UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels

MASTER THESIS

TITLE: Deployment and Validation of a Communication Suite using a NFV Service Platform

MASTER DEGREE: Master's degree in Applied Telecommunications and Engineering Management (MASTEAM)

AUTHOR: Juan Luis de la Cruz Cuevas

ADVISOR: Ricard Vilalta

SUPERVISOR: Salvatore Spadaro

DATE: July 7, 2019

Title : Deployment and Validation of a Communication Suite using a NFV Service Platform

Author: Juan Luis de la Cruz Cuevas

Advisor: Ricard Vilalta

Supervisor: Salvatore Spadaro

Date: July 7, 2019

Overview

Telecommunications are in permanent and constant change. The arrival of 5G brings new use cases to the table, wich drive the standarisation and where the efforts shall be directed. Also, Vertical Industries are demanding better performance in terms of network flexibility and computation power to Infrastructure Providers, and even vertical applications are requiring more complex scenarios (e.g., in Industrial, Immersive Media and Real Time Communication applications). Vertical industries do not only require a reliable, fast platform, but also to reduce on time-to-market and capability to manage Quality of Service (QoS) for certain applications. ETSI Network Functions Virtualisation (NFV) ISG is doing an effort to give solutions to these new problems aroused. In NFV nomenclature, these applications are abstracted by a concept called Network Service.
Some of these Network Services might be instantiated in complex scenarios, which must be connected across a Wide Area Networks (WAN), where recently Software Defined Networks (SDN) have been acquiring more importance, reducing Capex and Opex costs for WAN providers but also speeding up the process of network configuration, where Open and well-defined APIs are becoming a key to success. To solve the problem of instantiating services across a WAN, there exist the concept of WAN Infrastructure Manager (WIM), a component defined in ETSI NFV architecture and which currently Orchestration Platforms developers are bringing up some solutions with. WIM(s) communicates directly with the NFV Orchestrator, the main component governing a NFV Management and Orchestration Platform.
In this respect, this thesis has developed a solution within 5GPPP 5GTANGO project to extend SONATA NFV platform in order to be used to deploy network services over multiple Virtualisation Infrastructure Managers (VIM), which are interconnected through one or multiple WIM(s), using ONF Transport API (T-API) definition as WIM Southbound API Interface. Moreover, these instantiation has been validated and it has been analyzed the benefit of using a Point Of Presence (PoP) closer located at the 'edge' for instantiating a QoS reliant Network Service. T-API has been also extended in order to be used in application connectivity services, allowing an extension of what is called in SDN 'flow matching'.
Thus, the main output of this thesis is going to be included in new SONATA Service Platform release 5.0, and furthermore is going to have a critical importance to give functionality to one of 5GTANGO project pilots.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank Ricard Vilalta for his tireless support in doing this work and suggestions on how to best proceed to get over some stalled points, Salvatore Spadaro for being the supervisor of this thesis and always being so responsive, my work colleagues who gave me precious clues, all the partners of the 5GTANGO project who have been (and still are) working very hard to deliver a good output at the end of the project, my family and especially my parents Juan Luis and Rosa, and finally to all my friends and especially Daniel, who has been a great support week after week throughout the master.

Juan Luis de la Cruz Cuevas

INTRODUCTION

Software Defined Networks (SDN) has introduced high expectations in transforming telecom services. Because of that, software is becoming more and more important, both in controlling networks and their associated services, as well as for the actual execution of these service components.

These characteristics are essential to support many aspects of the expected functionality of 5th Generation (5G) networks. To this end, it is also necessary to provide a flexible Network Function Virtualisation (NFV) service platform, in order to support applications and services that quickly adapt to network tenants (vertical industries) necessities, while maintaining performance in terms of bandwidth and latency, and other capabilities such as fault recovery (self-healing networks) or Zero-touch configuration.

The interaction of SDN and NFV presents several research and implementation challenges, some of which will be analysed in this master thesis. Moreover, the multi-site Network Service (NS) instantiation across a Wide Area Network (WAN) is a 5G use case that has been discussed but there is no general and straightforward solution. This thesis tries to do an approach based in the use of a SDN orchestrator to find a solution to this challenge.

This master thesis has been performed during my duties as Research Assistant at Optical Network and Systems Department under the 5GPPP H2020 project 5GTANGO. 5GTANGO project provides an NFV Development & Operations (DevOps) network service architecture, and consist of three main blocks: SDK, Validation and Verification platform and Service Platform. This master thesis is mostly related to necessary extensions to Service Platform in order to support the requirements of a vertical industry such as our communication suite pilot.

Objectives

The main objectives of this work are:

- To deploy and adapt an infrastructure to support a service platform. Previously, ADRENALINE Testbed has been used as a topology with a single network branch, therefore there wasn't the possibility of use a end to end path without going through the optical domain. This proposed approach adds the possibility of using one pure packet switched path versus another hybrid (electro-optical-electrical) path

- Validate the capacity of the service platform to deploy NSs and Virtualised Network Function (VNF)s and the correct operation of the applications running on top of it

- Install and automate the deployment of a communication suite on the aforementioned infrastructure, taking into consideration specific requirements

- Experimental evaluation of benefits of communication suite placement

This document is structured as follows: Chapter 1 gives a high level glance about current efforts being developed in the NFV and SDN worlds, and also how they are highly related and depending within each other. Chapter 2 proposes the challenges that the Communications Pilot of 5GTANGO project proposes and which this work has analyzed. Chapter 3 explains and describes extensively the process of design and development that has been conducted. Then, the results obtained from different tests carried out to validate functionality of the service are evaluated in Chapter 4 gives a view of so. Finally, the main conclusions of this work and further possible future research guidelines are presented.

Publications

Some publications have been generated counting with contributions from this master thesis:

- Deliverables
  - 5GTANGO WP7 team, *D7.2 – Implementation of pilots and first evaluation*, February 2019.
  - 5GTANGO WP5 team, *D5.2 – Service Platform Final Release*, June 2019.
- Conference papers
  - Pol Alemany, Ricard Vilalta, Juan Luis de la Cruz, Ana Pol, Antón Román, Ramon Casellas, Ricardo Martínez, Raul Muñoz, *Experimental Validation of Network Slicing Management for Vertical Applications on Multimedia Real-Time Communications over a Packet/Optical Network*, ICTON 2019.
  - Ricard Vilalta, Juan Luis de la Cruz, Pol Alemany, Ramon Casellas, Ricardo Martínez, Abubakar Muqaddas, Reza Nejabati, Dimitra Simeonidou, *End-to-end network service deployment over multiple VIMs using a disaggregated transport optical network*, ICTON 2019.
  - Ana Pol, Anton Roman, Panagiotis Trakadas, Panagiotis Karkazis, Evgenia Kapassa, Marios Touloupou, Dimosthenis Kyriazis, Juan L. de la Cruz, Pol Alemany, Ricard Vilalta, Raul Munoz, *Advanced NFV Features Applied to Multimedia Real-Time Communications Use Case*, submitted to 2019 IEEE 5G World Forum (5GWF) - Track 5: 5G Trials, Experimental Results and Deployment Scenarios

CHAPTER 1. STATE OF THE ART

In this chapter we present some insights about the state of the art related to this Master Thesis. Section 1.1 gives an overview about how are NFV concepts being standardized, some of them used extensively in this thesis and later on evaluated. Section 1.2 introduces SDN and some concepts such ABNO and ONF Transport API which are very relevant within this thesis work.

1.1. Network Functions Virtualisation

1.1.1. ETSI Network Functions Virtualisation Industry Specification Group

The European Telecommunications Standards Institute (ETSI) is an independent and non-profit organization, whose main purpose is standardization in the field of telecommunications, and whose standards in information and communications technology (ICT) are globally adopted. Moreover, ETSI NFV is the most relevant standardization initiative arisen in the network function virtualization area. It was incepted at the end of 2012 by a group of top telecommunication operators, and has expanded rapidly to incorporate other operators, network vendors, ICT vendors and service providers. Up until now, the ETSI NFV industry specification group (ISG) can count on over 74 member companies plus 36 participants [2] and more than 100 publications. It represents a significant case of joint collaboration among heterogeneous and complementary kinds of expertise, in order to seek a common base for the multiple challenges related to NFV towards a solution as open and scalable as possible.

The ETSI NFV roadmap initially foresaw two major phases. First phase started at 2013 and finished at the end of 2014, when several specification documents were released conforming the foundations of the ETSI NFV specification [1], covering functional specification, data models, proof-of-concept (PoC) description, etc. During the first phase, the initial focus was:

- To drive convergence on network operator requirements for NFV
- To include applicable standards, where they already exist, into industry services and products
- To develop new technical requirements with the goal of stimulating innovation and creating an open ecosystem of vendors

The second phase (NFV RELEASE TWO), comprising years 2015 and 2016 released a new version of the ETSI NFV specification documents, addressing requirements which arised after the first phase and doing further developing in aspects such as Management of virtualized resources, lifecycle management of VNFs and NSs, package and software image management and modeling of the resources.

The third phase spanned during 2017 and 2018, progressing the work on architectural and evolutionary aspects, in some cases coming from vertical industries demands, such as

Information modelling, End-to-end multi-site services Management, Acceleration technologies, Accounting, Licensing, Security, Reliability, DevOps and Continuous Integration (CI), Testing and Policies, among others.

The fourth phase's Kick Off was recently during NFV#26 ETSI meeting [3]. Release 4 is going to focus in the following technical objectives:

- Enhance the Network Function Virtualisation Infrastructure (NFVI) for supporting modern lightweight virtualization technologies, reduce VNF-to-infrastructure coupling and optimize network fabric design
- Enhance automation, lifecycle management and orchestration
- Evolve Management and Orchestration (MANO) framework to optimize resource usage
- Improve NFV operational aspects, making easier to develop and to deploy it

The work of the ISG is further articulated into dedicated Working Group (WG)s. In phase 1, three WGs have been created, dealing with NFVI, MANO and Software Architectures (SWA). In phase 2, two additional WG were formed, Interfaces and Architecture (IFA) and Evolution and Ecosystem (EVE). In phase 3, even more new WGs has arised, regarding topics such as NS testing, security in NFV and others, and also new problems has been identified.

The currently acting specification of the ETSI NFV architecture was finalized in December 2014, and its high-level picture is shown in Figure 1.1.The ETSI NFV specification defines the functional characteristics of each module, their respective interfaces, and the underlying data model. The data model is basically made up by static and dynamic descriptors for both VNFs and NSs. More extensive definitions can be found below in Subsection 1.1.1.1.

The ETSI NFV framework specifies the architectural characteristics common to all the VNFs. It does, though, not rule out which specific network functions can or should be virtualised, leaving this decision up to the network function provider - apart from the use cases advised for the PoC.

1.1.1.1. ETSI Definitions

In this subsection there are some ETSI definitions with significant relevance in NFV and also related to the developed work. These definitions might serve as a reference guide when reaching to the following chapters:

A **Network Service** (NS) [4] is a composition of Network Function (NF)s which together act as a unique element, which connections between them can be specified in a VNF Forwarding Graph (VNFFG). A NS is defined in a file called Network Service Descriptor (NSD). Those Network Functions can be Physical Network Functions, or Virtual Network Functions. The Network Service contributes to the behaviour of the higher layer service, which is characterized by at least performance, dependability, and security specifications. The end-to-end network service behaviour is the result of the combination of the individual network function behaviours as well as the behaviours of the network infrastructure composition mechanism. Also, different Deployment Flavour (DF) can be included in a NSD to vary between differents levels of service, link capability requirements, or scalability for example. Also, a NSD includes Virtual Link Descriptor (VLD) to be used by the MANO to deploy Virtual Link (VL)s.

A **Virtual Network Function** (VNF) [4] is an implementation of a NF that can be deployed on a NFVI. A NF is functional block within a network infrastructure that has well-defined external interfaces and well-defined functional behaviour. VNFs run on top of the hardware networking infrastructure, which can include routers, switches, servers, cloud computing systems and more. A VNF is or could be conformed by one or more Virtual Deployment Unit (VDU)s, external connection points which are used to connect the VNF to others or to be accesed from the outside, and an internal VLD to define how VDUs are connected inside the VNF if it is composed by more than one VDU.

A **Virtual Link** (VL) [5, 4] is a set of connection points along with the connectivity relationship between them and any associated target performance metrics (e.g. bandwidth, latency, Quality of Service (QoS)) defined in a component called VLD that is included in the NSD to define connections between VNFs or inside a Virtualised Network Function Descriptor (VNFD) to define connections between VDUs.

The **Network Function Virtualisation Orchestration (NFVO)** is the functional block that takes care of the NS lifecycle and coordinates the management of NS lifecycle, VNF lifecycle which is further supported by the VNF Manager (VNF-M) and NFVI resources which are supported by the Virtualised Infrastructure Manager (VIM) ensuring an optimised allocation of the necessary resources and connectivity between them.

The **VNF-Manager** (VNF-M) is the functional block that is responsible for the lifecycle management of VNF, which includes instantiation, scaling, updating or upgrading, and termination of VNFs.

The **Virtualised Infrastructure Manager** (VIM) functional block is responsible for controlling and managing the NFVI compute, storage and network resources, usually within one operator´s Infrastructure Domain (e.g. NFVI/Point of Presence (PoP)).

The **WAN Infrastructure Manager (WIM)** functional block is responsible for controlling and managing the WAN network resources, cooperating within MANO framework in the lifecycle of VLs. WIMs provide connectivity services between the NFVI-PoPs and connectivity to Physical Network Functions. This functional block may be added as a specialized VIM, in particular for new greenfield virtualised deployments. Alternatively, a WIM can also be an existing component of the Operations and Business Support System (OSS/BSS) functional block. The WIMs are responsible for the following aspects related to the NFVI connectivity services:

- Path computation based on quality assurance factors such as jitter, Round Trip Time (RTT), delay & bandwidth arrangement.
- Establish connectivity over the physical network (e.g. set of Multiprotocol Label Switching (MPLS) tunnels).
- Provide a Northbound Interface (NBI) to the higher layers, e.g. NFVO, to provide connectivity services between NFVI-PoPs or to physical network functions.

- Invoke the underlying NFVI network Southbound Interface (SBI), whether they are Network Controllers or Network Functions, to construct the service within the domain.

1.1.1.2. ETSI Architecture

In general NFV ETSI can be summarized in Figure 1.1, which provide us with the position of each block inside MANO and asociated infrastructure, hierarchy, interaction between functional blocks and which are the names of each block and interfaces between them. On top of everything we have OSS/BSS which is the classical concept of operations and business support system. This management block communicates directly through Os-Ma-Nfvo interface with NFVO. Hanging from NFVO there exist catalogs for each available component ready to be deployed, and also current instances deployed in any of the registered NFVIs.

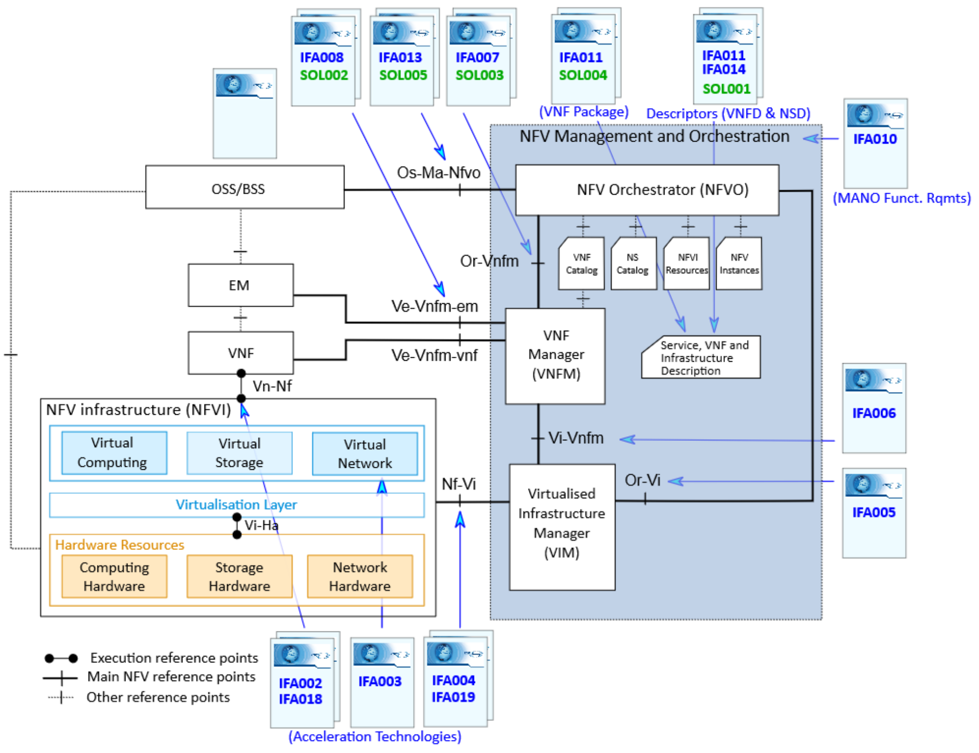


Figure 1.1: ETSI NFV architecture [1]

NFVI is directly controlled by VIM via Nf-Vi interface [6, 7]. There are a large amount of NFVI information that is or can be exposed to VIM so placement and lifecycle can be performed in the best way possible. NFVI has a virtualisation layer that in a way simplifies it physical resource database to make it easier for the MANO to control and manage.

1.1.1.3. Inter VIM connection

The same NFVI can have multiple PoPs managed by the same VIM, e.g. multiple compute nodes. Management of the resources is centralised and easier for the VIM to perform lifecycle operations on the NS or on the VNF without going up to the NFVO. When NFVO has to request a scale-out operation for a NS that is currently hosted in a NFVI which cannot host more VNFs, so a possibility, instead of migrate the whole NS to a new NFVI would be to expand the NS to another NFVI-PoP. This flexible NS has a great advantage from the point of view of Capital Expenditures (CAPEX) and Operation Expenditures (OPEX) when there is a rapid variability on traffic. This operation is described in [8], and implies for the NFVO to request a NS expansion over WAN. The VL is created across the backhaul by the WIM at a higher level and at a lower level by one or more Network Controllers, that could be SDN or not, and each NFVI-PoP's VIM manages the infrastructure internal networking using for example Virtual Extensible LAN (VXLAN) connections, while connectivity between PoPs can use other technologies such as MPLS.

A number of functional requirements for NFVO in multi-site scenarios have been enumerated and described on [9], and also requirements for WIM block. A WIM can manage its resources by using three different models: Reservation, Quota and On demand. The connection performed through WAN network and how the information model is defined is further explained in [10], where we have two new concepts:

- Multi-Site Connectivity Service (MSCS)
- Multi-Site Network Connection (MSNC)

A MSCS is a group of one or more MSNC and abstracts it details. The consumer can request the creation of a MSCS but only the WIM can internally create new MSNC that together form that MSCS, allowing the connectivity between two or more edge points of the WAN, and it provides provisioning and monitoring of the created connection. WIM domain ranges from the external port of one customer edge (NFVO-PoP edge nodes connected to wan) to the other, per each MSCS deployed to enable VL between the different VNFs deployed.

Furthermore, WIM has 4 functional requirements: MSCS Management, capacity Management, Fault Management and performance management. MSCS management is related with Creation, Request Update and Delete (CRUD) operations Application Programming Interface (API) related with them, and also notifications about resource reservations that are not yet operational. Capacity Management is related to querying available and total amount of resources that a WIM can rely on so it can be available for NFVO placement block. Fault Management is associated with the collection of fault information about the MSCS, creation of alarms for monitoring, and notification when there is an alarm that meets the condition on which it depends. Finally, Performance Management is about collecting and reporting performance metrics related with MSCS, for monitoring. Some of these interfaces can use a Subscriber-Consumer pattern to implement some of the functional requirements.

1.1.2. Network Function Virtualisation Orchestration and Virtual Network Function Management

In NFV, we could say that one of the most important components among all, due to its functionalities, is the NFVO, component in a MANO framework which is performing resource orchestration. It can manage NFVI through other components like VIM, WIM and VNF-M, or it could be a whole block but that is not as usual as a block architecture. ETSI architecture defines the NFV MANO architecture using the blocks approach.

There are some NFVO solutions that have different approaches to provide a NFVO implementation, such as SONATA which the result of SONATA [11] and 5GTANGO [12] european projects, Open Source MANO (OSM) that is an effort sponsorized by ETSI and Open Network Automation Platform (ONAP) which is supported by the linux foundation.

1.1.2.1. SONATA

SONATA [11] project was a 5GPPP H2020 phase 1 project, which had as major outcome the development of open source SONATA NFV framework. It consisted in three main blocks, the SONATA SDK, SONATA DevOps and SONATA Service Platform (SP). It was finished in 2017, and after this, 5GTANGO took over to further develop this SP. Since then, its currently finishing the second year of the project, with promising results in terms of NFV development, and the remaining project span will be devoted to pilot development and use case fulfilling.

To an external *user*, meaning provider, developer, tester, 3rd party, etc, we could say that the most relevant component to them is the Gatekeeper. The Gatekeeper, as shown in Figure 1.2, is the component responsible to make public the SP's APIs to the outside, ensuring that every access to the SP is done by authenticated and authorized users, first filtering invalid requests before they go further processed by the relevant internal component (web micro-services). The Gatekeeper is also responsible for get Descriptors from the Catalogue, enforce licensing models (through the Service Level Agreement (SLA) Manager) and enrich the requests with other data.

The Gatekeeper interacts internally with the remaining components of the SP:

- The Catalogue whenever descriptors of packages, services, functions, slices, SLAs and Policies are requested
- The MANO Framework whenever a service instantiation, service instance termination or service instance scaling is requested
- The Repository whenever records for instantiated services and functions are requested
- The Slice Manager whenever a slice template is created or requested and a slice template instantiation or termination is requested
- The SLA Manager whenever an SLA is created or associated with a service
- The Policy Manager whenever a policy is created or requested

- The Infrastructure Abstraction (IA) whenever there's the need to configure it with VIMs and WIMs, and also to know which infrastructure is available, and create or delete networks between services of the same slice
- the Monitoring Manager whenever a developer wants to extract some monitoring data related to the services and functions he has developed and that were instantiated

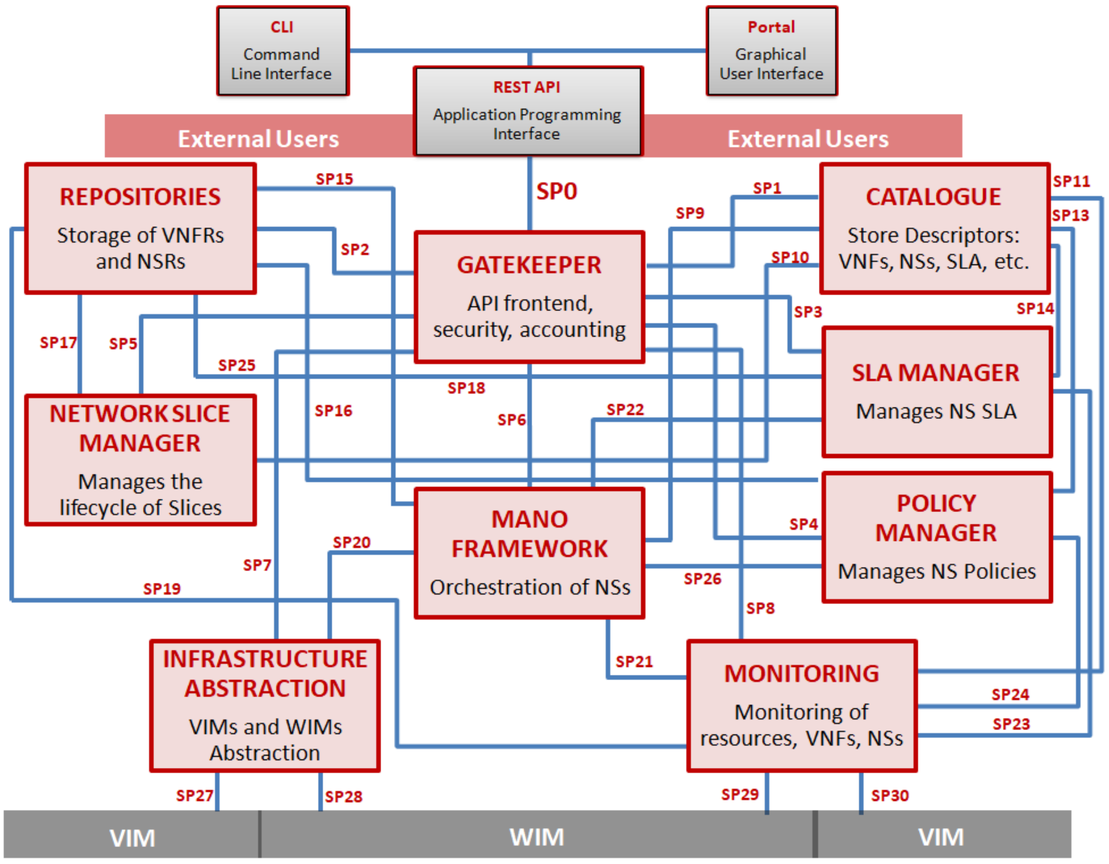


Figure 1.2: SONATA SP main components and internal interfaces

Another component, which relies on the Gatekeeper but is its user friendly interface, is the Portal, which is an Angular application responsible of showing not only information regarding every component in the SP but also giving the possibility of executing operations such as service and slice instantiation, SLA operations and attachment of new VIMs and WIMs among others. Release 5 is going to include a dashboard with useful monitoring data.

The usage of DevOps is one of the main features of SONATA, which is deployed easily using a combination of Ansible [13] Cookbooks and a Jenkins [14] automation server. Each different module runs inside a docker container, and all components are connected to the same virtual network bus, although not all components communicate with the rest of them. One of the main advantages from the developer point of view is that each component can use their own preferred language, (e.g. Java, Ruby, Python, Angular, R...) depending on the development team that has designed and developed the component.

Finally, it is worth mentioning that one of the efforts of this work has been extending IA, in order to allow SONATA SP to manage CTTC´s WIM, thus giving a solution to a multi-PoP through an SDN WAN.

1.1.2.2. OSM

OSM is an ETSI-supported effort to develop an Open Source NFV MANO software stack compliant with ETSI NFV. Novel features in last release (Release 6) include Network Slicing, support for Physical Network Function (PNF), policy and monitoring and support for VNF configuration. There are multiple VIM that can be registered coming from a variety of companies or foundations, such as Openstack or Open-Vim.

There are some design modules and runtime modules. From developer point of view, he can operate using a web Graphical User Interface (GUI) or a Command Line Interface (CLI). The OSM CLI client uses directly it´s NBI API to remotely interact with the platform. It also can be used as a python library, and develop using it. Design time components are those which provide DevOps support and also give some computer aid when creating a VNF or NS package.

Runtime modules are those which give functionality during lifecycle events or post-configuration tasks to OSM, are shown in Figure 1.3 and listed below:

- Unified Northbound Interface (NBI). The NBI component is providing the primary API endpoint into OSM
- Lightweight Life Cycle Manager (LCM). It is responsible for all aspects of service orchestration including lifecycle management or NS and network slices, and service primitive execution. It is effectively the 'master' orchestration component in the system. Also, it is responsible for supporting the concepts of multi-tenancy, projects, users, and enforcing role-based access controls
- Resource Orchestrator (RO) is responsible for managing and coordinating resource allocations across multiple distributed VIMs and multiple WIMs. Also, each VIM and WIM Plugins are responsible for connecting the Resource Orchestration Engine with the specific interface provided by the VIMs and SDN controllers
- VNF Construction and Abstraction (VCA) is responsible for enabling configurations, actions and notifications between VNFs and/or Element Managers
- Monitoring (MON) collects and also evaluates data retrieved both from VNFs and VIMs, and also can raise some alarms that trigger lifecycle related events
- Policy manager (POL) is a module where alarms triggered by monitoring events are defined
- A common Information Model (IM), which is common between much of the components in OSM. OSM, in fact, is based on a model-driven architecture.
- Databases: a Common NoSQL database, a Time-Series DB and an Object Storage

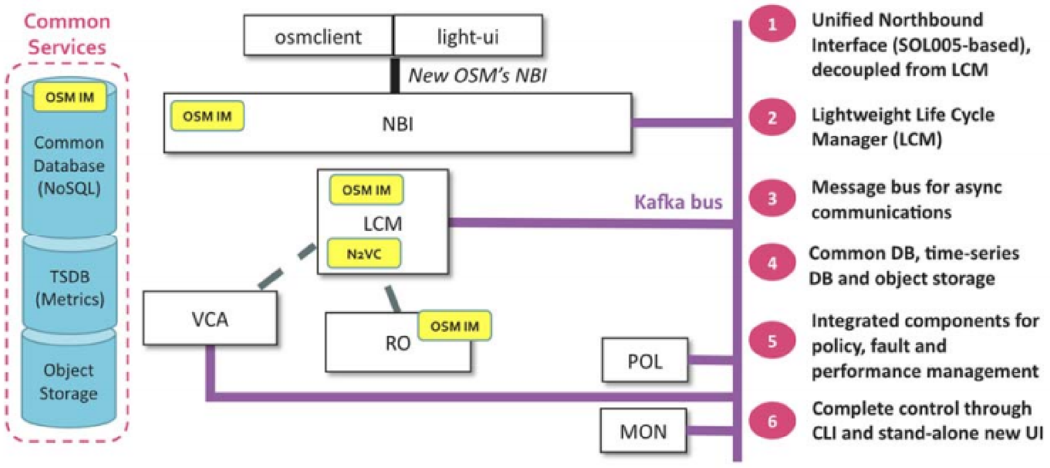

Common Services
OSM IM
Common Database (NoSQL)
TSDB (Metrics)
Object Storage
osmclient
light-ui
New OSM's NBI
OSM IM
NBI
OSM IM
LCM
N2VC
Kafka bus
VCA
OSM IM
RO
POL
MON
1 Unified Northbound Interface (SOL005-based), decoupled from LCM
2 Lightweight Life Cycle Manager (LCM)
3 Message bus for async communications
4 Common DB, time-series DB and object storage
5 Integrated components for policy, fault and performance management
6 Complete control through CLI and stand-alone new UI

Figure 1.3: OSM architecture diagram with different components

1.1.2.3. ONAP

ONAP is a Linux Foundation open source initiative with the support of AT&T and China Mobile among other companies. The ONAP project [15] addresses need for a common automation platform to deliver differentiated network services on demand, profitably and competitively, while leveraging existing investments. Figure 1.4 shows the deployment of a 5G network using ONAP. ONAP support container-based network functions, lighter than usual VNFs so the deployment time and resource usage becomes reduced.

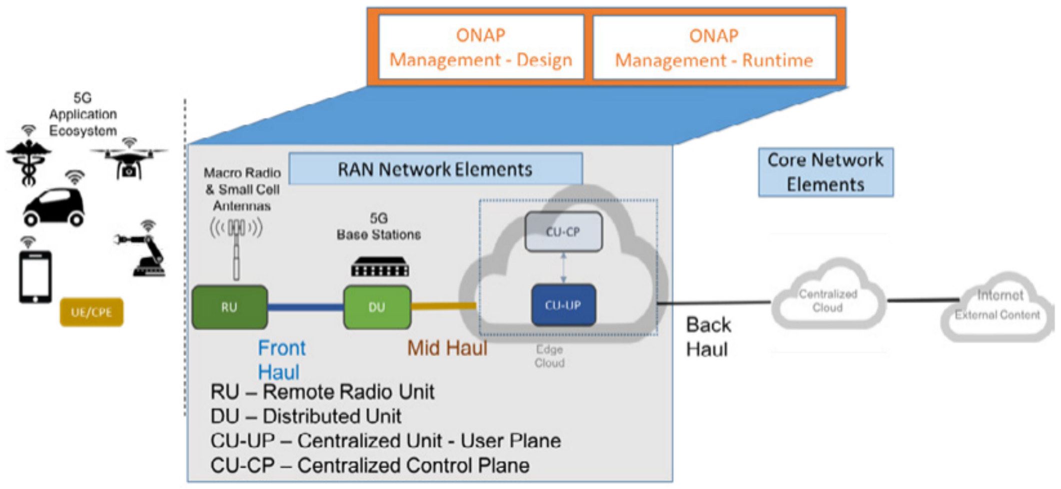


Figure 1.4: ONAP 5G Use Case examples

1.1.3. Virtualised Infrastructure Management

Virtualization is the first technology enabler of Cloud Computing. Server virtualization allows to share physical server resources (CPU, RAM, storage, ...) over different virtual instances. Server hypervisors are software appliances which can run over bare-metal hardware or over operating systems.

However, cloud systems are moving to a layer of software above the hypervisor so-called cloud orchestrator system software. While hypervisors are able to provide resource abstraction in a single server, cloud systems can abstract large pools of compute, storage and network resources over the same software platform, thus realizing the concept of IaaS described before. This resources can be managed through a set of APIs to provide a holistic view of an infrastructure. A cloud provider can manage its entire datacenter (DC) over this cloud orchestration system to offer different cloud services to different tenants. Among the cloud orchestration software available, OpenNebula, CloudStack and OpenStack, are found as the most complete and mature solutions in the open source domain.

OpenStack [16] is the leading open source cloud orchestration platform used within DCs operated by cloud service providers and large enterprises alike. Development is supported by a broad base of developers and a growing cohort of commercial software and hardware vendors.

OpenStack is an open source project carried out by the OpenStack Foundation whose purpose is to provide a rich software platform which be able to provide the creation of cloud applications with many virtual instances controlling distributed storage among different servers in a clustered environment. The framework has an extensive amount of modules, following a micro-services architecture as we can see in Figure 1.5. Some of them are *Nova*, *Glance*, *Keystone*, *Ironic* and *Neutron*; which are responsible of the orchestration of different resources: Computing, Storage, Identity management, Bare-metal support and Network resources, respectively. Other modules, such as *Heat*, can be used to deploy a series of related VNFs, connected between them, conforming in the end a NS.

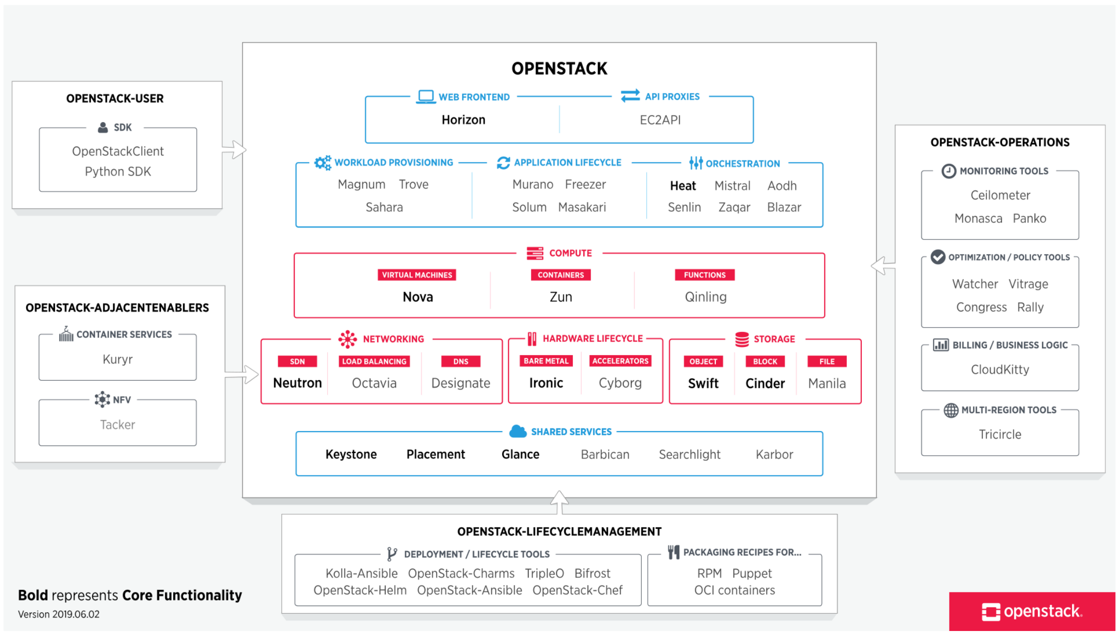


Figure 1.5: Openstack components

1.2. SDN-based Network Orchestration

Software Defined Networks (SDN) are becoming an esential part on how not only operators can manage their transport networks, but how functional blocks are connected in a DC. It also becames a principal actor when it comes to NFV enabler. A large number of VIM and WIM blocks uses SDN for enabling connections between different PoPs. The standard, beign defined by Open Networking Foundation (ONF), decouples the data plane from control plane, and also supports the programmability of how connections are created. That allows that a NFVO can use it to perform the so-called 'orchestration'.

1.2.1. Application-Based Network Orchestrator

When deploying services accross different network domains through per-domain SDN or active stateful Path Computation Element (PCE) controllers, it is needed an entity to support such end-to-end connectivity by orchestrating. This component is called Multi-Domain SDN Orchestrator (MSO) and it is depicted in Figure 1.6. The MSO considers the heterogeneous underlying network resources (e.g., multi-domain, multi-layer and multi-control network resources) and it assumes that the underlying SDN controllers can to provide network topology information and flow programming functions. The MSO enables multi-layer and multi-domain network orchestration and it has to implement different plugins for each SDN Controller, which are typically technology and vendor dependent since there is not an standarized one.

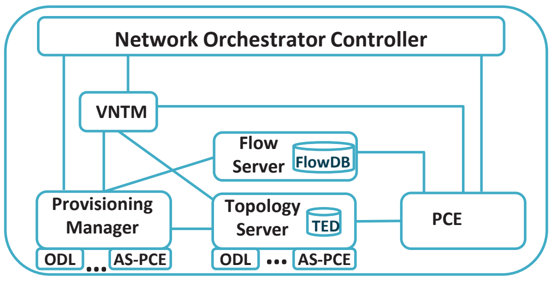


Figure 1.6: Multi-domain SDN Orchestrator architecture

The MSO architecture is composed by the following components: Network Orchestration Controller, Virtual Network Topology Manager (VNTM), Topology Server, Provisioning Manager and PCE. This architecture is based on the Application Based Network Orchestrator (ABNO) framework [17], which has been standardized by Internet Engineering Task Force (IETF). Following, we briefly describe the main role of the components:

- The Network Orchestration Controller is in charge of handling all the processes involved and to provision end-to-end connectivity services. It also exposes a NBI to offer its services to external applications
- The Topology Server is the component responsible for arrange the network topology, collecting each sub-topology components from each different domain and building the whole network topology which is stored in the Traffic Engineering Database (TED). The TED includes all the information about network links and

nodes, and is used by a dedicated PCE for calculating routes across the network. Moreover, it requires to have some inventory information for some operations

- VNTM is responsible for the multi-layer management. In the proposed architecture, the VNTM arranges the establishment of an optical connection, which is then offered it as a logical link to the upper layer to satisfy any incoming connectivity demand. Also takes care of Network Abstraction in a multi-hierarchical scheme
- The Provisioning Manager implements the different provisioning interfaces to command the forwarding rules and the establishment of connectivity segments into the data plane. An internal database stores each connection, and every network connectivity service along with the connections constructing it can be accesed using NBI API
- The Operations, Administration and management (OAM) takes care of notifications and events, consumed or produced from/to NBI or SBI, such as link failures or node additions, bringing the possibility of self-healing

1.2.2. ONF Transport API

Nowadays, the capacity to orchestrate multiple technologies and domains is key. As it was said before, usually SDN Controllers offer proprietary interfaces or, at best, open yet no standardized interfaces, to applications. These interfaces can be used by higher level controllers or other functional layers such as Network Orchestrators. Such SDN Controllers are arranged following an approach commonly referred to as 'vendor domains or islands'. This heterogeneity, due to having different controllers interfaces in a multi-domain context, forces the use of 'plugins' and it is difficult and costly to implement. As a driving motivation and clear problem statement, there is a need for a standard interface, with common models, to act as a controller NBI.

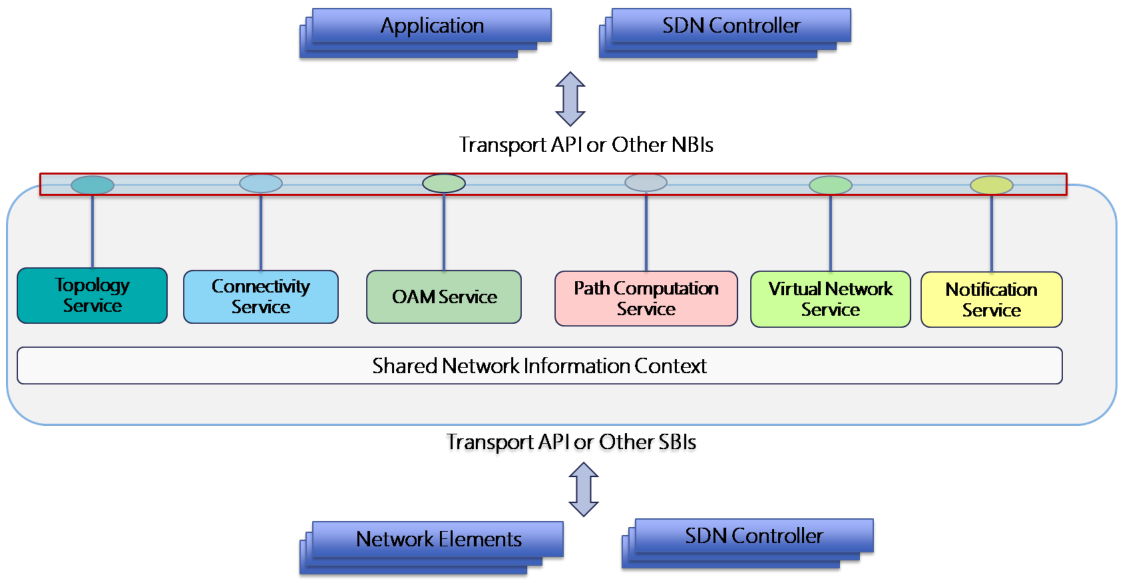


Figure 1.7: T-API high level functional architecture and interfaces

The Transport API (Transport Application Programming Interface (T-API)) [18] published by the ONF meets the main requirements to be a protocol and interface used between an orchestrator and multiple domain controllers. Figure 1.7 shows how a T-API based controller offers multiple services, being the more importants the topology and connectivity services. The services are modelled in the YANG modelling language.

The T-API context is the shared information between a T-API client and the T-API server, which controls a SDN network. A Service Interface Points (SIP) is a pair of a network node and a port of that node (an interface) where a service can be relied to a client. The model defines a T-API domain as being able to provide services between different SIP mainly characterized by their universally unique identifier (UUID). Most basic operation for a client is to 'retrieve' the context in order to obtain the list of SIPs, so connectivity services are established between two (or more) exported SIPs.

If a given T-API server supports topology model, it augments the T-API shared context with a list of topologyies. Each topology is composed of a list of nodes, which, in turn, have Node Edge Points (NEP). Links connect two NEPs. The model is flexible enough to support recursive topologies and different levels of abstraction. The level of detail exported is configurable by policy. A client is then able to obtain an more simplified graph of the topology and map T-API SIPs to external NEPs.

Finally, the instantiation of a Connectivity Service relies on the instantiation of several connections (e.g. one end-to-end and internal at each T-API node). To this purpose, Connection End Points (CEP)s are instantiated over NEPs (and contain information about the connections) and connections involve two or more CEPs. T-API will be used as a key interface between network controllers associated to different domains, allowing end-to-end transport connectivity services.

CHAPTER 2. END-TO-END DEPLOYMENT OF REAL TIME COMMUNICATIONS SERVICE: USE CASES DEFINITION

In this chapter we present how the problem was designed in early phases of 5GTANGO. How use cases drive us to solve different challenges is described in Section 2.1, and Section 2.2 further comments about how QoS enforcement is necesary in the pilot.

In the context of 5GTANGO project Work Package (WP) seven, there are 3 different pilots, all of them related with vertical output to industry, in basis of different usecases defined for each technology. Those pilots are:

- Smart Manufacturing Pilot is an effort for bringing new 5G technologies to old manufacturing industry processes
- Immersive Media Pilot is an approach to Augmented Reality, where a user can experience the watching of a video with a dashboard with social networks or live data
- Communications Pilot deploys a video-conference service that can host up to a high number of users using the same NS, with different levels of service depending on the license purchased by each client

This Master Thesis focuses on several aspects of the Communications Pilot. There are four different use case deployments, each one posing a different problem which have been considered for this pilot to cover, all of them further explained in the chapter 2 of deliverable 7.2 [19] of 5GTANGO project. These use cases are a Best Effort (BE) video-conference service, a premium video-conference service, an edge video-conference service and finally a video-conference service that offers the possibility of scaling when required.

2.1. Communications Pilot Use Cases

2.1.1. Centralized video-conference services

The first use case is BE video-conference, which does not takes advantage of QoS management, as shown in Figure 2.1. This use case can provide service to users that are connected to an access node, being the NS instantiated in the core, across an SDN fabric. When the service is instantiated, is necesary that the provider configure correctly which are the client border endpoints, for the placement and the WIM wrapper modules in the SP side and the WIM itself perform provisionining operations on the network to have it connected in a proper way. In the experimental deployment, up to four border nodes are available, meaning they are are ready to be used by this service.

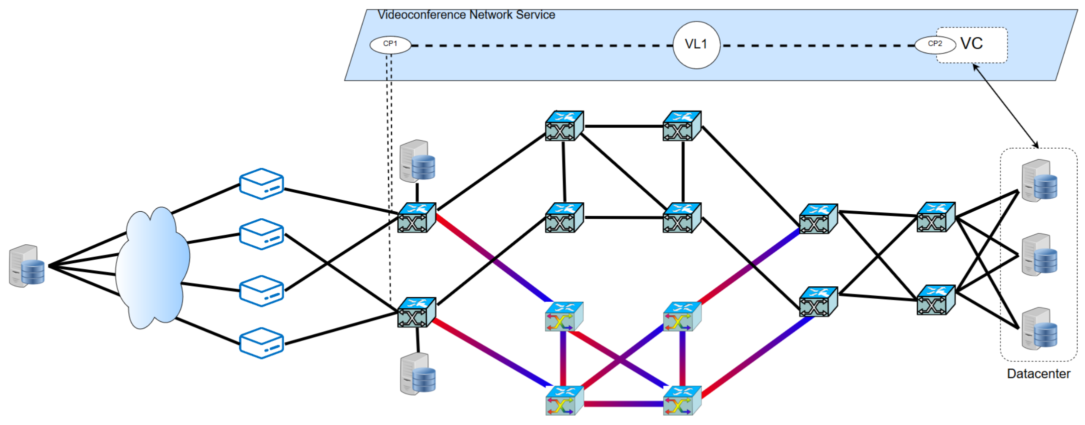


Figure 2.1: Use Case 1: deployment of one centralized service

2.1.2. Premium and best-effort QoS video-conference services

The second use case is similar to the previous one, but with the difference of using QoS enforcement if required, and is ilustrated in Figure 2.2. QoS can be parametrized by some conditions required by the client, and collected by a license and a SLA asociated to the service instantiation: a certain bandwidth needed for that service, or by a certain low-latency conditions, both of them can be critical in case of a high quality video streaming or other real time applications. In addition, other relevant QoS parameters are monitored by the SP, such as jitter, users connected, packet loss measured by the application, appart from those which are also monitored by WIM. These parameters can raise alarms at the SP due to SLA violation, and eventually trigger some actions regarding VNF life cycle such as scaling and VNF migration, but are not used in this use case.

Besides, this use case uses DF, following ETSI NFV SOL001 [20], to define different configurations within the same NS and have a different performance or behaviour profiles depending on the usage intended or SLA chosen by the provider to deploy it, and allowing variable resource assignation.

2.1.3. Edge video-conference services

As seen above, in some scenarios it may be required to deploy video-conference services closer to the end users in order to reduce the End-to-End (E2E) delay, since it may exceed the maximum desired threshold and the system may not fulfil customer SLAs. Under this assumption, in the third use case two scenarios (see Figure 2.3) are considered. In the first scenario, we consider that all the video-conference NSs are in an edge NFVI-PoP(e.g. small-DCs in central offices located in the aggregation/metro network).

In a second scenario, we consider that instead of deploying the whole video-conference NSs on an edge NFVI-PoP, we shall deploy only those NSs that are delay-sensitive for a proper operation and those which are not constrained remain in core-DC. In this case, the design of the NS must be defined to somehow tag those VNFs that are constraint to be deployed in the edge NFVI-PoP, and SP must give the posibility of having a multi-site

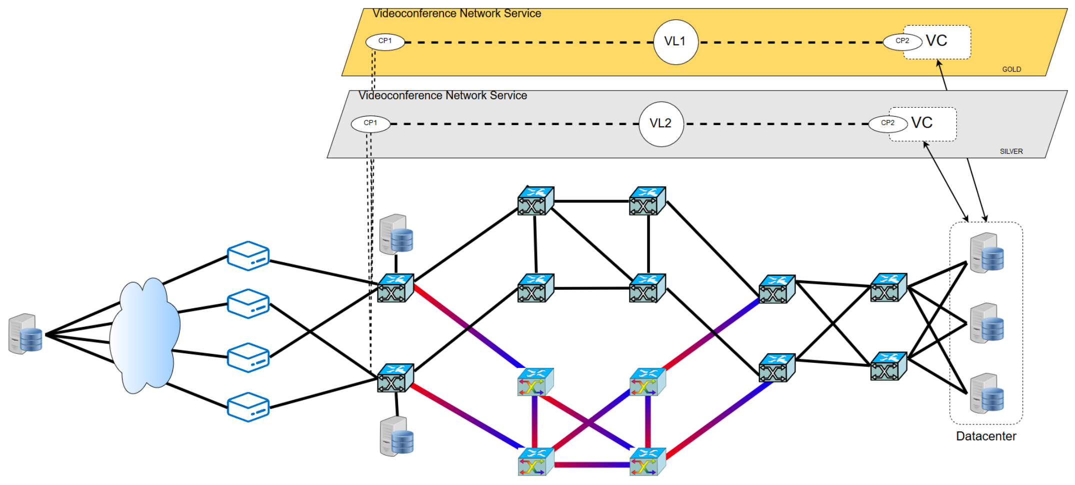

Videoconference Network Service
CP1
VL1
CP2
VC
GOLD
Videoconference Network Service
CP1
VL2
CP2
VC
SILVER
Datacenter

Figure 2.2: Use Case 2: deployment of two networks services, Gold and Silver

connection.

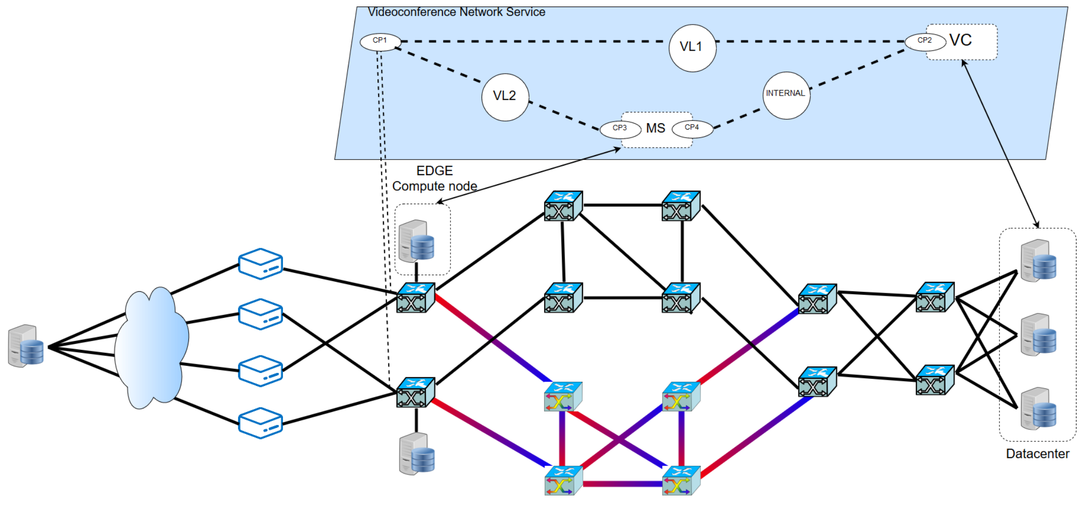


Figure 2.3: Use Case 3: deployment of VNFs closer to the edge

2.1.4. Video-conference scaling out/in services

Finally, the fourth use case is related with the introduction of dynamic scaling, which is add or remove strategicaly critical VNF replicas, that without this functionality would be the cause of a bottleneck in terms of performance. In the video-conference NS there are two VNFs that can be scaled in/out. These VNFs are the VNF-WAC and VNF-MS.

A monitoring service will check registered and connected users (from VNF-WAC), and also concurrent number of conferences being executed and bandwidth usage(from VNF-MS), and when certain conditions are reached, to keep Key Performance Indicators

(KPIs) under control, the scale in/out operation is performed by the SP without the user becoming aware of the process. This use case infrastructure is similar as the previous ones.

2.2. Communications Pilot and QoS enforcement

The ability to provide differentiated qualities of service is an old objective in network engineering. Different QoS levels are required for different customers (who subscribe different packages with different prices) or to accommodate the requirements for different services. Services like voice, video or more recently Virtual/Augmented Reality services have more strict requirements.

QoS parameters in NFV can be specified in a flexible manner either on the VLs or in connection points of the VNFs attached to them, referenced in NSD and/or VNFD. When the QoS is defined on connection points scope, it applies to the outgoing traffic (unidirectional QoS), while when the QoS is defined over the VLs, it applies to all connection points of that network, also for the outgoing traffic (bidirectional QoS).

The QoS parameters in descriptors (NSD and VNFD) can specify VL that later on will be provisioned 'internally' in a PoP by a VIM or provisioned 'externally' in a WAN, by a WIM. The scope of this project is the 'external' extended VLs, where the WIM performs the QoS enforcement.

This master thesis focuses only on use cases one and two, which refers to a BE video-conference service and a service that can provide QoS differentiating between premium and non-premium users.

CHAPTER 3. EXTENDED NFV ARCHITECTURE

In this chapter, we describe a number of tasks developed in order to give support to the pilot. Section 3.1 describes how NFVO platform has been modified adding the needed functionalities in other components not including the new WIM wrapper, while Section 3.2 gives an in-depth description of the new component developed, giving the capacity to perform placement across a SDN WAN. Section 3.3 dives into the video-conference service descriptors and components, and finally Section 3.4 details how are the workflows that take place when instantiating a network service with the new functionalities.

In the previous chapter, we presented several use cases stating several problems not yet resolved to which solutions have to be implemented, to achieve the objectives and expectations of 5GTANGO and more specifically the pilot.

Figure 3.1 shows a general view of key points that have been developed under this thesis effort. On the left side, we have two nodes named hereafter Client1 and Client2, which are outside the elements controlled by the WIM, although they are connected to the WAN so they can be provisioned by using it. The video-conference service (VC) NS is instantiated in the Core-DC PoP (in the right side of the figure), and external VL means an abstracted connection from ingress connection point (CP1) mapped to clients, with egress connection point (CP2) mapped with the NS. VL is created across the WAN, a network connection performed by the WIM functional block. Depending on QoS requirements, VL underlay will be optical-electrical for a more restrictive QoS or on the contrary it will be routed through only-electrical nodes. Thus, all elements are put together to work in order to achieve an operational service. Details of each component are further explained below.

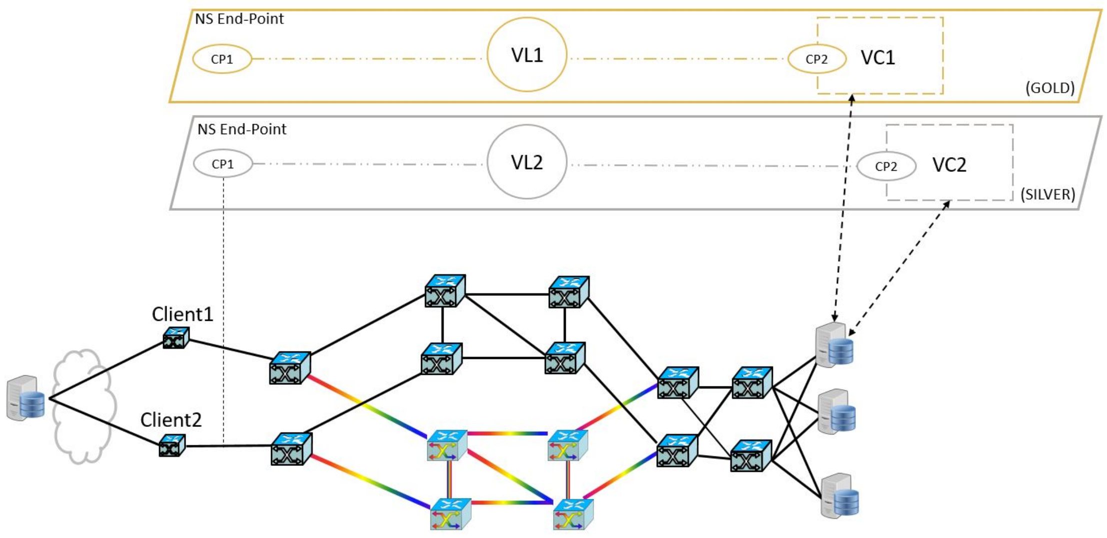


Figure 3.1: NS deployment with two network services, Gold and Silver and two shared clients each

3.1. Extensions to NFV Orchestrator

Currently there are two main deployment cases where a WIM wrapper can be used, subject of the actual WAN technology and management:

- Case 1: Programmatic Network Control (i.e. SDN based) This case is considered as the primary target for a WIM Wrapper, aligned with the view of ETSI NFV. Underlying networking infrastructure is controlled by Network Controllers that support SDN API interfaces, thus the WIM interfaces with those Network Controllers in order to provide multi-tenancy, programmability and isolation summarised as Network as a Service (NaaS).
- Case 2: Non-programmatic Network Control, controlled using 'legacy' Network Management. This case is considered where actual programmatic interfaces are not available (e.g. NETCONF) and where casual Network Management Systems are used for the resource management and operation of the WAN infrastructures. In this case it is expected that the service requests will be signalled through the OSS/BSS system and not directly through the WIM.

Also, it is convenient to say that usually, in a real scenario, the infrastructure is not purely 'Case 1' or 'Case 2' but something between them, islands of SDN among a sea of legacy networks or viceversa. It´s important to consider any 'in-the-middle' scenario.

TANGO´s NFVO, has been extended to support WAN management through a T-API interface. The new component is integrated in the IA module, which already had a Virtual Tenant Network (VTN) wrapper to interact with a WAN controlled by a single OpenDaylight (ODL) controller.

This solution is quite limited, since ODL is a pure SDN controller, thus it cannot operate over other kind of technologies, such as GMPLS/PCE optical networks, or more complex transport WANs. Moreover, this solution is not scalable, since one ODL controller will start to drop on performance (latency, control interface saturation) when controlled nodes start to rise, not mentioning a geographically distributed infrastructure where conditions are even worse.

As a result of previously mentioned problems, and new use cases planned to develop for communications pilot (See Chapter 2), the need of a new wrapper arises, one that can give support to an a distributed NS through a transport WAN. This can be done by NS extension, as it is described in [8, 10].

From a high level point of view, this extension to NFVO is the provision of a NaaS, but limited. WIM topology service is not fully available for SP operator, for this reason, there exists the possibility of using T-API NBI and the wrapper can create VLs between VNFs located in different NFVIs (multi-site NS) or create services on instantiation time to give customers E2E connectivity and reach the service external interface endpoint. How this connectivity services function and are designed is further explained in Subsection 3.2.1.

These VLs are referenced both from VNFDs and NSD corresponding to a service, in SONATA SP they can be grouped in three types: *E-Line*, *E-Tree* and *E-LAN*. These three connectivity types are named after MEF Carrier Ethernet specification[1], and summarizing:

[1]http://www.mef.net/mef-3-0-service-technology-certification

- *E-Line* is a point-to-point service
- *E-Tree* is a point-to-multipoint ethernet connection
- *E-LAN* is a multipoint-to-multipoint connection, behaviouring equally than an Ethernet HUB

SONATA MANO calculates placement in basis of those three kind of VLs, so it´s important to have a well defined VNF topology, to keep the minimal number of logical connections. That is key to improve performance (reducing overall traffic), and avoid possible security flaws, keeping out of the picture undesirable ‘doors’.

VL terminations are called connections points (*connection_points* field in the VNFD), a list in the VNFD with each interface that the corresponding VNF has. These connection points can be of three types: *internal*, *external* and *management*. *Internal* connection points are for connections between VNFs or VDUs within the same NS, *external* connection points are those which are exposed to access from outside, through dataplane and *management* is the interface where corresponding Function Specific Manager (FSM) accesses to perform configuration modifications. Having at least one *management* connection point is mandatory, while external and internal interfaces are not, but there is not a limit of how many connection points can have a VNF/VDU. After all, each connection point is mapped to a virtual interface inside the VNF image, when it is instantiated.

When using DFs, each flavour can vary on the number of VNF used (e.g. a High Availability flavour can have two external interfaces instead of one) and also have different QoS parameters. Supported QoS parameters are Bandwidth limit, Minimum Bandwidth, Maximum Latency and Differentiated Services Code Point (DSCP) Marking.

After every bit of information is mapped from the Yaml model by MANO, while on placement phase, information from VIMs and WIMs attached to the SP. In Communications Pilot, there are two Openstack deployments, thus only *Heat* wrapper takes action, while on the networking part there is an ABNO/T-API WIM. In the VIM part, a Heat Template is created, mapping each element found in the descriptors, to a Heat resource. After the NS is instantiated (this process might be long, depending on how many VDUs are forming it), FSM modules take action, performing configuration tweaks and checking that every VDU is working correctly.

3.1.1. T-API WIM IA Wrapper (*tng-sp-ia-wtapi*)

This leads to notifiying the WIM block, if it is necessary for it to take action. Furthermore, WANs must be SDN and controlled by attached WIM. There are two possible scenarios when this can occur:

- When the instantiated NS is distributed among two or more NFVI-PoPs
- When the instantiated NS external interfaces are to be connected to external network endpoints (customer nodes)

WIM wrapper is directly attached to the IA SONATA´s SP, and when one or more T-API WIM(s) is (are) registered in the SP, it retrieves useful information about the topology.

A new type of VIM element called *endpoint* has been defined, each of those is a SIP (a T-API object) stored and exported by WIM. These *endpoints* are stored in IA database, so it can access then and make them public to the rest of SP modules. This component has been developed using Python 3.6, which is a very spread, easy to maintain and develop language.

Some of the fields within SIP object record are name, geo-location (city and country), type and description. Name is an important field, since it is the field where endpoints corresponding with an specific VIM are correlated, to do subsequent operations. Type is also important, since it indicates if the SIP is a customer endpoint, SP management (see Subsection 3.1.3 or a PoP.

When a service is about to be instantiated from any interface of the SP, the customer can define these endpoints as ingress or egress attachements to the instantiation parameters that comes to MANO. It connects each external interface of the NS to each *endpoint*, also taking into account some qos parameters, that are processed and sent to the wrapper.

Communication between MANO, IA-NBI (acting as a broker) and WIM wrapper is done using Advanced Message Queuing Protocol (AMQP), with three routing keys configured to trigger the wrapper: *infrastructure.tapi.management.wan.list*, *infrastructure.tapi.wan.configure* and *infrastructure.tapi.wan.deconfigure*.

When a VL is to be created using the wrapper, a message comes through *infrastructure.tapi.wan.configure*, with a payload as shown in Appendix B, whose fields are explained below:

- service_instance_id: UUID corresponding with the one asigned by the SP to the NS instance whom this VL depends
- vl_id: String corresponding with the one specified in the NSD to identify this VL
- wim_uuid: UUID which identifies the WIM where the wrapper is going to provision the connections
- ingress: It has two fields, one is the endpoint UUID to which the connection has to be provisioned and the other is an ip segment, which will be allowed to connect to/from egress endpoint
- egress: Contains the analogous contrapart of ingress. It is the other endpoint of the virtual link
- qos: Contains parameters regarding QoS requirements (if any included in the descriptor), e.g. minimal or maximum bandwitdth, maximum latency, or other
- bidirectional: A boolean indicating whether the VL is connected as an undirected graph edge or as a directed one

For deconfiguration, a message comes through routing key *infrastructure.tapi.wan.deconfigure*, only *service instance id*, *vl id* and *wim uuid* fields are included in payload, and using them the connectivity services which enable the VL become to an end. This is done as first block in the *instance termination* pipeline of the NFVO. Both configure and deconfigure can be found in more detail in Section 3.4.

Both SONATA project and its extension 5GTANGO components are stored in this GitHub project, using a Continuous Integration/Continuous Delivery (CI/CD) approach on how components are developed. First, component is tested in a pre-integration environment, where a pull-request updated image is deployed over the current component container. When the component is well-tested, it follows to integration environment, and it follows more steps until is release-ready. Using this way of function, allows the whole team to check common logs and report bugs easily. Check T-API wrapper´s GitHub repository [21] for further information.

3.1.2. QoS support

QoS parameters coming in VL configuration are parsed and translated to a proper connectivity service by wrapper. As it was said before, available QoS parameters are in this case related with bandwidth and each of them are sent to the WIM to be processed. Bandwidth is used in our case to differentiate between *silver* and *gold* flavours. When bandwitdth is higher than a known threshold, if resources are available, the path will be routed through the optical domain, otherwise, the path will be electrical only. *gold* flavour then uses the higher minimum bandwidth parameter (see 3.3.1).

Latency is also a parameter to be taken into account. Low latency use cases are very common in 5G and one of the promises is the availability to make this latency come to a real case. However, latency is not a deterministic measurement, and after doing some testing, we decided to use only bandwidth for differentiating flavours, and while this maximum latency is still forwarded to the WIM, we lack of the monitoring framework for getting this measurements in real time.

Other QoS measurements, such us jitter or packet loss, have similar problems than latency, but one step further, since without application monitoring data (to have real time mesurements) or probes between customer and VNFs, is not possible to measure this parameters.

3.1.3. NFVI-PoP interconnection

As stated in 1.1.1.3, interconnecting NFVI-PoPs is a common use case in NFV. Dataplane connectivity is abstracted from descriptors, being possible to have a service of two VDUs and a single VL, instantiated in a single NFVI-PoP or in two different NFVI-PoPs without the customer realising this fact. This use case, as mentioned, can be due to lack of enough resources in single NFVI.

Moreover, if there is a WAN (which could be a single switch node to a complex transport network) connecting both NFVI-PoPs, a WIM intervention must take place in order to allow the correct operation of the running NS. This multi-site NS instantiation is not limited to two PoPs, but instead the number of PoPs that could take part in it is only limited by the number of VNFs that form part of the NS. For allowing SONATA to have this new functionality, more changes in modules were implemented.

Apart from the WIM T-API wrapper, several components where to be updated: *portal*, *gatekeeper*, *MANO* and *IA*, to include qos support, and new placement primitives. There was a notable work done by MANO and IA developers, which had also included QoS support. Descriptor models where also changed, and that included more components outside SP, the group called SDK, changed, to update the design phase. All this new changes are going to be included in a new SONATA framework upcoming release. Furthermore, in the specific WIM T-API wrapper, there were two specific problems to be tackled in order to allow this multi-site connectivity.

On one hand, Openstack deployments in SONATA are only compatible with the usage of an virtual internal router (an Openstack resource, dependant on Layer Three (L3) agent of *neutron* service) to give connectivity from/to the external network to/from each VNF is done by the use of a concept called *floating Internet Protocol (IP)s*, which in the end is a concept very similar to doing a Network Address Translation (NAT). Virtual router runs in a new namespace [22] inside Linux kernel, and new router interfaces can be created and attached to private tenant networks inside the openstack deployment (see Figure 3.2. In this figure, we can see a *provider network* which is accesible from outside without passing through other elements (blue bus), and other networks that are not directly accesible from outside, however, internal networks (colored in orange, green and red) are connected to the virtual router, and each VL is representd by each one of these internal networks.

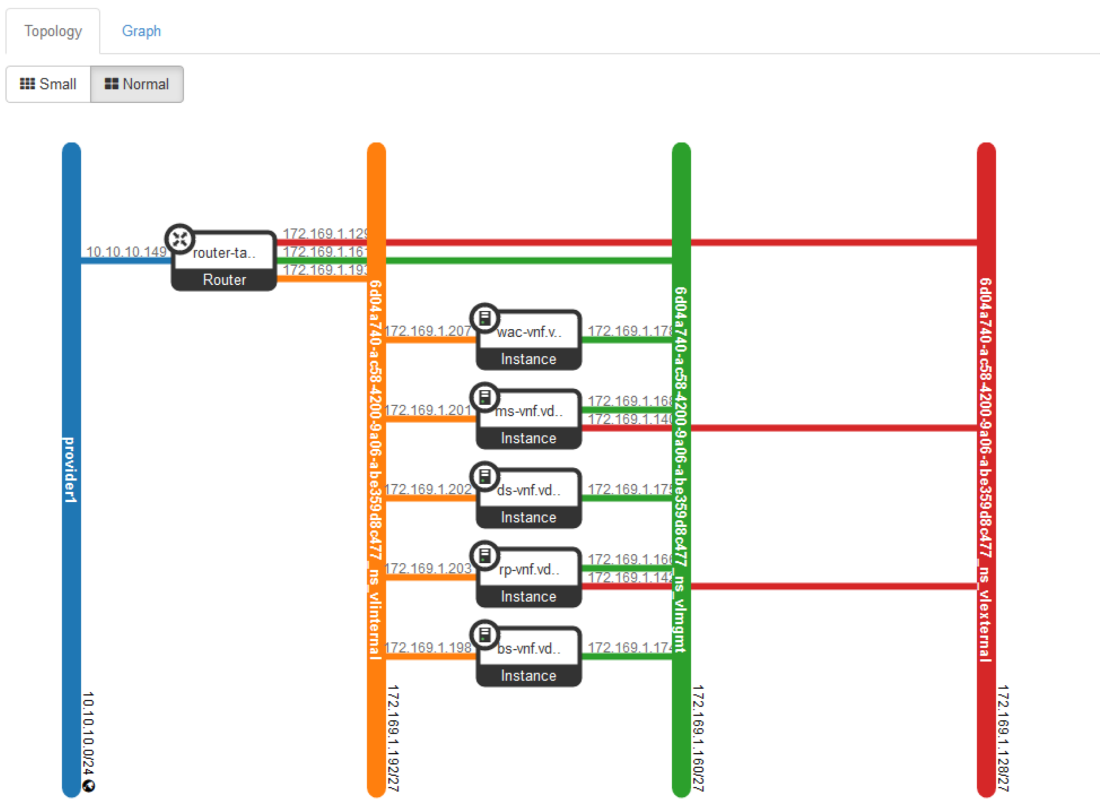


Figure 3.2: Openstack provider (blue) vs private (orange, green and red) networks comparison

As a consequence, *floating IPs* are assigned to each VNF interface which type is whether management or external. A *floating IP pool* is defined, and this pool has to be included by provider´s subnet range. *Floating IPs* are assigned by the L3 Agent of *Neutron* service. To allow connectivity to this addresses, customer endpoints must have connectivity to the virtual router, appart from to the VNF. This is the consequence of the router acting as proxy-Address Resolution Protocol (ARP) to the VNFs.

Besides, in our specific deployment, VNF management plane is shared with data plane. Because of this, a management service has to be created, which allows connectivity between every IP contained in the *floating IP pool*, also including the external IP assigned to the virtual router. A dedicated interface of the server running the SP is the contrapart of this 'management connectivity service'. Some SP components (e.g. FSM, Service Specific Manager (SSM), monitoring service, etc) use this connection to reach the target PoP and it hosted VNFs.

3.2. Network Management with WAN Infrastructure Manager

WIM model and definitions has been well defined by ETSI NFV work group, as explained in Subsection 1.1.1.1 entry for this functional block. CTTC premises are controlled by his own network orchestration suite [23], which takes care of topology retrieval for multiples topologies, and provisioning for the required connectivity services, using several plugins for each different domain. This plugins allows interfacing with:

- Pure SDN domains, where there are several Openflow Software Switches, controlled by a SDN controller, i.e. ODL, Open Networking Operating System (ONOS) or Ryu. Each controller has its own API
- Optical - Electrical transceiver nodes, which have their own API
- Optical colorless domain, controlled by a Active Stateful Path Computation Element (AS-PCE) and using Path Computation Element Protocol (PCEP) to communicate. Plugin uses an implementation of a Path Computation Client (PCC)
- Hierarchical control, where other controllers or orchestrators can be stacked in a multi-level architecture and use Control Orchestration Protocol (COP)

Also, there are plugins to allow connectivity between domains, defined in an Extensible Markup Language (XML) and also there is the posibility of having a complex network virtualized. In this work, network is not virtualized, but when an optical lightpath is created, which means that there is an optical circuit, a virtual link is created between the two border nodes involved. New connections can be groomed if the network is yet capable of supporting more bandwidth usage. When the link usage is almost at full capacity, a new optical link can be created, using spatial multiplexing.

In ADRENALINE Testbed there are currently three ODL controllers, one at the edge network close to the access nodes, one emulating a metro network between the edge and the core and one at the core emulating a core network and a DC. Also, in parallel to the metro network using packet switching, there is an optical circuit network using AS-PCE. All of them are considered the *Transport Network* in the Pilot.

In SDN domains controlled by a ODL controller, there´s a controller´s service that takes care about Topology Discovery. This service makes Switches to send Link Layer

Discovery Protocol (LLDP) packets through every port (as Spanning Tree Protocol (STP) does in legacy networks), but also an Openflow rule is created in each Software Switch (or Physical Openflow Switches), to report that packet to the controller. These packets contain information about sender´s identity, what is called OpenFlow Datapath ID (DPID), and is a unique identifier of each Openflow Switch. Thus, ODL controller can compute DPIDs connected within each other, and make a topology graph.

Edge sub-domain (see Figure 3.3) is composed by two software switch nodes, running OpenVswitch 2.6.6 over Ubuntu 14.04.6, which are, from the point of view of the ODL controller that governs over them, isolated, since there is no direct link between them. However, when this topology is imported, and conformed by the ABNO WIM, network graph connects each of this two nodes to an Optical Control-plane Controller (OCC), which translates into this node connected to the optical dataplane. Additionally, the physical connection is performed by the use of optical Myri PCI Cards [24] using 10 gigabit small form factor pluggable transceiver (XFP) modules. Also, each node has a link to two of the nodes which are integrated in metro sub-domain.

Moreover, Edge sub-domain nodes give connectivity to one edge cloud, containing an Openstack cluster composed by two nodes, representing a Central Office (CO), one of each is connected to one of the switch nodes. Also, four Next Unit of Computing (NUC)s act as client aggregators, and also are distributed among the two nodes.

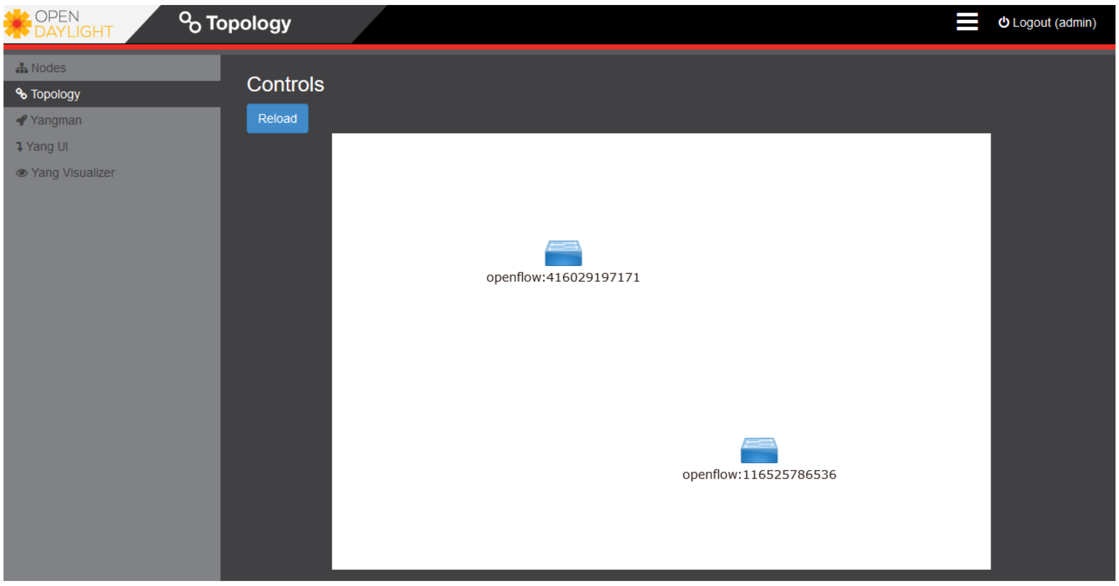


Figure 3.3: Edge sub-domain topology shown in Opendaylight web GUI

Metro sub-domain (see Figure 3.4) is composed by four Software Switch nodes, running OpenVswitch 2.10.0 over Ubuntu 18.04, which in this case are connected in a diamond topology, as shown in the figure. These nodes are fully packet switched, representing a metro network with ethernet connections. Each one of the four switches has an outgoing connection to edge or core subdomains. In this case, no other network or compute elements are connected to the switches. These nodes are also controlled by a ODL controller.

Core sub-domain (see Figure 3.4) is composed by four Software Switch nodes, in this case, two of them are running OpenVswitch 2.10.0 over Ubuntu 18.04 (*DC-Switches*) and the other two running OpenVswitch 2.6.6 over Ubuntu 14.04.6. These last two

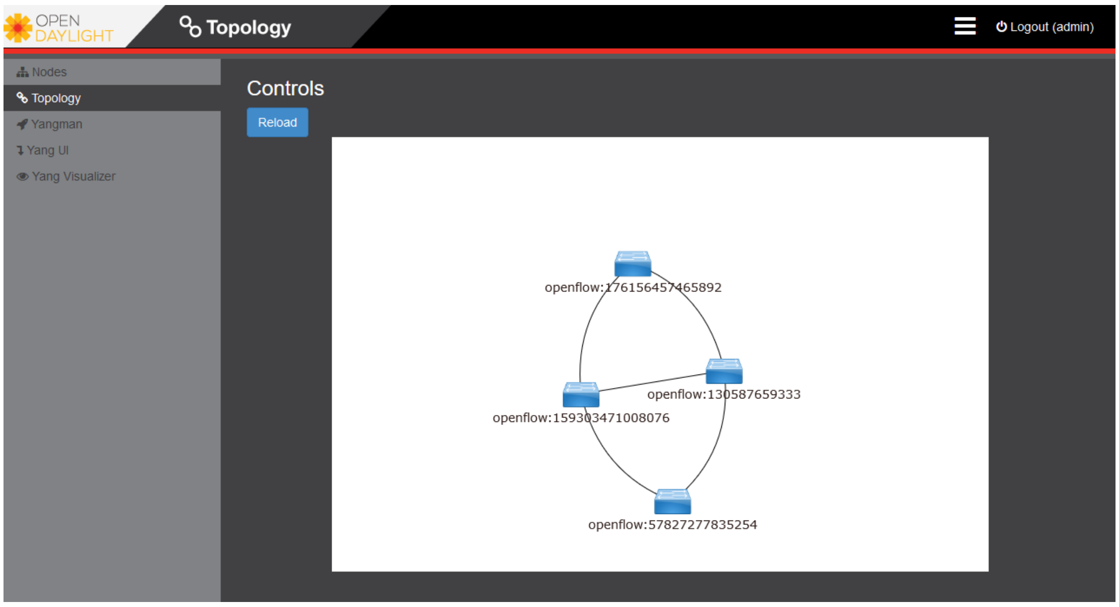


Figure 3.4: Metro sub-domain topology shown in Opendaylight web GUI

mentioned nodes also are connected physically to the optical domain using XFP in the same way as the edge nodes are. modules In this part of the WAN the *core-datacenter* NFVI is located, which is an openstack cluster of three nodes whose dataplane is connected to both of the *DC-switches*.

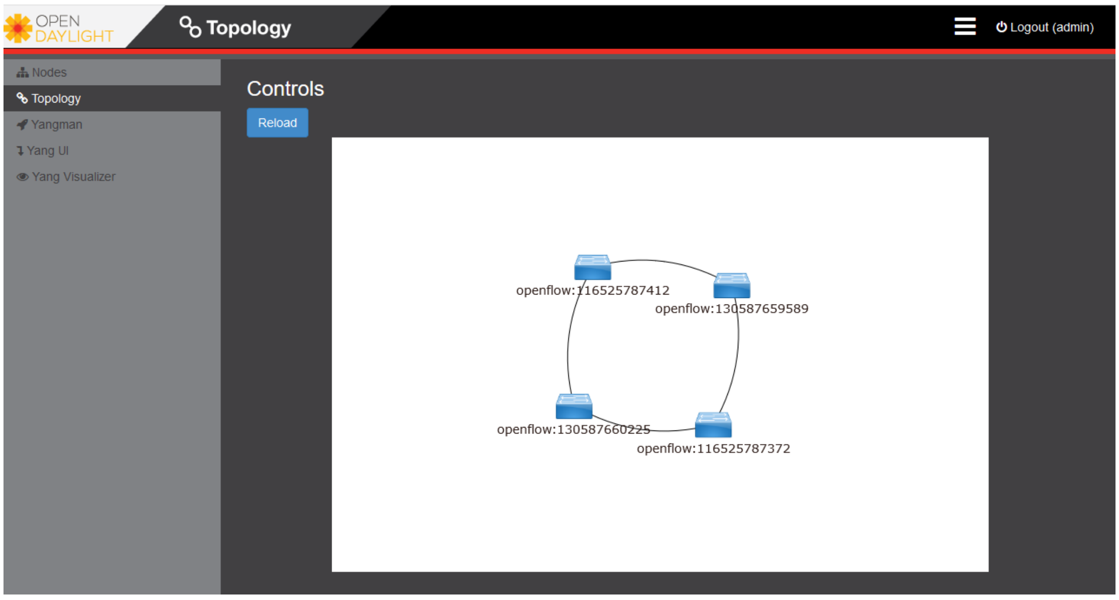
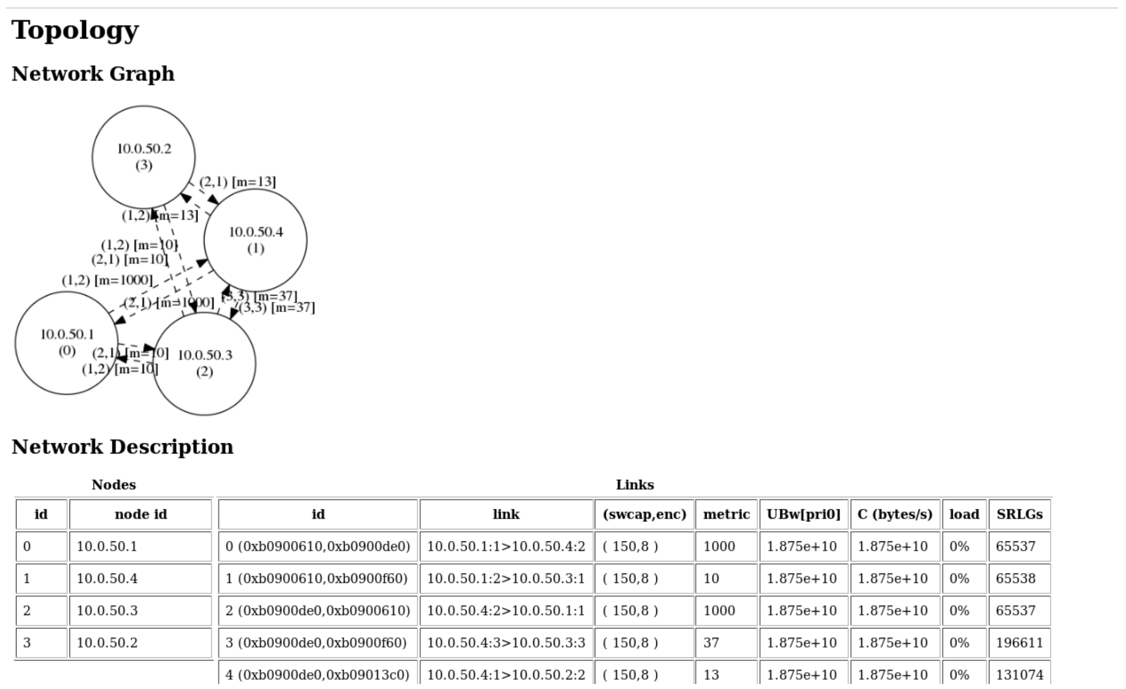


Figure 3.5: Core sub-domain topology shown in Opendaylight web GUI

Finally, an optical sub-domain (see Figure 3.6), which is formed by four optical routing elements (two reconfigurable optical add-drop multiplexer (ROADM)s and two optical cross-connect (OXC)s), whose control plane is attached to four corresponding OCCs, and all of them associated with a AS-PCE node, which takes care of both topology export and optical resources provisioning.

Overall, these four sub-domains are 'merged' by the ABNO WIM and a new cross-domain topology (*Application Topology*) is created, as shown in Figure 3.7. Optical nodes are



Topology

Network Graph

Network Description

Nodes

id	node id
0	10.0.50.1
1	10.0.50.4
2	10.0.50.3
3	10.0.50.2

Links

id	link	(swcap,enc)	metric	UBw[pri0]	C (bytes/s)	load	SRLGs
0 (0xb0900610,0xb0900de0)	10.0.50.1:1>10.0.50.4:2	(150,8)	1000	1.875e+10	1.875e+10	0%	65537
1 (0xb0900610,0xb0900f60)	10.0.50.1:2>10.0.50.3:1	(150,8)	10	1.875e+10	1.875e+10	0%	65538
2 (0xb0900de0,0xb0900610)	10.0.50.4:2>10.0.50.1:1	(150,8)	1000	1.875e+10	1.875e+10	0%	65537
3 (0xb0900de0,0xb0900f60)	10.0.50.4:3>10.0.50.3:3	(150,8)	37	1.875e+10	1.875e+10	0%	196611
4 (0xb0900de0,0xb09013c0)	10.0.50.4:1>10.0.50.2:2	(150,8)	13	1.875e+10	1.875e+10	0%	131074

Figure 3.6: Optical sub-domain topology shown in PCE web GUI

identified by the IP address of its corresponding OCC, and Openflow Switches by their DPID. This topology is the one that is available at WIM NBI API in JSON format, and also to perform connections, NFVO and furthermore OSS/BSS must use this topology or an abstracted version of it to perform routing algorithms and later placement on SIPs which have a PoP connected.

Service orchestration applies to the logic of the service as requested by the customer, identifying the functions needed to honour the customer request as well as the form in which these functions interrelate to complete the complete service. The service orchestrator will trigger the instantiation of the NFs in the underlying infrastructure in a dynamic way.

By the right combination of service and resource orchestration, the E2E management and orchestration functionalities will be responsible for a flexible mapping of services to topologies of VNFs, based on a dynamic allocation of resources to VNFs and the reconfiguration of VNFs according to a changing service demand during runtime.

3.2.1. T-API extension

As it was said in Subsection 1.2.2 T-API specification uses connectivity services to provision connections. These connectivity services can be created, modified or deleted using the NBI RESTful API, using a specific payload. Previously, only a connection between two SIPs can be created, as a result, every packet comming from SIP A to SIP B is forwarded, but that makes impossible a proper SDN path creation.

To give a solution to this problem, T-API was extended, to provide support to what is called a *match*. A *match*, in SDN protocols (e.g. OpenFlow [25]) is meeting a condition through extracting a target field, and comparing it to a parametrized value, extensible to a collection of *matches*. If the condition is met, the *action* or *actions* specified in a

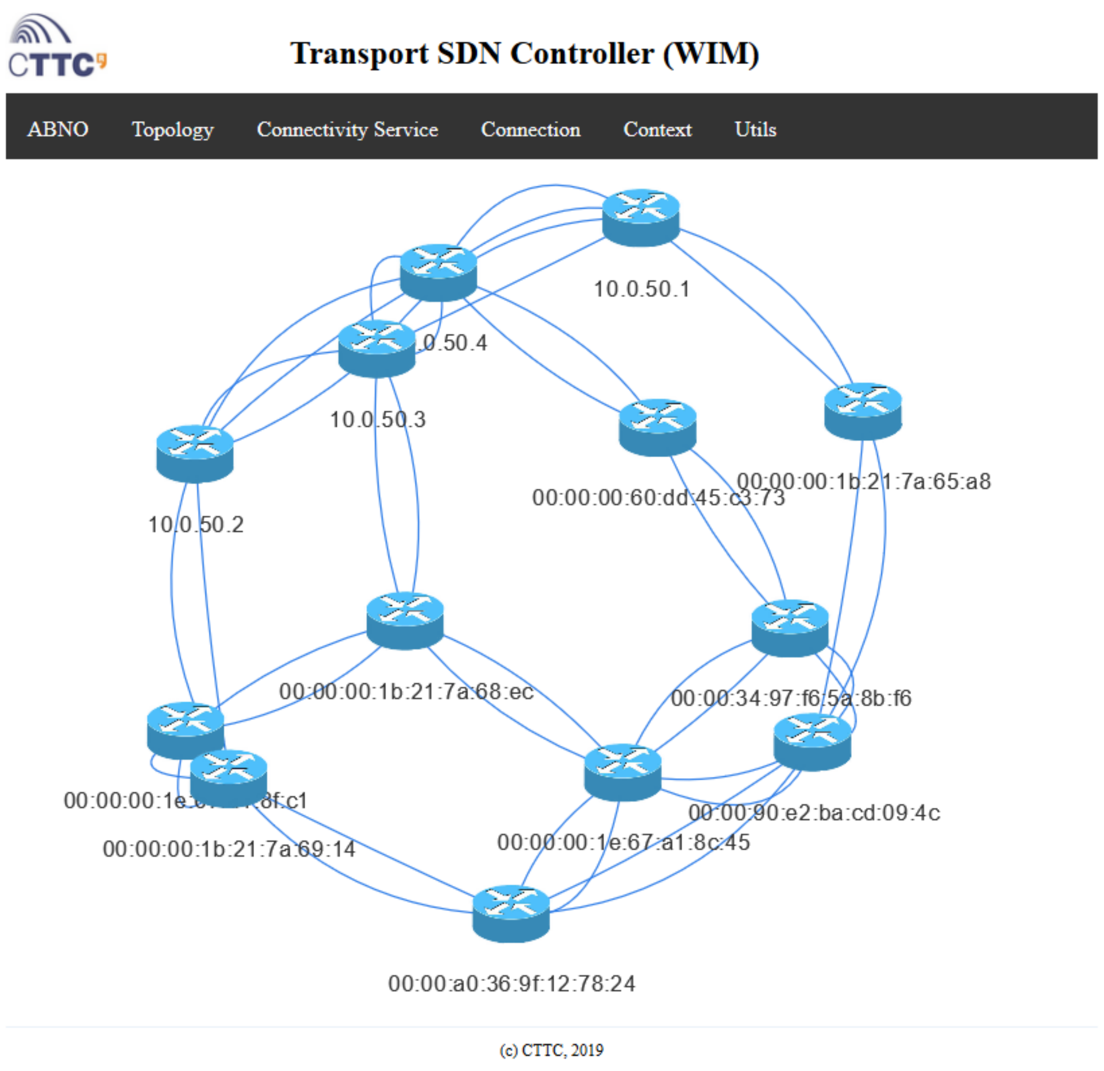


Figure 3.7: Application Topology shown in ABNO web GUI

Flow is/are executed. The YANG model tree of the extension is shown at Figure 3.8, and the whole YANG file can be found in Appendix A.

This is brought 'upwards', since Openflow is the protocol used by the SDN controllers SBI. Most previously presented *match* fields can be univocally mapped to a higher-level abstraction to allow for a connectivity service to use a *match*.

For this reason, T-API had to be extended. Chosen fields for being included as matches where Internet Protocol version 4 (IPv4) and MAC addresses. WIM manages to forward the match to the compatible plugins (in this case, subdomains controlled by ODL) and then several connectivity services can be deployed sharing the same resources.

Each bidirectional VL, in our implementation, makes use of four different unidirectional connectivity services. Two of them creates flows that are matching IP type packets, while the other two are ARP. This is necesary in order to avoid static arp entries in both of the endpoints which are to be connected. Each connectivity service is furthermore splitted

```
module: tapi-match-types
  augment /tapi-common:context/tapi-connectivity:connectivity-context/..
  tapi-connectivity:connectivity-service:
    +--rw match
       +--rw ipv4-source?              inet:ipv4-prefix
       +--rw ipv4-target?              inet:ipv4-prefix
       +--rw link-layer-addr-source?   yang:mac-address
       +--rw link-layer-addr-target?   yang:mac-address
       +--rw link-layer-type?          ethertypes:ethertype
```

Figure 3.8: Tree format of the extension of T-API YANG model which allow the use of matches in connectivity services

in connections, that are used by each plugin to perform provisioning in its corresponding domain.

We also decided to use MPLS to encapsulate packets through the transport network. This eases interoperability with other technologies, and makes faster matching, once the packets are labeled.

Also, SIP definition was extended, to include fields specific for SONATA SP, included in the SIP record without affecting previous specification. Each SIP with extended information, and augmented using a configuration JSON per sub-domain, defining which SIPs are to be made public and used by third parties.

3.3. Real Time Communications Network Service

Based on the requirements defined by 3rd Generation Partnership Project (3GPP) [26] for 5G(see Table 3.1), those which apply to communications have been addressed to be fulfilled by the communications pilot use cases (see Section 2.1, designed in previous stages of the project, and implemented during the past year (and still under development). These are Conversational Voice and Conversational Video.

The composition of the NS that provides real-time communications service is detailed in this section. There are five VNFs which compose the real-time communications NS as shown in the deliverable d72, chapter 2 [19] and the package is available in the github repository of the pilot [27]:

- **Reverse Proxy (VNF-RP)**: This VNF is composed by an Nginx with the purpose of receiving all the Hypertext Transfer Protocol (HTTP) and WebSocket traffic coming from the client side of the pilot deployment. A Reverse Proxy is useful, in this case, to provide caching to the rest of the NS which is slower, improving the Quality of Experience (QoE) perceived by the final user. It could be used to scale the service, as a load balancer, but this use case is not explored in this pilot.
- **WebRTC Application Controller (VNF-WAC)**: This VNF packages the Sippo Server and the Signaling Server which gives provision and coordinates the WebRTC communication. Besides, it also includes a Simple Network Management Protocol

5QI Value	Resource Type	Default Priority Level	Packet Delay Budget [ms]	Packet Error Rate	Default Averaging Window [ms]	Example Services
1	GBR	20	100	10^{-2}	2000	Conversational Voice
2		40	150	10^{-3}	2000	Conversational Video Live Streaming
3		30	50	10^{-3}	2000	Real Time Gaming Process automation - monitoring, V2X messages
4		50	300	10^{-6}	2000	Non-Conversational Video Buffered Streaming
5	Non-GBR	10	100	10^{-6}	N/A	IMS Signalling
6		60	300	10^{-6}	N/A	TCP-based Video (Buffered)
7		70	100	10^{-3}	N/A	Voice, Video (Live Streaming, Interactive Gaming)
8		80	300	10^{-6}	N/A	TCP-based Video (Buffered)
9		90				

Table 3.1: Subset of 5QI specification

(SNMP) drone service used to provide to the SP's monitoring service user related data, such as the user registrations and user events

- **Backend Services (VNF-BS)**: Backend support services run on the same VNF, in this case a database and a messaging queue are needed, in this case using a MongoDB database which is NoSLQ and RabbitMQ which uses AMQP. It is accessible for all VNFs requiring of these services

- **Dispatcher (VNF-DS)**: This VNF asks for *media rooms* to create multimedia sessions. Any new Media Server added to the system will register its inclusion in the Dispatcher configuration and thus be eligible to create new chat rooms

- **Media Server (VNF-MS)**: This VNF is composed of a Janus [28] and a Wrapper in charge of receiving and relaying the media during the videoconferences between the clients using the service. Moreover, it includes an SNMP service used to monitor network parameters (bandwidth, jitter and packet loss) affecting the QoS of the communication

Each VNF inside the NS is connected in bus, using private tenant networks [2] which use VXLAN. QoS is not applicable to the traffic internally addressed between VNFs the same way we do in WAN connections, since those VLs are out of the control of the WIM, meanwhile there are some references to configure QoS policies in Openstack [29] to make an approach to it, they have been not tested in our deployment, but they may be tested in the future.

[2] Openstack tenant networks is the typical way of connecting VNFs located in the same NFVI

Finally, there are two more components to consider that are relevant for NFVO to manage the communications NS:

- FSM: Are scripted modules which perform day-0 or day-1 operations in the configuration of each VNF if that´s needed, and also manages life cycle events on that VNF[30].
- SSM: It allows the VNFs to extend their own information with data from the other VNFs or allow them to have "neighbour awareness" within the whole VNF ecosistem, and location in case it is a multi-PoP deployment.

Several iterations were made by pilot developers, changing software inside the images, descriptors, FSM, SSM and monitoring probes; so an effort to snapshot and create new versions of the images have been done regularly. Moreover, due to how these snapshots are done internally, they grow in size with each generation, e.g. from an image with a size of ~1 GB, after some iterations it was sized to almost 20 GB. To avoid this, and automate the process to be more efficient, we developed an automated script, which search in the Openstack deployment for the images, having as input whether the name or the UUID of each instance to be 'versioned', and it takes the snapshot using Openstack CLI client, and after the new snapshot (an image of the current state of the instance) is done, the image is processed using a tool called *virt-sparsify* [3]. This tool shrinks the virtual image, and as an example, an ms image of 3.89 GB can be compressed to occupy only 1.14 GB, which is a considerable reduction of more than two thirds. After all target images have been processed, they are uploaded again using openstack CLI client. The script is created in bash, and being executed in the same node which hosts *Glance* service (Image service), it can take more than 15 min per image.

3.3.1. Network Service and Function Descriptors

The process of designing a NS starts by defining the descriptors. In Communications Pilot, as it was said before, there are five VNFDs, with one VDU each. All of them have a FSM attached, which performs health check and configuration modifications on instantiation. Two of the VNFDs include some parameters which are exported to the monitoring engine, using both prometheus and SNMP objects to export useful monitoring data. It´s important to review some specific variables inside the descriptor, such as *md5*, that must be the same of the image that is available at the VIM. Each SONATA descriptor is referenced by its UUID (once it is uploaded to the SP catalogue) or by the tuple *name-vendor-version*, which must be unique. Each VDU has some 'hardware' requirements for the VNF to be deployed, e.g CPU, RAM memory, Disk Size, among others.

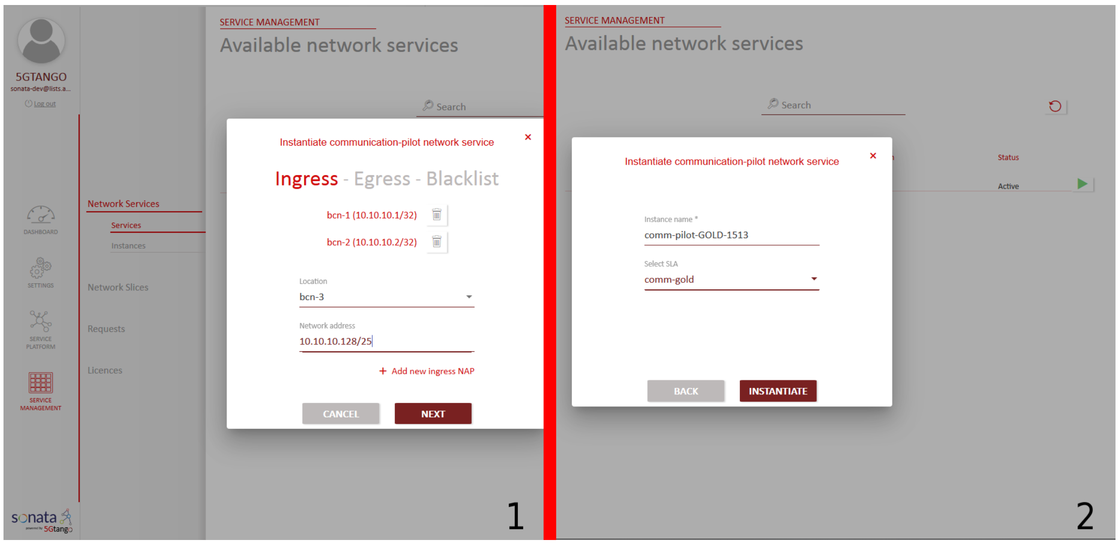
The NSD references each VNFD and all VLs are defined here, and also their endpoints. Endpoints which are not part of the NS elements, such as management endpoint or external endpoints, must be also defined in the NSD. In case of the communications pilot, these endpoints are named *nscpmgmt* and *nscpexternal*. External endpoints represent zero or more external customers, which will be using the NS. Communications pilot, due to QoS requirements, requires the use of DF, and there are two of them defined in the

[3]`http://libguestfs.org/virt-sparsify.1.html`

NSD, called '*gold*' and '*silver*'. *Gold* DF has a requirement of 1000 Mbps as minimum bandwitdth, while *silver* has only 500 Mbps. As a result, when this requirements are forwarded to the WIM, *gold* flavour is provisioned through the optical network, and *silver* is not. There are no further modifications in these DFs, but we could vary any parameter, change VNFs, include more/less functions... whatever we could imagine.

It is worth mentioning that every descriptor must follow its corresponding schema, and all of them are defined in *tng-schemas*[31]. Once the service design is finished, it is grouped as a project, and every included file must be reference in the *project.yml* file of the project. SONATA SDK provides a tool to validate and pack SONATA objects. When the service is packed, a *.tgo* package is generated with all the files included in the project, which is the file that can be uploaded to the SP, and will be unpacked there to be used.

The service instantiation, is executed using the portal, as we can see in Figure 3.9. This allows the operator to choose between a number of endpoints, which previously has been populated by wrapper (as explained in Subsection 3.1.1) to use them as ingress/egress, and also a network segment is also included as parameter. After that, the operator has to choose a name for the instance and an available SLA.



Instantiate communication-pilot network service

Ingress - Egress - Blacklist

bcn-1 (10.10.10.1/32)
bcn-2 (10.10.10.2/32)

Location: bcn-3

Network address: 10.10.10.128/25

+ Add new ingress NAP

Instantiate communication-pilot network service

Instance name *: comm-pilot-GOLD-1513

Select SLA: comm-gold

Figure 3.9: Instantiation of a NS using SONATA web GUI, showing Input parameters (first step) and instantiation commitment (second step)

Instantiation process can take several minutes, after every involved VIM(s) and WIM(s) finish their their actions, the NS instance will be shown as **READY**. If the final status is **ERROR**, that means that something went wrong in the process (e.g. there is no VIM with enough resources to deploy the NS, descriptors are not well formatted, a component with a buggy behaviour...) and we should check logs or send a message to the component developer so it can be fixed.

3.4. Workflows

This section gives a closer look to the processes and workflows of the wrapper module described in Section 3.1.

3.4.1. Management and Setup

On startup, there are two relevant actions performed before the wrapper is ready to be used; this workflow is shown in Figure 3.10. These actions are management connections creation and sip retrieval and export. SP must have a static connection to reach VNFs that are instantiated using one of the VIMs which are managed by it. To do so, WIM wrapper creates this management service on startup, based on the configuration of the WIM. There are tho relevant fields for this task, *management_flow_ip* and *floating_ip_ranging* stored in VIM configuration of IA database. The field *management_flow_ip* refers to the interface of the SP that is connected to the PoP to reach VNFs instantiated there, and *floating_ip_ranging* is a list of IP ranges that are susceptible to be assigned to VNFs, so the connection is pre-allocated.

Additionally, during startup step, SIPs from different T-API WIMs are identified and correlated with the current database, and stored or updated. Some checks are done to correlate SIPs with PoPs, and also new client endpoints are added to the database so it can be used to connect them to new instantiated NSs. Both management and endpoints information are refreshed when a message comes to *infrastructure.tapi.wan.list*, and then data is sent to MANO and also stored in the database.

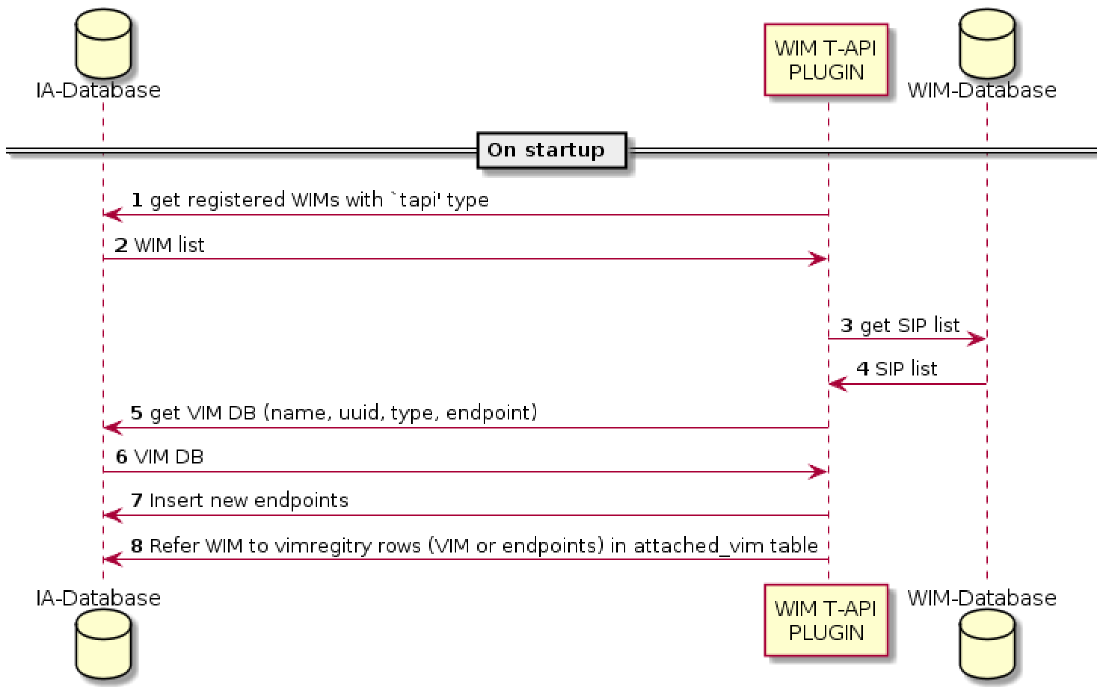


Figure 3.10: Startup endpoint population and management connection configuration

Furthermore, the wrapper is connected to the AMQP routing keys to be interacting with the rest of SP components and more precisely with IA which is the 'parent' component.

3.4.2. Network Service Instantiation

When a NS is instantiated and needs WIM intervention, a message reaches *infrastructure.tapi.wan.configure* and the corresponding method is triggered in the wrapper. The diagram presented in Figure 3.12. A new VL is needed, so first the wrapper recovers assigned WIM, ingress endpoint and egress endpoint information from the database, whose uuids are provided in the payload.

Then, it interacts with the WIM matching the ingress and egress SIPs by the public name (which is the one stored in IA database), meaning that a name for a SIP/endpoint is unique. If the new connection requires also an additional connection to the PoP virtual router, it is also created, and then every new connection (from the selected endpoint to the selected router, if using the same addresses) will be saved until it is needed no more. Both router and service connections are tunneled using MPLS, being the labels assigned automatically by WIM.

Figure 3.11 shows an example of the payload delivered to the WIM when a connectivity service is created, using a HTTP POST to the proper url. The connection is performed between the two SIPs, referenced by their respective uuid, and also QoS information, match parameters and 'layer' used are included there .

After the process finishes, relevant data about connection is stored in the WIM registry for using it when the termination message comes.

3.4.3. Network Service Termination

Just as it was before during the configuration workflow, for the termination a message comes through *infrastructure.tapi.wan.deconfigure* to terminate an existing VL (see Figure 3.13. Data previously stored is recovered, and then the connections are deconfigured. If it was using a router connection, and there are no more connections using that resource, that extra connection is deconfigured aswell. The message delivered to the WIM is a HTTP DELETE to the same url used to create the target connectivity service.

After that, old database entries are deleted and finally the VL termination is communicated to the MANO.

Payloads for both configure and deconfigure operations can be found at Appendix B.

```
{
  "direction": "UNIDIRECTIONAL",
  "requested-capacity": {
    "total-size": {
      "unit": "MBPS",
      "value": "500.0"
    }
  },
  "uuid": "1013",
  "layer-protocol-name": "MPLS",
  "end-point": [
    {
      "service-interface-point": "/restconf/config/context/
      service-interface-point/44c03813-0e33-4f50-88d2-3ffd6804dd11/",
      "direction": "BIDIRECTIONAL",
      "layer-protocol-name": "ETH",
      "role": "SYMMETRIC"
    },
    {
      "service-interface-point": "/restconf/config/context/
      service-interface-point/7be67c30-2bf9-4545-825e-81266cfff645/",
      "direction": "BIDIRECTIONAL",
      "layer-protocol-name": "ETH",
      "role": "SYMMETRIC"
    }
  ],
  "match": {
    "ipv4-target": "10.10.10.1/32",
    "link-layer-type": "2048",
    "ipv4-source": "10.10.10.141/32"
  }
}
```

Figure 3.11: Payload of the connectivity service message from T-API wrapper to WIM

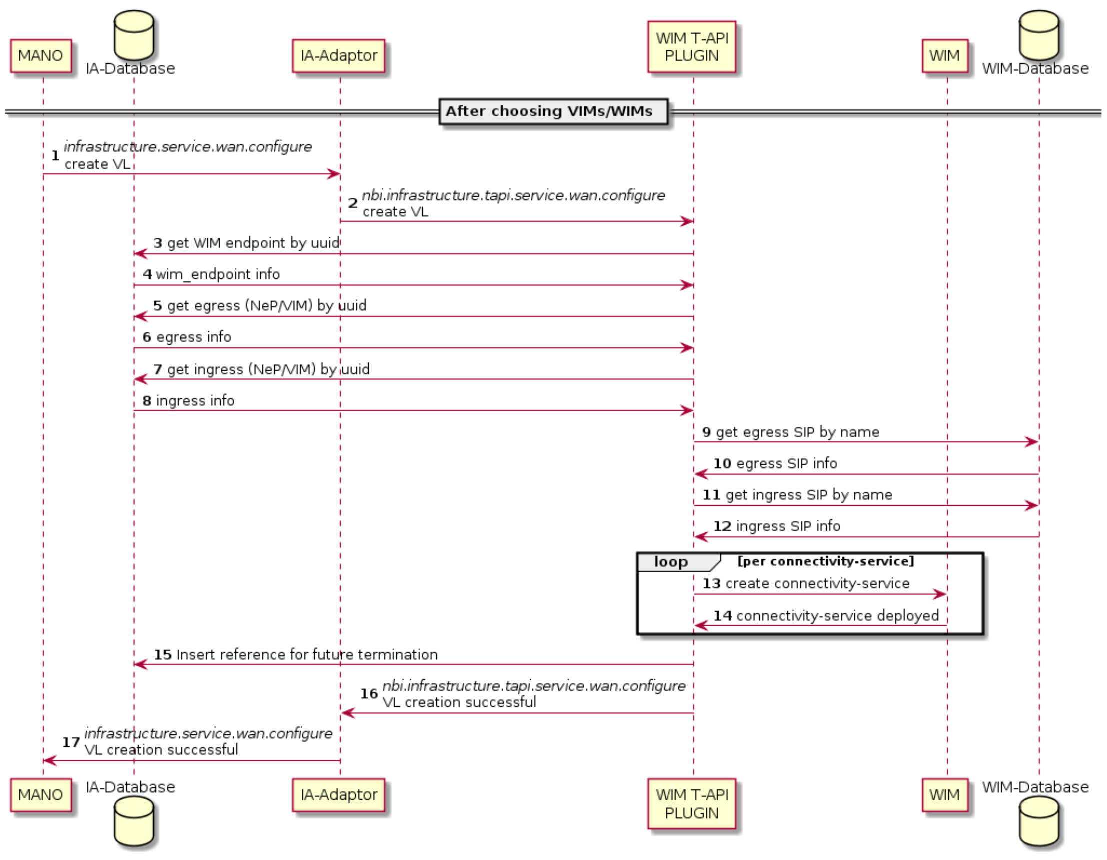

MANO
IA-Database
IA-Adaptor
WIM T-API PLUGIN
WIM
WIM-Database
After choosing VIMs/WIMs
1 infrastructure.service.wan.configure create VL
2 nbi.infrastructure.tapi.service.wan.configure create VL
3 get WIM endpoint by uuid
4 wim_endpoint info
5 get egress (NeP/VIM) by uuid
6 egress info
7 get ingress (NeP/VIM) by uuid
8 ingress info
9 get egress SIP by name
10 egress SIP info
11 get ingress SIP by name
12 ingress SIP info
loop [per connectivity-service]
13 create connectivity-service
14 connectivity-service deployed
15 Insert reference for future termination
16 nbi.infrastructure.tapi.service.wan.configure VL creation successful
17 infrastructure.service.wan.configure VL creation successful

Figure 3.12: Network Service external network connections creation process

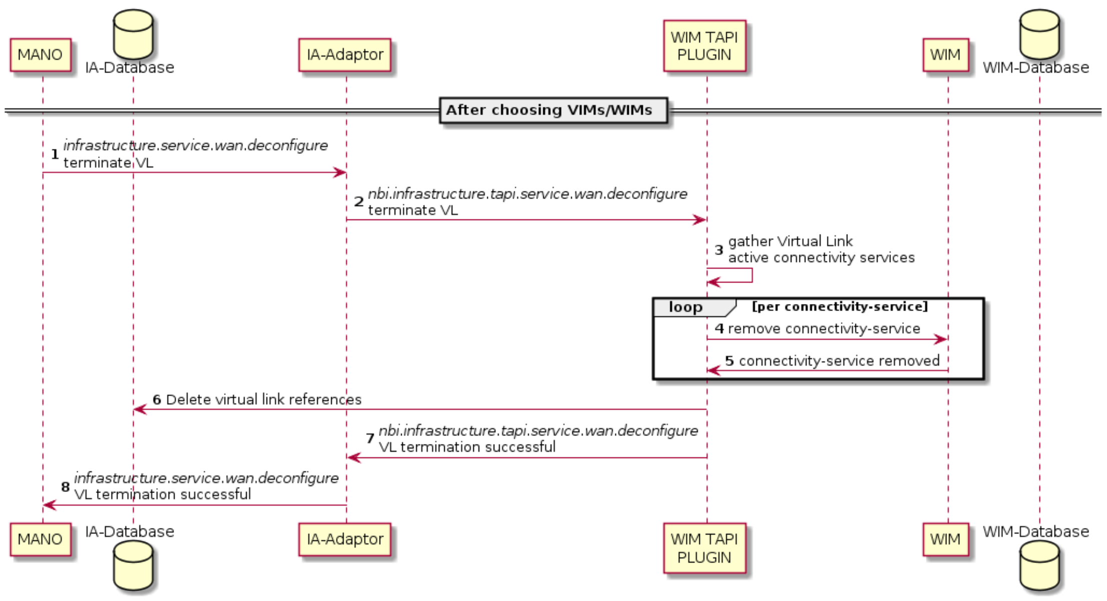

MANO
IA-Database
IA-Adaptor
WIM TAPI PLUGIN
WIM
WIM-Database
After choosing VIMs/WIMs
1 infrastructure.service.wan.deconfigure terminate VL
2 nbi.infrastructure.tapi.service.wan.deconfigure terminate VL
3 gather Virtual Link active connectivity services
loop [per connectivity-service]
4 remove connectivity-service
5 connectivity-service removed
6 Delete virtual link references
7 nbi.infrastructure.tapi.service.wan.deconfigure VL termination successful
8 infrastructure.service.wan.deconfigure VL termination successful

Figure 3.13: Network Service external network connections termination process

CHAPTER 4. EVALUATION

In this chapter, we provide the evaluation of the proposed architecture functionalities. Section 4.1 describes further actions in regarding the infrastructure. Section 4.2 analyzes a network capture of an instantiation workflow of a service using SONATA. Finally, some results are analized and commented in sections 4.3 and 4.4.

4.1. Testbed Description

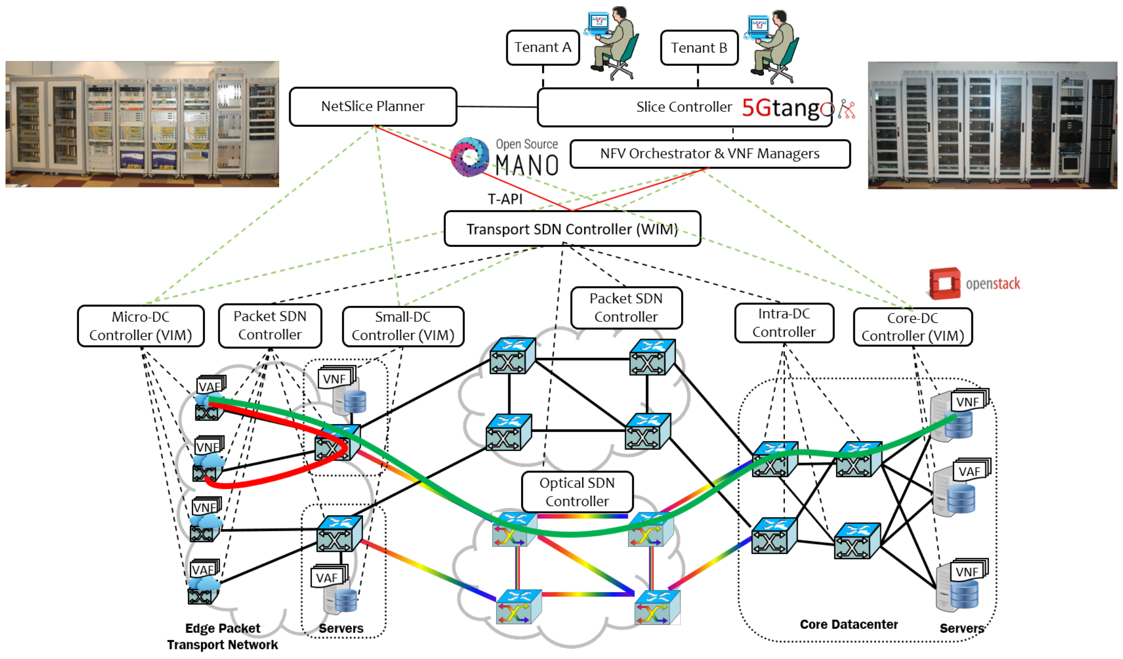


Figure 4.1: ADRENALINE Testbed architecture and deployment of Network Service A (in green) and Network Service B (in red)

In order to validate video-conference service, the cloud-computing platforms and transport network of the CTTC ADRENALINE Testbed [32] have been used. Figure 4.1 shows the ADRENALINE testbed, which is composed of multiple components and prototypes, with the objective of offering end-to-end services. Moreover, customers and applications are connected across an heterogeneous network and multiple technologies which apply cloud computing concepts are used for the development and validation of 5G services in conditions close to production environments.

ADRENALINE includes the following technologies and capabilities:

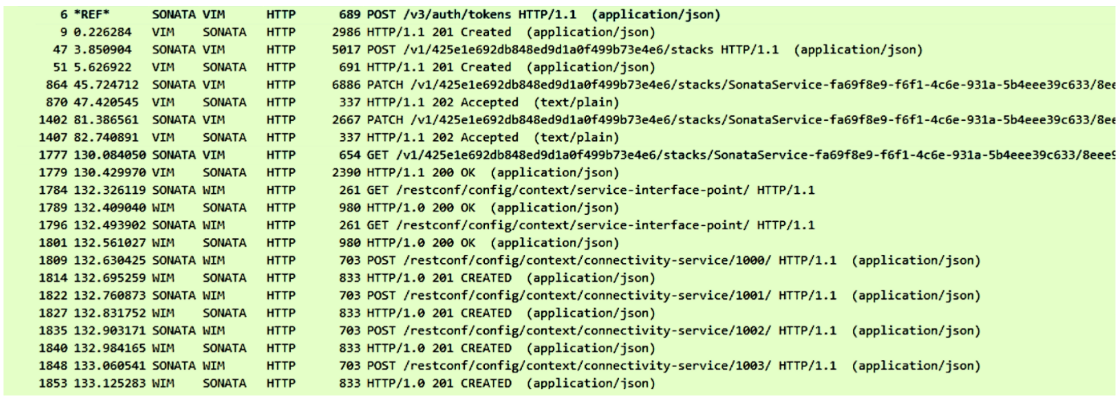
- A fixed/flexi-grid Dense Wavelength Division Multiplexing (DWDM) core network interconnected with white box ROADM/OXC nodes and Software defined optical transmission (SDOT) technologies for deploying sliceable-bandwidth variable transceiver (S-BVT) modules and programmable optical systems, with 610 km of optical fiber deployed
- A packet-based transport network for the edge (access) and metro segments for traffic aggregation and switching of Ethernet and MPLS flows with QoS, and alien wavelength transport to the optical core network

- A distributed core and edge cloud platform for the deployment of VNFs. The core cloud infrastructure composed of a core-DC with high-performance computing (HPC) servers and an intra-DC packet network with alien wavelength transport to the optical core network. The edge cloud infrastructure is composed of micro-DCs in the edge nodes and small-DCs in the COs
- An SDN/NFV control and orchestration system for the joint orchestration of the multi-layer (packet/optical) network resources and the distributed cloud infrastructure resources. This SDN/NFV orchestrator is able to provide life-cycle management for end-to-end services (also including service function chaining of VNFs) and multi-tenancy
- Network Slices are offered as interconnected NSs [33]

4.2. Workflow validation

This section focuses on demonstrating the proposed scenario in Section 3.4. The objective is to validate the workflow diagrams involving the deployment of a Network Service over multiple inter-connected VIMs.

Figure 4.2 shows a Wireshark capture between SONATA and VIMs/WIM. It can be observed that once the network service request is received, SONATA requests to VIMs the deployment of heat templates, and finally it interacts with the WIM using SONATA WIM wrapper [21] in order to inter-connect VNFA and VNFB. All these interaction is using VIM/WIM RESTful NBI APIs.



```
   6 *REF*      SONATA VIM    HTTP    689 POST /v3/auth/tokens HTTP/1.1  (application/json)
   9 0.226284   VIM    SONATA HTTP   2986 HTTP/1.1 201 Created  (application/json)
  47 3.850904   SONATA VIM    HTTP   5017 POST /v1/425e1e692db848ed9d1a0f499b73e4e6/stacks HTTP/1.1  (application/json)
  51 5.626922   VIM    SONATA HTTP    691 HTTP/1.1 201 Created  (application/json)
 864 45.724712  SONATA VIM    HTTP   6886 PATCH /v1/425e1e692db848ed9d1a0f499b73e4e6/stacks/SonataService-fa69f8e9-f6f1-4c6e-931a-5b4eee39c633/8ee
 870 47.420545  VIM    SONATA HTTP    337 HTTP/1.1 202 Accepted  (text/plain)
1402 81.386561  SONATA VIM    HTTP   2667 PATCH /v1/425e1e692db848ed9d1a0f499b73e4e6/stacks/SonataService-fa69f8e9-f6f1-4c6e-931a-5b4eee39c633/8ee
1407 82.740891  VIM    SONATA HTTP    337 HTTP/1.1 202 Accepted  (text/plain)
1777 130.084050 SONATA VIM    HTTP    654 GET /v1/425e1e692db848ed9d1a0f499b73e4e6/stacks/SonataService-fa69f8e9-f6f1-4c6e-931a-5b4eee39c633/8eee9
1779 130.429970 VIM    SONATA HTTP   2390 HTTP/1.1 200 OK  (application/json)
1784 132.326119 SONATA WIM    HTTP    261 GET /restconf/config/context/service-interface-point/ HTTP/1.1
1789 132.409040 WIM    SONATA HTTP    980 HTTP/1.0 200 OK  (application/json)
1796 132.493902 SONATA WIM    HTTP    261 GET /restconf/config/context/service-interface-point/ HTTP/1.1
1801 132.561027 WIM    SONATA HTTP    980 HTTP/1.0 200 OK  (application/json)
1809 132.630425 SONATA WIM    HTTP    703 POST /restconf/config/context/connectivity-service/1000/ HTTP/1.1  (application/json)
1814 132.695259 WIM    SONATA HTTP    833 HTTP/1.0 201 CREATED  (application/json)
1822 132.760873 SONATA WIM    HTTP    703 POST /restconf/config/context/connectivity-service/1001/ HTTP/1.1  (application/json)
1827 132.831752 WIM    SONATA HTTP    833 HTTP/1.0 201 CREATED  (application/json)
1835 132.903171 SONATA WIM    HTTP    703 POST /restconf/config/context/connectivity-service/1002/ HTTP/1.1  (application/json)
1840 132.984165 WIM    SONATA HTTP    833 HTTP/1.0 201 CREATED  (application/json)
1848 133.060541 SONATA WIM    HTTP    703 POST /restconf/config/context/connectivity-service/1003/ HTTP/1.1  (application/json)
1853 133.125283 WIM    SONATA HTTP    833 HTTP/1.0 201 CREATED  (application/json)
```

Figure 4.2: Network capture of a Network Service instantiation

4.3. Evaluation of Instantiation on Edge-DC and comparison

A series of tests were executed in order to measure performance and QoS parameters along the WAN, using a similar scenario than the one decipted in Figure 4.1. The results can be found out in Table 4.1, and while them indicate similar results for both electrical-only and electrical-optical-electrical paths, it is worth mentioning that the optical link

	NS A	NS B	NS C
VNF	2	2	2
VLD	1	1	1
NS	1	1	1
Latency	200ms	5ms	200ms
Path	Electrical-optical -electrical	Edge nodes	Electrical
NV tech.	MPLS & wavelength	MPLS	MPLS
RTT (min, avg, max)	0.766/0.985/ 2.586 ms	0.153/0.209/ 4.914 ms	0.562/0.810/ 4.825 ms
Jitter	0.063 ms	0.038 ms	0.072 ms

Table 4.1: NS properties and results

used in this case is 70 km long, so the comparison is not totally fair since ethernet path is not that long. To perform a fair comparison, we shall need a huge number of ethernet hops (which is not realistic) or introduce the additional latency using software to do an equivalent test. Actually, even with that unfair competence, RTT for both client-Core paths are pretty similar, with an average RTT of 0.810 ms for a pure electrical path and 0.985 for a electrical-optical-electrical one, with a difference of less than 0.2 ms, but also both paths are compliant with KPIs.

Nevertheless, when comparing with edge path, latency improvement is noticeable, with an average RTT of only 0.209 ms in the Edge-DC link. This improvement is not very relevant for a video-conference service, but it may become relevant for other services with more stringent requirements, e.g. Vehicle to everything communications (V2X) or IP Multimedia Subsystem (IMS) use cases.

Other relevant QoS parameters were also measured across the WAN: bandwidth, jitter and loss. These measurements have been carried out using *iperf* tool from one of the clients in the left side to each of the DC installed (see Figure 4.1). There are no remarkable differences between measurements using electrical or optical path. The bottleneck when using User Datagram Protocol (UDP) is in the packets per second measure. Using a 1470 byte payload, we have 806 Mbps for both Core-DC and Edge-DC and a jitter of 0.02 ms and 0.26% of packet loss. Transmission Control Protocol (TCP) has better performance, reaching 936 Mbps

4.4. Evaluation of Application results

In our test, it was possible to deploy both NS over the architecture in Figure 4.1 and establish a video-conference in which two users can talk with each other as real-time video through webcam devices.

During the videoconference sessions, several measures of bitrate and packet loss were measured for both audio (Figure 4.3-I) and video (Figure 4.3-II). By looking at the measures, it can be seen that the service reached the desired performance and the values met the expectations, showing just a marginal packet loss of 2 packets until the moment this information was taken. This value is considered negligible for the videoconference service. Besides, the audio bitrate remains by a stable value of 25 kbps for outgoing

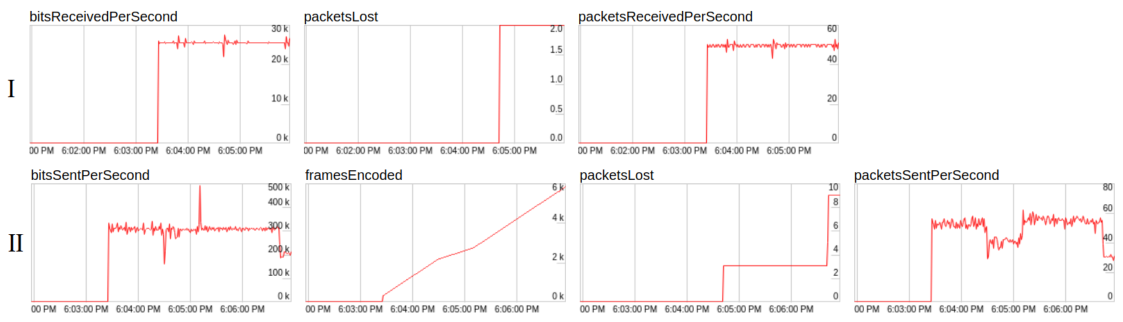


Figure 4.3: WebRTC application graphs captured at the client using a browser

and incoming directions and, the video bitrate remains stable at 300 Kbps for outgoing directions. The peaks observed are due to the different types of frame used in the VP8 video flow.

CONCLUSIONS

Telco and Information Techonology (IT) industries have been searching for several years the open alternative solutions to redesign the current virtualisation model, in order to keep up with the constant growth of the resource-hungry cloud applications market. In this sense, the concept of orchestration has as a disruptive potential approach. It is worth to note that many companies are actively contributing to open solutions by efforts in the standardization process, and developing Open Source SDN and NFV architectures.

Additionally SONATA is pushing forward to have a reliable SP that will have specific novel functionalities not found in other Open Source Cloud Orchestration Platforms. The development of a T-API wrapper has been a contribution that allows SONATA performing placement across a SDN WAN, also adding QoS capabilities and allowing other higher level functionalities that are WIM such as Network Slicing and multi-site placement.

The deployment of a service consisting of several virtual (and physical) network functions, with connectivity/computation/storage requirements, needs a detailed description of all functions, their configurations, deployment characteristics, inter-connections, and QoS requirements. This information is included (as specified by ETSI NFV) in the 'service descriptor' in order to map them over network services. This level of detail goes beyond what is needed for many services.

Software networks are at a transition moment from already existing PoCs in laboratory environments to larger operational deployments. However, multiple barriers exist such as the complexity of initial NFV solutions, concerns about interoperability between them, shortage of adequate skill sets for software networks, difficulty in integrating with legacy, brown-field scenarios, initial performance issues of first generation virtualized infrastructure, etc.

The DevOps model is already an established concept in IT industry enabling a close and agile collaboration between service developers and service operators. Companies that practice DevOps have reported significant benefits, including noticeably shorter time-to-market, improved customer satisfaction, better product quality, more reliable releases, improved productivity and efficiency, and an increased success when building a product by fast experimentation. To apply DevOps to network services, tools for integration, monitoring, and reporting are needed, in order to provide developers with feedbacks from the qualification environment and from operational part.

Furthermore, to develop this thesis, an extensive uses of devops has been done, not only to develop the WIM T-API wrapper, but for other tasks, such as VNF image versioning, node cleanup, real-time monitoring, environment setup and WAN components behaviour testing and debugging.

Besides, it has been proved that Edge placement can have benefits for network services, specially for those use cases which are uRLLC. Thus, application ecosystem on NFV can be extended to a wider range of applications. The measured experimental latency was noticeable, with an average RTT of 0.209 milliseconds, giving the efforts to deploy on the edge extra value, while the extra latency to the core can be critical for mentioned uRLLC, such as autonomous and conected cars, mission critical applications and emergency services.

While development on the component which has been the main output of this thesis work, 5GTANGO´s WP7 Communications Pilot is taking momentum, and now the real test is coming. Some use cases are still not covered, and to have a valuable output and cover them with assurances, for which the video-conference service development time is not yet finished.

Sustainability Considerations

The SDN/NFV approach focuses on the decoupling of control from dataplane, thus it is possible to disconnect nodes that are unused or perform VNF migration to reduce node balancing in favor of less energy consumption, using a 'green' placement policy executed by the NFVO. Moreover, a similar operation could be performed in the WAN by putting into low-consumption mode nodes that are not used, but always mantaining them ready to be used in case of necesity.

However, it must be taken into account the cost and energy requirements of the compute nodes, that could be optimised for a lower power-budget reduction, driven by the open equipment. Also, using more energy efficient edge nodes (that can use ARM processors), or well geo-located nodes to reduce refrigeration costs, should be taken into consideration.

Ethical Considerations

Network security is a critical topic since the developed architecture must preserve the confidentiality and integrity of the application data. In this sense, the creation of well secured VNFs and also secured VLs is key. Thus, companies will rely on a secured platform, remaining their private or compromising data securely stored and also using secured communications. Both SP and video-conference service (VNF-RP) use secure web applications, by the use of certificates and HTTPS/SSL.

Moreover, using Open Source to develop the project, and making public the code and the work done is a good way to show to the scientific and industrial community how the work has been performed, and also make anyone able to contribute to the project.

Future work

There are some future guidelines interesting for coverage, after this thesis, apart from develop the use cases defined by each pilot, which is something that is under the project objectives. One of the possible fields that would be interesting to develop is network service self-healing, not only from the functional point of view but from the monitorization of the dataplane. That would require to develop new software or search for existing solutions to develop over them.

In adition to that, the extended model of T-API is rather limited, with only 5 types of matches. Increasing the number of available matches could lead to more complex usecases and enrich the WIM NBI available capabilities.

Finally, in while in 5GTANGO project there have been some efforts to give support to cloud-native VIMs, more specifically Kubernetes, and proving that cloud-native network functions are instantiated noticeably quicker, an interesting action would be to include a new VIM in the testbed based in Kubernetes, and also extend the WIM wrapper (if necesary) to give support to a new cloud-native PoP. Besides, this innovation line could lead to an hybrid network service that could use some cloud-native blocks and other virtualised, a solution that could be very benefitial to some use cases if it is implemented.

ACRONYMS

3GPP 3rd Generation Partnership Project 34

5G 5th Generation 3, 13, 19, 27, 34, 41

ABNO Application Based Network Orchestrator 15, 25, 30, 31

AMQP Advanced Message Queuing Protocol 26, 35, 38

API Application Programming Interface 9, 10, 12, 14, 16, 24, 29, 32, 42

ARP Address Resolution Protocol 29, 33

AS-PCE Active Stateful Path Computation Element 29, 31

BE Best Effort 19, 22

CAPEX Capital Expenditures 9

CEP Connection End Points 17

CI Continuous Integration 6

CI/CD Continuous Integration/Continuous Delivery 27

CLI Command Line Interface 12, 36

CO Central Office 30, 42

COP Control Orchestration Protocol 29

CRUD Creation, Request Update and Delete 9

DC datacenter 14, 15, 20, 23, 29–31, 42, 43

DevOps Development & Operations 3, 6, 10–12, 45

DF Deployment Flavour 7, 20, 25, 37

DPID OpenFlow Datapath ID 30, 32

DSCP Differentiated Services Code Point 25

DWDM Dense Wavelength Division Multiplexing 41

E2E End-to-End 20, 24, 32

ETSI European Telecommunications Standards Institute 5, 6, 8, 10, 12, 20, 24, 29, 45

EVE Evolution and Ecosystem 6

FSM Function Specific Manager 25, 29, 36

GUI Graphical User Interface 12

HPC high-performance computing 42

HTTP Hypertext Transfer Protocol 34

IA Infrastructure Abstraction 11, 24–27, 38, 39

ICT information and communications technology 5

IETF Internet Engineering Task Force 15

IFA Interfaces and Architecture 6

IMS IP Multimedia Subsystem 43

IP Internet Protocol 28, 29, 32, 33, 38

IPv4 Internet Protocol version 4 33

ISG industry specification group 5, 6

IT Information Techonology 45

KPIs Key Performance Indicators 21, 43

L3 Layer Three 28

LLDP Link Layer Discovery Protocol 29

MANO Management and Orchestration 6–8, 10, 12, 25–27, 38, 40

MPLS Multiprotocol Label Switching 7, 9, 34, 39, 41

MSCS Multi-Site Connectivity Service 9

MSNC Multi-Site Network Connection 9

MSO Multi-Domain SDN Orchestrator 15

NaaS Network as a Service 24

NAT Network Address Translation 28

NBI Northbound Interface 7, 12, 15, 16, 24, 26, 32, 42, 46

NEP Node Edge Points 17

NF Network Function 6, 7

NFV Network Function Virtualisation 3–6, 8, 10, 12, 15, 20, 22, 24, 27, 29, 42, 45, 46

NFVI Network Function Virtualisation Infrastructure 6–10, 20, 24, 25, 27, 31

NFVO Network Function Virtualisation Orchestration 7–10, 15, 23, 24, 26, 32, 36, 46

NS Network Service 3, 5–7, 9, 12, 14, 19–21, 23–27, 34–39, 42, 44

NSD Network Service Descriptor 6, 7, 22, 24, 26, 36, 37

NUC Next Unit of Computing 30

OAM Operations, Administration and management 16

OCC Optical Control-plane Controller 30–32

ODL OpenDaylight 24, 29, 30, 33

ONAP Open Network Automation Platform 10, 13

ONF Open Networking Foundation 15, 16

ONOS Open Networking Operating System 29

OPEX Operation Expenditures 9

OSM Open Source MANO 10, 12

OSS/BSS Operations and Business Support System 7, 8, 24, 32

OXC optical cross-connect 31, 41

PCC Path Computation Client 29

PCE Path Computation Element 15, 16

PCEP Path Computation Element Protocol 29

PNF Physical Network Function 12

PoC proof-of-concept 5, 6, 45

PoP Point of Presence 7–9, 12, 15, 20, 22, 23, 25–27, 29, 32, 36, 38, 39, 47

QoE Quality of Experience 34

QoS Quality of Service 7, 19, 20, 22, 23, 25–27, 35, 37, 41, 43, 45

ROADM reconfigurable optical add-drop multiplexer 31, 41

RTT Round Trip Time 7, 43, 45

SBI Southbound Interface 8, 16, 33

S-BVT sliceable-bandwidth variable transceiver 41

SDN Software Defined Networks 3, 4, 9, 12, 15–17, 19, 23–25, 29, 32, 33, 42, 45, 46

SDOT Software defined optical transmission 41

SIP Service Interface Points 17, 26, 32, 34, 38, 39

SLA Service Level Agreement 10, 11, 20, 37

SNMP Simple Network Management Protocol 35, 36

SP Service Platform 10–12, 19, 20, 22, 24–26, 28, 29, 34–38, 45, 46

SSM Service Specific Manager 29, 36

STP Spanning Tree Protocol 30

SWA Software Architectures 6

T-API Transport Application Programming Interface 16, 17, 24–28, 32, 33, 38, 45, 46

TCP Transmission Control Protocol 43

TED Traffic Engineering Database 15

UDP User Datagram Protocol 43

UUID universally unique identifier 17, 26, 36

V2X Vehicle to everything communications 43

VDU Virtual Deployment Unit 7, 25, 27, 36

VIM Virtualised Infrastructure Manager 7–12, 15, 22, 25, 26, 36–38, 42, 46, 47

VL Virtual Link 7, 9, 22–28, 33, 35, 36, 39, 40, 42, 46, 57

VLD Virtual Link Descriptor 7

VNF Virtualised Network Function 3, 5–7, 9, 12–14, 20–22, 24, 25, 27–29, 32, 34–38, 42, 45, 46

VNFD Virtualised Network Function Descriptor 7, 22, 24, 25, 36

VNFFG VNF Forwarding Graph 6

VNF-M VNF Manager 7, 10

VNTM Virtual Network Topology Manager 15, 16

VTN Virtual Tenant Network 24

VXLAN Virtual Extensible LAN 9, 35

WAN Wide Area Network 3, 9, 12, 22–25, 27, 31, 35, 43, 45, 46

WG Working Group 6

WIM WAN Infrastructure Manager 7, 9–12, 15, 19, 20, 22–33, 35, 37–39, 42, 45–47

WP Work Package 19, 46

XFP 10 gigabit small form factor pluggable transceiver 30, 31

XML Extensible Markup Language 29

BIBLIOGRAPHY

[1] Ultan Mulligan. ETSI - Network Functions Virtualisation (NFV). Online at `https://www.etsi.org/technologies/689-network-functions-virtualisation`. VII, 5, 8

[2] NFV Membership. Online at `https://portal.etsi.org/TBSiteMap/NFV/NFVMembership.aspx`. 5

[3] Technology blogs, NFV, MEC, NGP, ZSM, ENI - NFV#26 - Kicking off the Release 4. Online at `https://www.etsi.org/newsroom/blogs/entry/nfv-26-kicking-off-the-release-4`. 6

[4] ETSI. *Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Network Service Templates Specification*, 4 2019. 6, 7

[5] ETSI. *Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV*, 8 2018. 7

[6] ETSI. *Network Functions Virtualisation (NFV) Release 2; Acceleration Technologies; Management Aspects Specification*, 2 2018. 8

[7] ETSI. *Network Functions Virtualisation (NFV); Acceleration Technologies; Acceleration Resource Management Interface Specification; Release 3*, 7 2017. 8

[8] ETSI. *Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Report on Management and Connectivity for Multi-Site Services*, 4 2018. 9, 24

[9] ETSI. *Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Functional requirements specification*, 4 2019. 9

[10] ETSI. *Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Interface and Information Model Specification for Multi-Site Connectivity Services*, 4 2019. 9, 24

[11] Home | SONATA NFV. Online at `http://www.sonata-nfv.eu/`. 10

[12] 5gtango. Online at `https://www.5gtango.eu/`. 10

[13] Ansible Hat, Red. Ansible is Simple IT Automation. Online at `https://www.ansible.com`. 11

[14] Jenkins. Online at `https://jenkins.io/index.html`. 11

[15] Introduction — master branch documentation. Online at `https://onap.readthedocs.io/en/latest/guides/onap-developer/architecture/onap-architecture.html#{}g-blueprint`. 13

[16] Omar Sefraoui, Mohammed Aissaoui, and Mohsine Eleuldj. Comparison of multiple IaaS Cloud platform solutions. page 6. 14

[17] A. Aguado, V. Lopez, J. Marhuenda, O. Gonzalez de Dios, and J. P. Fernandez-Palacios. Abno: a feasible sdn approach for multivendor ip and optical networks. *Journal of Optical Communications and Networking vol. 7 issue 2, 2015*. 15

[18] ONF. *Transport API (TAPI) 2.0 Features Overview*, 6 2017. Online at `https://www.opennetworking.org/wp-content/uploads/2017/08/TAPI-2.0-Updates-Overview.pdf`. 16

[19] D7.2 implementation of pilots and first evaluation, 2 2019. 19, 34

[20] ETSI. *Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; NFV descriptors based on TOSCA specification*, 5 2019. 20

[21] 5gtango Service Platform Infrastructure Adapter WIM TAPI repository: sonata-nfv/tng-sp-ia-wtapi, June 2019. Online at `https://github.com/sonata-nfv/tng-sp-ia-wtapi`. 27, 44

[22] namespaces(7) - Linux manual page. Online at `http://man7.org/linux/man-pages/man7/namespaces.7.html`. 28

[23] Arturo Mayoral, Ricard Vilalta, Raul Muñoz, Ramon Casellas, and Ricardo Martínez. Sdn orchestration architectures and their integration with cloud computing applications. *Optical Switching and Networking*, 26:2–13, 2017. 29

[24] Myricom 10g-PCIE-8b-S-10g-PCIE-8b-S. Online at `https://shop.stordis.com/adapter-cards/network-interface-cards/myricom-1x-10gbe-sfp+-low-profile-nic-10g-pcie-8b-s`. 30

[25] ONF. *OpenFlow Switch Specification Version 1.3.5 (Protocol version 0x04)*, 3 2015. 32

[26] Specification # 23.501 - System architecture for the 5g System (5gs). Online at `https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3144`. 34

[27] 5gtango Communications pilot repository. Contribute to sonata-nfv/tng-communications-pilot development by creating an account on GitHub, May 2019. Online at `https://github.com/sonata-nfv/tng-communications-pilot`. 34

[28] Janus WebRTC Server: About Janus. Online at `https://janus.conf.meetecho.com/`. 35

[29] Chapter 10. Configure Quality-of-Service (QoS). Online at `https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/14/html/networking_guide/qos`. 35

[30] D5.1 service platform operational first release. 36

[31] 5gtango descriptor, record, and package specifications and schemas (data models): sonata-nfv/tng-schema, June 2019. Online at `https://github.com/sonata-nfv/tng-schema`. 37

[32] R. Muñoz, L. Nadal, R. Casellas, M. S. Moreolo, R. Vilalta, J. M. Fàbrega, R. Martínez, A. Mayoral, and F. J. Vílchez. The ADRENALINE testbed: An SDN/NFV packet/optical transport network and edge/core cloud platform for end-to-end 5g and IoT services. In *2017 European Conference on Networks and Communications (EuCNC)*, pages 1–5, June 2017. 43

[33] A. Mayoral, R. Vilalta, R. Casellas, R. Martinez, and R. Munoz. Multi-tenant 5g Network Slicing Architecture with Dynamic Deployment of Virtualized Tenant Management and Orchestration (MANO) Instances. In *ECOC 2016; 42nd European Conference on Optical Communication*, pages 1–3, September 2016. 44

APPENDICES

APPENDIX A. MATCHING YANG EXTENSION OF T-API SPECIFICATION

The YANG model can be found in Figure A.1.

```
module tapi-match-types {
    namespace "urn:onf:otcc:yang:tapi-match-types";
    prefix tapi-match-types;
    import ietf-inet-types {prefix inet; revision-date "2013-07-15";}
    import ietf-yang-types {prefix yang; revision-date "2013-07-15";}
    import ietf-ethertypes {prefix ethertypes; revision-date "2017-10-03";}
    import tapi-common { prefix tapi-common; }
    import tapi-connectivity { prefix tapi-connectivity; }

    revision "2019-02-19" { description "Initial revision of match types"; }

    augment "/tapi-common:context/tapi-connectivity:connectivity-context/
    ../tapi-connectivity:connectivity-service" {
        container match {
            uses match;
        }
    }
    grouping match {
        leaf ipv4-source {
            type inet:ipv4-prefix;
            description "Ip source address and network prefix.";
        }
        leaf ipv4-target {
            type inet:ipv4-prefix;
            description "Ip target address and network prefix.";
        }
        leaf link-layer-addr-source {
            type yang:mac-address;
            description "Ethernet source address.";
        }
        leaf link-layer-addr-target {
            type yang:mac-address;
            description "Ethernet target address.";
        }
        leaf link-layer-type {
            type ethertypes:ethertype;
            description "Ethernet frame type.";
        }
        description "Matches to be configured in internal
            connections of the connectivity service";
    }
}
```

Figure A.1: Tree format of the extension of T-API YANG model which allow the use of matches in connectivity services

APPENDIX B. T-API WRAPPER PAYLOADS FOR CONFIGURE AND DECONFIGURE

Below it can be found the Schemas for both configurartion (Figure B.1) and deconfiguration (Figure B.2) messages for creation and deletion of VL.

```
service_instance_id: <uuid>
vl_id: <id>
wim_uuid: <uuid>
ingress:
  location: <endpoint or vim uuid>
  nap: <ip segment>
egress:
  location: <endpoint or vim uuid>
  nap: <ip segment>
qos:
  bandwidth: <integer>
  bandwidth_unit: <unit, if absent default Mbps>
  latency: <integer>
  latency_unit: <unit, if absent default ms>
bidirectional: boolean
```

Figure B.1: Payload for the configuration message during a Network Service instantiation

```
service_instance_id: <uuid>
vl_id: <id>
wim_uuid: <uuid>
```

Figure B.2: Payload for the deconfiguration message during a Network Service termination