, they will also provide the necessary tools for reconfiguring the network in case any given actuation is triggered for slice maintenance. In this case, the PM would contact the correspondent controller after identifying potential problems at a specific network segment (e.g., DCN, WAN), so they can push required configurations.

- **Orchestration Layer:** On the other hand, orchestration components gather and expose metrics from the deployed functions (e.g., VNF, VM, container). CPU/RAM consumption in this way, is an example of a metric that could be retrieved by the MM through the orchestrator. Then, the PM would take charge of analysing this data to identify potential performance problems due to the lack of allocated compute resources. Actions at this particular level could represent the resizing or scaling of a VM, or even the live migration of a VNF.

- **MANO Layer:** At the MANO level, the collection of metrics is not very much expected. In any case, mostly data related to the provisioning of NSSIs is stored at the MANO component, which may not be relevant during slice maintenance. However, considering actuations, operations triggered through this component would enable performing high-level reconfigurations of a particular slice. In this way, the NSSI composition can be modified, by adding VNFs or changing their chaining order as an example. Besides, modifications to the entire NSI could also be achieved, by instantiating new NSSIs or eliminating them to rearrange the distribution of resources over the underling infrastructure, in some cases even considering altering the slice performance as required by the service.

During the maintenance stage, and considering the given per-layer analysis, MM and PM modules would keep a constant interaction with other components of the architecture. Such interactions, depending on the layer of operation and the component characteristics, would use specific communication channels such as REST interfaces, RPCs or direct access through management networks to collect

metrics and trigger actuations when required. At the end of the slice lifecycle then, and after runtime stage finishes, the decommissioning stage begins. It basically entails the deactivation of the NSI, where management components must reclaim all the resources allocated for the slice, hence terminating network functions and eliminating the configured network data paths. Moreover, dependencies in shared resources must be also reconfigured or removed to guarantee the correct operation of other slices/services running over the same infrastructure [45].

8.5 Conclusions

Following the proposal for a 5G architecture in *Chapter 7*, this chapter presented an analysis on the sensing and actuation tasks related to the maintenance of slices during runtime, considering the collection of monitoring parameters (i.e., metrics) and the execution of response actions (i.e., actuations). To this end, the use of a policy-based approach has been described along with the possibilities of using gathered data to accomplish not only QoS guaranteeing, but to evaluate the perceived service QoE by processing this information at a higher level.

Then, a version of the framework tailored for the maintenance stage has been also analysed, showing the required interactions across architectural layers to maintain service performance by adapting the slice to the current state of the network, until the time of its decommission. In this regard, next chapter introduces the experimental validation of some of the analysed actuation cases to prove the correct operation of the NFV-C and the rest of the components at this stage.

8.6 Publications

The work in the scope of this thesis chapter has generated the following scientific journal and conference publications:

1. **R. Montero**, F. Agraz, A. Pagès and S. Spadaro, “Enabling Multi-segment 5G Service Provisioning and Maintenance through Network Slicing,” Journal of Network and Systems Management (2020). <https://doi.org/10.1007/s10922-019-09509-9>

2. **R. Montero**, F. Agraz, A. Pagès and S. Spadaro, “Real-time Maintenance of Latency-sensitive 5G Services through Network Slicing,” Photonic Network Communications Journal [**Under Second Review** - June 2020].

3. **R. Montero**, F. Agraz, A. Pagès and S. Spadaro, "Actuation Framework for 5G-Enabled Network Slices with QoE/QoS Guarantees," 2019 21st International Conference on Transparent Optical Networks (ICTON), France, 2019, pp. 1-4. <https://doi.org/10.1109/ICTON.2019.8840548>

4. **R. Montero**, F. Agraz, A. Pagès and S. Spadaro (2020), "End-to-end Network Slicing in Support of Latency-Sensitive 5G Services," In: Tzanakaki A. et al. (eds) Optical Network Design and Modeling (ONDM 2019). Lecture Notes in Computer Science, vol 11616. Springer, Cham. <https://doi.org/10.1007/978-3-030-38085-4_5>

5. **R. Montero**, A. Pagès, F. Agraz, and S. Spadaro, “Supporting QoE/QoS-aware end-to-end network slicing in future 5G-enabled optical networks", Proc. SPIE OPTO vol. 10946, Metro and Data Center Optical Networks and Short-Reach Links II, 109460F, San Francisco, California, United States, February 2019. <https://doi.org/10.1117/12.2508579>

6. **R. Montero**, F. Agraz, A. Pagès and S. Spadaro, "End-to-End 5G Service Deployment and Orchestration in Optical Networks with QoE Guarantees," 2018 20th International Conference on Transparent Optical Networks (ICTON), Bucharest, 2018, pp. 1-4. <https://doi.org/10.1109/ICTON.2018.8473996>

Chapter 9. Experimental Testing

The last chapter of this thesis part describes the use of an experimental multi-segment testbed to prove the concepts presented in *Chapter 7* and *Chapter 8*, in a way to validate the 5G architecture for slice provisioning and maintenance.

9.1 Multi-Segment Experimental Testbed

The setup used for the experimental validation of this work is depicted in Fig. 9-1.

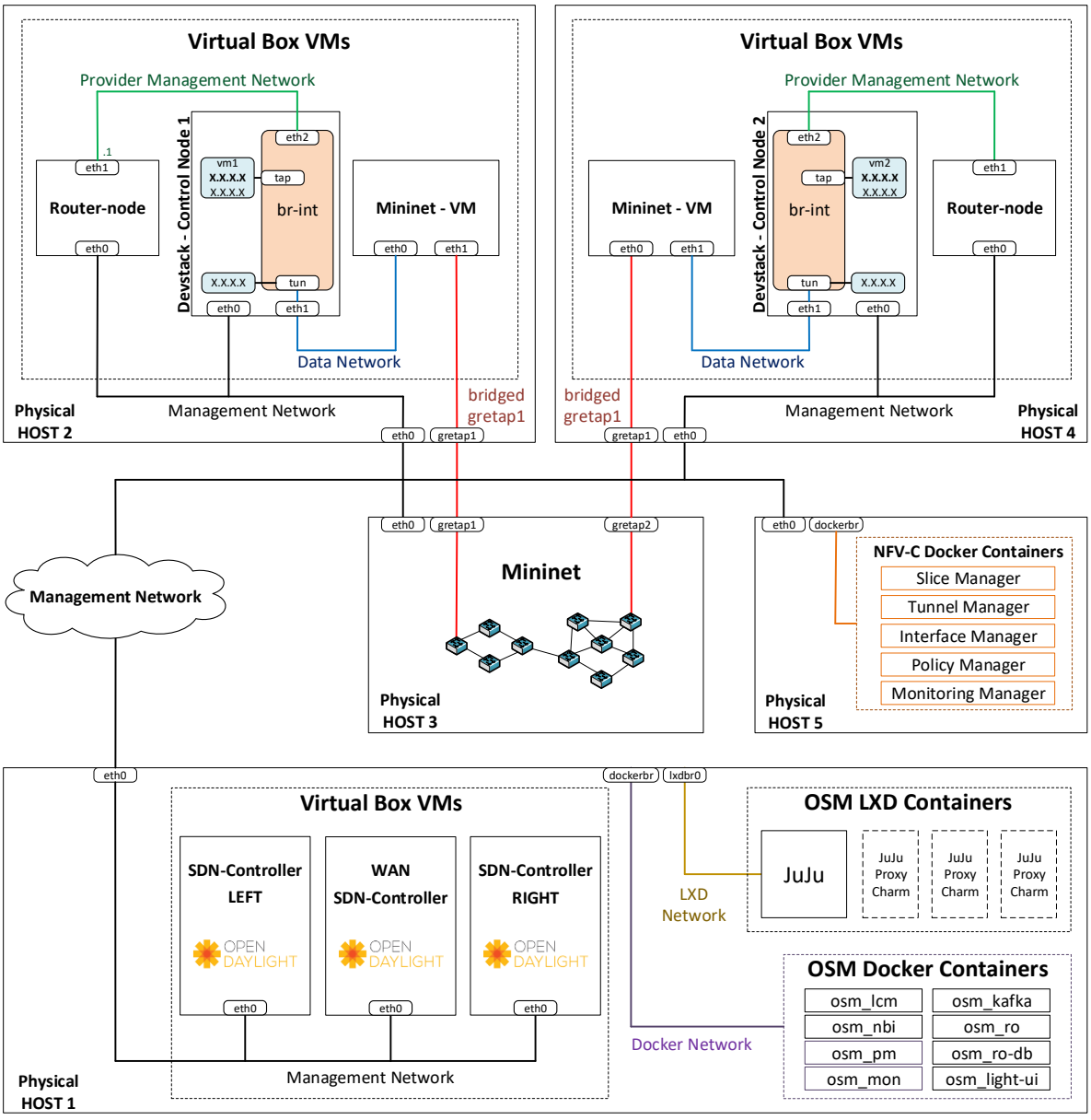


Fig. 9-1. Experimental testbed for 5G service provisioning and maintenance.

The given figure, presents an experimental testbed based on the principles investigated for a single-domain multi-segment 5G architecture. In this manner, the emulated scenario allows deploying service slices, along with the required sensors or sensing configurations. Besides, it also enables testing the response actions in case service performance is not compliant with defined SLAs/KPIs. The elements conforming the testbed are then distributed across 5 physical servers.

As shown in high detail, the different servers comprise the required software components for supporting slice provisioning and maintenance. In particular, the scenario represents two DC segments interconnected through an emulated WAN network, which are then managed by higher level entities such as VIM-Controllers, VIM-Orchestrators, the MANO component and the NFV-C. Server 1 then, allocates an instance of Open Source MANO (OSM) [31] and a set of Opendaylight (ODL) SND-Controllers [55], one for each emulated section of the network. Server 2 and 4 in turn, contain three components each, an instance of Mininet [81] to emulate the local DCN, an instance of OpenStack orchestrator [16] and a router-node VM to provide external access to VNFs. The WAN segments of the network (e.g., Metro/Access, Core) are then emulated with Mininet in Server 3, to serve as the connection point between Servers 2 and 4. At last, Server 5 allocates the NFV-C modules in the form of Docker containers. In respect of network connectivity, a Management Network is fixed across the testbed for inter-server communication, while internal networks are also configured to establish intra-server connections.

The next sections describe the steps taken to validate the correct operation of the proposed 5G architecture by means of experimental testing on the previously analysed testbed scenario, considering slice provisioning and maintenance tasks.

9.2 5G Service Provisioning

To begin, tests required to validate 5G service provisioning through Network Slicing were performed, considering the provisioning of an end-to-end slice (NSI-1), whose resources are allocated across the multiple segments of the emulated scenario. In this way, the slice is partitioned in NSSI-1 and NSSI-2 sub-slices at the NFV-C to manage the de-composed slice deployment through OSM as depicted in Fig. 9-2.

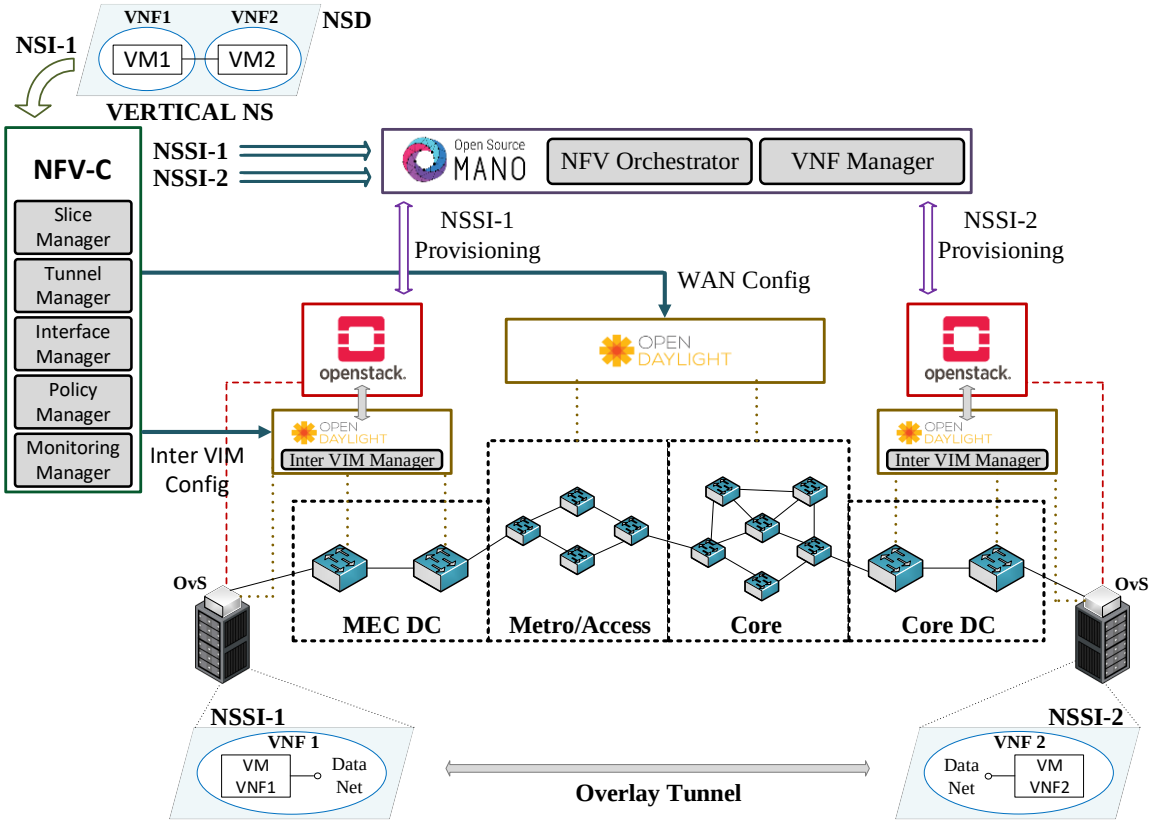


Fig. 9-2. 5G Service Provisioning through the NFV-C.

Following the figure exposed, the defined NSSIs are composed of the VNFs set to enable the required service operation, with VNF1 on NSSI-1 and VNF2 on NSSI-2 respectively. The deployment of NSSIs, at this point mapped to NSs, is then triggered by OSM, so OpenStack orchestrators in charge of managing computational resources at Multi-access Edge Computing (MEC) DC and Core DC segments can allocate/instantiate the corresponding VMs for each VNF. Besides, orchestrators will also request the SDN-controllers at these segments to configure the local Mininet-based DCN and the Open Virtual Switch (OVS), in order to enable intra-DCN connectivity. In turn, to enable the communication across the WAN, the NFV-C will set up an overlay tunnel across emulated WAN network segments (i.e., Metro/Access, Core), and request the configuration of the network devices at the Mininet-based WAN through the WAN SDN-controller. Then, it will also send specific configurations to the DC segment controllers, via the Inter-VIM Manager modules, to establish the routing connection of VNF1 to VNF2 across the overlay tunnel and through the emulated data network. Finally, and after all configurations are pushed, the composed end-to-end NSI-1 will become operative.

9.2.1 Slice Composition Validation

The use of the Slice Composition technique in this case, is validated in Fig. 9-3, where the graphical interfaces of both OSM and OpenStack are depicted. In turn, in (a), the OSM dashboard shows the provisioned NSSI-1 and NSS-2 over the two registered VIMs (i.e., openstack-left, openstack-right), considering the request by OSM for the provisioning of compute and network resources at DC segments via OpenStack and Opendaylight components. On (b), the instantiated VMs belonging to VNF1 and VNF2 are shown in the OpenStack dashboard. As it can be seen, each VM is connected to the Data Network (i.e., data) for internal connectivity as well as to the Management Network (i.e., provider) for external access.

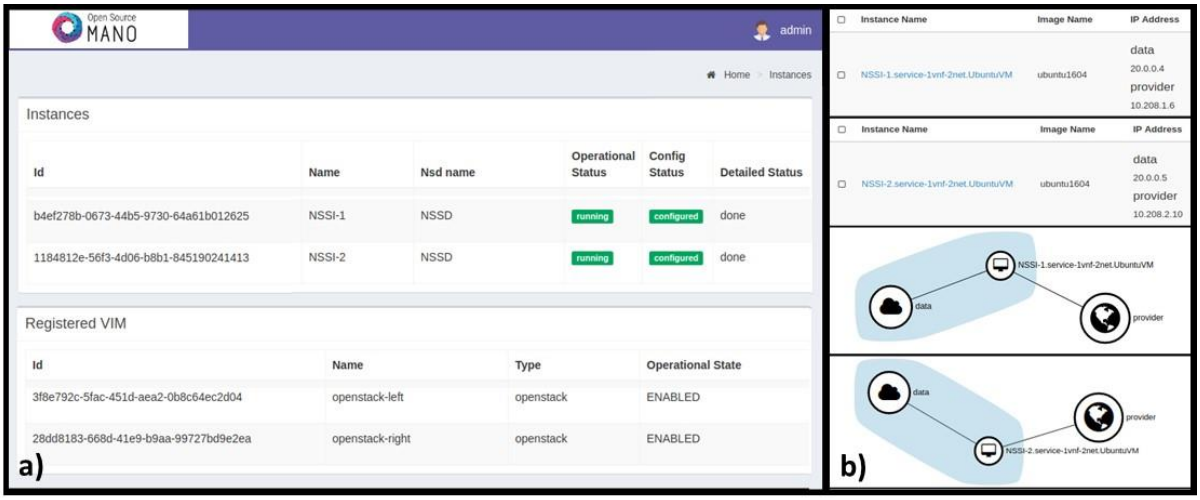


Fig. 9-3. View on OSM (a) and OpenStack (b) dashboards with deployed NSSIs.

The analysed NSSI deployments prove the correct operation of the NFV-C internal components, based on the use of Java-based modules and scripts running on software containers, considering particularly the SM operations related to the provisioning of NSI-1 by its further de-composition into NSSI-1 and NSSI-2.

9.2.2 Latency Sensor Validation

Taking into account the previously analysed case, the latency sensor can be added to NSI-1 either at slice provisioning (i.e., by slice composition) or during runtime (i.e., by slice-modification). In this regard, Fig. 9-4 next, describes how the sensor is placed between VNF1 and VNF2 at the time NSI-1 is provisioned.

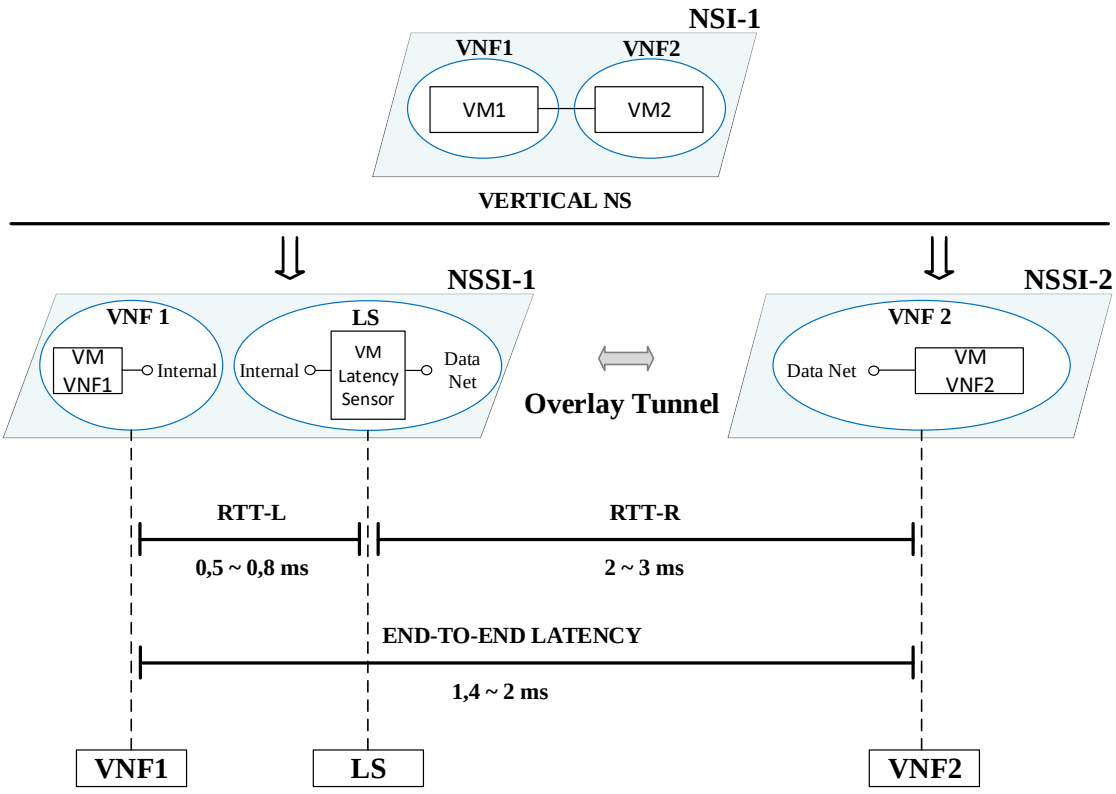


Fig. 9-4. End-to-end latency sensing in multi-segment de-composed NSI.

Thanks to its strategical allocation, the sensor is able to use the sensing mechanism to analyse the work traffic between VNFs and calculate the real-time latency on this data path. As depicted, the NSSI-1 now contains VNF1 and the Latency Sensor (LS) connected together via an Internal Network, while the latter is also connected to VNF2 at NSSI-2 through the Data Network.

After the sensor gets deployed, it will start getting the time stamps of the analysed messages, and will be able calculate the round-trip-times (RTT) at both sides. From the measurements achieved at the testbed, the RTT-L between VNF1 and the LS shows times around 0,5 to 0,8 ms, taking in consideration that both VMs are allocated over the same OpenStack instance. On the other side, RTT-R given between the LS and VNF2, shows higher times around 2 to 3 ms, which in turn relates to the higher existent delay in packet transmission over the set overlay tunnel connection between OpenStack instances (i.e., through the Data Network). By using this results then, the LS calculated an end-to-end latency between VNF1 and VNF2 around 1,4 to 2 ms. In this matter, it is important to consider the fact that results could vary between testing over an emulated testbed as in this case, and

analysing latency on a more real-case scenario. However, and for the scope of this work, the obtained results serve to prove the correct functionality of the LS.

As for the NFV-C related operations, the MM is the component in charge of retrieving the calculated latency data from deployed sensors via the management network. At the time this information gets to the MM, it is sent first to the PM so it can be processed, thus checking compliance with existent policies. Next, it is used to calculate the average latency on each monitored data path, where a sensor has been deployed, so it can be stored at the MM local database. The objective of this task is to maintain the awareness on the current state of configured data paths, in terms of latency, so this data can be used in case any slice modification is required.

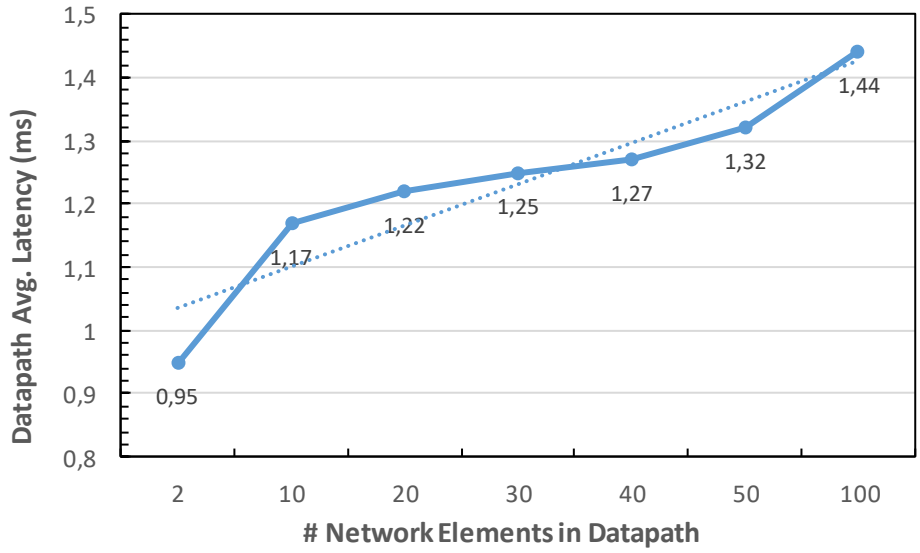


Fig. 9-5. Datapath Avg. Latency vs Number of Network Elements in Datapath.

The graph presented in Fig. 9-5 provides an example of the stored latency information at the MM. In this case, a set of currently monitored data paths with a different number of network elements for inter-VNF interconnection is considered. In turn, in the event that PM identifies the need to modify the configuration of a connection to guarantee the agreed latency levels, it will check the MM database to get the average latency on all monitored data paths. This data is represented in the figure, and will be used by the PM to analyse and decide on the new data path to be configured. As a result, actions will be triggered considering this data, so the best way to optimize the overall service latency is selected and considering the required configurations. These particular cases are analysed in next section.

At last, it is also important to consider that this data should be continuously retrieved and updated by the MM, so real-time latency information of the physical network can be maintained. To this end, latency sensors are required across all running services/slices, in order to allow the monitoring on all active connections.

9.3 5G Service Maintenance

Focusing on the maintenance stage, and considering the operations required to guarantee the slices QoS in case of compromised service performance, a set of actuation cases have been tested over the experimental testbed. Being analysed in *Section 8.3*, these cases follow the definition of an ECA model, so the retrieved monitoring information can be compared to the established thresholds, to then act upon the event these are reached or surpassed.

Depending on the use case and the defined model, specific parameters may be monitored either directly from resources or via management components across the scenario. Response actions in this matter, would also consider different levels of actuation (e.g., NSI level, NSSI/NS level, VM/VNF level), according to the scope of the identified affected resources. A view on the tested actuation cases follows.

9.3.1 VM Resizing

The first tested use case relates to the enhancement of VM compute resources after identifying high CPU or RAM consumption that could affect directly the normal service operation. The defined ECA model in this given case is described next:

- **EVENT (high CPU/RAM consumption)**: Following the slice provisioning tests, VM1 and VM2 are deployed at MEC DC and Core DC segments. The MM then, triggers the collection of CPU/RAM consumption metrics via the correspondent orchestrators. The PM in turn, analyses the retrieved data to identify potential high levels on these specific parameters.

- **CONDITION (CPU>40%, RAM>80%)**: The guard thresholds defined at the model during slice provisioning, establish that a normal CPU consumption level should be below 40 percent, while in case of RAM it should be under 80 percent.

Any identified measure above these specified levels would result in preventive actions to be triggered over the VMs.

- **ACTION (Resize VM)**: Actions in this use case require the PM of sending a request to the orchestrator managing the affected VM, so it can re-provision (i.e., resize) it with added computational resources. This means, increasing the allocated RAM and/or number of virtual CPUs for the given VM.

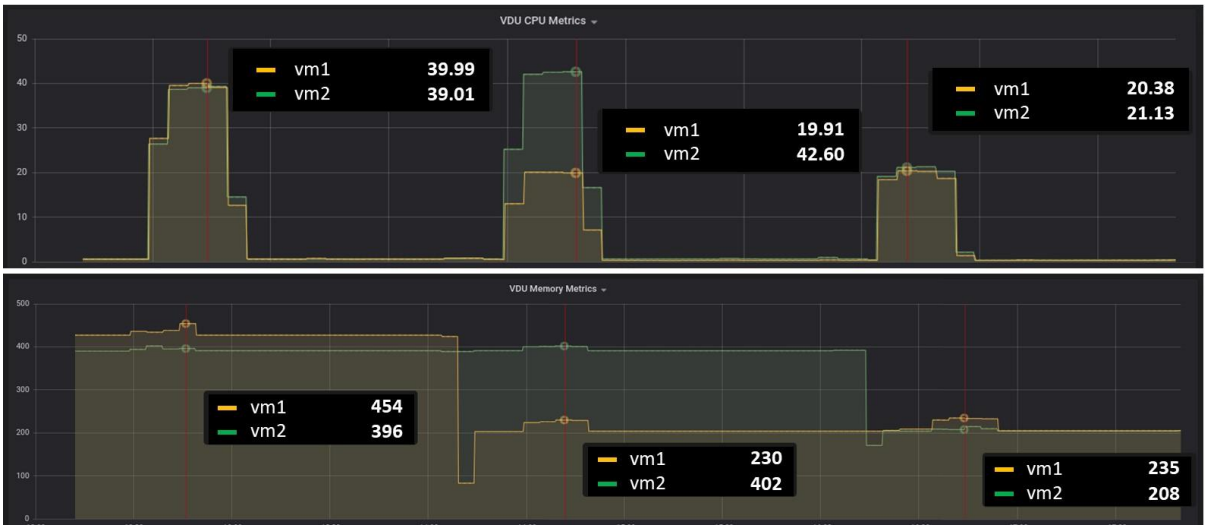


Fig. 9-6. CPU (top) average usage [%] and RAM (bottom) consumption [MB] on VM1/VM2 as illustrated in Grafana dashboard.

Fig. 9-6 then, illustrates the results obtained for this actuation case by using the Grafana monitoring and data visualization platform [82]. The first graph (top) shows the monitored CPU consumption percentage of running Virtual Data Units (VDU), VM1 and VM2 in this case, while the second (bottom) depicts their retrieved RAM consumption. To analyse the impact of allocated computational resources on these levels, the analysed graphs consider 3 different states for the VMs:

- 1st State (left):

  - VM1&2 @ 1vcpu, 512 MB RAM, 5GB disk.

- 2nd State (middle):

  - VM1 @ 2vcpu, 512 MB RAM, 5GB disk.
  - VM2 @ 1vcpu, 512 MB RAM, 5GB disk.

- 3rd State (right):

  - VM1 @ 2vcpu, 512 MB RAM, 5GB disk.
  - VM2 @ 2vcpu, 512 MB RAM, 5GB disk.

On the first state, both VMS are running and work traffic is being emulated between them by means of the iPerf network tool [83]. In this way, it is possible to examine how CPU consumption reaches around 40 percent while RAM goes up to 80 percent, thus meeting the established thresholds at the ECA model. Then, in order to check if an augmentation of computational resources would lower the VMs CPU/RAM levels, VM1 resizing is triggered, to re-provision it with an added virtual CPU. The results obtained with this modification are exposed in the second state, where once the work traffic starts flowing again between VMs, it can be seen how the CPU usage of VM1 lowers to 20 percent while RAM goes down to 40 percent. The third and last state, describes results after performing the same modification to VM2, showing similar CPU/RAM consumption levels for both VMs.

Thanks to the performed tests, the execution of the VM resizing actuation case has proved to lower the CPU/RAM consumption levels of a VM back to safe standards (i.e., below the established guard threshold). Other cases in turn, could involve testing the increase of RAM or disk memory on the VM. As a final concern, the impact of the VM resizing time on the service downtime has not been analysed on the presented tests, as there already exist studies on this subject [84].

9.3.2 Throughput Shaping

The tests considered for the second use case, present an actuation based on the monitoring of throughput between VM1 and VM2, taking into account the Packet Loss Ratio (PLR) for the calculations. The model defined is the following:

- **EVENT (high PLR)**: As in the previous case, VM1 and VM2 are deployed and work traffic is emulated between them. The data regarding the VMs rate of transferred/received bytes and packets is then collected from the MM through the VIM-Orchestrators. Using this data, the PM is able to calculate the PLR in the connection to check for losses due to channel bandwidth saturation.

- **CONDITION (PLR>1%)**: A guard threshold for PLR is set to 1 percent, so the normal service operation can be guaranteed. In the case where this condition is met, the PM would trigger preventive actions to maintain PLR on safe levels.

- **ACTION (increase bandwidth)**: The response actions set to overcome high PLR values, consider the PM requesting corresponding SDN-controllers for an increase of bandwidth at the data path set for inter-VM communication. The controllers in turn, would reconfigure the network devise to raise the assigned bandwidth for this connection or may modify the interfaces configuration of both VMs in case bandwidth is restricted at this level.

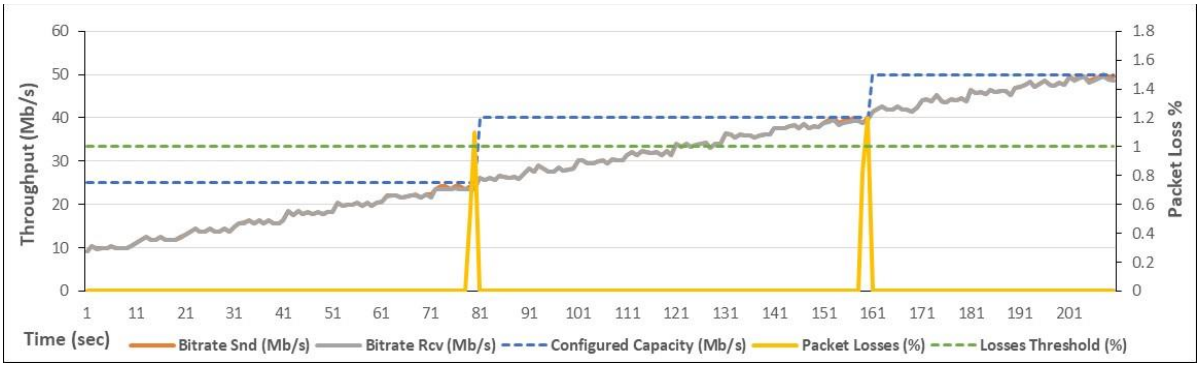


Fig. 9-7. Throughput [Mb/s] versus PLR [%] with policy-based actuations.

The graph presented in Fig. 9-7 shows tests performed for this specific use case, where bandwidth limitation was set at the VMs interfaces to control the inbound traffic. The communication channel then, was saturated using the iPerf tool, in a way to get packet losses from the transmitted data and reach the PLR threshold of 1 percent. As illustrated, tests considered two threshold violation events, where response actions were executed to lower the PLR and prove the defined model:

-1st Threshold Violation (left):

- At the initial state, the measured throughput rises gradually from 10 Mb/s as the VMs interfaces are configured with a limit of 25 Mb/s for allowed inbound traffic. After the data transfer rate reaches this limit, the PLR begins to rise. At the time it reaches 1 percent, the PM triggers the reconfiguration of the VMs interfaces, establishing a new traffic limit of 40 Mb/s and consequently lowering the PLR back to 0 percent.

-2nd Threshold Violation (right):

- As the PLR is normalized, the measured throughput continues to grow, now above the previously set 25 Mb/s limit. Then, after it reaches the 40 MB/s limit, the PLR rises again to meet the 1 percent threshold. Upon this event, the VMs interfaces are configured once again now with a limit of 50 Mb/s in bandwidth capacity. PLR in turn, gets back to safe levels, thus stabilizing service operation considering the agreed performance standards.

9.3.3 Path Reconfiguration

The third use case, defines a scenario where the slice must maintain a required standard for the end-to-end latency between VM1 and VM2. To this end, a Latency Sensor (LS) is deployed along them at provisioning, so it can analyse traffic flowing through and calculate the latency. The MM then, collects this data and implements the following model in coordination with the PM:

- **EVENT (high latency)**: The LS, as introduced in *Section 8.2.1*, stores and exposes the calculated latency between VMs. The MM reads this data, so the PM can check when values rise up to service performance threatening levels.

- **CONDITION (latency>1ms)**: In this case, the limit for end-to-end latency is set to 1 ms, following the requirements fixed for 5G communications.

- **ACTION (reconfigure data path)**: In the event that conditions are met, the PM requests the controllers to trigger data path reconfiguration. In this manner, the required configurations are pushed to network devices, so the data path used for traffic flow between VMs is changed. By setting up an alternative connection, the latency levels can be maintained below the guard threshold in case these rise due to physical network malfunctions or to the overutilization of data paths.

The tests performed for this actuation case, are described in Fig 9-8, where VM1-LS-VM2 are deployed and connected at the experimental testbed. Then, by augmenting the traffic volume at some points of the scenario, network congestion can be simulated as current links of the used data path get saturated. This will result in a rise of the measured latency which can reach the established conditions.

The PM then, would trigger the switching of the current path to an alternative one in response to the event, performing flow reconfiguration at virtual switches via the SDN-Controllers. After the new data path becomes active, the end-to-end latency measured at the sensor get back to safe levels. By means of these tests, the path reconfiguration use case for slice maintenance was proven, considering specially services that are highly dependent on strict latency levels for its normal operation.

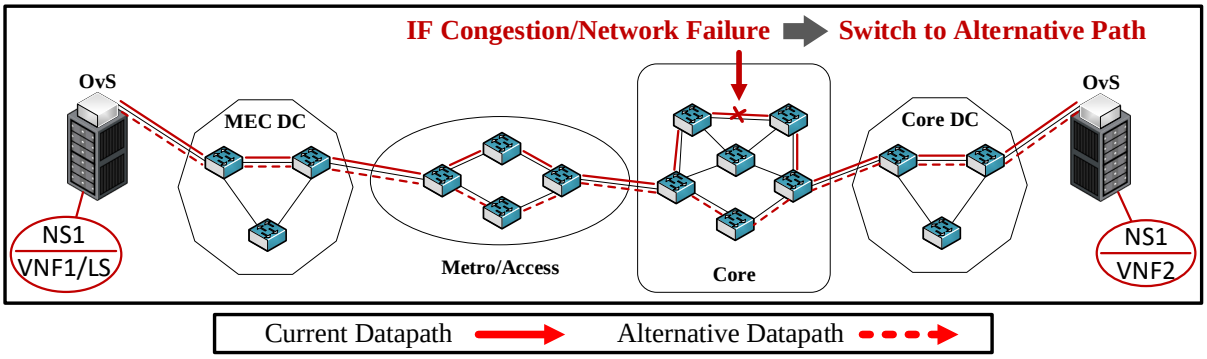


Fig. 9-8. Latency-sensitive Service Maintenance through Path Reconfiguration.

9.3.3.1 Inter-Module Messaging Workflow

Fulfilling the analysed actuation case requires a set of interactions at the NFV-C level and towards the involved components. To clarify this process, a step-by-step workflow is described next along with a graphic representation as in Fig. 9-9.

- **Step 1 - (PM <-> MM)**: While the MM is collecting latency data, the PM performs the necessary checks to validate if latency levels satisfy the defined policies. In the event the given model conditions are met, the PM checks the MM database for the current state of monitored data paths in respect of their average latency.

- **Step 2 - (PM -> TM)**: If an alternative path is selected as a potential solution for the high latency event, the PM then triggers the reconfiguration of the data path by requesting the TM to coordinate the required actions.

- **Step 3 - (TM -> WAN-Controller)**: The TM in turn, takes charge in this case of contacting the WAN-Controller, so it can obtain the reconfiguration instructions.

- **Step 4 - (WAN-Controller -> Network Devices)**: Then, configurations required to set up the new connection get pushed towards the network devices. In this way, the alternative path gets configured and the configurations related to the identified affected path are deleted. Once everything is set, the path becomes active and the measured latency is expected to go back to safe levels.

- **Step 5 - (MM -> PM)**: In order to validate the correct application of the actuation and as a way to reload the sensing-actuation cycle, the MM starts again the continuous collection of latency data from the sensor, so the PM can furtherly check policies compliance. In the event a new violation to defined thresholds is found, the process would start all over again from Step 1.

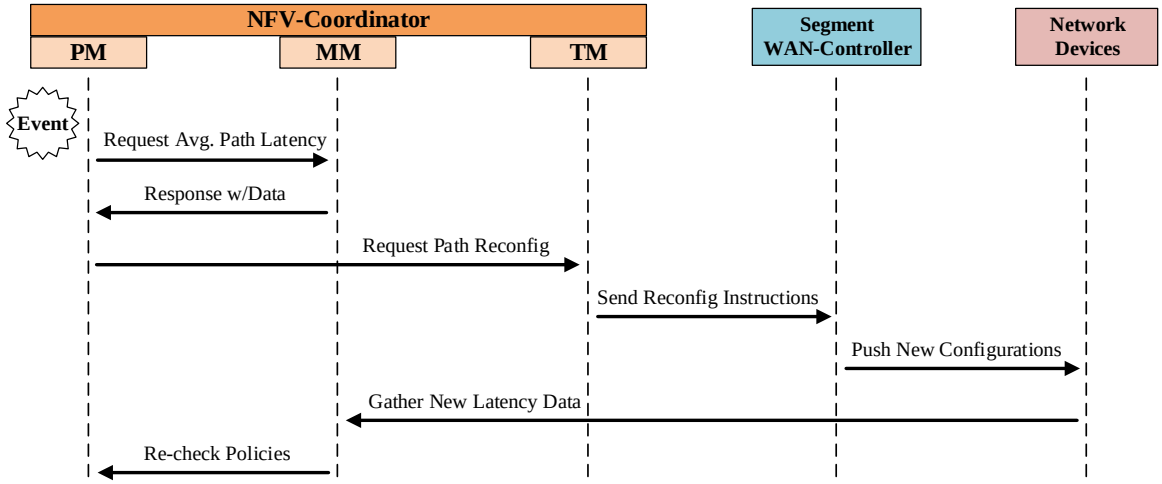


Fig. 9-9. Component Interaction Workflow for Path-Reconfiguration Process.

9.3.4 Slice Reallocation

The last actuation case considered at the experimental tested is associated to the reallocation of a particular slice, by rearranging network configurations to achieve better performance in terms of latency. Although in theory it may seem similar to the path reconfiguration case, a slice reallocation may also represent the migration of VNFs between different VIM segments [85], besides the usual configuration of new connections/data paths. In this regard, Fig. 9-10 describes the analysed case where two services/slices are deployed with two VNFs each, which are connected from MEC DC to Core DC2 respectively. The VNFs at the latter segment in turn, share the same physical location which could derive into potential congestions if

there is a large volume of traffic being transmitted to this section of the network. In the event that this occurs, an evaluation of other possible locations to perform VNF migration is conducted. As illustrated, the resultant action in this case involves the migration of VNF2 of NS2 to the Core DC1 segment. In this manner, the traffic flow gets distributed so the expected latency levels can be guaranteed for both slices.

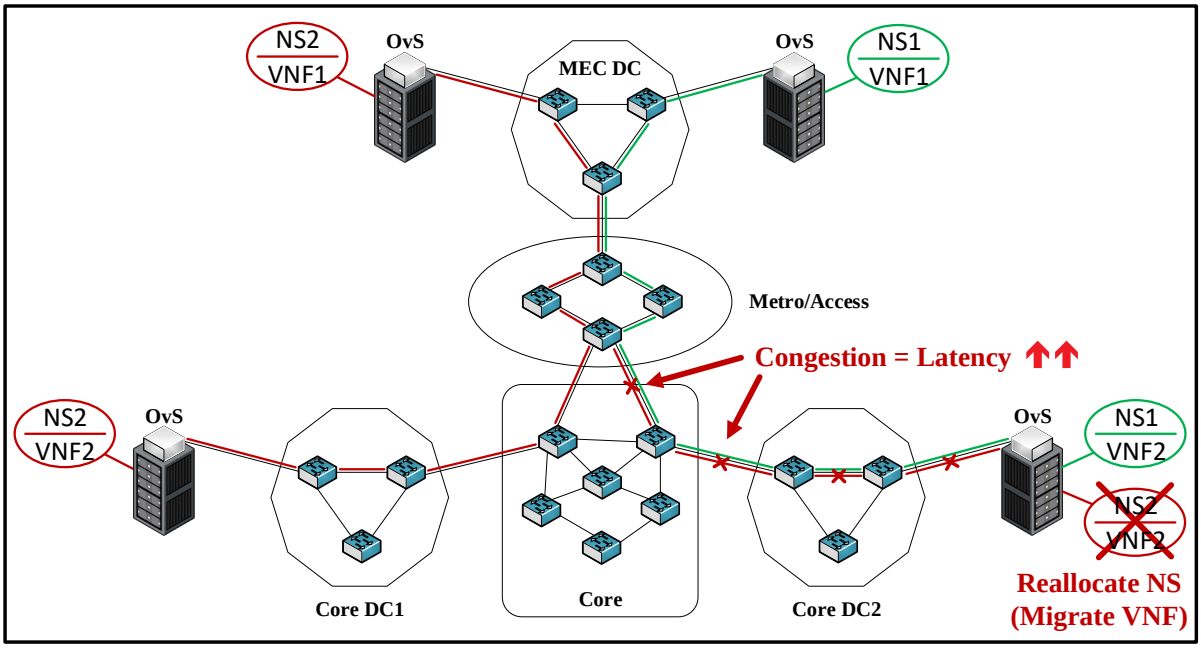


Fig. 9-10. Latency-sensitive Service Maintenance through Slice Reallocation.

9.3.4.1 Inter-Module Messaging Workflow

In respect of the necessary interactions between architectural components, the slice reallocation process requires more steps than the previously analysed path reconfiguration. Fig. 9-11 presents a representation of such required steps, while a more detailed description follows:

- **Step 1 - (PM <-> SM)**: Upon the violation of established latency thresholds, the PM sends a request to the SM to evaluate if there exist an option to reallocate one or many of the running slices (i.e., by VNF migration). The response from the SM would provide the possible location options (if there is any) taking into account the available computational resources.

- **Step 2 - (PM <-> MM)**: The PM then, would calculate the best candidate paths towards these locations, using the monitored latency data stored at the MM.

- **Step 3 - (PM -> SM)**: In case an option fits for slice reallocation, the PM sends a request the SM to trigger this specific actuation.

- **Step 4 - (PM -> TM)**: At the same time, the PM would send another request to the TM asking for the configuration of the selected data path towards the new location, while also considering the deletion of the one used before.

- **Step 5 - (PM -> IM)**: The IM in turn, should also become aware of slice changes so it can set up required routing configurations to and from newly placed VNFs at the corresponding VIM segments.

- **Step 6 - (SM -> OSM)**: After the SM receives the request for changing the slice configuration, it sends specific instructions to OSM, so it can fulfil them.

- **Step 7 - (OSM -> VIM-Orchestrators)**: OSM then, pushes the required set of configurations towards the orchestrators at the specified VIMs, so these can go on with the VNF migration process.

- **Step 8 - (TM -> WAN-Controller)**: The TM in turn, contacts the WAN-Controller to send the new network configuration instructions.

- **Step 9 - (WAN-Controller -> Network Devices)**: Following these instructions, the controller pushes the required flows to network devices to set up the new routes towards the locations where VNFs were migrated, configuring data paths and deleting the original configurations afterwards.

- **Step 10 - (IM -> VIM-Controllers)**: The IM on the other hand, contacts the VIM-Controllers at the correspondent segments to give them the instructions needed for inter-VNF routing. After everything is set, the configured data paths become active and latency levels are expected to stabilize on the involved segments.

- **Step 11 - (MM -> PM)**: To test the correct application of the actuation and to reload again the sensing-actuation cycle, the MM starts collecting again latency data from the deployed sensors and checking policies compliance through the PM. In the event that new measured latency values surpass the permitted levels, the process starts all over again from step 1.

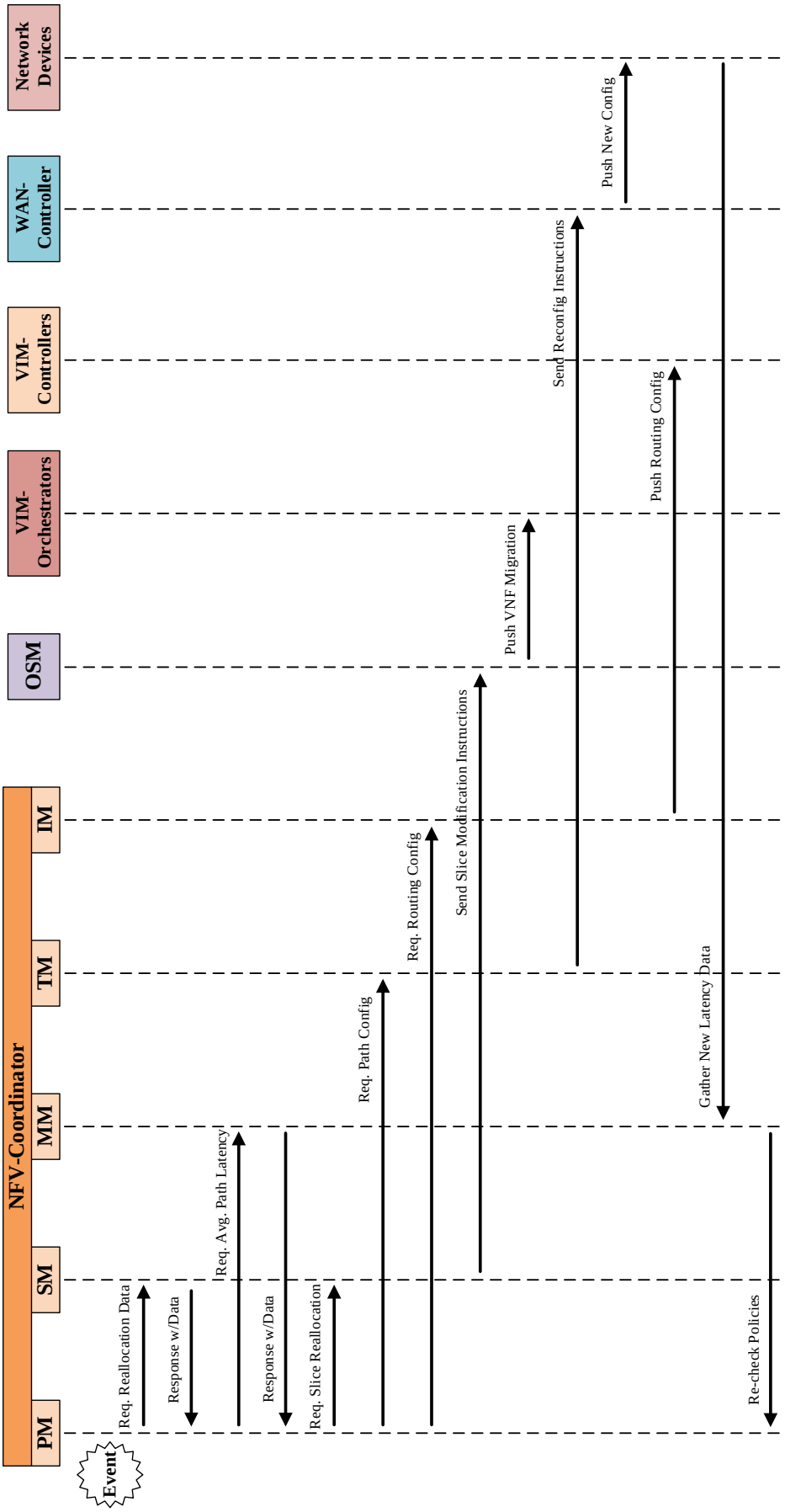


Fig. 9-11. Component Interaction Workflow for Slice-Reallocation Process.

9.4 Conclusions

The experimental single-domain multi-segment testbed presented in this chapter, follows the base architecture defined in *Chapter 7* and *Chapter 8*. The scenario in turn, considered the emulation of network resources across different segments of an end-to-end inter-DCN, besides the use of virtual resource orchestrators, SDN-controllers, a MANO entity and the practical implementation of the NFV-C modules. In this way, slice provisioning and maintenance related tests were performed.

The provisioning stage, was proved by the definition of a multi-segment NSI de-composed into a set of NSSIs to allocate its resources by means of the slice composition technique. Besides, the required network configurations driven from the NFV-C were also proven. In respect of the maintenance stage, some of the previously analysed sensing parameters and actuation cases were tested, where results validating the correct application of policy-based models were described.

By means of the tests performed on the given testbed scenario, the support given by the NFV-C for the required provisioning and maintenance tasks related to the management of 5G services/slices was verified, thus validating the practical usage of the proposed 5G common architecture.

9.5 Publications

The work realized throughout this thesis chapter has been exposed in the following scientific journal and conference publications:

1. **R. Montero**, F. Agraz, A. Pagès and S. Spadaro, “Enabling Multi-segment 5G Service Provisioning and Maintenance through Network Slicing,” Journal of Network and Systems Management (2020). <https://doi.org/10.1007/s10922-019-09509-9>

2. **R. Montero**, F. Agraz, A. Pagès and S. Spadaro, “Real-time Maintenance of Latency-sensitive 5G Services through Network Slicing,” Photonic Network Communications Journal [**Under Second Review** - June 2020].

3. **R. Montero**, F. Agraz, A. Pagès and S. Spadaro (2020), "End-to-end Network Slicing in Support of Latency-Sensitive 5G Services," In: Tzanakaki A. et al. (eds) Optical Network Design and Modeling (ONDM 2019). Lecture Notes in Computer Science, vol 11616. Springer, Cham. <https://doi.org/10.1007/978-3-030-38085-4_5>

4. **R. Montero**, A. Pagès, F. Agraz, and S. Spadaro, Supporting QoE/QoS-aware end-to-end network slicing in future 5G-enabled optical networks", Proc. SPIE OPTO vol. 10946, Metro and Data Center Optical Networks and Short-Reach Links II, 109460F, San Francisco, California, United States, February 2019. <https://doi.org/10.1117/12.2508579>

PART IV

RESULTS DISSEMINATION & CONCLUSIONS

Chapter 10. Scientific Dissemination

This chapter exposes the research projects and conference/journal publications, either published or in an accepted/submitted state, that served to disseminate the work achieved during the realization of this thesis on the scientific community.

10.1 Scientific Journals

The scientific journals published during this thesis, considering 3 author-led works, 2 published and 1 submitted, besides 1 collaboration work, as detailed next.

10.1.1 Author-led Publications

1. **R. Montero**, F. Agraz, A. Pagès, J. Perelló and S. Spadaro, “SDN-based parallel link discovery in optical transport networks,” Transactions on Emerging Telecommunications Technologies, volume 30, e3512, 2019. <https://doi.org/10.1002/ett.3512>

2. **R. Montero**, F. Agraz, A. Pagès and S. Spadaro, “Enabling Multi-segment 5G Service Provisioning and Maintenance through Network Slicing,” Journal of Network and Systems Management (2020). <https://doi.org/10.1007/s10922-019-09509-9>

3. **R. Montero**, F. Agraz, A. Pagès and S. Spadaro, “Real-time Maintenance of Latency-sensitive 5G Services through Network Slicing,” Photonic Network Communications Journal [**Under Second Review** - June 2020].

10.1.2 Collaborations

1. A. Pagès, F. Agraz, **R. Montero**, G. Landi, M. Capitani, D. Gallico, M. Biancani, R. Nejabati, D. Simeonidou, S. Spadaro, “Orchestrating virtual slices in data centre infrastructures with optical DCN,” Optical Fiber Technology, vol. 50, 2019, pp 36-49. <https://doi.org/10.1016/j.yofte.2019.02.011>

10.2 Conference Papers

The conference papers presented in the scope of the thesis refer to the following publications, considering 6 author-led and 5 collaboration works.

10.2.1 Author-led Publications

1. **R. Montero**, F. Agraz, A. Pagès, J. Perelló and S. Spadaro, "Dynamic topology discovery in SDN-enabled Transparent Optical Networks," 2017 International Conference on Optical Network Design and Modeling (ONDM), Budapest, 2017, pp. 1-6.
   <https://doi.org/10.23919/ONDM.2017.7958525>

2. **R. Montero**, F. Agraz, A. Pagès and S. Spadaro, "End-to-End 5G Service Deployment and Orchestration in Optical Networks with QoE Guarantees," 2018 20th International Conference on Transparent Optical Networks (ICTON), Bucharest, 2018, pp. 1-4.
   <https://doi.org/10.1109/ICTON.2018.8473996>

3. **R. Montero**, N. Calabretta, F. Agraz, A. Pagès and S. Spadaro, "SDN-controlled and Orchestrated OPSquare DC Architecture enabling Network Slice Deployment with QoS guarantees," 2018 Photonics in Switching and Computing (PSC), Limassol, Cyprus, 2018, pp. 1-3.
   <https://doi.org/10.1109/PS.2018.8751416>

4. **R. Montero**, A. Pagès, F. Agraz, and S. Spadaro, Supporting QoE/QoS-aware end-to-end network slicing in future 5G-enabled optical networks", Proc. SPIE OPTO vol. 10946, Metro and Data Center Optical Networks and Short-Reach Links II, 109460F, San Francisco, California, United States, February 2019.
   <https://doi.org/10.1117/12.2508579>

5. **R. Montero**, F. Agraz, A. Pagès and S. Spadaro (2020), "End-to-end Network Slicing in Support of Latency-Sensitive 5G Services," In: Tzanakaki A. et al. (eds) Optical Network Design and Modeling (ONDM 2019). Lecture Notes in Computer Science, vol 11616. Springer, Cham.
   <https://doi.org/10.1007/978-3-030-38085-4_5>

6. **R. Montero**, F. Agraz, A. Pagès and S. Spadaro, "Actuation Framework for 5G-Enabled Network Slices with QoE/QoS Guarantees," 2019 21st International Conference on Transparent Optical Networks (ICTON), France, 2019, pp. 1-4. <https://doi.org/10.1109/ICTON.2019.8840548>

10.2.2 Collaborations

1. S. Spadaro, A. Pagès, F. Agraz, **R. Montero** and J. Perelló, "Resource orchestration in SDN-based future optical data centres," 2016 International Conference on Optical Network Design and Modeling (ONDM), Cartagena, 2016, pp. 1-6. <https://doi.org/10.1109/ONDM.2016.7494057>

2. A. Pagès, F. Agraz, **R. Montero**, G. Landi, R. Monno, J.I. Aznar, A. Vinez, C. Jackson, D. Simeonidou and S. Spadaro, "Experimental Assessment of VDC Provisioning in SDN/OpenStack-based DC Infrastructures with Optical DCN," ECOC 2016; 42nd European Conference on Optical Communication, Dusseldorf, Germany, 2016, pp. 1-3. <https://ieeexplore.ieee.org/document/7767632>

3. S. Sarmiento, **R. Montero**, J.A Altabas, D. Izquierdo, A. Pagès, J. Perello, J. Gene, M. Alonso, A. Pascual, I. Garces, S. Spadaro and J. A. Lazaro, "SDN-enabled flexible optical node designs and transceivers for sustainable metro-access networks convergence," 2016 18th International Conference on Transparent Optical Networks (ICTON), Trento, 2016, pp. 1-4. <https://doi.org/10.1109/ICTON.2016.7550659>

4. S. Spadaro, A. Pagès, F. Agraz, **R. Montero** and J. Perelló, "Orchestrated SDN-based VDC provisioning over multi-technology optical data centre networks," 2017 19th International Conference on Transparent Optical Networks (ICTON), Girona, 2017, pp. 1-4. <https://doi.org/10.1109/ICTON.2017.8025181>

5. A. Pagès, F. Agraz, **R. Montero** and S. Spadaro, "Dynamic Service Reallocation in NFV-based Transport WDM Optical Networks," 2018 Photonics in Switching and Computing (PSC), Limassol, Cyprus, 2018, pp. 1-3. <https://doi.org/10.1109/PS.2018.8751237>

6. S. Spadaro, F. Agraz, A. Pagès, and **R. Montero**, “Autonomic 5G and beyond network management” [**Accepted**], IEEE 22nd International Conference on Transparent Optical Networks (ICTON) 2020, 19-23 July 2020, Bari, Italia.

10.3 Research Projects

Throughout the development of the thesis, efforts were made in relation to specific Spanish and European research projects. In this regard, a brief description of each project is given next, besides analysing their relationship with this PhD thesis.

10.3.1 Spanish Projects

10.3.1.1 SUNSET Project

This SUNSET project [86] looks forward to solve bottlenecks issues in existing transport networks by means of implementing novel network architectures based in advanced optical technologies controlled and orchestrated from a SDN control plane, searching for the dynamic provisioning of traffic loads to guarantee the optimal utilization of networks and computational resources.

This thesis followed the proposed scenario in SUNSET to identify potential issues and set goals to optimize resource utilization, focusing on the management of optical related data between devices and controller. In this manner, the works presented during this thesis addressed the required SDN control for specific optical devices, allowing the joint management and automation of both network and IT resources, particularly in intra-DCNs when considering the relation to this project.

10.3.1.2 ALLIANCE-B Project

The ALLIANCE-B project [87] defines as its goal, the design and implementation of a network architecture that can be integrated into the 5G infrastructure. To this end, the project bases on SDN and NFV concepts to provide support for the control, management and orchestration of heterogeneous network and IT resources, taking into account the use of all-optical solutions for inter/intra-DCNs. At management and orchestration level, cognition-based techniques are considered to optimize the user's quality of experience (QoE) and the network resource utilization. Moreover, ALLIANCE-B focuses also on the design of new optical nodes to provide efficient signal switching, besides the use of novel transmission techniques so an ultra-high capacity and flexible optical network can be provided in support of 5G services.

Research done in this PhD thesis, studied the architectural needs to address the goals defined in this project, proposing a multi-segment architecture capable of handling the management of network and IT resources across inter-DCNs. In this way, the high-level analysis of monitored parameter data was also considered, presenting in this case the use of an actuation-decision table to guarantee service performance closer to the user experience (i.e., at the QoE level).

10.3.2 European Projects

10.3.2.1 FP7 COSIGN Project

The COSIGN project [52], aims to define and implement a flat and scalable DCN architecture, powered by the use of optical technologies and SDN, in a way to overcome the drawbacks of current intra-DCN architectural solutions. To this end, several key use cases for deploying the architecture were defined on the scope of the project [7], focused on the virtualization and orchestration of DC infrastructures.

The work presented in this thesis, showed a significant contribution to this project, as described in *Chapter 4*. More specifically, support was provided to fulfil the Virtual Data Centre (VDC) use case, where new software components at the SDN level were developed, while others were extended, to provide a fully functional architecture capable of managing the configuration and virtualization of hybrid

electrical/optical intra-DCNs. In this way, multiple VDC allocation over a commonly shared infrastructure was enabled, thus optimizing physical resources utilization.

10.3.2.2 H2020 SLICENET Project

SLICENET H2020 project [88] looks forward to address management and control planes for the support of network slicing technologies across the 5G infrastructure, in a way to facilitate the adoption of 5G slices for verticals, to then achieve an efficient and flexible resource usage in view of their demanding use cases. In addition, the configurable warranties required in terms of service QoS and/or QoE are also aimed, to secure the expected service operation throughout its lifecycle.

Following the objectives of the SLICENET project, this thesis studied the requirements for a common 5G architecture, taking into account the mapping of 3GPP 5G network resource model and the ETSI NFV information model. In this way, an architecture tailored for managing 5G slices provisioning and maintenance was presented, considering the introduction of a coordination element at the Slice Management level so all the required configurations for the composition of an end-to-end slice could be performed. Besides, the gathering of specific data related to service defined SLAs/KPIs was addressed to further guarantee established service QoS levels through a set of analysed and experimentally proven actuation cases.

Chapter 11. Final Conclusions and Considerations

The last chapter of this document summarizes the contributions achieved during the realization of this thesis, considering as a base the defined objectives in order to evaluate the presented thesis. Besides, a view on the future work on the studied subject is also given to analyse the next steps of this research.

11.1 Conclusions

Following the objectives defined in *Chapter 3*, and considering the scope of this thesis as related to the optimization of resource utilization in inter/intra DCNs, enhanced by the application of novel technologies such as Optical, SDN, NFV and Network Slicing in the context of the 5G framework. This thesis presented a specific set of solutions to address the need for accommodating current infrastructures to the requirements set for next-generation network scenarios, as in the case of 5G technologies and beyond. With this goal, the document was divided in four parts, where *Part II* and *Part III* in particular, included the main contributions regarding the optimizations achieved in intra-DCN and inter-DCN scenarios, respectively.

Beginning with the introduction of optical technologies in DCNs, required to cope with the demands of high traffic loads and fast connectivity in 5G, the SDN-based control of optical devices was defined as the first objective in this thesis. In this regard, work presented in *Chapter 4* and *Chapter 6* specifically demonstrated the control and management of optical devices for different intra-DCN scenarios through the use of extended southbound protocols, the design/extension of SDN-level applications and the experimental validation of these implementations. In this manner, the control of hybrid electrical/optical networks was achieved, proving the potential of the SDN technology, in terms of programmability, flexibility and network optimization when applied over current and modern network infrastructures.

The work presented in *Chapter 5* then, analysed the necessity of providing the correct mapping of the underlying network topology at the SDN-level, especially considering the recognition the optical links/topology, so the consumers depending

of this information can correctly operate. To this end, two mechanisms for optical topology discovery where designed, following a sequential and a parallel approach, considering all the required implementations at the devices agents and at the SDN controller to support the discovery process. In this manner, a new SDN application was designed while extensions were also made to other components to support optical link recognition without exchanging discovery related information at the data layer, thus not affecting network traffic. Both methods in turn, where proven in an emulated testbed, where results showed not only the correct mapping of the underlying network configuration for both methods, but also the advantages of applying the parallel approach in respect of the sequential one considering the achieved topology discovery time, thanks to the use of specific link-binding data.

Moving to the next part of this thesis, *Chapters 7-9* were focused on defining and proving a common architecture in support of 5G, taking the inter-DCN scenario as a base along with the application of SDN, NFV and Network Slicing technologies to manage the provisioning and maintenance of network slices. To accomplish this, the current standards and community approaches towards the definition of a 5G architecture were studied, identifying the need for enabling the mapping between defined frameworks and information models. In this way, a single-domain multi-segment architecture was defined, including the design of a coordination entity at the Slice Management level capable of managing the provisioning of slices across network segments. Besides, sensing and actuations operations were also studied, to allow executing slice maintenance related tasks from the introduced component. To validate the proposed framework, an experimental testbed was setup, where the provisioning of slices by means of the slice composition technique was proven, besides testing the deployment of specific VNF-based sensors and the gathering of monitoring data from different levels of the architecture. Finally, a set of actuation cases were defined and tested, demonstrating the capacity of the given framework for guaranteeing QoS agreed levels for a particular slice/service.

The results and the proposals presented throughout this thesis document, introduced novel resource optimization solutions for inter/intra DCN scenarios, considering the control, orchestration, virtualization and management of resources at different architectural levels and at different segments of the 5G network. Taking

into account the current push towards the use of next-generation technologies to fulfil the requirements of future network infrastructures. The work achieved in this PhD thesis can be considered as a valuable addition to the state of the art, serving as proof-of-concept for the implementation of these technologies while considering as well the definition and validation of specific resource optimization techniques.

11.2 Future Work

This PhD thesis addressed challenges considered crucial for the management of resources in future network scenarios, such as the SDN-control of optical devices, the dynamic discovery of optical topologies and the definition of an architecture in support of 5G slice provisioning and maintenance. In this regard, further efforts still have to be considered to follow the research line of this thesis and take it to the next level of implementation and validation.

More specifically, the validation of the designed optical topology discovery methods, although proven using an emulated testbed based on software agents of optical devices, has to be furtherly tested considering a more real-case scenario, so other factors difficult to analyse in an emulated environment, can be addressed.

As for the proposal of a common 5G architecture, it becomes necessary to perform more testing to better define the intelligence and the use of the required cross-layer communication interfaces at the introduced coordinator component, in order to validate its full automation. Besides, adding Service Management level components to the experimental validation is also required, so the full service cycle starting with the request from the 5G client/vertical can be proven.

In addition to these specific considerations, the support of SDN control for optical technologies still demands extra efforts from the community, in order to analyse and develop more use cases so these can be implemented at the industry-level. To this end, the definition of a standard southbound communication protocol to manage the exchange of optical related data between network devices and the controller becomes decisive.

Appendix

I. ORVM Main YANG file

The base YANG file code for the ORVM module can be seen next, where all set of variables required to define the datastore and java classes of the application are defined, and where RPCs functions are also described.

Code 1 - open-resource-virtualization-manager.yang

```
module optical-resource-virtualization-manager {
    yang-version 1;
    namespace "urn:opendaylight:params:xml:ns:yang:orvm";
    prefix "orvm";

    import yang-ext {prefix ext; revision-date "2013-07-09";}
    /* OpenDaylight controller */
    import opendaylight-inventory {
        prefix inv;
        revision-date 2013-08-19;
    }

    revision "2016-03-07" {
        description "First stable version of the ORVM model. Based on
references to the elements in the inventory";
    }

    /* Virtual Nodes */
    typedef vnode-ref {
        type instance-identifier;
        description "A reference that points to a orvm:vnode-cont/vnode";
    }
    typedef vnode-lref {
        type leafref {
            path "/vnode-cont/vnodes/vnode-id";
        }
        description "A type for an absolute reference to a Vnode instance.";
    }
    typedef tp-lref {
        type leafref {
            path "/vnode-cont/vnodes/vnode-connector/tp-id";
        }
        description "A type for an absolute reference to a TP instance.";
    }
identity vnode-context {
        description "A vnode-context is a classifier for node elements which
allows an RPC to provide a service on behalf of a particular element in the
data tree.";
    }

grouping vnode-attributes {
        description "Representation of a virtual node";

        leaf vnode-id {
            type inv:node-id;
            description "Virtual node identifier = 'node-id:vnode-
instance'.";
        }
```

```
        leaf vnode-instance {
            type int32;
            description "Identifier of the virtual instance of the node.
Automatically increased";
        }
        leaf phy-node {
            type inv:node-ref;
            description "Reference to the physical node";
        }
        list vnode-connector {
            key "vnode-connector";
            leaf tp-id {
                type inv:node-connector-id;
                description "Identifier of the Node Connector";
            }
            leaf vnode-connector {
                type inv:node-connector-ref;
                description "Reference to the physical node connector";
            }
        }
    }

    grouping vnode-context-ref {
        description
        "A helper grouping which contains a reference to a node classified
with a vnode-context. This allows RPCs in other yang files to refine their
input to a particular node instance.";

        leaf vnode {
            ext:context-reference "vnode-context";
            type vnode-ref;
            description "A reference to a particular vnode.";
        }
    }
    /* Virtual Links */
    typedef vlink-ref {
        type instance-identifier;
        description "A reference that points to a orvm:vlink-cont/vlink";
    }
    typedef virtual-link-id {
        type string;
        description "Identifier of the virtual link.;
    }
    identity vlink-context {
        description "A vlink-context is a classifier for vlink elements
which allows an RPC to provide a service on behalf of a particular element
in the data tree.";
    }

    grouping vlink-attributes {
        description "Representation of a virtual link";

        leaf vlink-id {
            type virtual-link-id;
            description "Virtual link identifier";
        }
        leaf connection-id {
            type string;
            description "Connection Id of the associated Optical Path from
the Optical Provisioning Manager";
        }
         container source {
            leaf src-vnode {
                //type vnode-ref;
                type vnode-lref;
                description "Reference to the source virtual node";
            }
```

```
             leaf src-tp {
                //type inv:node-connector-ref;
                type tp-lref;
                description "Reference to the physical node connector";
            }
        }
        container destination {
            leaf dst-vnode {
                //type vnode-ref;
                type vnode-lref;
                description "Reference to the destination virtual node";
            }
            leaf dst-tp {
                //type inv:node-connector-ref;
                type tp-lref;
                description "Reference to the physical node connector";
            }
        }
    }

    grouping vlink-context-ref {
        description
        "A helper grouping which contains a reference to a vlink classified
with a vlink-context. This allows RPCs in other yang files to refine their
input to a particular vlink instance.";

        leaf link {
            ext:context-reference "vlink-context";
            type vlink-ref;
            description "A reference to a particular vlink.";
        }
    }
    /* Virtual Topologies */
    typedef vtopo-ref {
        type instance-identifier;
        description "A reference that points to a orvm:vtopo-cont/vtopo";
    }
    identity vtopo-context {
        description "A vtopo-context is a classifier for vtopo elements
which allows an RPC to provide a service on behalf of a particular element
in the data tree.";
    }

    grouping vtopo {
        description "Representation of a virtual topology";
         leaf vtopo-id {
            type string;
            description "Identifier of the virtual instance of the link.";
        }
        list vnode {
            description "The list of virtual nodes of the virtual topology";
            key "vnode-id";
            uses vnode-attributes;
        }
        list vlink {
            description "The list of virtual links of the virtual topology";
            key "vlink-id";
            uses vlink-attributes;
        }
    }

    grouping vtopo-context-ref {
        description
         "A helper grouping which contains a reference to a vtopo
classified with a vtopo-context. This allows RPCs in other yang
files to refine their input to a particular vtopo instance.";
```

```
        leaf topo {
            ext:context-reference "vlink-context";
            type vtopo-ref;
            description "A reference to a particular vtopo.";
        }
    }
    /* Virtual Node Container */
    container vnode-cont {
        description "The root container of all vnodes.";
        list vnodes {
            //key "phy-node vnode-instance";
            key "vnode-id";
            ext:context-instance "vnode-context";
            description "A list of the virtual nodes";
            uses vnode-attributes;
        }
    }
    /* Virtual Link Container */
    container vlink-cont {
        description "The root container of all vlinks.";
        list vlinks {
            key "vlink-id";
            ext:context-instance "vlink-context";
            description "A list of the virtual links ";
            uses vlink-attributes;
        }
    }
    /* Virtual Topology Container */
    container vtopo-cont {
        description "The root container of all vlinks.";
        list vtopos {
            key "vtopo-id";
            ext:context-instance "vtopo-context";
            description "A list of the virtual topologies";
            uses vtopo;
        }
    }

    // RPCs

    rpc create-vnode {
        description "Create a new virtual node instance of the given
physical node";
        input {
            //uses inv:node-context-ref;
            leaf pnode-id {
                type inv:node-id;
                description "Identifier of the physical node";
            }
         }
     }
    rpc get-vnode {
        input {
            uses vnode-context-ref;
        }

        output {
            uses vnode-attributes;
        }
    }
    rpc get-all-vnodes {
        output {
            list vnodes {
                uses vnode-attributes;
            }
        }
    }
```

```
    rpc create-vlink {
        description "Create a new virtual link instance";
        input {
            leaf vl-id {
                type virtual-link-id;
                description "Identifier of the virtual link";
            }
            leaf src-vnode-id {
                type inv:node-id;
                description "Identifier of the source virtual node";
            }
            leaf src-node-connector-id {
                type inv:node-connector-id;
                description "Identifier of the port of the source physical
node";
            }
            leaf dst-vnode-id {
                type inv:node-id;
                description "Identifier of the destination virtual node";
            }
            leaf dst-node-connector-id {
                type inv:node-connector-id;
                description "Identifier of the port of the destination
physical node";
            }
        }
    }
    rpc get-all-vlinks {
        output {
            list vlinks {
                uses vlink-attributes;
            }
        }
    }
    rpc create-vtopo {
        description "Create a new virtual topology";
        input {
            leaf vt-id {
                type string;
                description "Identifier of the virtual topology";
            }
            list vt-links {
                leaf vl-id {
                    type virtual-link-id;
                }
            }
            list vt-nodes {
                leaf vn-id {
                    type inv:node-id;
                    description "Identifier of the virtual node";
                }
            }
        }
    }
     rpc create-vtnlink {
        description "Creates VTN-links in the vtn inventory";
        input {
            leaf link-id
            {
                type string;
                description "Link identifier of type topo:link-id";
            }
            leaf vtn-src
            {
                type inv:node-connector-id;
                description "Source connector id of the source node";
            }
```

```
             leaf vtn-dst
            {
                type inv:node-connector-id;
                description "Destination connector id of the destination
node";
            }
        }
    }
}
```

II. ORVM Host-tracker YANG file

The YANG file code for the ORVM module host-tracker functionality can be seen next, considering all required variables and RPCs for augmenting the topology DS.

Code 2 – orvm-host-tracker.yang

```
module orvm-host-tracker {
    yang-version 1;
    namespace "urn:opendaylight:params:xml:ns:yang:orvm-host-tracker";
    prefix "orvm-host.track";
    import ietf-inet-types {prefix inet; revision-date 2010-09-24; }
    import ietf-yang-types {prefix yang; revision-date 2010-09-24; }
    import opendaylight-inventory {prefix inv;}
    import network-topology {prefix "topo"; revision-date "2013-10-21"; }
    import yang-ext {prefix ext;}

    revision "2016-06-10" {
        description "First version of the ORVM Host-Tracker model.";
    }

    grouping address-node-connector {
        list addresses {
            key id;
            leaf id {
                type uint64;
                description "A 64-bit key for this observation. This is
opaque and should not be interpreted.";
            }
            leaf mac {
                type yang:mac-address;
                description "MAC address";
            }
            leaf ip {
                type inet:ip-address;
                description "IPv4 or IPv6 address";
            }
            leaf vlan {
                type uint32;
                description "VLAN id";
            }
        }
    }
    typedef host-id {
        type string;
    }
    grouping host {
        description "List of addresses and attachment points";
        uses address-node-connector;
        leaf id {
            type host-id;
        }
```

```
        list attachment-points {
            description "the assumption is that all addresses can be reached
at all attachment points";
            uses topo:tp-attributes;
            key tp-id;
            leaf corresponding-tp {
                type topo:tp-ref;
            }
            leaf active {
                type boolean;
            }
        }
    }
    augment "/topo:network-topology/topo:topology/topo:node" {
        ext:augment-identifier "host-node";
        uses host;
    }
    rpc create-orvm-host {
        description "Creates hosts in the network-topology datastore";
        input {
            leaf host-id
            {
                type string;
                description "identifier of Host:host-id";
            }
            leaf tp-id
            {
                type string;
                description "identifier of Attachment Point:tp-id";
            }
        }
    }
    rpc create-orvm-link {
        description "Creates links in the network-topology datastore";
        input {
            leaf host-id
            {
                type string;
                description "identifier of Host:host-id";
            }
            leaf node-id
            {
                type string;
                description "identifier of Node Id:node-id";
            }
            leaf tp-id
            {
                type string;
                description "identifier of Attachment Point:tp-id";
            }
        }
    }
    rpc delete-orvm-host {
        description "Deletes host in the network-topology datastore";
        input {
            leaf host-id
            {
                type string;
                description "identifier of Host:host-id";
            }
            leaf tp-id
              {
                type string;
                description "identifier of Attachment Point:tp-id";
            }}
    }
}
```

References

[1] *NGMN Alliance 5G White Paper, Version 1.0, February 2015.*

[2] *5G-PPP, View on 5G Architecture, Version 1.0, July 2016.*

[3] *5G-PPP, "Advanced 5G Network Infrastructure for the Future Internet", Public Private Partnership in Horizon 2020, November 2013.*

[4] *W. Xia, P. Zhao, Y. Wen and H. Xie, "A Survey on Data Center Networking (DCN): Infrastructure and Operations," in IEEE Communications Surveys & Tutorials, vol. 19, no. 1, pp. 640-656, Firstquarter 2017.*

[5] *C. Kachris and I. Tomkos, "A Survey on Optical Interconnects for Data Centers," in IEEE Communications Surveys & Tutorials, vol. 14, no. 4, pp. 1021-1036, Fourth Quarter 2012.*

[6] *ONF White Paper, "Software-Defined Networking: The New Norm for Networks", Open Networking Foundation, 3-7 (2012).*

[7] *COSIGN FP7, WP1 Deliverable D1.1, "Requirements for next generation intra-Data Centres", July 2014.*

[8] *ONF TR-522, "SDN Architecture for Transport Networks", March 2016.*

[9] *D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," in Proceedings of the IEEE, vol. 103, no. 1, pp. 14-76, Jan. 2015.*

[10] *P. Goransson and C. Black, "How SDN Works", Software Defined Networking - A Comprehensive Approach, 59-61 (2014).*

[11] *ONF, Open Networking Foundation <https://www.opennetworking.org/>.*

[12] *ONF, "OpenFlow-enabled SDN and Network Functions Virtualization", ONF Solution Brief, February 2014.*

[13] *N. McKeown et al., "OpenFlow: Enabling innovation in campus networks". ACM Communications Review, vol. 38, num. 2, pp. 69-74, April 2008.*

[14] *ONF TR-502, "SDN Architecture 1.0", Issue 1, June 2014.*

[15] *C. Ebert, G. Gallardo, J. Hernantes and N. Serrano, "DevOps," in IEEE Software, vol. 33, no. 3, pp. 94-100, May-June 2016.*

[16] *OpenStack Cloud Operating System <https://www.openstack.org/>.*

[17] *P4 Protocol, "Programming Protocol-independent Packet Processors" <https://p4.org/>.*

[18] *ONF TS-002, "OpenFlow Switch Specification", Version 1.1.0, Open Networking Foundation, (2011).*

[19] *ONF TR-508, "Requirements Analysis for Transport Openflow/SDN", v1.0, August 2014.*

[20] *S. Das, "Extensions to the OF Protocol in support of Circuit Switching", addendum v0.3, June 2010.*

[21] *ONF TS-022, "Optical Transport Protocol Extensions", v1.0, March 2015.*

[22] *CALIENT Technologies, "Optical Circuit Switch OpenFlow Protocol Extensions", rev 0.4, June 2015.*

[23] *F. Pakzad, "Comparison of Software Defined Networking (SDN) Controllers", Aptira, July 2019. <https://aptira.com/comparison-of-software-defined-networking-sdn-controllers-part-7-comparison-and-product-rating/>.*

[24] *OSGi Alliance, Architecture <https://osgi.org/developer/architecture/>.*

[25] *Docker, The Modern Platform for High-Velocity Innovation <hhtps://www.docker.com>.*

[26] *A. S. Thyagaturu, A. Mercian, M. P. McGarry, M. Reisslein and W. Kellerer, "Software Defined Optical Networks (SDONs): A Comprehensive Survey," in IEEE Communications Surveys & Tutorials, vol. 18, no. 4, pp. 2738-2786, Fourthquarter 2016.*

[27] *ETSI, "Network Functions Virtualisation - Introductory White Paper", Issue 1, October 2012.*

[28] *ETSI GS NFV 002, "Network Functions Virtualization (NFV), Architectural Framework", v1.2.1, December 2014.*

[29] *ETSI GR NFV 001, "Network Functions Virtualisation (NFV), Use Cases", v1.2.1, May 2017.*

[30] *ETSI GS NFV-MAN 001, "Network Functions Virtualisation (NFV), Management and Orchestration", v1.1.1, December 2014.*

[31] *Open Source MANO Project <https://osm.etsi.org/>.*

[32] *ONAP, Open Network Automation Platform <https://www.onap.org/>.*

[33] *Open Baton Framework <https://openbaton.github.io/>.*

[34] *OPNFV, Open Platform for NFV <https://www.opnfv.org/>.*

[35] *G. M. Yilma, F. Z. Yousaf, V. Sciancalepore, X. Costa-Pérez, "Tutorial Paper Benchmarking Open-Source NFV MANO Systems: OSM and ONAP", 2019.*

[36] *Kubernetes Container Orchestration System <https://kubernetes.io/>.*

[37] *OSM White Paper, "OSM Deployment and Integration", Issue 1, February 2020.*

[38] *3GPP TR 28.801, "Study on management and orchestration of network slicing for next generation network", version 15.1.0, January 2018.*

[39] *NGMN Alliance, "Description of network slicing concept", January 2016.*

[40] *3GPP TS 28.541: "Management and orchestration of networks and network slicing; NR and NG-RAN Network Resource Model (NRM); Stage 2 and stage 3", v16.3.0, December 2019.*

[41] *ETSI GR NFV-IFA 015: "Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Report on NFV Information Model", v3.1.1, September 2018.*

[42] *ETSI GR NFV-IFA 024: "Network Functions Virtualisation (NFV) Release 3; Information Modeling; Report on External Touchpoints related to NFV Information Model", v3.2.1, April 2019.*

[43] *ETSI TS 128 533 V15.0.0 , 3GPP TS 28.533 version 15.0.0 Release 15, "5G; Management and orchestration; Architecture framework", October 2018.*

[44] *5G-PPP, 5G Architecture White Paper, Version 2.0, December 2017.*

[45] *5G-PPP, 5G Architecture White Paper, Version 3.0, June 2019.*

[46] *N. Cvijetic, "Optical network evolution for 5G mobile applications and SDN-based control," 2014 16th International Telecommunications Network Strategy and Planning Symposium (Networks), Funchal, 2014, pp. 1-5.*

[47] *ITU-R M.2083-0, IMT Vision - Framework and overall objectives of the future development of IMT for 2020 and beyond, September 2015 <http://www.itu.int/dms_pubrec/itu-r/rec/m/R-REC-M.2083-0-201509-I!!PDF-E.pdf>.*

[48] *5G-PPP, 5G empowering vertical industries, February 2016 <https://5g-ppp.eu/wpcontent/uploads/2016/02/BROCHURE_5PPP_BAT2_PL.pdf>.*

[49] *5G-PPP, 5G-Infrastructure PPP Vision Document, March 2015 <https://5g-ppp.eu/wp-content/uploads/2015/02/5G-Vision-Brochure-v1.pdf>.*

[50] *K.-K. Nguyen, M. Cheriet and M. Lemay, "Enabling infrastructure as a service (IaaS) on IP networks: from distributed to virtualized control plane", IEEE Communications Magazine, vol. 51, no. 1, pp. 136-144, January 2013.*

[51] *S. Spadaro et al., "Virtual Slices Allocation in Multi-tenant Data Centre Architectures Based on Optical technologies and SDN" [invited], Asia Communications and Photonics Conference (ACP), November 2015.*

[52] *COSIGN FP7 Project, Advanced Optical Hardware and SDN for Next-Generation Data Centres <https://www.fp7-cosign.eu/>.*

[53] *N. Calabretta, et al., "System performance assessment of a monolithically integrated WDM cross-connect switch for optical data centre networks", 42nd European Conference and Exhibition on Optical Communications (ECOC 2016), Düsseldorf (Germany), 2016.*

[54] *N. Parsons, et al., "High Radix All-Optical Switches for Software-Defined Datacentre Networks", 42nd European Conference and Exhibition on Optical Communications (ECOC 2016), Düsseldorf (Germany), 2016.*

[55] *OpenDaylight SDN-Controller <https://www.opendaylight.org/>.*

[56] *IETF RFC-7950, "The YANG 1.1 Data Modeling Language", August 2016.*

[57] *ITU-T recommendation G.7714.1/Y.1705.1: protocol for automatic discovery in SDH and OTN networks. 2017.*

[58] *ITU-T recommendation G.7714/Y.1705: generalized automatic discovery for transport entities. 2015.*

[59] *J. Oliveira et al., "Experimental testbed of reconfigurable flexgrid optical network with virtualized GMPLS control plane and autonomic controls towards SDN," 2013 SBMO/IEEE MTT-S International Microwave & Optoelectronics Conference,* Rio de Janeiro, Brazil, 2013.

[60] *S. Khan, A. Gani, A.W.A. Wahab, M. Guizani and M.K Khan, "Topology discovery in software defined networks: threats, taxonomy, and state-of-the-art," IEEE Communications Surveys & Tutorials, vol. 19, no. 1, pp. 303-324, 2017.*

[61] *OpenFlow Discovery Protocol and Link Layer Discovery Protocol. <http://groups.geni.net/geni/wiki/OpenFlowDiscoveryProtocol>.*

[62] *F. Pakzad, M. Portmann, W. L. Tan and J. Indulska, "Efficient topology discovery in software defined networks," 2014 8th International Conference on Signal Processing and Communication Systems (ICSPCS), Gold Coast, QLD, 2014, pp. 1-8.*

[63] *A. Azzouni, N. T. Mai Trang, R. Boutaba and G. Pujolle, "Limitations of openflow topology discovery protocol," 2017 16th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net), Budva, 2017, pp. 1-3.*

[64] *T. Alharbi, M. Portmann and F. Pakzad, "The (in)security of Topology Discovery in Software Defined Networks," 2015 IEEE 40th Conference on Local Computer Networks (LCN), Clearwater Beach, FL, 2015, pp. 502-505.*

[65] *R. Egorov et.al., "Next generation ROADM architecture and design," 2013 Optical Fiber Communication Conference and Exposition (OFC), Anaheim, CA, 2013.*

[66] *F. Yan, W. Miao, O. Raz and N. Calabretta, "Opsquare: A flat DCN architecture based on flow-controlled optical packet switches," in IEEE/OSA Journal of Optical Communications and Networking, vol. 9, no. 4, pp. 291-303, April 2017.*

[67] *W. Miao et al., "SDN-enabled OPS with QoS guarantee for reconfigurable virtual data center networks," in IEEE/OSA Journal of Optical Communications and Networking, vol. 7, no. 7, pp. 634-643, July 2015.*

[68] *ONF TR-526, "Applying SDN Architecture to Network Slicing", Issue 1, April 2016.*

[69] *K. Samdanis, X. Costa-Perez and V. Sciancalepore, "From network sharing to multi-tenancy: The 5G network slice broker," in IEEE Communications Magazine, vol. 54, no. 7, pp. 32-39, July 2016.*

[70] *X. Costa-Perez, A. Garcia-Saavedra, X. Li, T. Deiss, A. de la Oliva, A. di Giglio, P. Iovanna and A.Moored, "5G-Crosshaul: An SDN/NFV Integrated Fronthaul/Backhaul Transport Network Architecture," in IEEE Wireless Communications, vol. 24, no. 1, pp. 3.*

[71] *R. Ravindran, A. Chakraborti, S. O. Amin, A. Azgin and G. Wang, "5G-ICN: Delivering ICN Services over 5G Using Network Slicing," in IEEE Communications Magazine, vol. 55, no. 5, pp. 101-107, May 2017.*

[72] *Q. Ye, J. Li, K. Qu, W. Zhuang, X. S. Shen and X. Li, "End-to-End Quality of Service in 5G Networks: Examining the Effectiveness of a Network Slicing Framework," in IEEE Vehicular Technology Magazine, vol. 13, no. 2, pp. 65-74, June 2018.*

[73] *R. Munoz, R. Vilalta, R. Casellas, R. Martinez, T. Szyrkowiec, A. Autenrieth, V. Lopez and D. Lopez, "Integrated SDN/NFV management and orchestration architecture for dynamic deployment of virtual SDN control instances for virtual tenant networks,",* Journal of Optical Communication Networks 7, 62-70, 2015.

[74] *CogNet deliverable D2.2, "CogNet final requirements, scenarios and architecture", April 2017.*

[75] *P. Neves, R. Cale, M. R. Costa, C. Parada, B. Parreira, J. M. A. Calero, Q. Wang, J. Nightingale, E. ChirivellaPerez, W. Jiang, H. D. Schotten, K. Koutsopoulos, A. Gavras and M. J. Barros, The SELFNET Approach for Autonomic Management in an NFV/SDN N.*

[76] *ETSI GS NFV-SOL 005 V2.4.1 (2018-02), "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; RESTful protocols specification for the Os-Ma-nfvo Reference Point".*

[77] *ETSI Open Source MANO, "OSM VNF Onboarding Guidelines" <http://osm-download.etsi.org/ftp/Documentation/vnf-onboarding-guidelines/>.*

[78] *ETSI Open Source MANO, OSM Release FIVE Technical Overview, 1st Edition, 2019.*

[79] *Gnocchi - Metric as a Service, Gnocchi 4.2.1.dev96 documentation <https ://gnocc hi.xyz/>.*

[80] *IETFdraft ietf-supa-generic-policy-info-model-03: "Generic Policy Information Model for Simplified Use of Policy Abstractions (SUPA)", May 30, 2017.*

[81] *Mininet Network Emulator <http://mininet.org>.*

[82] *Grafana Monitoring & Data Visualization Platform <https://grafana.com/>.*

[83] *iPerf Measurement Tool <https://iperf.fr/>.*

[84] *A. Pagès et. al., "Experimental Assessment of VDC Provisioning in SDN/OpenStack-based DC Infrastructures with Optical DCN," ECOC 2016; 42nd European Confence on Optical Communication, Dusseldorf, Germany, 2016, pp. 1-3.*

[85] *A. Pagès, F. Agraz, R. Montero and S. Spadaro, "Dynamic Service Reallocation in NFV-based Transport WDM Optical Networks," 2018 Photonics in Switching and Computing (PSC), Limassol, Cyprus, 2018, pp. 1-3.*

[86] *SUNSET Spanish Project, "Sustainable Network Infrastructure Enabling the Future Digital Society" <http://www.tsc.upc.edu/sunset/>.*

[87] *ALLIANCE-B, "Architecting a knowLedge-defined 5G-enabLed network Infrastructure towArd the upcomiNg digital soCiEty-B" <https://sunset.upc.edu/en/alliance-concept>.*

[88] *SLICENET H2020 Project, "End-to-End Cognitive Network Slicing and Slice Management Framework in Virtualised Multi-Domain, Multi-Tenant 5G Networks" <https://slicenet.eu/>.*

[89] *ONF Info-1002, "Open Disaggregated Transport Network (ODTN)", Open Networking Foundation, version 4.0, March 2019.*

Biography

Rafael Montero Herrera was born in Sucre, Bolivia in 1987. Following his interests in technology he started a career in the telecommunications field, receiving in 2010 his Bachelor's Degree [BSc] in Network & Telecommunications Engineering from Universidad Privada de Santa Cruz de la Sierra-UPSA (Bolivia). After this, he continued developing his skills, by working in the telecom field during 2011-2013 as a Customer Support Engineer for a national Internet Service Provider.

In the year 2013, his interests led him to pursue a higher-level education abroad, moving to Spain where he received a Master's Degree of Science [MSc] in Telecommunication Engineering & Management from Universitat Politècnica de Catalunya-UPC (Spain) in 2015, developing his Master thesis at the Optical Communications Group (GCO) under the supervision of Prof. Salvatore Spadaro. After the Master, he continued his relationship with the group through a set of collaborations, enrolling in 2016 to the PhD program of the UPC Department of Signal Theory and Communications (TSC).

As a Ph.D. candidate in UPC during 2016-2020, he developed his doctoral thesis focusing on network programmability and its application over electronic and optical networks as his main research area, exploring the use of novel technologies such as Software Defined Networking (SDN), Network Function Virtualization (NFV) and Network Slicing in the context of 5G and beyond. During this period, he collaborated in the research field through FP7 COSIGN and H2020 SLICENET European projects and with SUNSET and ALLIANCE-B Spanish projects.

As future goals, he looks forward in using all gained knowledge and skills to improve life quality standards by means of the application of telecommunication technologies and the research on innovative software-based solutions, considering a growing interest for working with international development and environmental projects aiming to address current issues worldwide.