UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Integrated IT and SDN orchestration of multi-domain multi-layer transport networks

Arturo Mayoral López de Lerma

Integrated IT and SDN Orchestration of multi-domain multi-layer transport networks

TESIS
para la obtención del título de
Doctor por la Universitat Politècnica de Catalunya

por
Arturo Mayoral López de Lerma

Co-directores: Dr. Ricard Vilalta Cañellas, Dr. Raül Muñoz González
Centre Tecnològic de Telecomunicacions de Catalunya - CTTC/CERCA

Ponent: Dr. Gabriel Junyent Giralt
Universitat Politècnica de Catalunya
Departament de Teoria del Senyal i Comunicacions

Composició del Tribunal

President: David Larrabeiti López
Secretari: Salvatore Spadaro
Vocal: Ignacio De Miguel Jimenez

Para mis padres.

Summary

Telecom operators (Telcos) networks' management and control remains partitioned by technology, equipment supplier and networking layer. In some segments (e.g., in IP) the network operations are highly costly due to the need of individual, and even manual, configuration of the network equipment by highly specialized personnel. In multi-vendor networks, extremely expensive and never ending integration processes between Network Management Systems (NMSs) and the rest of systems (e.g., Operations Support Systems - OSSs and Business Support Systems - BSSs) is a common situation, due to lack of adoption of standard interfaces in the management systems of the different equipment suppliers. Moreover, the increasing impact, on Telco's transport networks, of the new traffic flows introduced by the deployment of massive Data Centers (DCs) is also imposing new challenges that traditional networking is not ready to overcome.

To complete the picture, the Fifth Generation of Mobile Technology (5G) is also introducing stringent network requirements such as the need of connecting to the network billions of new devices in Internet of Things (IoT) paradigm, new ultra-low latency applications (i.e., remote surgery) and vehicular communications. All these new services, together with enhanced broadband network access, are supposed to be delivered over the same network infrastructure.

In this PhD Thesis, an holistic view of Network and Cloud Computing resources, based on the recent innovations introduced by Software Defined Networking (SDN), is proposed as the solution for designing an end-to-end multi-layer, multi-technology and multi-domain cloud and transport network management architecture, capable to offer end-to-end service management and control, from the DC networks to customers access networks, and the virtualization of network resources, opening the door to new ways of slicing the network resources for the forthcoming 5G deployments.

The first contribution of this PhD Thesis deals with the design and validation of different SDN-based network orchestration architectures capable to improve the current state-of-the art solutions for the management and control of multi-layer, multi-domain backbone transport networks. The multi-layer and multi-domain network problems have been assessed and progressively solved by different control and management architectures which has been designed and evaluated in real evaluation environments. One of the major findings of this work has been the need of developed a common information model for transport network's management, capable to describe the resources and services of multi-layer networks. In this line, the Control Orchestration Protocol (COP) has been proposed in this Thesis, as a first contribution towards an standard management interface based on the main principles driven by SDN.

Furthermore, this PhD Thesis introduces a novel architecture capable to coordinate the management of IT computing resources together with inter- and intra-Data Center (DC) networks. The provisioning and migration of virtual machines together with the dynamic reconfiguration of the network has been successfully demonstrated in a feasible timescale. Moreover, a resource optimization engine is introduced in the architecture to introduce optimization algorithms capable to solve different allocation problems such the optimal deployment of Virtual Machine Graphs (VMGs) over different DCs locations minimizing the inter-DC network resources allocation. A solution for this problem is also presenting and a baseline blocking probability measurement over different network loads was obtained.

The third major contribution is the result of the previous two. With a converged cloud and network infrastructure controlled and operated jointly, the holistic view of the network allows the on-demand provisioning of network slices consisting of dedicated network and cloud resources over a distributed DC infrastructure interconnected by an optical transport network. The last chapters of this thesis discuss the management and orchestration of 5G network slices based over the control and management components designed and developed in the previous chapters. The design of one of the first network slicing architectures and the deployment of a fully operational 5G network slice in a real Testbed, is one of the major contributions of this thesis.

Keywords: Software-Defined Networking (SDN), GMPLS/PCE, Cloud computing, SDN Orchestration, Network Virtualization, 5G Network Slicing.

Resum

La gestió i el control de les xarxes dels operadors de telecomunicacions (Telcos), encara avui, està segmentat per tecnologia, per proveïdors d'equipament i per capes de xarxa. En alguns segments (Per exemple en IP) l'operació de la xarxa és tremendament costosa, ja que en molts casos encara es requereix de configuració individual, i fins i tot manual, dels equips per part de personal altament especialitzat. En xarxes amb múltiples proveïdors, els processos d'integració entre els Sistemes de gestió de xarxa (NMS) i la resta de sistemes (per exemple, Sistemes de suport d'operacions - OSS i Sistemes de suport de negocis - BSS) són habitualment interminables i extremadament costosos a causa de la falta d'adopció d'interfícies estàndard per part dels diferents proveïdors de xarxa. A més, l'impacte creixent en les xarxes de transport dels nous fluxos de trànsit introduïts pel desplegament massius de Data Centers (DC), introdueix nous desafiaments que les arquitectures de gestió i control de les xarxes tradicionals que no estan llestes per afrontar.

Per acabar de descriure el context, la cinquena generació de tecnologia mòbil (5G) també presenta nous requisits de xarxa altament exigents, com la necessitat de connectar a la xarxa milers de milions de dispositius nous, dins el context de l'Internet de les coses (IOT), o les noves aplicacions d'ultra baixa latència (com ara la cirurgia a distància) i les comunicacions vehiculars. Se suposa que tots aquests nous serveis, juntament amb l'accés millorat a la xarxa de banda ampla, es lliuraran a través de la mateixa infraestructura de xarxa.

Aquesta tesi doctoral proposa una visió holística dels recursos de xarxa i cloud, basada en els principis introduïts per Software Defined Networking (SDN), com la solució per al disseny de una arquitectura de gestió extrem a extrem per a escenaris de xarxa multi-capa, multi-domini i consistents en múltiples tecnologies de transport. Aquesta arquitectura de gestió i control de xarxes transport i recursos IT, ha de ser capaç d'oferir serveis d'extrem a extrem, des de les xarxes intra-DC fins a les xarxes d'accés dels clients i oferir a més virtualització dels recursos de la xarxa, obrint la porta a noves formes de segmentació a les xarxes de transport i la infrastructura de cloud, pels propers desplegaments de 5G.

La primera contribució d'aquesta tesi doctoral consisteix en la validació de diferents arquitectures d'orquestració de xarxa basades en SDN capaces de millorar les solucions existents per a la gestió i control de xarxes de transport troncals multi-domini i multicapa. Aquests problemes (gestió de xarxes multicapa i multi-domini), han estat avaluats de manera incremental, mitjançant el disseny i l'avaluació experimental, en entorns de proves reals, de diferents arquitectures de control i gestió. Un dels principals troballes d'aquest treball ha estat la necessitat de dissenyar un model d'informació

comú per a les interfícies de gestió de xarxes, capaç de descriure els recursos i serveis de la xarxes transport multicapa. En aquesta línia, el Protocol de Control Orchestration (COP, en les seves sigles en anglès) ha estat proposat en aquesta Tesi, com una primera contribució cap a una interfície de gestió de xarxa estàndard basada en els principis bàsics de SDN.

A més, en aquesta tesi presentem una arquitectura innovadora capaç de coordinar la gestió de els recursos IT juntament amb les xarxes inter i intra-DC. L'aprovisionament i la migració de màquines virtuals juntament amb la reconfiguració dinàmica de la xarxa, ha estat demostrat amb èxit en una escala de temps factible. A més, l'arquitectura incorpora una plataforma per a l'execució d'algorismes d'optimització de recursos, capaços de resoldre diferents problemes d'assignació, com el desplegament òptim de Grafs de Màquines Virtuals (VMG) en diferents ubicacions de DC que minimitzen la assignació de recursos de xarxa entre DC. També es presenta una solució bàsica per a aquest problema, així com els resultats de probabilitat de bloqueig per a diferents càrregues de xarxa.

La tercera contribució principal és el resultat dels dos anteriors. Amb una infraestructura de xarxa i cloud convergent, controlada i operada de manera conjunta, la visió holística de la xarxa permet l'aprovisionament sota demanda de "network slices" que consisteixen en subconjunts de recursos d' xarxa i cloud, dedicats per a diferents clients, sobre una infraestructura de Data Centers distribuïda i interconnectada per una xarxa de transport òptica. Els últims capítols d'aquesta tesi tracten sobre la gestió i organització de "network slices" per a xarxes 5G en funció dels components de control i administració dissenyats i desenvolupats en els capítols anteriors. El disseny d'una de les primeres arquitectures de "network slicing" i el desplegament d'un "slice" de xarxa 5G totalment operatiu en un Testbed real, és una de les principals contribucions d'aquesta tesi.

Paraules clau: Xarxes definides per programari (SDN), GMPLS / PCE, Cloud computing, OpenFlow, orquestració SDN, Network Virtualization, Particionament de la Xarxa (Network Slicing).

Resumen

La gestión y el control de las redes de los operadores de telecomunicaciones (Telcos), todavía hoy, está segmentado por tecnología, por proveedor de equipamiento y por capa de red. En algunos segmentos (por ejemplo en IP) la operación de la red es tremendamente costosa, ya que en muchos casos aún se requiere configuración individual, e incluso manual, de los equipos por parte de personal altamente especializado. En redes con múltiples proveedores, los procesos de integración entre los Sistemas de gestión de red (NMS) y el resto de sistemas (por ejemplo, Sistemas de soporte de operaciones - OSS y Sistemas de soporte de negocios - BSS) son habitualmente interminables y extremadamente costosos debido a la falta de adopción de interfaces estándar por parte de los diferentes proveedores de red. Además, el impacto creciente en las redes de transporte de los nuevos flujos de tráfico introducidos por el despliegue masivo de Data Centers (DC), introduce nuevos desafíos que las arquitecturas de gestión y control de las redes tradicionales no están preparadas para afrontar.

Para acabar de describir el contexto, la quinta generación de tecnología móvil (5G) también introduce nuevos requisitos de red altamente exigentes, como la necesidad de conectar a la red miles de millones de dispositivos nuevos, dentro del contexto del Internet de las cosas (IoT), o las nuevas aplicaciones de ultra baja latencia (como por ejemplo la cirugía a distancia) y las comunicaciones vehiculares. Se supone que todos estos nuevos servicios, junto con el acceso mejorado a la red de banda ancha, se entregarán a través de la misma infraestructura de red.

Esta tesis doctoral propone una visión holística de los recursos de red y cloud, basada en los principios introducidos por Software Defined Networking (SDN), como la solución para el diseño de una arquitectura de gestión extremo a estremo para escenarios de red multi-capa, multi-dominio y consistentes en múltiples tecnologías de transporte. Dicha arquitectura de gestión y control de redes transporte y recursos IT, debe ser capaz de ofrecer servicios de extremo a extremo, desde las redes intra-DC hasta las redes de acceso de los clientes y ofrecer ademas virtualización de los recursos de la red, abriendo la puerta a nuevas formas de segmentación en las redes de transporte y la infrastructura de cloud, para los próximos despliegues de 5G.

La primera contribución de esta tesis doctoral consiste en la validación de diferentes arquitecturas de orquestración de red basadas en SDN, capaces de mejorar las soluciones existentes para la gestión y control de redes de transporte troncales multi-dominio y multi-capa. Estos problemas (gestion de redes multi-capa y multi-dominio), han sido evaluados de manera incremental, mediante el diseño y la evaluación experimental, en entornos de pruebas reales, de diferentes arquitecturas de control y gestión. Uno de los principales hallazgos de este trabajo ha sido la necesidad de diseñar un modelo de

información común para las interfaces de gestion de redes, capaz de describir los recursos y servicios de la redes transporte multi-capa. En esta línea, el Protocolo de Control Orchestration (COP, en sus siglas en inglés) ha sido propuesto en esta Tesis, como una primera contribución hacia una interfaz de gestión de red estándar basada en los principios basicos de SDN.

Además, en esta tesis presentamos una arquitectura novedosa capaz de coordinar la gestión de los recursos IT junto con las redes inter e intra-DC. El aprovisionamiento y la migración de máquinas virtuales junto con la reconfiguración dinámica de la red, ha sido demostrado con éxito en una escala de tiempo factible. Además, la arquitectura incorpora una plataforma para la ejecución de algoritmos de optimización de recursos, capaces de resolver diferentes problemas de asignación, como el despliegue óptimo de Grafos de Máquinas Virtuales (VMG) en diferentes ubicaciones de DC que minimizan la asignación de recursos de red entre DC. También se presenta una solución básica para este problema, así como los resultados de probabilidad de bloqueo para diferentes cargas de red.

La tercera contribución principal es el resultado de las dos anteriores. Con una infraestructura de red y cloud convergente, controlada y operada de manera conjunta, la visión holística de la red permite el aprovisionamiento bajo demanda de "network slices" que consisten en sub-conjuntos de recursos de red y cloud dedicados para diferentes clientes, sobre una infraestructura de DCs distribuida e interconectada por una red de transporte óptica. Los últimos capítulos de esta tesis discurren sobre la gestión de estos "network slices" para redes 5G en función de los componentes de control y administración diseñados y desarrollados en los capítulos anteriores. El diseño de una de las primeras arquitecturas de "network slicing" y el despliegue de un "slice" de red 5G totalmente operativo en un Testbed real, es una de las principales contribuciones de esta tesis.

Palabras clave: Redes definidas en Software (SDN), GMPLS / PCE, Cloud computing, OpenFlow, Orquestración SDN, Network Virtualization, Particionamiento de la Red (Network Slicing).

Contents

List of Figures

List of Tables

Abbreviations

ABNO	Applications-based Network Operations
API	Application Programming Interface
ADC	Analog to Digital Converter
BBU	BaseBand Unit
BER	Bit Error Ratio
BGP	Border Gateway Protocol
BSS	Business Support System
CAPEX	Capital Expenditures
CDN	Content Delivery Network
CLI	Command Line Interface
COP	Control Orchestration Protocol
COTS	Commercial Off The Shelf
CRAN	Cloud Radio Access Network
DAC	Digital to Analog Converter
DC	Data Center
DEMUX	Demultiplexer
DP	Dual Polarization
DSP	Digital Signal Processing
DWDM	Dense Wavelength Division Multiplexing
E2E	End-to-End
EDFA	Erbium Doped Fibre Amplifier
EON	Elastic Optical Network
ERO	Explicit Route Object
FEC	Forward Error Correction
GMPLS	Generalized Multi Protocol Label Switching
HT	Holding Time
HW	HardWare
IaaS	Infrastructure as a Service
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
LDP	Label Distribution Protocol
LLDP	Link Layer Discovery Protocol
LSP	Label Switched Path

IT	Information Technology
ITU	International Telecommunications Union
JSON	JavaScript Object Notation
LAN	Local Area Network
MEC	Mobile Edge Computing
MEF	Metro Ethernet Forum
MEMS	Micro-Electro Mechanical Systems
MPLS	Multi Protocol Label Switching
MSO	Multi-domain SDN Orchestrator
MTU	Maximum Transmission Unit
MUX	Multiplexer
NBI	NorthBound Interface
NFV	Network Function Virtualization
NFVi	Network Function Virtualization infrastructure
OCS	Optical Circuit Switching
ODU	OTN Data Unit
ODL	OpenDayLight
OEO	Optic-Electric-Optic
OF	OpenFlow
OFDM	Orthogonal Frequency Division Multiplexing
OFDMA	Orthogonal Frequency Division Multiple Access
OIF	Optical Internetworking Forum
OMS	Optical Multiplex Section
ONF	Open Networking Foundation
OPEX	Operational Expenditures
OPM	Optical Performance Monitor
OPS	Optical Packet Switching
OSI	Open Systems Interconnection
OSNR	Optical Signal to Noise Ratio
OSPF	Open Shortest Path First
OSS	Operations Support System
OTN	Optical Transport Network
OTSi	Optical Tributary Signal
OTSiA	Optical Tributary Signal Assembly
OTS	Optical Transmission Section
OTU	Optical Transport Unit
OVS	Open Virtual Switch
OXC	Optical Cross Connect
PCE	Path Computation Element
PCEP	Path Computation Element Protocol
PoP	Point of Presence
PWE	PseudoWire Emulation Edge to Edge
PXC	Photonics Cross-Connects
QAM	Quadrature Amplitude Modulation
QoS	Quality of Service
QPSK	Quadrature Phase Shift Keying
REST	Representational State Transfer
ROADM	Reconfigurable Optical Add/Drop Module
RoI	Return of Investment
RPC	Remote Procedure Call

RSVP	Resource Reservation Protocol
SAL	Services Abstraction Layer
SDH	Synchronous Digital Hierarchy
SDN	Software Defined Networking
SDO	Standards Defining Organization
SLA	Service Level Agreement
SNMP	Simple Network Management Protocol
SONET	Synchronous Optical NETworking
TDM	Time Division Multiplexing
TE	Traffic Engineering
TED	Traffic Engineering Database
URI	Uniform Resource Identifier
VM	Virtual Machine
VNF	Virtualized Network Function
VON	Virtual Optical Network
VPN	Virtual Private Network
VTN	Virtual Tenant Network
WDM	Wavelength Division Multiplexing
WSON	Wavelength Switched Optical Networks
WSS	Wavelengths Selective Switches
YANG	Yet Another Next Generation
XML	eXtensible Markup Language

Acknowledgements

In the first place I would like to thank my parents Raquel y Carlos for all the support given during all these years of growth, new challenges and hard work. They always have had the right word to encourage me to continue pushing harder when it was needed. I can just thanks them for teaching that hard work, passion and perseverance are they right ingredients to succeed in anything you want to achieve.

This thesis would not have been possible without the help, support and patience of my PhD Advisors, Dr. Ricard Vilalta and Dr. Raul Muñoz, I am really grateful for all what I learned from them during these years. It would also have not been possible without the help of the rest of the Optical Networks and Systems Department colleagues: Dr. Ramon Casellas, Dr. Ricardo Martínez, Dr. Michela Svaluto, Dr. Josep M. Fàbrega, Dr. Laia Nadal, Dr. Laura Martín and Javier Vílchez, and also the rest of colleagues and friends at the CTTC.

I would like to acknowledge Dr. Gabriel Junyent for his help and guidance in Universitat Politècnica de Catalunya (UPC).

I won't be here writing these lines without the guidance of my M.Sc. advisor Victor López, who introduced me in the research world. Not only he was an excellent supervisor who teach me a lot about work ethic, passion for technology, leadership and compromise, but he is also a friend who has always helped me during all this journey, thanks. I also want to thanks to Alejandro Aguado, Oscar Gonzalez and Juan Pedro Fernandez-Palacios, who also helped me from the very beginning of my professional career until today.

Also I would like to thanks my cousins Manuel and Nico with who I shared my concerns, my ambitions and all the amazing moments that only happens when you live together. I gladly share this Thesis with you brothers, as I know that part of the merit to arrive here is yours.

I also want to thanks to Chiara with who I shared this difficult path that is the PhD. I wish you the best luck with the last steps you miss in yours (PhD).

I wouldn't want to finish these lines without greeting all my friends from Madrid and Barcelona without who I would get crazy during this path. Jose, Javi, Cantero, Ricardo, Leti, Inigo, Ausin, Hector, Onur, Achileas, Juanma, Mikel, Marco, Joan, Xantal and the ones I miss but I love, thanks.

Madrid, Spain - May 15, 2019 Arturo Mayoral

Part I

Introduction

Chapter 1

Background and Motivation

To provide a clear perspective of the problems and challenges this thesis addresses, this section will firstly introduce the motivation and later the background work which support this PhD Thesis. As part of the background we will first introduce the transport networks technology and later the current control plane technologies applied to the transport networks nowadays. After identifying the context for the application of SDN Orchestration, the current management solutions employed to provide E2E connectivity services across the network will also be discussed. Later on the current Cloud computing paradigm is presented and linked to one of the main motivations of this work which is the holistic management and control of cloud and network resources for seamless Data Center's (DC) resources interconnection and migration. To conclude this chapter, network virtualization is presented as the last technology enablers towards the fifth generation of mobile technology (5G).

1.1 Motivation

Network operators face a continuous growth of the traffic demand which is not reflected in an increase of their revenues at the same pace. As a consequence, one the their main objective in the last years has been the reduction in their Capital Expenditures (CAPEX) and Operational Expenditures (OPEX), to maintain their competitiveness at the same time they invest in other market sectors with a better Return of Investment (RoI), as content delivery, digital services or data analysis. Accordingly, the trend in the industry has been to simplify network's architectures by converging the different services offered over the same network infrastructures. The digitization of voice services and its convergence into the IP data networks is a good example that has effectively contributed to this objective.

Transport networks remain managed partitioned by technology and also by the equipment supplier. The main reasons of this fragmentation are: a) the different nature of different transport technologies (microwaves, optics) and communication layers (L0-L3 in the OSI model [3]): b) the lack of adoption of standardized management interfaces by different equipment providers: and c) the lack of inter-operability among different control solutions. On the other hand, the control of these networks is tightly coupled with the switching hardware, implying that every network element has to be configured independently, in most cases by human intervention, resulting in a highly complex and extremely slow network operations. These two realities are the main motivations of emerging Software Defined Networking (SDN), a paradigm shift, aimed to change the traditional way of controlling and manage the networks. SDN has been designed and developed during the last decade by the networking industry, starting by the scientific community and progressively being adopted by tier 1 network operators supplied by large network equipment providers.

Another crucial aspect of the transformation process of the network is the emergence of the virtualization technologies and its massive adoption. The increasing impact, on operator's transport networks, of the new traffic flows introduced by the deployment of massive Data Centers (DCs) is also imposing new challenges that traditional networking is not ready to overcome. For instance, the dynamic nature of cloud-based services requires to decrease the time-to-market deployment of new networking services and/or to adapt existing connectivity services' capacity to the actual traffic demand in a flexible and dynamic way. These traffic flows may traverse heterogeneous network domains from the metro/aggregation segments or intra-DC networks, with fine-grained packet-based traffic control requirements, to long-haul optical transport networks with carrier-grade, multi-domain control requirements. This reality introduces the need of introducing end-to-end (E2E) service management and control, which is one of the central concepts which are going to be assessed in this thesis. An E2E transport service is defined here as a cross-domain (across different technology network domains) service between Layer 2 service points .

On the other hand, Network Function Virtualization (NFV) has emerged as a logical step in this network transformation process, which fundamentally consists on removing networking functions such Access Control Lists (ACLs), firewalls, load balancers, from dedicated hardware and implement them as software appliances, which can be virtualized and deployed in the cloud. The clear objective is to reduce the number of different network elements present in the networks today and making them more homogeneous and easy to operate. Equally important are the inherent benefits introducing the virtualization on scalability and robustness, allowing network functions to be deployed, replicated or migrated, where and when are needed. SDN again plays a fundamental role in providing the required dynamism and flexibility to interconnect NFV appliances in distributed Data Centers (DCs), and it is also another important problem being assessed in this thesis.

The increasing impact in operators transport networks traffic flows introduced by the deployment of massive DCs, driven by the lower operational costs introduced by computing and storage infrastructure virtualization and the massive demand of cloud-based services, is also imposing to network

operators new challenges which current network architectures are not ready to overcome. For instance, the dynamic nature of cloud-based services requires to decrease the time-to-market deployment of new networking services and/or to adapt existing connectivity services capacity to the actual traffic demand in a flexible and dynamic way. Moreover, these traffic flows may traverse heterogeneous network domains from the metro/aggregation segments or intra-DC networks, with fine-grained packet-based traffic control requirements, to long-haul, optical transport networks with carrier-grade, multi-domain control requirements. This reality introduces the need of introducing E2E service management and control, which is one of the central concepts which are studied in this work.

In summary, the convergence of different technologies under an integrated E2E network control and management architecture has been identified by network operators, as the one of the critical points of action to enhance network operability and automation, and it is also the fundamental aspect on which this thesis is focused. This first chapter will introduce first the traditional network architecture before the emergence of SDN and NFV technologies, to give to the reader the background and starting point of the work we are presenting in this thesis.

1.2 Traditional Internet and Transport Network architectures

Large scale networks are becoming very complex and expensive to operate with the current control and management technologies. Traditionally, high capacity and reliable delivery of connectivity services in metro, aggregation and core segments of the network have relied upon over-provisioning of the network capacity and periodic (and static) network planning phases.

Current transport networks are stratified in two different layers: (i) Layer 3 networks based on the Internet Protocol (IP) and Multi-protocol Label Switching (MPLS) control, are responsible for cost effective user traffic delivery, fostering the statistical multiplexing nature of packet networks; and (ii) Layer 1/ Layer 0 circuit-based transport networks, based on Time-Division Multiplexing (TDM) and Wavelength Division Multiplexing (WDM) technologies, which provide high capacity, point-to-point connectivity circuits to the upper layer network.

These two networks have traditionally remained separated from the operators perspective, they are planned, designed and operated independently, by separated departments with none or scarce collaboration between them. The main consequences of this approach are:

- Over provisioning of the IP Layer links derived by the need of protecting the traffic against failures and packet losses. It leads in a low exploitation of the optical network links (30-40%) [4].
- Costly and complex network operation due to the lack of automatic programmability of the network, thus leading into long time-to-market for new network services to be deployed.

Thus, the IP over WDM architecture (Figure 1.1) has been the preferred approach followed by Telecom operators to increase network capacity in core transport networks. The IP traffic is routed over MPLS tunnels for traffic engineering (TE) and QoS purposes, which are transported into SONET/SDH or Optical Transport Network (OTN) frames for extended OAM capabilities, and then carried on a dedicated wavelength through DWDM technology. This solution allows the progressive introduction of point-to-point optical DWDM links, with IP routers connected to DWDM-based transponders transmitting IP traffic over dedicated wavelengths.

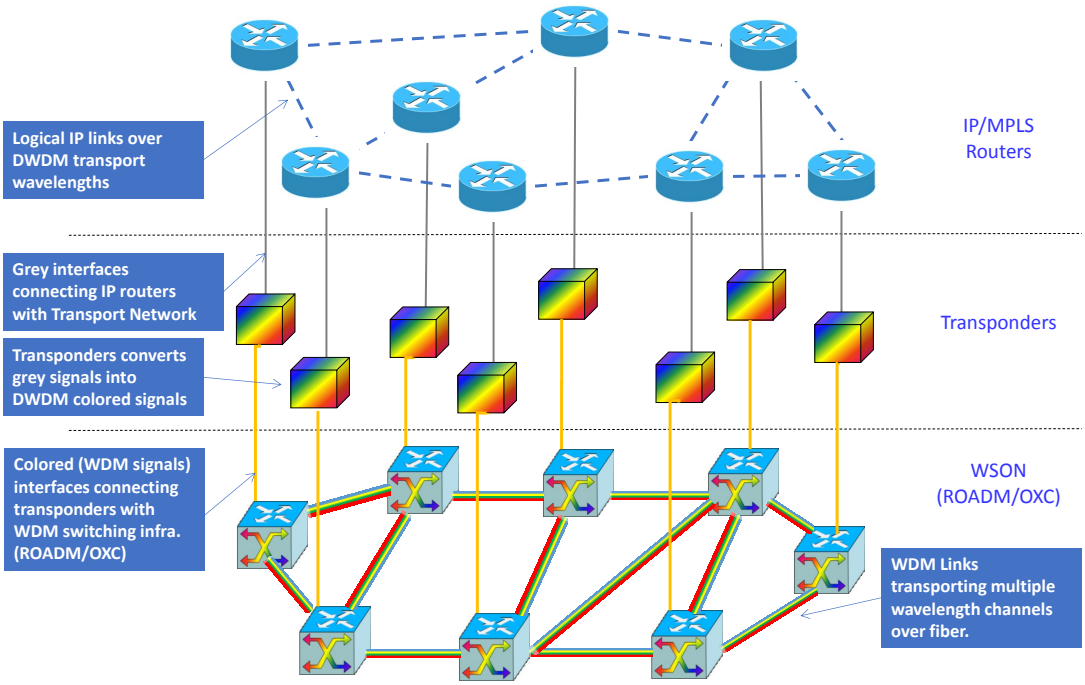


Figure 1.1: IP over WDM transport network architecture.

1.2.1 Wavelength Switched Optical Networks

The main objective of transport networks is to provide connectivity from geographically diverse locations. Transport networks support multiple client networks such IP core network, Mobile Backhaul (MBH) or private enterprise lines/networks.

The transport network itself is subdivided into multiple layers, in the bottom the WDM physical layer is built over optical fibers operating in the C-Band (1530 nm-1565 nm) or L-band (1565 nm-1625 nm), providing up to 160 wavelength channels on Dense WDM (DWDM) systems or even more when flexible ITU-T grid is supported . The G.694.1 ITU-T [5] defines the spectrum division of the optical bands for WDM applications (grids), being the most common channel spacing options for DWDM systems 50 and 100GHz.

This optical transport architecture allows to bypass the IP/MPLS layer and eliminates the need of costly Opto-Electro-Optical (OEO) transponders by introducing new transparent optical switching technologies, such as, Optical Cross-Connects (OXC) and Re-configurable Optical Add Drop Multiplexers (ROADM) [6]. An OXC is an optical node able to switch DWDM channels among different fibers, demultiplexing the optical channels from any incoming fiber and routing them to the required output port. Optical switching is performed by optical filters generally based on Micro-Electrical Mirror Systems (MEMS)[7] or Liquid-Crystal On Silicon (LCoS)[8] technologies. A more advance optical node is the ROADM which adds the capability of Add-and-Drop DWDM channels from the incoming/outgoing fibers.

These emerging optical systems, in combination with the adoption of advanced modulation formats and tunable transceivers, have allowed that optical connections can be automatically switched entirely within the optical domain between source and destination nodes, optically bypassing the IP/MPLS routers. Accordingly, the processing at higher layers is avoided at the intermediate routers, delegated to those points where the header processing is needed. Therefore, this new dynamic network

infrastructure delivers the high-bandwidth transport and deterministic performance of the optical circuit switched technology (i.e., WSON) along with the efficient aggregation and statistical multiplexing of a packet switched technology (e.g., IP/MPLS).

However, the main drawback of this solution is that all the routing capabilities rely in the IP layer, thus increasing the number of IP router ports and pushing packet processing capacity of expensive IP routers.

1.2.1.1 Optical Transport Networks

Often network providers want to provide sub-wavelength capacity for client connections, given that not always all the wavelength bandwidth capacity (10, 40, 100 and up to 400Gbps nowadays) is consumed by a single client connection. This flexibility can be introduced by TDM technology.

The Optical Transport Network (OTN) standard, defined by the G.872 ITU-T [9] recommendation, comprises the transport, aggregation, routing, supervision and survivability of client signals that are processed in the digital domain and carried across the optical domain. The OTN architecture is divided on three layers:

- The digital layer, subdivided into Optical Data Unit (ODU) and Optical Transport Unit (OTU) sub-layers, provides a framework for the multiplexing of digital signal and its mapping into optical signals.

- The optical signal layer, defines the Optical Tributary Signal (OTSi) an the Optical Tributary Signal Assembly (OTSiA) constructs, which are management/control abstractions representing one or multiple optical signals transporting a single OTU.

- The media layer, includes the Optical Multiplex Sections (OMS), which represents the link between two ports of entities able to multiplex/de-multiplex OTSi, and the Optical Transmission Sections (OTS), which represents the lower layer abstraction of the photonic network infrastructure which is the output media port of one amplifier and the input media port of the next amplifier. Finally, the media layer also includes the media constructs which allows the definition of the media channels, which are defined as a serial concatenation of frequency slots, allowing the transmission of an OTSi/OTSiA end-to-end across the media layer.

In brief, the OTN architecture provides a formal definition for the multiplexing and mapping of digital payloads into optical signals, and its transmission along the photonic media layer.

The enhancements introduced by OTN are better illustrated by an example (Figure 1.2). In the figure, two client digital services (DS_1, DS_2) starting at Router 1 are multiplexed and mapped into a single wavelength λ_1, thus saving one wavelength thanks to OTN multiplexing. At Router 2, DS_1 and DS_2 are de-multiplexed, while DS_1 has reached its destination and it is delivered to Router 2, DS_2 is multiplexed again together with DS_3 in the OTN switch and transported over a new wavelength (λ_2) towards Router 3, where both signals are de-multiplexed and delivered to their final destination Router 3. By introducing the OTN switching layer, the service deliver is achieved with two tributary wavelengths instead of three (an express wavelength between Router 1 and Router 3 is saved).

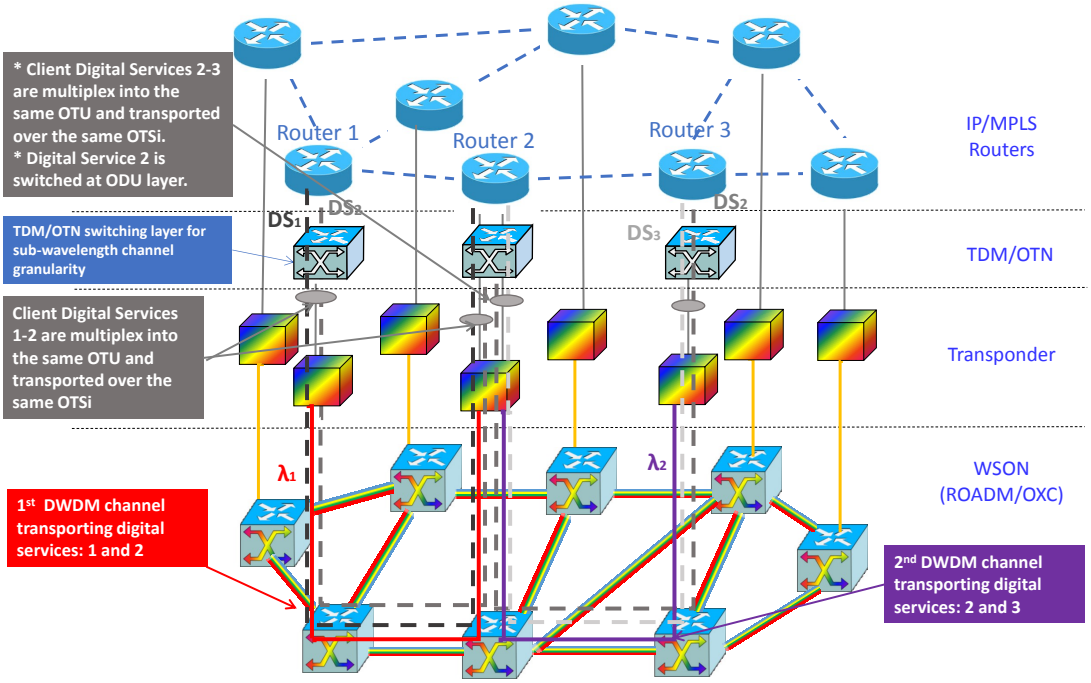


Figure 1.2: IP over OTN/WDM transport network architecture.

1.2.1.2 Elastic Optical Networks

The rigid granularity driven by the DWDM fixed ITU-T wavelength grid present two main problems for future networks. Firstly, fixed assignment of spectral resources to the optical channels led into a poor utilization of the network equipment when the traffic flow through them is low. And secondly, the maximum traffic rate an optical channel can achieve, when the bandwidth assigned to it, is fixed and limited by the modulation schemes can be applied to transmission depending on the maximum reach the of the light-path without signal degradation.

Driven by these limitations and the ever-increasing bandwidth demand into the transport networks, in 2012 the ITU-T G.694.1 [5] was updated to define a new flexible grid with a lower central frequency spacing (6.25GHz), the required amount of optical bandwidth can be dynamically and adaptively allocated in multiples of a given slot width granularity (12.5 GHz), determined by the signal modulation format and its data rate. This new grid is known as Flexible Grid or Flex-grid and it has led the development of a new optical architecture know as Switched Spectrum Optical Networks (SSON) or a more widespread term Elastic Optical Networks (EONs).

EONs have been extensively researched in the last decade [10][11]. New transmission schemes have been explored for the realization of the EON concepts, i.e. Orthogonal Frequency Division Multiplexing (OFDM) and Nyquist DWDM [12][13]. Also, the problem of computing a path/route and allocate spectrum resources (i.e., contiguous frequency slots) known as Routing and Spectrum Allocation (RSA), has been addressed thoughtfully in the last years [14]. Based on these works some analysis has shown the potential CAPEX reduction of introducing Flexgrid technology in the following years [15].

1.2.2 IP/MPLS Networks

The IP layer is composed by independent networks or routing domains interconnected through distributed routing protocols such External Border Gateway Protocol (E-BGP) which allows to adver-

tise IP addressing reach-ability information and to choose routes across routing domains, known as Autonomous Systems (AS) [16]. The objective of this section is not to present the whole internet architecture but to have a closer look on how IP networks are managed.

Each router composing an IP network implements a at least an Interior Gateway Protocols (IGP) such Open Shortest Path First (OSPF) or Intermediate System to intermediate System (IS-IS), for the exchange of routing information, one or more signaling protocols, which allow the reservation of resources for service provisioning, such Label Distribution Protocol (LDP) or Resource Reservation Protocol (RSVP). All these protocols are implemented on each IP router constructing a fully distributed and automated set of control mechanism. Thus both the control and the data planes are integrated into the same box, as we were mentioning in the previous sections.

However a router is not autonomous until it is configured. The router's configuration can be divided in three steps: (i) startup configuration, which is done when the router is installed and consists on configuring the management interface to operate it remotely; (ii) initial configuration, which consist on configuring the basic set of routing protocols (OSPF, BGP, LDP...) and the initially connected interfaces. Last step (iii), is the service's configuration which involve the creation of network interfaces (e.g., VLANs) and the configuration of tunneling or overlay control mechanisms such MPLS tunnels or pseudorwires for L2VPNs or Virtual Routing Instances (VRFs) for L3VPN services. After steps (i) and (ii), the router's control plane is enabled and its main functions, such automatic neighbors discovery, exchange of routing information and packets forwarding, are enabled. Network services, step (iii), are also usually implemented in a distributed way, through different configurations of the control plane protocols. The way the different IP routers vendors (i.e., Cisco or Juniper) implement these services sometimes vary on how those interactions with the control plane are done, e.g., the configuration of MPLS-TE - Traffic Engineer functionality varies from one vendor implementation to another.

Aforementioned configurations are done typically via a Command Line Interface (CLI) while Simple Network Management Protocol (SNMP) is typically employed for network monitoring. Specially for multi-vendor scenarios, Network Operators requires highly qualified operation teams which deal with the complexity of multiple vendor management languages, different service's implementation and individual device configuration. Thus, IP management is perceived as a complex and labor intensive task.

Another important observation is how performance is preserved and guaranteed. Network's congestion is usually avoided by over-provisioning the packet networks, it improves performance, helps to meet Service Level Agreements (SLAs) and end-user experience. It is often to find that intra-domain IP core-networks are typically 2-4 times over-provisioned. The over-provisioned problem is even worse when it is view in common with the optical transport layer where is very common to provide 1+1 optical protection to assure resiliency, leading into an over-provisioning factor of 4-8 times.

To conclude the overview of IP networks, the increasing requirements of service performance, reliability and efficiency pushed the service providers to introduce Traffic Engineering (TE) techniques that allows finer control of traffic routing across the IP networks.

The ability of design and operate backbone networks directly relays on the ability of assuring performance for given services by assigning to them dedicated bandwidth to address the service requirements. MPLS enables the introduction of enhanced control capabilities to enable TE in IP networks [17]. MPLS supports explicit explicit label-switched paths (LSPs) which allow constraint-based routing to be implemented efficiently in IP networks. When combined with differentiation services (DiffServ) it becomes a powerful technique to enable QoS in IP networks.

MPLS operates as follows: at an ingress node within a MPLS domain, a Label Switching Router (LSR) classifies the traffic based on a combination of the information carried in the IP headers of the

packets and the routing information stored in the LSR. Based on this classification, the LSR tags the traffic with an MPLS label and then forwards the packets. The next hop in the LSP route either forwards the packet, or strip the label based on the information associated to the label prepending the packet. The incoming label may be replaced by an outgoing label, and the packet may be switched to the next LSR until it reaches its final destination [18].

1.3 Network management and control technologies for transport networks

Once it is presented the macroscopic view of the data plane architecture of today's transport networks, it seems clear that there is a wide heterogeneity of technologies co-existing together in the network and this fact increases the complexity and the cost to operate E2E services.

Several solutions have been proposed to solve this problem, during the last decade a huge effort of the industry to standardize common solutions which will provide unification of control of the network. Distributed solutions, such as Generalized Multi-Protocol Label Switching (GMPLS), as a common routing and signaling protocol for multiple transport technologies and BGP for inter-domain communications has appeared as a consequence of this efforts. In the last years, SDN has emerged as a promising candidate to unify the control in multi-layer networks.

In the following subsections, these technologies which represent the current state-of-the-art of the control plane, will be described in detail.

1.3.1 Generalized Multiprotocol Label Switching protocol

ITU-T G.805 recommendation defines a transport network as: "the functional resources of the network which conveys user information between locations" [19]. The term user information provides a clear layering structure on the network. GMPLS provides a common control plane for managing different network technologies and enabling service provisioning across the network [20]. The GMPLS control plane consists of a set of routing and signaling protocols. The signaling protocols are responsible for the establishment of transport plane paths, while the routing protocols are responsible for dynamically distributing connectivity and reachability information.

The GMPLS protocol suite extends MPLS to manage further classes of interfaces and switching technologies, such as TDM or WDM, among others. GMPLS is based on the concept of generalized labels, which are abstract labels which can represent either a single fiber in a bundle, a single waveband within fiber, a single wavelength within a waveband (or fiber), or a set of time-slots within a wavelength (or fiber). GMPLS consists of three main protocols:

- Resource Reservation Protocol with Traffic Engineering extensions (RSVP-TE) signaling protocol.
- Open Shortest Path First with Traffic Engineering extensions (OSPF-TE) routing protocol.
- Link Management Protocol (LMP).

The provisioning of end-to-end connections requires distributed coordination among the nodes, performed by the signaling protocol (e.g., RSVP-TE) and employing a hop-by-hop mechanism from the source to the destination node. The routing protocol (e.g., OSPF-TE) is responsible for disseminating any change occurring in the network state, allowing the nodes to update their local TEDs.

The GMPLS control plane offers a carrier-grade support solution for the establishment of optical circuits based on Fixed or Flexible grid granularity and also accounts for existing network deployments. GMPLS-based control solutions are still largely deployed in current core optical networks and network operators need to assure the return of investment of their current deployments.

1.3.1.1 Path Computation Element

Distributed routing presents important scalability problems when the size and complexity of the controlled network increase. As a consequence, a centralized path computation entity, named Path Computation Element (PCE) [21] has been defined and standardized within the IETF to alleviate this problem. The PCE standard also defines a dedicated protocol PCE-Protocol (PCEP) to communicate potential Path Computation Clients (PCCs) nodes (GMPLS nodes) and PCEs. The PCE introduction inside the GMPLS control plane was motivated to solve the complex problem of network routing across multiple transport layers (multi-layer) and several domains (multi-domain), where the optimal route calculation exceeds the capabilities of the nodes in the distributed solution OSPF.

In this context, an Active Stateful Path Computation Element (AS-PCE) is defined as a PCE, which maintains not only the traffic engineering information (link and node states), but also the state of the active connections in the network in a Label Switched Path Database (LSPDB). It is not only used as an input to the path computation process, but also for the control of the state (e.g. increase of bandwidth, rerouting) of the stored LSPs. Since under distributed control LSPs are only managed by the GMPLS controllers, this approach requires that the GMPLS controllers temporally delegate the control of a set of active LSPs to an Active Stateful PCE. A delegation mechanism for PCEP, based on the PCRpt, is proposed in [22].

Moreover, an Active Stateful PCE with instantiation capabilities has also been considered that is able to provision/release new/existing LSPs [23]. This message includes the endpoints and the computed Explicit Route Object (ERO), defining the route and resources to be traversed and allocated by the LSP. After the connection is successfully established, a PCEP Report Message (PCRpt) is generated to notify to the AS-PCE the successful LSP establishment and its management (e.g., deletion, modifying attributes, etc.). Therefore, the AS-PCE implicitly enables the switching programmability within the optical domain by acting as an interface between the SDN controller and the GMPLS control plane.

The AS-PCE has been demonstrated as an effective and comprehensive control solution in GMPLS-based distributed control plane for DWDM networks [24].

1.3.1.2 Applications-based Network Operations

A further step on management and control standardization was done by the International Engineering Task Force (IETF) by the definition of the Application Based Network Operations (ABNO) [1] architecture to efficiently provide a solution for implementing control functions over heterogeneous networks (IP, WDM, TDM).

The ABNO architecture combines a number of technology components, mechanisms and procedures. The main component of the ABNO architecture is the Path Computation Element (PCE). The IETF ABNO architecture is based on existing standard blocks defined within the IETF (PCE, ALTO, VNTM...), which could be implemented either in a centralized or distributed way according to network operator requirements. This architecture includes policy control for the different entities together with applications for managing requests for network resource information and connections.

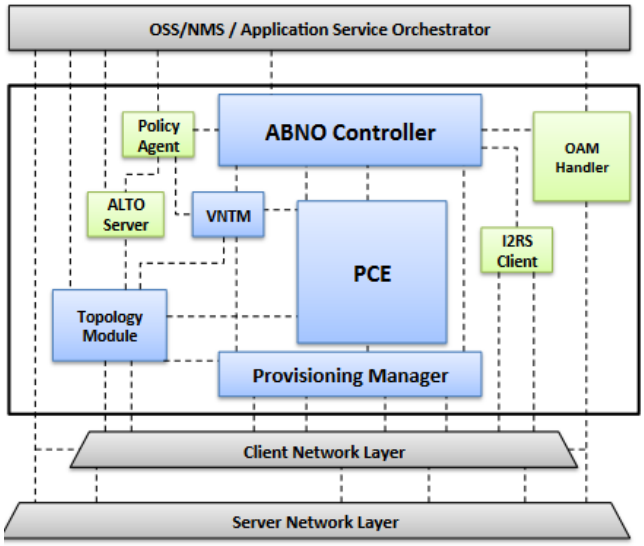


Figure 1.3: ABNO Architecture (`https://tools.ietf.org/html/rfc7491`) [1].

Also, network discovery, including multi-layer resources, using multiple interfaces depending on the underlying technology (i.e. IGP, BGP protocols). The proposed architecture also enables network virtualization and abstraction, by managing the creation of virtual links, and the mapping between topological representation and underlying network resources. It handles the path computation requests and responses, provisioning and reserving network resources. Finally, it verifies the connection and resource setup.

The solution's initial requirements included all the control plane features, in particular: network orchestration, network resources abstraction to external entities, policy (authorization, authentication and accounting), network element provisioning, path computation and routing engineering, QoS control, operations and maintenance (OAM), performance monitoring, multi-domain coordination, multi-layer coordination and, finally, discovery and storage of network resources. Figure 1.3 depicts the ABNO architecture.

This modular architecture allows that its different building blocks can be deployed by different vendors or third parties and even by a single provider. This modularity and the standard interfaces between the modules solve the problem of vendor lock-in for the operators. On the other hand, ABNO is specifically adapted to multi-domain and multi-vendor networks, enabling inter-operability between control plane based and OF based domains.

1.3.2 Software-Defined Networking

The distributed control and transport network protocols running inside switches and routers conforms the traditional technology employed to allow the data traffic to be reliably delivered through the Internet across the world. Despite the widespread adoption of the IP networks, they are complex and hard to operate [25]. Distributed control implies that every forwarding device must be configured with a set of supported network protocols in order to apply a determined network policy. This leads into a tremendous effort by network operators in configuring every network elements in their network (often using vendor-specific command line interfaces) when they need to create or modify a end-to-end network service.

Software Defined Networking (SDN) is an emerging network paradigm has attracted the attention of network operators for its promise of changing the limitations of the traditional distributed network

architecture by separating the control plane logic from the data plane forwarding infrastructure, which, traditionally, has been bound together or vertically integrated in the majority of network equipment. Moreover, SDN can infer important CAPEX savings by replacing dedicated hardware network equipment by software-driven switches installed on cheaper Commercial Off the Self (COTS) servers. SDN has been accompanied with new open standard interfaces which allow to program the forwarding logic in the physical devices from remote, centralized entities generally defined as SDN controllers, allowing the underlying infrastructure control functions to be abstracted and used by applications and network services as a virtual entity.

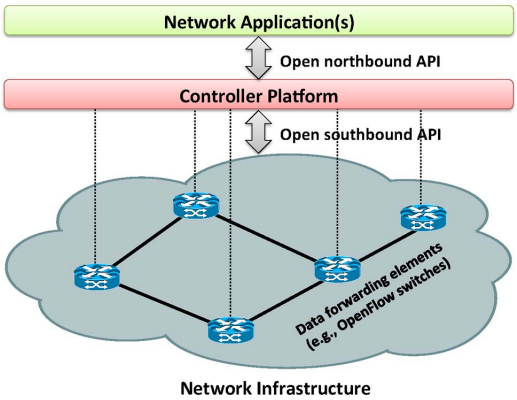


Figure 1.4: Simplified view of an SDN architecture.

SDN proposes a centralized management and control approach where the network intelligence is removed from the network and placed in logically centralized software-based controllers. Figure 1.4 shows a simplified view of the SDN architecture, open standard Application Programming Interfaces (APIs) allow the SDN controller to infer the forwarding behavior to the switching infrastructure and also to allow different network applications to define network services over the network global view offered by the SDN controller. SDN can reduce the traditional vendor-dependency of network operations while increasing automation and manageability. Other benefits of common interfaces introduction and control functions abstraction are: increased inter-operability between technologies, fast new network service and capability delivery and reductions of operational costs.

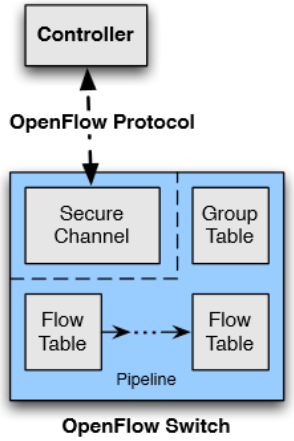


Figure 1.5: OpenFlow Architecture (`https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf`).

1.3.2.1 OpenFlow

OpenFlow, first published in 2008 [26], is an open standard, vendor and technology agnostic protocol which allows to program forwarding rules into OpenFlow-enabled switches (OFSs) through the

definition of flows [27]. It enables a higher flexibility to manage the life-cycle of the network services (creation, modification and release), and allows a simplified control plane in contrast to the independent protocol configurations needed in traditional IP networks.

Over the past few years, OF has become the predominant protocol in SDN for packet-based networks, and it has had a special impact in Data-Center (DC) networks. OpenFlow was born as an attempt to integrate packet and circuit-based network control [28]. However, until its latest releases (OpenFlow 1.5), its applicability for circuit-based connections in DWDM optical networks requires proprietary extensions to describe the lambda-based switching capabilities of optical ROADMs [29]. The prove of the little penetration of OpenFlow in the optical market is the reduced number of commercial optical transport networks deployments based on OpenFlow nowadays [30].

Although initially, the OpenFlow protocol was conceived for packet switched networks, recent extensions to OTN [31] have enabled also L1/L0 circuit switching, positioning OF as a feasible solution for Unified Control Plane (UCP) realization [32].

OpenFlow is based on the flow abstraction concept, which consists on representing the switching capabilities of a Network Element (NE) as a flow forwarding table. A flow entry, which represent a NE forwarding configuration, can be defined as any combination of L2-L4 packet header combination. These flows are defined in a separate control entity (i.e. OF controller) and pushed into the OFSs through the OpenFlow protocol FLOW_MOD message (OFPT_FLOW_MOD). This capability allows to create external software/user/defined routing, control and management applications, thus realizing one of the main objectives of SDN, the separation of the control logic from the data forwarding infrastructure.

The OpenFlow relays into an SDN architecture 1.5 where a controller exchanges TCP packets with an OpenFlow agent through a secure channel, which controls the packet forwarding of an OpenFlow Switch. Internally, the OpenFlow Switch maintains several flow tables which performs the packet processing pipeline.

The Flow Table is the building block of the logical switches, each packet entering into a switch may pass through several flow tables, each flow table contains entries consisting of six components:

- Match fields: are used to select those packets intended to being processed. The match fields consist of the following required fields: (i) Ingress Port, (ii) Ethernet Source and Destination addresses, (iii) IPv4/IPv6 protocol number, (iv) IPv4/IPv6 Source and Destination addresses, (v) TCP Source and Destination ports, and (vi) UPD Source and destination ports. Addresses may be exact matches or wildcards values. OpenFlow allows further optional match fields such Ethernet Type, VLAN IDs or MPLS tags.
- Priority: relative to priorities among table entries.
- Counters: updated with matching packets and used for traffic monitoring purposes.
- Instructions: are the actions to be taken when a packet match. These actions are: (i) *Output* the packet to a physical port; (ii) *Set-Queue* id for a packet in order to allow scheduling forwarding at the output port (QoS); (iii) *Group* packets for further common processing; (iv) *Push-Tag/Pop-Tag* for VLAN or MPLS packets; (v) *Set-Field* to allow packet header modifications; and (vi) *Change-TTL* (Time-To-Live) of a IPv4 packet, the IPv6 Hop Limit, or the MPLS TTL.
- Timeouts: Maximum amount of idle time before a flow is expired by the switch.
- Cookie: is an opaque data value chosen by the controller, can be used by the controller to filter flow statistics.

The OpenFlow protocol defines a large set of messages to exchange flow rules, transfer packets and switch features and configuration states between an OpenFlow controller and an OpenFlow switch. Typically, the protocol is implemented on top of Secure Sockets Layer (SSL) or Transport Layer Security (TLS), providing a secure OpenFlow channel. It supports three types of messages:

- Controller-to-Switch, messages initiated by the controller to request OFS information about its capabilities, configuration state, statistics... to push state changes into the OFS configuration, e.g., the *OFPT_FLOW_MOD*, which is one of the main OpenFlow messages, it used to create or modify flow entries into the switch. The Controller also can direct packets to the OFS through the *OFPT_PACKET_OUT* message, is used for instance for network adjacency discovery in OpenFlow networks in conjunction with the Link Layer Discovery Protocol (LLDP) protocol.
- Asynchronous messages sent by the switch to the controller. Here the main message is the *OFPT_PACKET_IN* which is used to send packets received by a OFS to the Controller. This may happen under some circumstances, such the packet does not match any defined match rule, or there is an specific instruction (action) defined under a flow rule which indicates the packet to be sent to the controller for further processing.
- Symmetric messages are sent without solicitation from either the controller or the switch. This is the case of the *HELLO* message used when the protocol connection is established or the *ECHO* message used for protocol synchronization purposes such in the KeepAlive message in TCP protocol.

In summary, the OpenFlow brought SDN control paradigm with a powerful, vendor-independent approach to manage complex networks and its specially relevant for this PhD Thesis given its relevancy in DC networking.

1.3.2.2 SDN Controllers

An SDN Controller is a software application in software-defined networking (SDN) that manages the control of flows in the network enabling intelligent networking. The controller can be considered the core (and brain) of an SDN network. An SDN Controller lies between network devices at one end and applications at the other end. Any communications between applications and devices have to go through the controller. SDN controllers are based on protocols, such as OpenFlow.

OpenDaylight (ODL) [33] is an open source project carried out by the Linux Foundation, which is a non-profit consortium which provides a framework to implement control and management services in a centralized network manner. ODL includes a Network Controller implementation called ODL controller. The ODL controller contains different management/control protocols plugins to provide programmability of heterogeneous data plane devices. OpenFlow, Path Computation Element Protocol (PCEP) and Border Gateway Protocol (BGP), among others are included as southbound plugins in the ODL controller distribution.

The ODLs North Bound Interface (NBI) is implemented by a Representational State Transfer (REST) API [34]. It exposes all ODL controller functionalities, allowing external applications, to get configuration information or program operations into the data plane. The southbound interface includes a Service Abstraction Layer (SAL) which connects the internal services with the specific networking protocol plugins. The communication between providers (generally networking protocol plugins which exposes functionalities to other ODL internal modules) and consumers is based on modelled APIs defined using the YANG modelling language. This data modeling allow abstract network configuration details (protocol specific messages) to upper SDN layers.

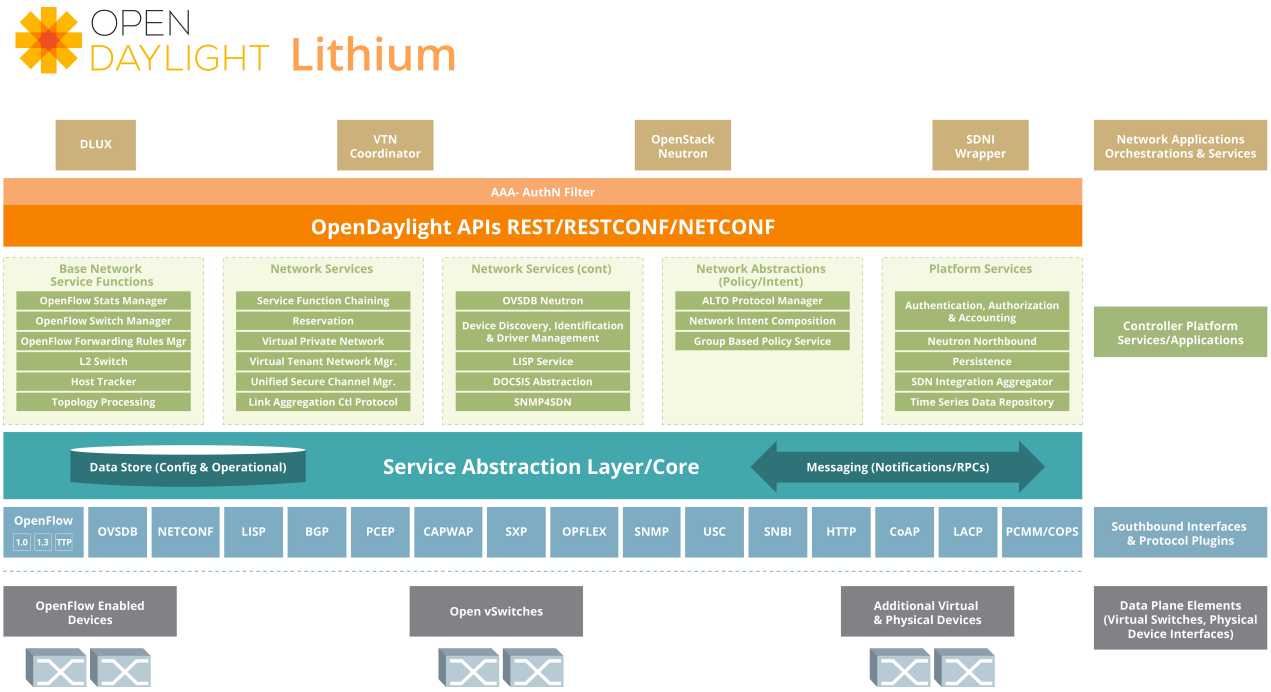


Figure 1.6: ODL Controller (`https://www.opendaylight.org/what-we-do/current-release/lithium`)

Internally the ODL controller is built up of different components (Figure 1.6). Among these, the Topology Manager is responsible for building and storing the network topology inside the ODL controllers domain. The Switch Manager handles the information about the switching devices. It exposes a NBI through the REST API from which the external applications can retrieve detailed information about the switches, including the detailed port's description used to define the flow requests at the later stage. Finally the Host Tracker abstracts the information about the hosts within the network domain.

The programmability of the network devices is done through different southbound plugins. OpenFlow 1.0 and 1.3 are already fully operational and BGP-LS and PCEP libraries are available to implement the corresponding plugins. The Forwarding Rules Manager is the module responsible for validating the flow descriptions received from the NBI. Afterwards it sends the flow configuration data to the Flow Programmer Service integrated into the SAL. This service is responsible for routing the flow establishment request to the corresponding southbound plugin. The architecture is designed to allow multiple southbound protocols.

1.3.2.3 Software Virtual Switching

Guided by the recent innovations on resource virtualization that have taken place in the Information Technology (IT) industry in the last years, and motivated by the need of extending this flexibility on the resource partitioning of the network infrastructure and decrease switching fabric costs, a new technology family of software switching has emerged as one of the most promising solutions for DC networking.

Software switches are designed to be deployed over very cost effective and energy-efficient Commodity x86/IA (Intel Architecture)-based servers, and they are designed for two purposes: (i) to

support packet forwarding between VMs within a server, and (ii) as a replacement for physical switches that interconnect servers. Solutions in the first category include; *Open vSwitch (OVS)* [35], *Switch Light* [36] Linux-based thin switching software solution developed by Big Switch (Floodlight Controller), *ofsoftswitch13* OpenFlow 1.3 compatible user-space software switch implementation by Ericsson. On the other hand, for replacement of physical switches we found Lagopus [37], flexible software-based OpenFlow 1.3 switch in userspace leverages the state of the art of multi-core CPUs and OS technology for high-speed network I/O; and Pantou/OpenWRT [38] developed by Stanford University, turns a wireless router into an OF-enabled switch.

OVS is the most widespread virtual switch implementation. It is a network virtualization platform designed to create virtual switches instances as software processes in general purpose fabric computers. OVS support the OpenFlow protocol (until version 1.5) and can be deployed in user space (for lightweight applications) or in kernel space implementations for data plane acceleration through Intel Data Plane Development Kit (DPDK) or Linux Kernel integration.

1.3.3 Application Programming Interfaces for network management

Any API definition is composed by two parts: (a) the transport protocol (e.g., REST), which defines the syntax, the communication paradigm (Remote Procedure Calls (RPCs), REST...) and in general all the rules which allow the communication between the entities interacting through the protocol; and (b) the information model, which defines the semantics of the API, the language and content of the messages exchanged through the interface.

Typically the SDN controllers' NBI APIs implementations have been designed based on the RESTful paradigm (or REST) over HTTP-based application protocol. REST encodes data into a uniform media type such as JavaScript Object Notation (JSON)[39] or Extensible Markup Language (XML) [40], that is specified into the message header and every resource exchanged is uniformly described using an Uniform Resource Identifiers (URIs).

The REST paradigm is convenient for the APIs implementation due to the need of stateless communication between client and server entities. It is also convenient because of the flexibility, scalability and commodity for practical implementation.

1.3.3.1 NETCONF and RESTCONF

REST practices and architecture have been adopted by NETwork CONFiguration protocol (NETCONF) [41] and RESTconf [42] and considering the benefits of both communication schemes (REST and RPCs)).

NETCONF is an IETF network management protocol designed to manipulate configuration data information of network devices. It provides the mechanisms to install, manipulate, and delete the configuration and operational data of network devices.

RESTCONF is the lightweight version of NETCONF. It has been standardized to bring network configure-ability from Web based applications. RESTCONF follows the REST HTTP methods to provide Create, Read, Update, Delete (CRUD) operations over a NETCONF data store containing YANG-defined data. Configuration data and state data are exposed as resources that can be retrieved with the HTTP GET method. Resources representing configuration data can be modified with the HTTP DELETE, PATCH, POST, and PUT methods. Data is encoded in either XML or JSON.

The RESTCONF interface structures the YANG information model in the following tree:

- /restconf/data : Data (configuration/operational) accessible from the clients.
- /restconf/operations : Set of operations (YANG-defined RPCs) supported by the server.
- /restconf/streams: Set of notifications supported by the server.

1.3.3.2 YANG

The impact of NETCONF and RESTCONF has notably increased with the introduction of YANG modeling language [43]. YANG provides a simple, user legible language to define the NETCONF data models exchanged by client and server entities. The YANG models are structured in a hierarchical tree structure of data that can be used for NETCONF operations, including configuration, state data, RPCs and notifications:

- Configuration Datastore which is organized in YANG as a hierarchical tree data structure where each node contains a name identifier and a set of child nodes. Each node supports the Create, Retrieve, Update and Delete (CRUD) operations.
- RPCs can be defined in YANG as independent, self-contained operations which can be invoked by the API client. They are defined by an input and output data structures, which are normally (but not necessarily) a subset of the configuration datastore.
- Notifications can be received asynchronously by the API client to update the state of any configuration parameter in case of changes in the network. This feature is a key-requirement for efficient resilient mechanisms.

YANG data models are organized in modules which can be imported and augmented by other YANG models, providing the flexible and modular environment required for the definition of management interfaces in SDN.

YANG is a data modeling language designed to describe the configuration, interactions and state data managed by the NETCONF and RESCONF protocols.

1.4 Data Center virtualization orchestration and networking

Several definitions can be found in internet of cloud computing, one of my favorites is the one proposed by the US National Institute of Standards and Technology (NIST) which says: "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [44].

It is undeniable that cloud computing has transformed drastically the IT industry, and it has become an essential part of any enterprise IT infrastructure. Cloud computing has introduced a new application paradigm where storage and server infrastructures are hosted and shared, allowing cost reduction and innovation for the development of new services and applications. This revolution has benefit the creation a new start-up business models where entrepreneurs with innovative ideas no longer require the large capital expenditures in hardware to deploy their services or the human expense to operate them.

New pay-as-you-go models where the IT resources can be offered "as a service" in different models. At the lower level, the cloud computing providers exploits virtualization to offer infrastructure components such as compute, storage and network resources over a shared pool of servers and a shared network infrastructure owned by the cloud providers, this model is known as Infrastructure as a Services (IaaS). Users can access to IaaS platforms which provide the whole environment to create new virtual machines, scale them or making replicas of its own customized virtual machines. Generally, the IaaS model provides the user its own infrastructure which can be managed completely by a software application and accessed remotely through the Internet.

There are two other cloud computing models widely exploited now by cloud provides: Platform as a Service (PaaS) and Software as a Service (SaaS). These two models offers not only the virtualized infrastructure but different software appliances to enhance business development over the virtualized infrastructure (PaaS) or/and software appliances for end users (SaaS) such online text editors or online storage services.

With the growth of the volume of data needed and the variety of Internet applications consuming cloud services, has drove the need of optimizing the way Data Centers (DCs) are exploited and managed. A DC refers to any large, dedicated cluster of computers that is owned and operated by a single organization. In the next sections we focus on presenting the DC orchestration and virtualization technologies with the focus on the networking side of the problem which is the focus of the present work.

1.4.1 Data Center Orchestration

Server virtualization is the first technology enabler of Cloud computing. Server virtualization allows to share physical server resources (CPU, RAM and ROM memory) over different virtual instances (Virtual Machines). Server hypervisors are software appliances which can run over bare-metal hardware (VMware ESXi [45]) or over operating systems (KVM [46]).

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 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.

Apache CloudStack is an open source cloud computing software, which is used to build private, public and hybrid IaaS clouds by pooling computing resources [47]. CloudStack requires at least three system VMs (secondary storage, console proxy and virtual router) to work. These VMs are used for robustness and reliability. Consequently, the VMs system reduce the resources available for run instances [48].

OpenNebula software is designed to work with driver concept for implementing the IaaS layer [49], it allows to use multiple storage back ends such as Logical Volume Manager (LVM) and Internet Small Computer System Interface (iSCSI), and different hypervisors, such as VMware, Xen, and KVM. This project, driven by OpenNebula Systems, is focused on user experience feedback and its main design principle is present a very flexible and open core system which adapt to its user deployment needs.

Following, OpenStack is described in more detail due to its specially relevancy for this work, as it has been extensively used for developing the studies and experimental demonstrations presented in this PhD Thesis.

1.4.1.1 OpenStack

OpenStack [50] 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 aimed is 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 cluster environment. The project is built among different modules: *Nova*, *Glance*, *Keystone* and *Neutron*; which are responsible of the orchestration of different resources: Computing, Storage, Identity management and Network resources, respectively.

OpenStack's networking is managed by a specific module called *Neutron* within OpenStack. Neutron exposes an API which provides network connectivity and addressing in the cloud. This API contains three main fields:

1. Network: A L2 broadcast domain virtualized inside the cloud.
2. Subnet: A block of IPv4/v6 network addresses. It is used to assign IP addresses to virtual instances. Each subnet must have a CIDR and must be associated with a network. Additionally, a subnet can also have associated a gateway, a list of DNS servers and hosts' routes.
3. Port: A connection point for attaching a single device, is a general definition of an endpoint within the network. It is defined by a physical address (MAC) and an IP of the devices plugged into it. When a port is associated to an IP, inherently has associated to a subnet, as the IP is picked up from an allocation pool of one subnet, always.

1.4.2 Distributed Data Center interconnection

DCs interconnection is one of the major problems that service providers have to face. DCs have been spread geographically to reduce services' latency to the end user, and that has led into an exponential growth of the inter DC traffic [51]. The expenses derived from the DCs interconnection across the Wide Area Networks (WANs) represents a big portion of the budget for large DCs operators, thus how cutting costs and improving the efficiency of the network are becoming some of their major concerns [4].

In order to support a large number of application services highly available for consumers from around the world, Cloud infrastructure providers have deployed DCs in multiple geographical locations to provide redundancy by ensuring reliability in case of site failures and to reduce services' latency to the end user. This situation has led into an exponential growth on the inter DC traffic [51].

Currently, cloud operators expect their customers to express their preference about where placing their services. However, this approach present many shortcomings, which include it is difficult to know in advance for Cloud customers where their applications are going to have a bigger impact and consequently where to place their services. As the traffic received by a internet application is highly dynamic, is difficult to accurately estimate in advance the amount of network resources a Cloud-operator has to provisioned for a service. This implies cloud services must be able to be scaled and migrated between DCs, moreover these operations shall be quick and cheap.

This paradigm requires building mechanisms for seamless federation of data centers of a Cloud provider or providers supporting dynamic scaling of applications across multiple domains in order to meet QoS targets of Cloud customers [52]. In summary, the DC interconnection cannot rely upon static, coarse granular and expensive connection pipes, but need to be adjusted to the traffic demand for both Cloud and Network providers can take full advantage of the existing network infrastructure [53].

The integration of the network control and management systems with cloud-based applications is directly related with the development of open, standard interfaces between network control/orchestration systems and upper applications. Open and extensible APIs allow gathering the relevant information from both domains, Cloud and Network, to deploy, manage and control the network and cloud infrastructure in a coordinated manner. Standard interfaces allow IT and Network applications being developed independently hiding the internal implementation details and offering abstracted services, such as the virtual machine creation/deletion or the E2E connectivity provisioning.

1.5 The fifth generation of mobile technology (5G) paradigm

The fifth generation (5G) of mobile network technology is not only about the development of new radio interfaces or waveforms to cope to with the expected 1000x increase and very stringent latency in mobile traffic [54]. 5G is also about an end-to-end converged network and cloud infrastructure for not only traditional human based services, but also for emerging Internet of Things (IoT) services. It is envisioned that billions of heterogeneous IoT devices will be connected to the 5G network, enabling a wide variety of applications in different vertical industries such as automotive, energy, media and entertainment, e-health and factories of the futures [55].

At the cloud level, 5G requires massive distributed computing and storage infrastructures in order to perform IoT analytics (e.g. BigData) from the data collected of sensors and actuators (e.g., temperature monitoring, energy consumption measurement, etc.,). Additionally, the adoption of Cloud Radio Access Network (CRAN), Network Functions Virtualization (NFV) and Mobile Edge Computing (MEC) requires cloud services for the deployment of virtualized network functions (VNFs) that are typically deployed in specialized and dedicated hardware (e.g., mobile Evolved Packet Core -EPC-, firewall, Content Delivery Networks -CDN-, Baseband Units -BBU- , etc.). It includes core cloud (e.g., in core/metro data centers DC-) for high-computational capability and long-term response time, but also edge cloud closer to the end-user (i.e., devices and terminals) with lower capabilities but fast response time (e.g. in Central Offices).

At the network level, 5G requires the integration and convergence of all multi-technology network segments (radio and fixed access, metro and core networks) in order to provide end-to-end connectivity services with Quality of Service (QoS). The heterogeneity of the foreseen 5G applications sets different requirements in terms of QoS, ranging from ultra low latency and high reliability for mission-critical applications to highly dynamic bandwidth allocations for video surveillance. Thus, 5G requires to dynamically allocate computing and storage resources to flexibly deploy virtualized functions (i.e.,VNFs and IoT analytics) in distributed cloud infrastructures, and to provide the required end-to-end connectivity among end users and the virtualized functions.

1.5.1 Network Function Virtualization

Network Function Virtualization (NFV) is an emerging paradigm which consists on displacing network functions traditionally performed by dedicated hardware middleboxes into software appliances which can be dynamically deployed, scaled and migrated in and between Commercial Off The Shelf (COTS) servers. This initiative responds to the need of network operators of overcome the increase of their operational and capital expenditures due to the flattening of their revenues driven by their infrastructure.

The NFV ETSI Industry Specification Group (ISG) presented the first definition of the NFV use cases in ETSI GS NFV 001 [56]. The problem and the definition of the architectural guidelines were assessed in ETSI GS NFV 002 [2]. The main objectives of NFV defined by the ETSI are:

- Improve CAPEX efficiencies by employing COTS hardware to provide Network Functions (NFs) through software in stead of on dedicated hardware appliances.
- Improve flexibility on the assignment of the NFs to hardware. NFV improves scalability and deployability of NFs from site locations (referred as NFV Infrastructure Points-of-Presence NFVI-PoPs in the ETSI architecture), allowing migration of the NFs where and when are needed.
- Standardization of open interfaces between Virtualized Network Functions (VNFs) and the infrastructure and management entities dedicated for their deployment.

Figure 1.7 shows the NFV architectural framework depicting the functional blocks and reference points in the NFV framework. The NFV archictectural framework main building blocks are:

- NFV Infrastructure (NFVI). Consists of the hardware resources including computing, storage and network to provide processing, storage and connectivity to VNFs, and the Virtualization Layer which abstracts the hardware resources and decouples the VNF software from the underlying hardware.
- NFV Management and Orchestration (MANO). Is composed by: (i) the Virtualised Infrastructure Manager (VIM), which comprises the functionalities that are used to control and manage the interaction of a VNF with computing, storage and network resources; (ii) the VNF Managers, responsible of the entire lifecycle management of each VNF; and (iii) the NFV Orchestrator (NFV-O), responsible of the orchestration and management of NFV infrastructure and software resources, and realizing network services on NFVI.
- Virtualized Network Functions (VNF) and Element Managers (EMs). VNFs are the logical entities representing the NFs being performed over a non-virtualised network and the EMs performs the management over one or several VNFs.
- Operations Support Systems and Business Support Systems (OSS/BSS). Represents the traditional network operator systems in charge of the operations, administration and maintenance (OAM) and the network monetization.

This architectural guidelines are the very starting point of the solutions proposed and experimentally assessed in Chapters 9, 10, 11 and 12, which cover different aspects of the aforementioned architectural framework.

1.5.2 Network Slicing

With a converged cloud and network infrastructure, controlled and operated together, there has appear the opportunity to serve different network services together. However, different network services have different network requirements. For instance, broadband access Internet connectivity for intelligent devices require a big amount of bandwidth as a best-effort basis with high bandwidth, high mobility and high traffic/connection density. On the other side emerging internet of things (IoT) will introduce a huge amount of new devices which shall be connected to services applications installed in different cloud sites, network requirements in this use case are completely different, IoT services targets low cost, long range and low power networks.

The Next Generation Mobile Networks (NGMN) alliance envisions a 5G architecture that leverages the structural separation of hardware and software, as well as the programmability introduced by SDN

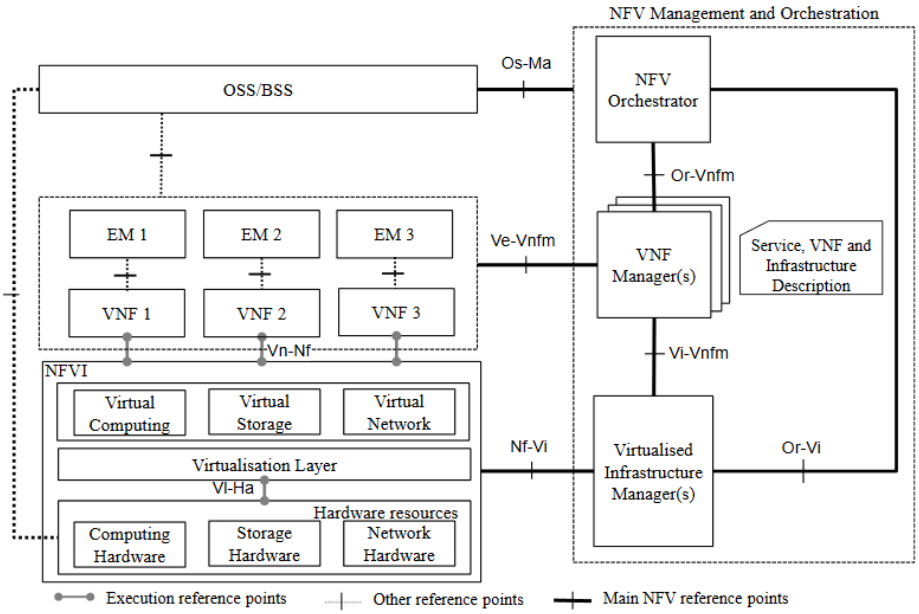


Figure 1.7: ETSI NFV reference architectural framework [2].

and NFV paradigms. As such, the 5G architecture is a native SDN/NFV architecture covering aspects ranging from devices, (mobile/fixed) infrastructure, virtualized functions, and all the management functions to orchestrate the 5G system [57]. The 5G architecture shall accommodate a wide range of use cases derived from the new needs of vertical industries, customers and enterprises which will impose new stringent network requirements in terms of: user experience (mobility, latency, and data rate), performance (connection and traffic density, spectrum efficiency and coverage), security, resilience and network deployment, operation and management.

In order to face this challenge, the NGMN has proposed the concept of '5G network slicing' [58]. It provides the main architectural framework for 5G, which allows multiple dedicated infrastructure instances or slices (involving shared network, cloud and virtualized functions) to co-exist in parallel, as shown in Chapter 11.1. These slices can adopt different architectures, involving different configuration of infrastructure resources, control plane architectures and also customized management systems to serve the purpose of giving customized performance for each specific use case

Chapter 2

PhD Thesis Objectives

This chapter presents the PhD Thesis objectives. It is divided in the three parts on which the Thesis contributions is divided: (i) End-to-End service provisioning for multi-domain, multi-layer Transport Networks; (ii) Integrated Orchestration of Cloud and Transport Network services; and (iii) 5G Network Slicing.

2.1 End-to-End service provisioning for multi-domain, multi-layer transport networks

One of the main objectives of this PhD Thesis is the design of an scalable and efficient SDN-based orchestration architecture to offer end-to-end service provisioning over multi-layer, multi-domain transport networks. To accomplish this objective, it is proposed to explore different SDN orchestration alternatives which can accomplish this task with especial emphasis on the interaction with heterogeneous control plane technologies. The work required has been divided into two sub-objectives:

– SDN Orchestration Architecture definition, design and implementation.

– Common Orchestration Protocol definition and validation.

2.1.1 SDN Orchestration architecture design

This sub-objective will cover the exploration of different design and implementation alternatives on SDN Orchestration architectures, with the final objective of proposed a final solution over which, the rest of the research activities of this PhD. Thesis are going to be sustained. The following tasks will describe different SDN orchestration architectures, including experimental validation in the ADRENALINE Testbed of all of them:

– SDN controller for multi-domain multi-layer network orchestration.

– SDN Orchestrator for multi-domain multi-layer networks.

– Hierarchical SDN orchestration architecture.

– Peer SDN orchestration architecture.

These objectives will be assessed in Chapters 4-8 (Part II).

2.1.1.1 SDN controller for multi-layer network orchestration

In this sub-objective, a single SDN controller based orchestration architecture is presented. The scope of this architecture will be discussed and an experimental validation of the architecture implementation the ADRENALINE Testbed is shown.

2.1.1.2 SDN Orchestrator for multi-domain, multi-layer networks

In this sub-objective, an Multi-domain SDN orchestration (MSO) architecture is defined to extend the scope of the previously presented solution to more complex network scenarios such multi-domain and multi-layer networks. The rest of the tasks and research activities, in the PhD Thesis will be supported by this SDN orchestration architecture. An experimental validation of the architecture will be presented as well as a comprehensive comparison with the preliminary solution as a conclusion.

2.1.1.3 Hierarchical/Peer SDN orchestration architectures

Finally, this sub-objective refers to present an hierarchical orchestration approach based on parent/child ABNO components, introducing scalability and security consideration to the SDN orchestration problem. Network virtualization and abstraction are key-technology enablers for accomplishing the so-desired scalability and manageability of large and complex network scenarios. Therefore this task will also propose a solution to integrate this features in the proposed MSO components.

2.1.2 Control Orchestration Protocol

Once defined the architecture which will serve as the foundation of this PhD. Thesis, the second objective is the functional requirement identification, modeling, design and implementation of a Common Orchestration Protocol (COP). The main objective of this protocol will be provide a common interface between the orchestration, and the control and application layers.

– COP requirements identification, modeling and design.

– COP validation.

2.1.2.1 COP requirements identification, modeling and design

This sub-objective has as a main target to present the need of the COP, identifying the functional requirements and translating them to a comprehensive model definition which will led into the design and implementation of the COP.

2.1.2.2 COP validation

In this sub-objective, the objective is achieving a functional implementation of the COP and its experimental validation. The main objectives of this protocol is to help the integration of the increasing number of SDN technologies implementation under the same principles following the same principle OpenFlow has achieved to integrate multiple data forwarding technologies under the same control principles. Therefore, in this task, it is intended to achieve an experimental validation of the COP integrating different SDN technologies (SDN controllers and orchestration architectures).

2.2 Integrated Orchestration of Cloud and transport network services

The second main topic of research covered during the development of this thesis, is the integration of the management of IT/Cloud computing resources within the SDN Orchestration. The organization of the work within this section will be the following.

– Integrated SDN IT and Network Orchestration architecture

– Geographically distributed DC interconnection

– E2E service orchestration for cloud computing

These objectives will be assessed in Chapters 9 and 10 (Part III).

2.2.1 Integrated IT and SDN Orchestration architecture

This task's objective is the definition, design and validation of an integrated IT and SDN Orchestration architecture. For this task, firstly the needs and requirements of network connectivity services in the Cloud and Fog Computing environments will be assessed. Secondly, a northbound API for the SDN Orchestration architecture proposed in the Section 3.1.1 need to be defined to provide E2E connectivity service abstraction to upper applications. Finally, an integrated IT and SDN orchestration architecture will be proposed, assessed and evaluated.

2.2.2 Geographically distributed Data Center interconnection

In the second task within this section, the geographically distributed DC interconnection across a multi-layer, multi-domain network will be demonstrated. For this task an IT and SDN Orchestration application will be designed and evaluated within the ADRENALINE Testbed.

2.2.3 E2E service orchestration for cloud computing

This last section will cover different cloud computing services provisioning by integrated IT and SDN orchestration. The main objectives we want to demonstrate and achieve within this last task are: seamless virtual machine migration (minimizing service disruption time), flexible bandwidth reservation for cloud services, and fast network responsiveness to traffic changes (BW steering).

2.3 5G Network Slicing

The third topic which will be covered in this PhD. Thesis is the applicability of the previously researched topics into the implementation of the Network Slicing concept. In this third part of the thesis, we will focus on the extension of the previously developed architectures for the inclusion of Network Function Virtualization (NFV) with the goal of creating per-tenant dedicated computing, storage and network resources to conform customized network slices for different according to the different requirements imposed by the network services being implemented over the slice.

These objectives will be assessed in Chapters 11 and 12 (Part IV).

2.3.1 Architecture definition

First goal in this part is to define how the different building blocks developed in the previous sections, together with new components will conform the proposed 5G Network Slicing architecture. The main requirements for the proposed architecture are:

– Multi-tenancy context aware orchestration for computing, storage and network resources.

– Dedicated virtualized control instances (SDN controllers and Cloud orchestrators) for each slice.

– Management and orchestration of VNF instances.

2.3.2 5G Network Slicing architecture validation: Virtual Network Operator use case.

We will conclude the work with an implementation of the Network slicing concept by providing a custom network slice for the implementation of the Virtual Network Operator use case. The objective is to see how the different concepts proposed in this PhD. Thesis play an important role in the implementation of the network slicing concept, which pave the way for the upcoming 5G era.

Chapter 3

The Cloud Computing Platform and Transport Network of the ADRENALINE Testbed

The demonstration of the concepts, theories and architecture presented in this PhD Thesis have been developed and tested over the Cloud Computing Platform and Transport Network ADRENALINE Testbed, a test platform designed and developed by the CTTC Optical Networking and Systems research group for experimental research on high-performance and large-scale intelligent optical transport networks. This chapter include a complete description the components and platforms that conforms the Testbed.

3.1 GMPLS/PCE enabled Ethernet over WSON platform of the ADRENALINE Testbed

The ADRENALINE testbed is composed of an all-optical DWDM mesh network (Figure 3.1) with two colour-less ROADM nodes and two OXC nodes, providing reconfigurable (in space and in frequency) end-to-end lightpaths, transparent to the format and payload of client signals (e.g., SONET/SDH, Gigabit Ethernet). Each optical node has two DWDM transceivers up to 2.5 Gb/s and one at 12.5 Gb/s with fully tuneable laser sources. Arrays of power meters and Variable Optical Attenuators (VOAs) are used for optical power equalization at output fibers. ADRENALINE deploys a total of 610 km of G.652 and G.655 optical fiber divided in 5 bidirectional links, in which optical amplifiers (Erbium-Doped Fiber Amplifiers -EDFAs-) are allocated to compensate power losses during optical transmission and switching at C-band. ADRENALINE transport plane also includes non-intrusive Optical Performance Monitors (OPM) to obtain spectral information tapping a 5% of all the input and output fibers, namely, channel and in-band Optical Signal to Noise Ratio (OSNR), channel and aggregate optical power, and wavelength drift.

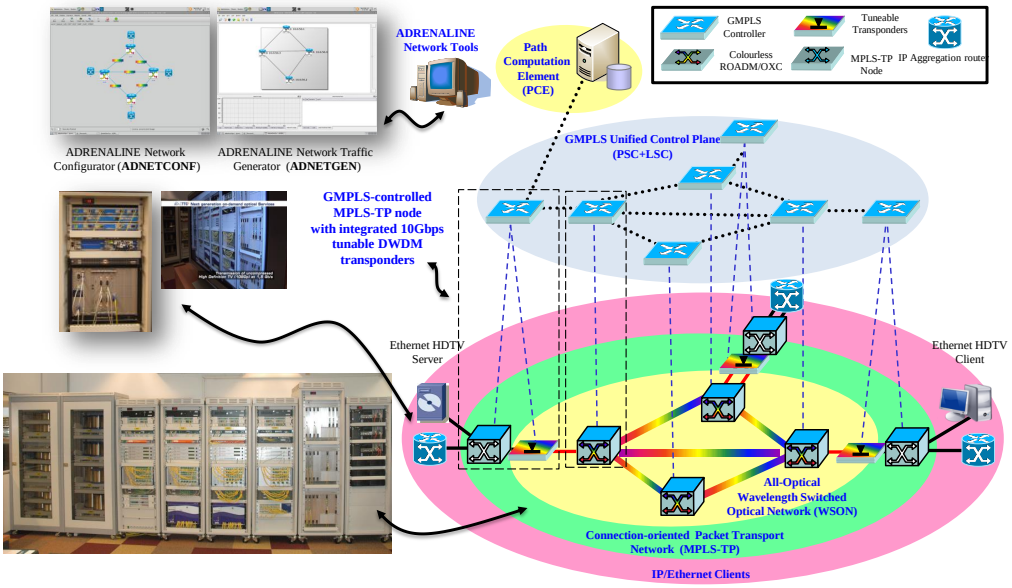


Figure 3.1: GMPLS/PCE enabled Ethernet over WSON platform of the ADRENALINE Testbed

Each optical node is equipped with a GMPLS controller for implementing a distributed GMPLS-based distributed control plane [59]. Such a control plane is responsible for handling, dynamically and in real-time, the resources of the optical node in order to manage the automatic provisioning and survivability of lightpaths (using the RSVP-TE signaling protocol for wavelength reservation, and the Open Shortest Path First Traffic Engineering (OSPF-TE) routing protocol for topology and optical resource dissemination), allowing traffic engineering algorithms with QoS. The system running each GMPLS controller is based on a Linux-based router with an Intel Core 2 Duo E6550 2,33 GHz processor. The Data Control Network (DCN) is based on IPCC carried at 1310 nm and C-band at the optical fiber with a line rate of 100 Mb/s using point-to-point links.

The ADRENALINE network includes a PCE, which is a dedicated network entity responsible for doing advanced path computations. The PCE serves requests from PCCs, and computes constrained EROs over the topology that constitutes the optical transport layer. The selected PCE deployment model is based on deploying a single PCE per OSPF-TE area, co-located in a GMPLS-enabled controller node and coupled to a Routing Controller. The preferred synchronization mechanism, by which the PCE constructs a local copy of the TED is non-intrusive: by sniffing OSPF-TE traffic, it constructs a dedicated (i.e. not shared) database using stateful inspection of the TE Link State Advertisements (LSAs) contained within the OSPF-TE Link State (LS) update messages, thus passively reusing the OSPF-TE dissemination mechanism, and not requiring the creation of an additional listener adjacency.

In the second phase of the Adrenaline testbed development, an AS-PCE was introduced over the GMPLS control plane to enhance the DWDM network programmability and its integration on a wider SDN control architecture for multi-layer (Ethernet/MPLS over DWDM) networks. This second phase is completed with the introduction of Cloud Computing Platform introduced in the next section.

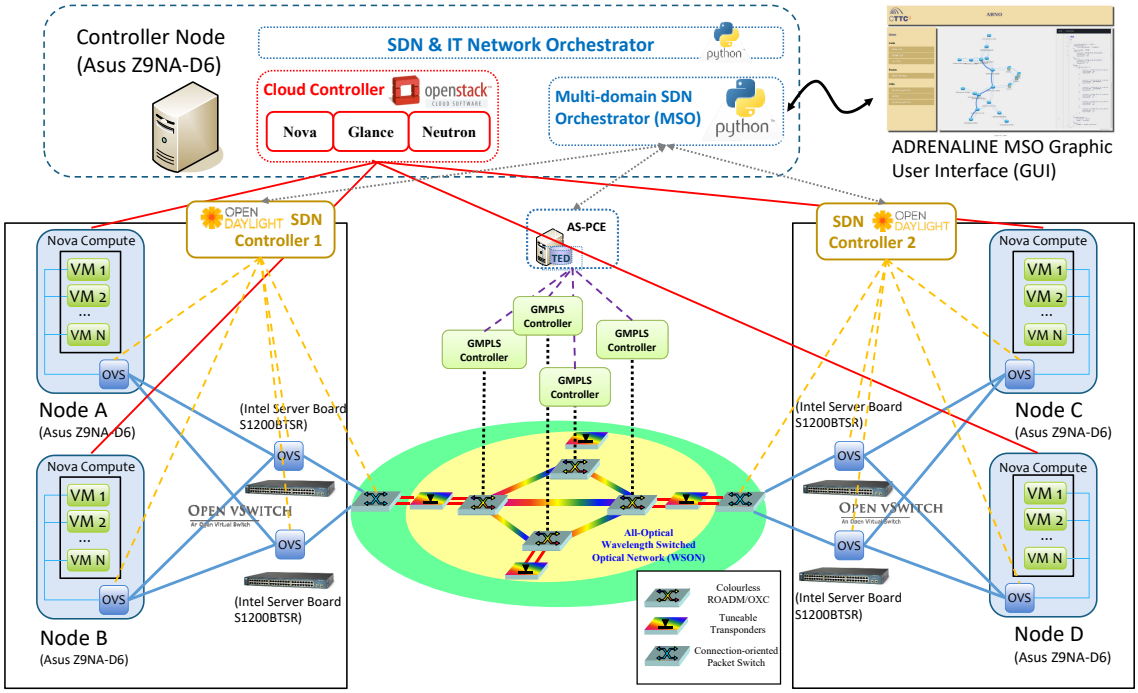


Figure 3.2: The Cloud Computing Platform and Transport Network of the ADRENALINE Testbed

3.2 The Cloud Computing Platform of the ADRENALINE Testbed

The Cloud Computing platform of the ADRENALINE Testbed (Figure 3.2) consist on two geographically distributed datacenters separated by the optical Transport Network detailed in the previous section. The Cloud Computing platform has been deployed with OpenStack (Liberty release) into five ASUS Z9NA-D6 Commercial Off The Shelf (COTS) servers equipped with 2 x Intel Xeon E5-2420 and 32GB RAM each. An Openstack controller node and four compute nodes have been setup in different network availability zones.

For each DC location an intra-data center network composed by four OpenFlow switches have been deployed using standard COTS hardware with multiple 1Gb Ethernet Network Interface Cards (NICs) and running OpenVSwitch 2.5.0 (OVS) implementing OpenFlow v1.3. Each Data Center border switch has been equipped with a 10 Gb/s XFP tunable transponder and a custom SDN-enabled XFP agent to dynamically configure the tunnable laser through a REST API.

The control plane architecture, which is explained in detailed in the chapters of this PhD Thesis, includes two OpenDaylight Berylium SR2-Release SDN controllers for the two intra-DC networks. On top of the SDN controllers the Multi-domain SDN Orchestrator (MSO) implemented mostly in Python 2.7 is the responsible of the coordination of the different control instances to provide end-to-end connectivity services. This piece of software is a proprietary code entirely developed by CTTC based on the IETF ABNO architecture. Last but not least, an SDN & IT Network Orchestrator has been implemented entirely by CTTC in Python 2.7 and its the software responsible of the joint orchestration of the Cloud Computing platform and Transport Network of the ADRENALINE Testbed.

Part II

End-to-End service provisioning for multi-domain, multi-layer Transport Networks

Chapter 4

Multi-layer SDN End-to-End service provisioning

In this chapter, as the first approximation of this PhD Thesis to Software Defined Networking (SDN) paradigm. An hybrid SDN and GMPLS control architecture is presented to provide end-to-end (E2E) connectivity services in multi-layer (MPLS/Ethernet over DWDM) networks. A centralized SDN controller coordinates an Active Stateful Path Computation Element (AS-PCE), which dynamically instantiates the provisioning of optical circuits within GMPLS-controlled DWDM networks, and directly programs the forwarding behavior of an OpenFlow-enabled Layer 2 network by the definition of flows.

This chapter is structured as follows: in section 4.1, the overall network architecture is presented. In section 4.2, the SDN controller extensions implemented to support multi-layer E2E connectivity are discussed. Finally in section 4.3, the experimental scenario and the obtained results are exposed to validate the E2E connectivity provisioning with the proposed extensions to a centralized SDN controller.

4.1 Multi-layer SDN architecture

As previously introduced in section 1.3.2, SDN has attracted the attention of network operators for its promise of infer important CAPEX savings replacing dedicated hardware network equipment by software-driven switches installed on cheaper Commercial Off the Self (COTS) servers (section 1.3.2.3).

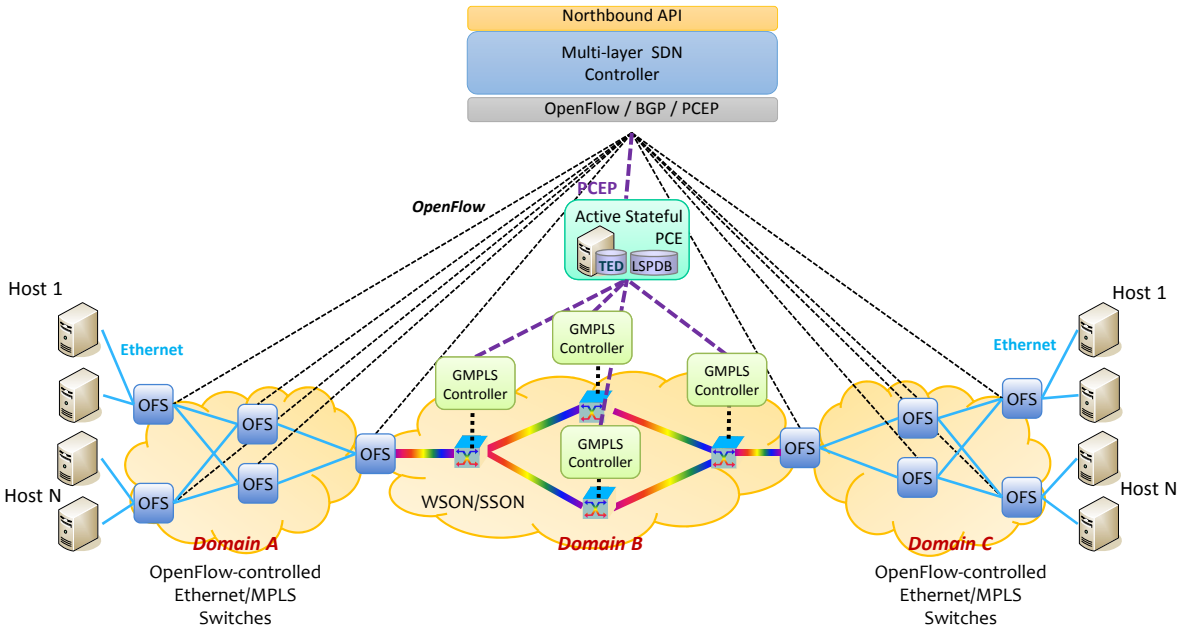


Figure 4.1: Multi-layer network and SDN control architecture

Over the past few years, the OpenFlow is the most associated protocol to SDN principles for packet-based networks, and it has an special impact in Data Centers (DC) environments. On the other hand, Generalized Multi-protocol Label System (GMPLS) in combination with the Path Computation Element (PCE) is the most wide-spread and mature control plane technology for automatic circuit provisioning in optical networks.

Bearing this in mind, our approach in this initial chapter, is to achieve the integration of this two control paradigms under a centralized SDN controller, which combines circuit and packet switching to provided dynamic control of E2E connectivity services in multi-layer networks.

The figure 4.1 shows a multi-layer network consisting on two different electrical packet domains inter-connected by an optical DWDM transport network domain conforming a MPLS/Ethernet over DWDM multi-layer network. The switching infrastructure within the packet network domains is implemented by OF-enabled switches (OFSs). The interconnection between packet domains is done by a GMPLS-controlled optical DWDM transport network, relying on the dynamic establishment of optical circuits or Label Switched Paths (LSPs) using GMPLS terminology. An AS-PCE (described in section 1.3.1.1), is employed to perform the Traffic Engineering (TE) path computations and centralize the LSP provisioning and monitoring. This way, the AS-PCE implicitly enables the switching programmability within the optical domain by acting as an interface between the SDN controller and the GMPLS control plane.

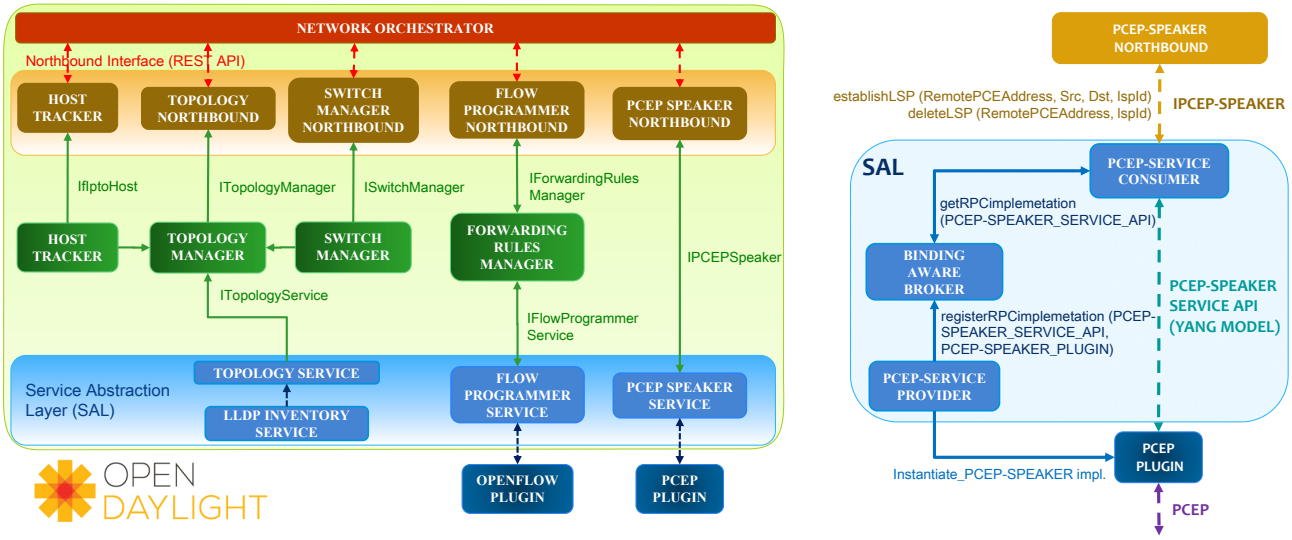
The aggregated packet traffic is groomed by the nodes, which are hybrid electrical/optical nodes equipped with a colored interfaces equipped with tuneable XFP transceivers. An SDN-enabled agent is needed to allow lambda tuneability on this hybrid electric/optical aggregation nodes. In this work, a dedicated REST API has been implemented as part of the control suite of the node to enable this feature, however other protocols, such NETCONF or SNMP, could be also employed to configure these devices.

On top of the described architecture, the multi-layer SDN controller unifies the control plane functions by implementing the PCEP and OF protocols with dedicated southbound plugins. The

optical network topology management, the path computation and the programmability of optical switching devices tasks, remain a responsibility of the AS-PCE.

4.2 Proposed extended multi-layer SDN controller

The proposed multi-layer SDN controller implements different management/control protocols to provide programmability of heterogeneous data plane devices. OpenFlow, Path Computation Element Protocol (PCEP) and Border Gateway Protocol (BGP) among others are included as southbound interface plugins in the SDN controller distribution, in this work OpenDaylight (ODL) controller distribution [33] has been approached to implement our solution.



(a) (b)

Figure 4.2: Extended Multi-layer SDN controller archiecture:(a) SDN controller internal components, (b) PCEP-Speaker block diagram.

The Figure 4.2a shows a block diagram representing the set of ODL controller's internal services involved in the E2E connectivity provisioning of hosts in the previously described network scenario.

On the top of the architecture, a Network Orchestrator (ORCH) process requests arriving through the northbound interface (NBI) and triggers the creation of the required services to the ODL controller. The controller's NBI is implemented by a Representational State Transfer (REST) Application Programming Interface (API). It exposes all controller features to external applications through HTTP basic operations (GET, POST, PUT, and DELETE). Each internal service (term to refer a basic architectural block inside the controller in Figure 4.2a) can expose its own NBI through the REST API.

The SDN controller includes a set of internal services including a Topology Manager for network topology discovery, a Switch Manager in charge of the monitoring of the network elements (packet stats, forwarding tables...), a Forwarding Rules Manager responsible of translating NBI flow requests into OpenFlow specific flow rules syntax and a dedicated PCEP-Speaker module (Figure 4.2b) responsible of implementing the PCEP protocol, including the PCEP-Initate and PCEP-Report message handlers.

The southbound interface includes a Service Abstraction Layer (SAL) which connects the internal services with the specific networking protocol plugins. The communication between providers (generally networking protocol plugins which expose services to other ODL internal modules) and consumers (components which consume a service from one or more providers) is based on modeled APIs defined using the YANG modeling language [43]. YANG is used to model configuration, state data, Remote Procedure Calls (RPCs) and notifications of network configuration protocols (e.g., NETCONF, OF or PCEP). The APIs generated from the services modeled in YANG, define the data structures exchanged and the methods that can be called by a consumer or the notifications that can be received. These APIs are registered inside the SAL by the southbound plugins (providers) and are available for ODL services. In this way, network configuration details (protocol specific messages) are abstracted to upper SDN layers.

4.2.1 Muti-layer orchestration

On top of the NBI of the SDN controller, an internal orchestration application (ORCH) has been created in order to handle the required mechanisms to solve the E2E provisioning in the multi-layer network. The E2E provisioning workflow diagram can be seen in Figure 4.3.

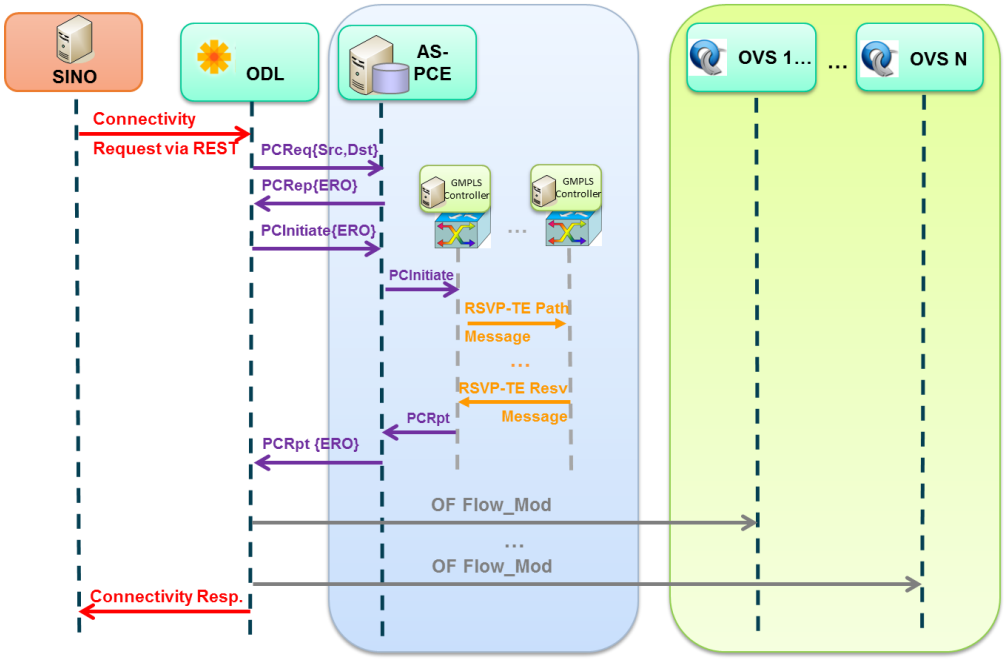


Figure 4.3: E2E provisioining workflow.

Firstly, the ORCH application recovers the network topology from the SDN controller through the NBI. The inter-domain connectivity is dynamically loaded from an internal configuration file when the SDN controller starts. Consequently, the path computation, inside the optical domain, is delegated to the AS-PCE. The optical domain topology is abstracted as a single-node, whose ports represent the border nodes of the optical transport network.

Then, when the E2E connectivity request arrives the ORCH calculates the shortest path between endpoints using Dijkstra's algorithm. If the computation fails to obtain a feasible route in the network, in this case if it discovers that there is no connectivity between packet domains (Figure 4.1, Domains A and C), the ORCH explores the inter-domain connectivity information to detect if the E2E service

request failed due to the lack of connectivity through the optical domain. If so, the ORCH requests the creation of a new Layer 0 connection by sending a Path Computation Request (PCRequest) to the AS-PCE to obtain a feasible route between the border OFS. After a successfully response from the AS-PCE, a LSP establishment request with the pre-calculated route encoded into an Explicit Route Object (ERO) is sent to the AS-PCE through the PCEP-Speaker service. After the AS-PCE notifies the PCEP-Speacker Service the effective LSP creation, the ORCH configures the border OFS node's SDN-enabled XFP agent with the corresponding channel information. After the border XFP interface is tuned into the target wavelength the a new link is discovered by the Topology Manager between the disconnected OFSs through the Link Layer Discovery Protocol (LLDP) mechanism described in the subsection 4.2.2.

Finally, the SDN controllers provisioning the E2E connection by calculating the route into the OF network topology managed by the TM between the E2E request's endpoints and it configures the OFS forwarding tables through OFPT_FLOW_MOD messages containing the match rules (source and destination hosts MAC addresses) and the action (input port/output port).

4.2.2 ODL internal services

The Topology Manager is the responsible for building and storing the network topology inside the SDN controllers domain. The Topology Manager receives the topology updates through the SALs Topology Service, which, in turn, receives LLDP packets advertising the network devices identities. This mechanism consist on sending a OFPT_PACKET_OUT message containing an LLDP packet, to a specific node (OFS) and is forwarded according with the forwarding rule included in the OF message. When another OFS receives a LLDP packet and there is no specific flow action for that packet, the OFS executes the OFPT_PACKET_IN action and encapsulating the LLDP message and re-send it to the controller. This way the SDN controller can determine the network topology.

The programmability of the network devices is done separately through the OF and PCEP plugins. The electrical domains' switches are configured by the OF protocol. The Forwarding Rules Manager is the module responsible of validating the flow description received from the NBI. Afterwards it sends the flow configuration data to the Flow Programmer Service integrated into the SAL. This service is responsible of routing the flow establishment request to the corresponding southbound plugin. The architecture is designed to allow multiple southbound protocols. In our case, the OF plugin is called in this last step.

4.2.3 PCEP-Speaker Service

The PCEP-Speaker consists of three different components: the NBI of the component which exposes the LSP establishment and deletion capabilities to the external applications (ORCH); the PCEP-Speaker Service, coupled into the SAL, which translates REST calls into the southbound plugin YANG modeled RPCs; and the PCEP Speaker Plugin, which implements the PCEP protocol, establishing the PCEP session with the external AS-PCE and sending the necessary PCEP messages to setup or delete a LSP into the data plane.

The new YANG model is created to define the LSP establishment and release capabilities thought a new data structure's set: LSP establishment Input/Output and LSP deletion Input/Output objects. It defines the interface (PCEP-SERVICE API) between the PCEP-Speaker Service and the PCEP-Speaker Plugin.

The PCEP-Speaker Service is broken in two SALs modules: PCEP-Speaker Consumer and Provider. The consumer calls the YANG modelled RPC implemented in the PCEP-Speaker Plugin, and the provider is responsible of register the implementation into the SALs resource broker

(BINDING AWARE BROKER) responsible to route all the RPCs requests from SALs consumers to providers.

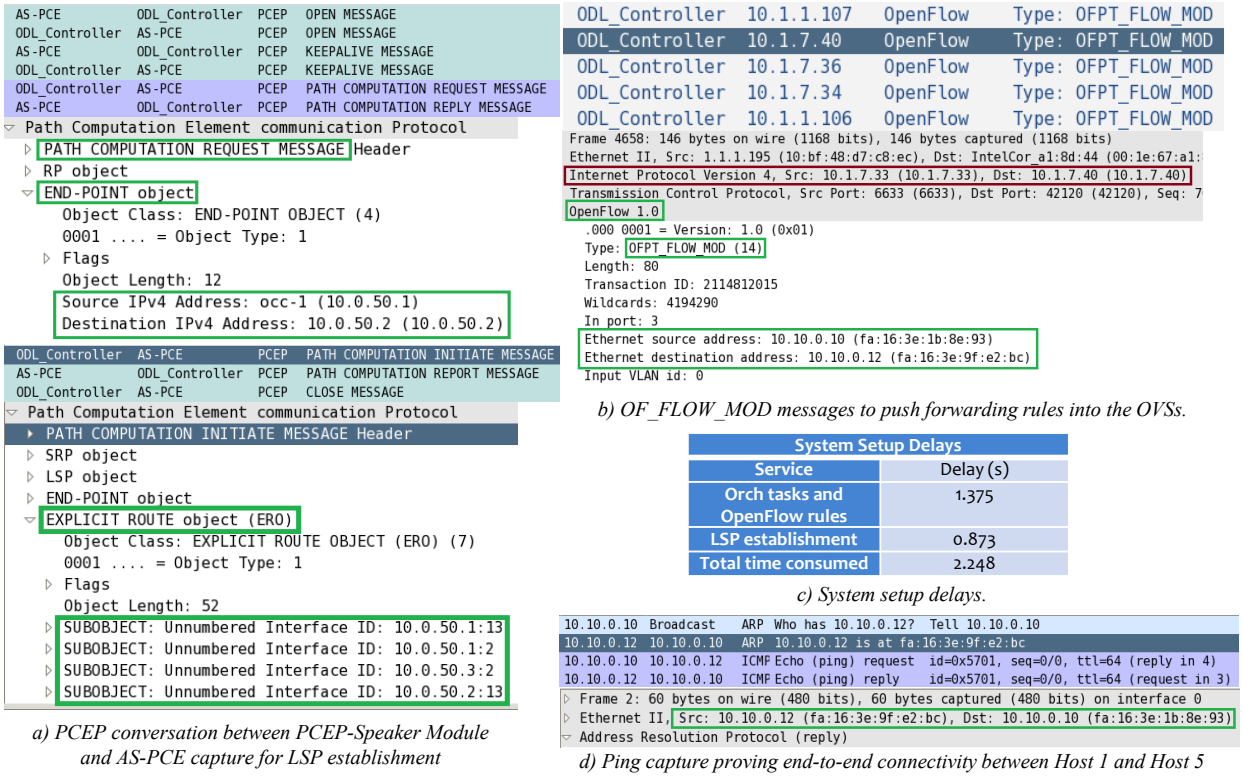


Figure 4.4: Experimental validation results

4.3 Experimental demonstration and results

The proposed architecture has been validated in the Cloud Computing platform of the ADRENALINE Testbed (Chapter 3). The network scenario (Figure 4.1) is composed by two packet switching domains (A,C) controlled both by the proposed extended OpenDaylight SDN Controller based on the Hydrogen release. OpenDaylight (ODL) [33] is an open-source project carried out by the Linux Foundation dedicated to the development of an open-source SDN controller.

In order to validate the proposed architecture, an E2E connectivity provisioning is requested, between Host1 (fa:16:3e:1b:8e:93) and Host5 (fa:16:3e:9f:e2:bc). Figure 4.4(a) shows the PCEP conversation between the ODL controller and the external AS-PCE for a path computation between the end-points (top), and for a LSP establishment request (bottom) through the PC-Initiate message which explicitly indicates the path to establish in the GMPLS control plane. The OF switches are configured through OFPT_FLOW_MOD messages (Figure 4.4(b)) sent from the ODL controller. We can observe that the Flow-rules establish the forwarding action only for the traffic from source to destination hosts MAC addresses.

In Figure 4.4(c) are presented the setup delays of the different processes involved in the E2E connectivity provisioning, i.e., the multi-layer orchestration, including the path computation and driver selection which together with the flow rules provisioning consumed 1.375s; and the LSP establishment

along the GMPLS control plane which consumed 0.873s. In Figure 4.4(d) the ARPs and ICMP messages exchange, between the two previously presented hosts through the multi-layer network, is depicted.

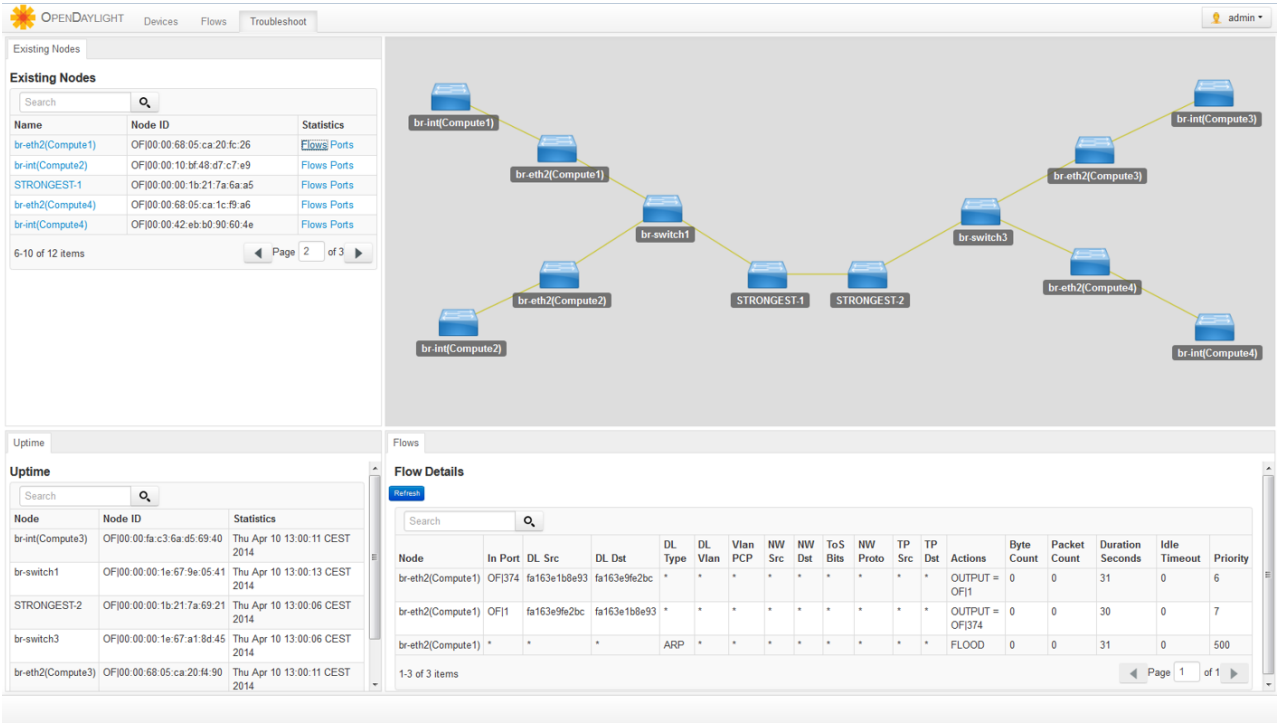


Figure 4.5: OpenDaylight controller Graphic User interface topology view after Optical LSP creation between nodes *STRONGEST_1* and *STRONGEST_2*.

Finally, in 4.5 the OpenDaylight controller Graphic User Interface (GUI) vision of the network topology and the flows created in one of the nodes is displayed (note that the *dl_src* and *dl_dst* OF Match fields correspond to Host1 and Host5 MAC addresses). The topology shows a link between nodes *STRONGEST_1* and *STRONGEST_2* which is dynamically discovered by the OpenDaylight controller right after the LSP in the optical network is created, by the LLDP/OpenFlow mechanism explained in the subsection 4.2.2.

4.4 Conclusions

A novel SDN multi-layer network control architecture has been presented featuring OpenFlow protocol for the L2 layer and the PCEP protocol for the dynamic instantiation of LSP within a distributed GMPLS control plane over the L0 optical layer. The OpenDaylight SDN controller has been extended to implement the PCEP protocol, by embedding a new service (PCEP_SPEAKER) within the ODL's MD-SAL, to dynamically create and delete optical Label Switched Paths in the GMPLS-controlled optical network.

Chapter 5

SDN orchestration of multi-domain multi-layer networks

In the previous chapter, a novel SDN control approach for end-to-end (E2E) service provisioning in multi-layer networks was presented. In that case, the architecture was based on a single SDN controller entity which was coordinating a multi-layer network consisting of OpenFlow-controlled Ethernet switches conforming the Layer 2 network and a DWDM optical network controlled by the GMPLS/PCE paradigm. The use of a single SDN controller simplifies the network discovery when new optical circuits were creating thanks to the use of the correlation of LLDP packets by the packet-in mechanism of OpenFlow.

Now the problem of E2E provisioning across multi-layer and multi-technology in SDN is extended to multi-domain scenarios. In this chapter, the concept of SDN orchestration is firstly introduced as the coordination and automation of the establishment and release of multiple independent network connections (usually performed by different control instances) to conform end-to-end connectivity services through heterogeneous network domains (which might consist of different network technologies). The SDN Orchestration is one of the central concepts of this PhD Thesis, the successive topics that will be presented in later chapters are supported by the SDN orchestration architecture introduced here.

This chapter is organized as follows: the section 5.1 presents the problem description of multi-domain and multi-layer control, and the SDN Orchestration concept as the proposed solution. In section 5.2, the SDN Orchestration approach is defined and thoughtfully described by analyzing the proposed solution from different angles. Section 5.3 presents the Multi-domain SDN Orchestration (MSO) architecture with the main building blocks and functions. In section 5.4, the experimental validation of the proposed architecture is presented by two different use cases: (i) SDN orchestration of Traffic Engineering (TE)-aware multi-domain, multi-layer networks; (ii) Automatic Provisioning of Fixed and Mobile Services. Finally, in section 5.5 presents a performance evaluation of the Multi-domain SDN Orchestration architecture in the ADRENALINE Testbed addressing Key Performance Indicators (KPIs).

5.1 Introduction

Network management and control systems are suffering an unprecedented evolution since Software Defined Networking (SDN) networking paradigm emerged almost ten years ago. As it was highlighted in the previous chapter, SDN proposes separating the control logic from the switching infrastructure by removing the 'intelligence' from the forwarding elements and placing it into a logically centralized SDN controller. OpenFlow (OF) allows to remotely program the forwarding behavior of the network infrastructure by characterizing the traffic as a combination of flow rules based on the packet headers. OpenFlow has become the preferred SDN interface, between control and data planes, for packet-based networks.

Ideally, as it was presented in the previous chapter, the network consists on a single control domain comprising multiple network elements featuring diverse technologies which are exposed to the SDN controller by standard interfaces. However, it is not realistic in the short term in transport networks since different vendor's transport equipment does not inter operate at the data plane level (only at the grey interface level) unlike regular Ethernet switches or IP routers, which causes the network to be fragmented into multiple vendor domains. Moreover, each vendor offers its own control plane technology (e.g., SDN with some proprietary OF extensions or GMPLS and PCE) because of the need of configuring vendor-proprietary parameters (e.g., FEC), generating isolated vendor domains.

Co-existence of GMPLS and OF control technologies for different network domains seems to be a realistic scenario in the mid-term. In this context, it arises the need of coordinating or orchestrating

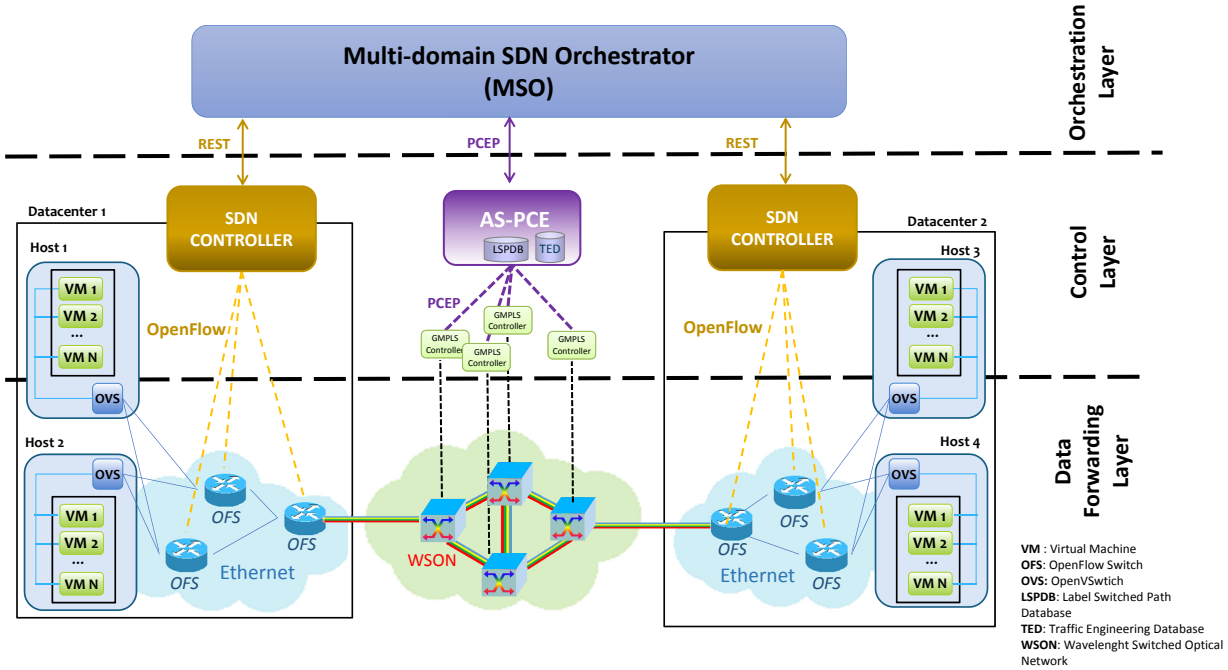


Figure 5.1: Multi-domain SDN Orchestration (MSO) architecture.

multiple, heterogeneous control planes arranged in different control domains. Inter-working between different control planes requires a higher, master entity (referred here as Multi-domain SDN Orchestrator MSO) which automatically coordinates the processes to establish and release E2E connections through different network domains controlled by different control instances. A graphical description of this control architecture can be viewed in 5.1.

In this line, the ABNO architecture [1] has been designed within the IETF, based on standard protocols (PCEP, BGP-LS) and components (PCE) to efficiently provide a network orchestration solution for multi-layer and multi-domain networks. In this chapter, it is presented a Multi-domain SDN Orchestration architecture which is conceptually aligned to the IETF ABNO proposal but focused on integrating multiple control plane technologies under the same orchestration entity. The implementation is based on a modular, plugin-based, architecture to orchestrate heterogeneous control instances with diferent different northbound interfaces. Its northbound Application Programmable Interface (API) has been designed following the Representational State Transfer (REST)-ful principles to allow external IT applications [34] (i.e. Cloud Computing management systems) to directly request E2E connectivity services into the network.

5.2 SDN Orchestration procedures and status

Network orchestration can be defined as the coordination and automation of the establishment and release of multiple independent network connections (usually performed by different control instances) to conform E2E connectivity services through heterogeneous network domains (which might be composed of different network technologies).

The SDN orchestration approach is based on an hierarchical architecture where a software-based, logically centralized entity provides E2E communication through different transport networks (Ethernet/DWDM) and/or control technologies (SDN/OF, GMPLS/PCE). This hierarchical approach has

been proposed before in [60] where multiple OpenFlow-enabled PCEs are orchestrated by a parent-PCE to provision E2E connections. In [61], the ABNO architecture is used in a international Testbed to orchestrate E2E connectivity services across several optical network transport technologies (Optical Packet Switching - OPS, Elastic Optical Networks - EON and WSON) controlled by different SDN controller distributions.

Differently, in this thesis another perspective is observed by assuming a network scenario composed by multiple control technologies: SDN/OF network domains interconnected by a GMPLS/PCE controlled optical DWDM network, with a logically centralized Multi-domain SDN Orchestrator (MSO). Network domains are interconnected by border links shared between two nodes which belong to different domains. This heterogeneity, among the control technologies considered, introduces the need of implementing multiple southbound interfaces in the MSO and also introduce new elements of discussion on the design of the orchestration architecture (i.e. level of topology abstraction and multi-layer path computation). Now, the set of requirements that should be fulfilled by the orchestration layer in the previously presented scenario are summarized as follows:

1. Translation of the external application connectivity service requests to the configuration of the different control plane instances. Definition of a standard and extensible northbound API to support customer service requests and offering network control abstraction to customer applications.
2. Discovery and inventory of the physical devices and composition of the network topology. Full physical network topology information is not strictly required in the orchestration layer, but at least, the inter-domain connectivity and an abstracted view of the network domains are required.
3. Multi-domain and multi-layer path calculation across the different network domains. Domain selection or full path computation depending on the level of topology abstraction.
4. Provisioning and restoration of the E2E connectivity services. Programmability of the different network controllers by implementing the necessary provisioning interfaces and configuration of the inter-domain connections.
5. Event handling and notification support of changes in the network (failures, topological changes...).

In the following subsections, the different protocol/interface alternatives to design and implement an effective SDN orchestration architecture are presented. The discussion is focused on weather or not they can address the previous outlined requirements.

5.2.1 Topology discovery

The MSO may compose its network topology by the cooperation of the underlying network controllers which can advertise its intra-domain topologies using different protocol or interfaces. Most of the SDN controllers implement custom RESTful APIs to offer their network topology to external applications, but also other possible interfaces are attracting a lot of interest, this is the case of the NETCONF protocol [41] and its RESTful based version RESTCONF [42], both based on the YANG modeling language [43].

The topology recovery in the orchestration layer can be done in a reactive or proactive manner. In the proactive approach, the MSO requests the network topology to the control plane instance every time it need to refresh its working copy to perform a new path computation. RESTful interfaces are connection-less interfaces and they do not inherently support asynchronous notifications.

This feature may constrains some orchestration implementations to perform the topology discovery proactively. Regarding the reactive approach, the asynchronous notifications between the control and the orchestration layers can be supported by WebSocket transport technology (RFC 6455). The Websocket Protocol is a TCP-based protocol which uses the HTTP handshake mechanism which facilitates real-time data transfer between the control instance (SDN controller or AS-PCE) and the MSO.

Regarding the network inventory and topology discovery in the control layer, the preferred solution for most SDN controllers is the combination of the Link Layer Discovery Protocol (LLDP) and OF as it was explained in the previous chapter. Similarly, in the GMPLS control plane an Internal Gateway Protocol (IGP), i.e. Open Shortest Path First (OSPF) protocol, can be used to exchange the topological information between GMPLS nodes. The AS-PCE can include a IGP instance to listen as well network status information packets and building the Trafic Engineering Database (TED) based on the gathered information.

5.2.2 Path Computation

The E2E path computation can be performed in a different manner depending on the level of topology abstraction and the number of transport layers. Depending on the level of abstraction, the MSO can manage an abstracted view of the network that consists on the domain connectivity through inter-domain links and a node abstract representation of the network domains [62]. In this case, the intra-domain path computation is delegated to the lower, per-domain controllers and the MSO only performs the domain selection of the controllers involved in the E2E path calculation.

Another alternative is the discovery of the complete physical network topology by the MSO. In this case the MSO is responsible of calculating the full path across the network, which in the proposed scenario comprises different layers. In the multi-layer scenario, a separate path computation instance (i.e. a PCE) for each layer topology can be employed, or a single path computation instance, with network visibility of all the transport layer topologies, can use multi-layer aware algorithms to calculate the routes.

5.2.3 Connectivity provisioning

The E2E connectivity provisioning involves the orchestration of different control plane and forwarding technologies. The MSO is responsible for the implementation of the interfaces or protocols exposed by the control plane to forward the orders from the orchestration layer to the control layer. Also, the generalization of different transport technology connections into a flexible and common data structure is a key requirement to be able to offer abstracted information to upper layers through the NBI.

Most SDN controllers offers a custom provisioning REST API which input parameters are aligned with those used on the OFPT_FLOW_MOD messages to insert flows into the OF-enabled switching devices. A flow is defined by one or more matching rules which range from Layer 2 to Layer 4 packet headers, and an action (forward to a specific port, drop packet, forwarding to the controller, etc...) inserted into the forwarding device's OF table.

The AS-PCE can instantiate or remove LSPs into the network using the PCEP initiate request message (PCInitiate) [23]. This message includes the endpoints and the computed explicit route object (ERO), defining the route and resources to be traversed and allocated by the LSP. After the connection is successfully established, a PCEP Report Message (PCRpt) is generated to notify to the AS-PCE the successful LSP establishment and its management (e.g., deletion, modifying attributes, etc). The PCEP protocol can be use as a NBI of the AS-PCE to expose a programming interface to the orchestration entity.

5.3 Multi-domain SDN Orchestrator (MSO) architecture

Now, the Multi-domain SDN Orchestration (MSO) architecture is introduced. Firstly, the different modules that conforms the MSO architecture will be presented, describing how this approach addresses the set of requirements enumerated in section 5.2. As it was mentioned in the introduction of this chapter, the MSO architecture design principles are aligned with the IETF ABNO architecture proposal (RFC 7491).

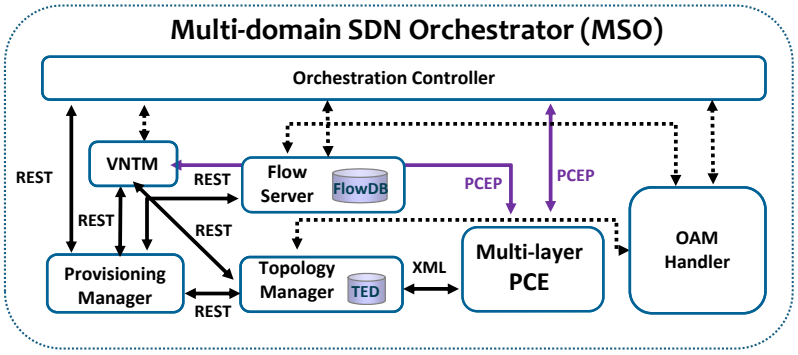


Figure 5.2: MSO architecture for multi-domain, multi-controller orchestration

5.3.1 Orchestration Controller

On top of the MSO architecture (Figure 5.2), the Orchestration Controller (OC) handles the internal workflow between different MSO modules and it processes the incoming requests from the Northbound Interface (NBI). The NBI offers, to other external applications, the CRUD operations for the E2E connectivity services, the multi-domain topology service and the path computation. It has been implemented as a RESTful API.

The OC is responsible of implementing different orchestration workflows depending on the transport layer of the service being requested. Currently, the MSO supports DWDM optical services, Ethernet L2 services and MPLS tunnels. Besides, the MSO supports multi-layer cooperation for the creation of services. This means that a upper layer connectivity services (Ethernet or MPLS) can trigger the creation of one or more Layer-0 (Optical DWDM lightpaths) connectivity services to provide connectivity between upper-layer's endpoints. The upper service provisioning is transparent to the operations involved into the creation of lower layer services, but internal correlation between the dynamically created Layer-0 connectivity services and the resulting virtual links exposed into the upper layer's topology are maintained by the Virtual Network Topology Manager (VNTM) component. The OC identifies the need of creating a new virtual link depending on the response in the path calculation. This feature will be explained in detail in following subsections.

Once the route is calculated between the service request endpoints, the OC is responsible of correlate the topology information to determine to which administrative network domain corresponds each part of the entire path. In the proposed architecture we assume that one domain cannot consist of multiple layers, thus the multi-domain path decomposition it is always performed in single layer paths.

5.3.2 Topology Manager

The Topology Manager (TM) is the component responsible of collecting the network topology from each control domain and building the Traffic Engineering Database (TED). The TED includes all the relevant information about network links and nodes, and it is used by the Path Computation Element (PCE) for calculating routes across the network. The TM recovers the physical network topology of each network domain and the inter-domain connectivity is loaded from configuration files. From this information, the TM builds a complete multi-domain topology and per-layer topologies (from now we will refer to them as TE topologies) built by filtering the whole topology based on the TE information of the links, for each transport layer technology. All this information is stored in the TED.

The TM design is based on a modular and extensible plugin based approach where for each administrative domain, a controller client instance is customized with its corresponding topology discovery interface based on whether the underlying control instance is a SDN Controller, and AS-PCE or other control technology. Currently, the TM implements dedicated plugins for: (1) different SDN controller RESTful APIs (OpenDaylight, RYU, ONOS); (2) a proprietary interface of the AS-PCE based on a raw socket TCP and XML/JSON encoding; (3) the Control Orchestration Protocol (COP) which will be presented in detail in Chapter 6.

The topology discovery is performed in a proactive manner, the general principle is that every time a new connectivity service request arrives to the OC, every domain controller is requested to refresh the MSO copy of their topology and the TE topologies are re-build.

The TE enforcement strategy consist in the proactive reservation of the network capacity and its dynamic representation into the TED network topologies. The MSO architecture allows to specify the bandwidth requirements for a specific E2E connectivity service request and proactively enforce the reservation of this bandwidth capacity in the network, this feature will be explained in detail in 5.3.5. An important requirement to support this capability is to store the Traffic Engineering information, only known by the orchestration layer in the TM TED. The relevant topological parameters necessary to include TED are: the maximum reservable bandwidth and the unreserved bandwidth of every link. The former is obtained from the topology information retrieved by the per-domain controllers and the latter must be dynamically updated after every new connection is effectively established.

5.3.3 Path Computation Element

The Path Computation Element (PCE) is a standard component standardized by the IETF [63] centralizes the path computation functions within the MSO architecture. The MSO architecture support two different approaches: a single unified multi-layer PCE which can perform a path computation accounting with the TE constrains of both layers (such as with wavelength continuity constrain in the WDM-based optical networks or bandwidth/delay constrained packet networks), or two separated PCEs dedicated for each layer (L0 and L2).

The PCE is used by the OC and the VNTM components within the MSO architecture. The communication between those entities is done through the Path Computation Element Protocol (PCEP) [21]. The PCE retrieves the corresponding TE topology from the TM. In our implementation, the TM implements a topology server which is accessible through a socket TCP and XML/JSON encoding for each TE topology. This way the PCE, every time it receives a new Path Computation Request (PCEPRequest), it can obtain a fresh version of the current MSO TE Topology.

The PCE can load different algorithms depending on the underlying transport network loaded in the TM TEDB. For the Multi-layer PCE, a Constrained Shortest Path First (CSPF) based algorithm

defined in [64], employs the delay as a TE-metric was presented. In the implementation used for the Multi-layer PCE based MSO architecture, the unreserved bandwidth on every link is taken as a constraint by the algorithm to calculate the E2E path across the multi-layer network. The CSPF output returned within the PCEP Response includes a multi-layer Explicit Route Object (ERO) representing the path combining packet links (physical or virtual) and/or optical hops where the layer adaptation is ensured. Optical segments are subject to the Wavelength Continuity Constraint (WCC) and the optical wavelength is included in the ERO if the E2E path traverses a WSON network segment.

5.3.4 Virtual Network Topology Manager

The Virtual Network Topology Manager (VNTM) is the responsible of the multi-layer management. The general purpose of the VNTM is two fold: (1) the provisioning of Layer-0 (L0)/DWDM optical connections to satisfy upper layer's connectivity demands; (2) the creation of Layer 2 virtual links that represent the physical connectivity provided by the created connectivity services in the optical layer. The VNTM workflow varies weather a single multi-layer PCE or multiple per-layer PCEs are employed. Following both workflow descriptions are detailed.

- Multi-layer PCE: when the path computation response of a new L2/Ethernet connectivity service consist on network elements from different layers, the VNTM extract the L0 sub-path and the inter-layer pair. To do this, the VNTM consults the transport layer information of every consecutive node included in the multi-layer path in the multi-layer TE topology obtained from the TM.
- Per-layer PCEs: first the L2-PCE handles the path computation of the new L2/Ethernet connectivity service. If the path computation fails, the VNTM requests uses the inter-domain topology information stored in the TM to find which inter-domain nodes are reachable from the service request's source and destination nodes. To do this, it sends a path computation request between each L2 inter-domain node and the source node, to conform the list of possible inter-domain source nodes. Then, it repeats the procedure with the destination node. When the operation finishes, if there is at least one node in each list (at least one inter-domain source and destination nodes) it selects the first possible solution and, based on the inter-domain topology information, selects the Layer 0 source and destination nodes, conforming the inter-layer pair.

Once the inter-layer pair is obtained, the VNTM triggers the creation of a new L0 connection between the L0 inter-layer endpoints through the Provisioning Manager. After the successfully establishment of the L0 connection, the VNTM notifies the TM the creation of a new virtual link (VLink) and the related virtual ports on the border nodes into the L2 network topology. The responsibility of mapping the created L0 connections and the generated L2 virtual links remains in the VNTM too. Finally, when the new VLink is created, it is characterized with the TE information of the underlying connection (i.e. total available bandwidth, aggregated delay).

5.3.5 Provisioning Manager

The Provisioning Manager (PM) is the module responsible of translating the connectivity requests, processed by the OC and the VNTM, into the specific syntax of the underlying network control instance northbound provisioning interface.

The PM design follows the same principles of the topology manager. It implements a provisioning plugin for each different network controller technology and it creates a dedicated customized controller

instance with the corresponding plugin interface implementation. Currently, the PM implements dedicated plugins for: (1) each the custom SDN controller's provisioning REST API (OpenDaylight, RYU and ONOS); (2)the PCEP protocol with Stateful and PCE-initiated LSP Setup extensions [23]; and (3) the Control Orchestration Protocol (COP). All the established connections (both L0 and L2) are stored in the Flow server by the provisioning manager.

The *Connection* is the data structure characterizing the internal connectivity constructs created in the underlying transport domains, and which is send to the PM to characterize the provisioning requests. It consists on the following parameters:

- *Endpoints*. Source and destination nodes, described as: {Router_ID, Interface_ID}
- *Path*. List of hops traversed by the connection, each one described as: {Router_ID, Interface_ID}
- *Transport_layer*. (L0, L2)
- *Forwarding_rules*. Matching_Rules and Action similar to the OpenFlow equivalents [27]
- *Connection_type*. (Unidirectional, Bidirectional)

In chapter 6, the connection construct will be introduced as part of the COP information models

5.3.5.1 Multi-layer bandwidth reservation in SDN orchestration

TE-aware traffic aggregation requires the guarantee of the effective bandwidth reservation into the E2E service provisioning. While in a WSON GMPLS domain a fixed amount of physical resources (50 GHz wavelength) are reserved for each Label Switched Path (LSP), in a packet-based OF domain it is necessary to limit the bandwidth assigned to each flow in order to preserve a certain QoS (i.e. bandwidth) assured to each E2E services. OF 1.3 introduces a new message (i.e. OFPT_METER_MOD) which enables the specification of traffic meters into the OF-switches, with an associated data rate and a QoS-enabling strategy, such as dropping packets at a determined Drop-rate or Differentiated Services Code Point (DSCP) packet tagging to allow DiffServ. Flows can be attached to these predefined meters, associating a maximum rate to each flow. A Meter instruction has to be included inside the OFPT_FLOW_MOD message indicating the Meter-Id desired to be attached to.

5.3.6 OAM Handler

The Operation, Administration and Maintenance (OAM) handler receives asynchronous notifications, such as topology updates, flow statistics, or failure alarms. Based on them, it triggers the corresponding internal workflows, which start from updating topology information stored in the TM's TED, then to discover the affected E2E connections by the network update and finally depending on the event it would led into a restoration or update of the E2E services installed by the MSO.

The internal architecture of the OAM Handler is based on a modular plugin approach. The OAM Handler subscribes the Notification Servers (if any) available on each domain controller, and implements custom notification decoders based on the information model of each specific domain controller. As a REST API does not allow notifications, websockets have been introduced in some SDN controller implementations (RYU) [65] in order to emit asynchronous messages in the opposite directio (bottoms-up). Websockets are also employed in the Common Orchestration Protocol (COP) designed in this PhD. Thesis, which will be carefully defined in Chapter 6. Currently the plugins implemented in the OAM Handler are based on the RYU SDN controller implementation and the COP.

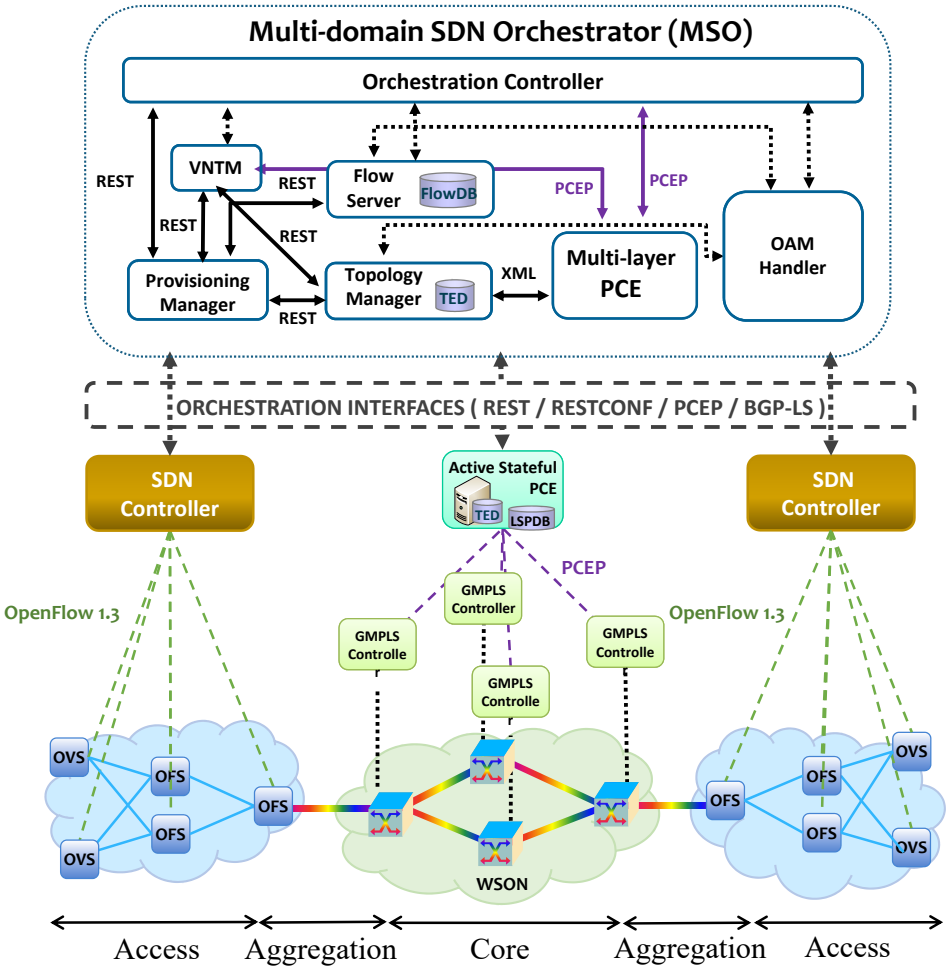


Figure 5.3: Multi-domain SDN Orchestration of the multi-layer, multi-domain network of the ADRENALINE Testbed.

5.4 Experimental evaluation

In order to experimentally evaluate the proposed MSO architecture, the multi-layer transport network of the ADRENALINE Testbed (Figure 5.3) has been chosen as the reference scenario for the different uses cases on whose the MSO is evaluated. Specifically two different uses cases are proposed: (1) the experimental validation of the TE-aware orchestration of multi-domain, multi-layer networks; and (2) the automatic provisioning of Fixed and Mobile Services through MPLS tunneling.

5.4.1 SDN orchestration of TE-aware multi-domain, multi-layer networks.

In this first evaluation, the objective is to experimentally validate the end-to-end service provisioning in the aforementioned network scenario. The MSO implementation employs a single multi-layer PCE which employs the Constrained Shortest Path First (CSPF) algorithm detailed in subsection 5.3.3.

The E2E service request is characterized by its service endpoints (ingress and egress interfaces to connect), the transport layer of the connection (MSO supports L0/DWDM links, L2/Ethernet or MPLS services), and a set of forwarding constrains such as source/destination MAC or IP addresses, TCP ports, VLAN IDs or a MPLS label. Moreover, the service request may include a set of traffic parameters such a target reserved bandwidth or a latency threshold.

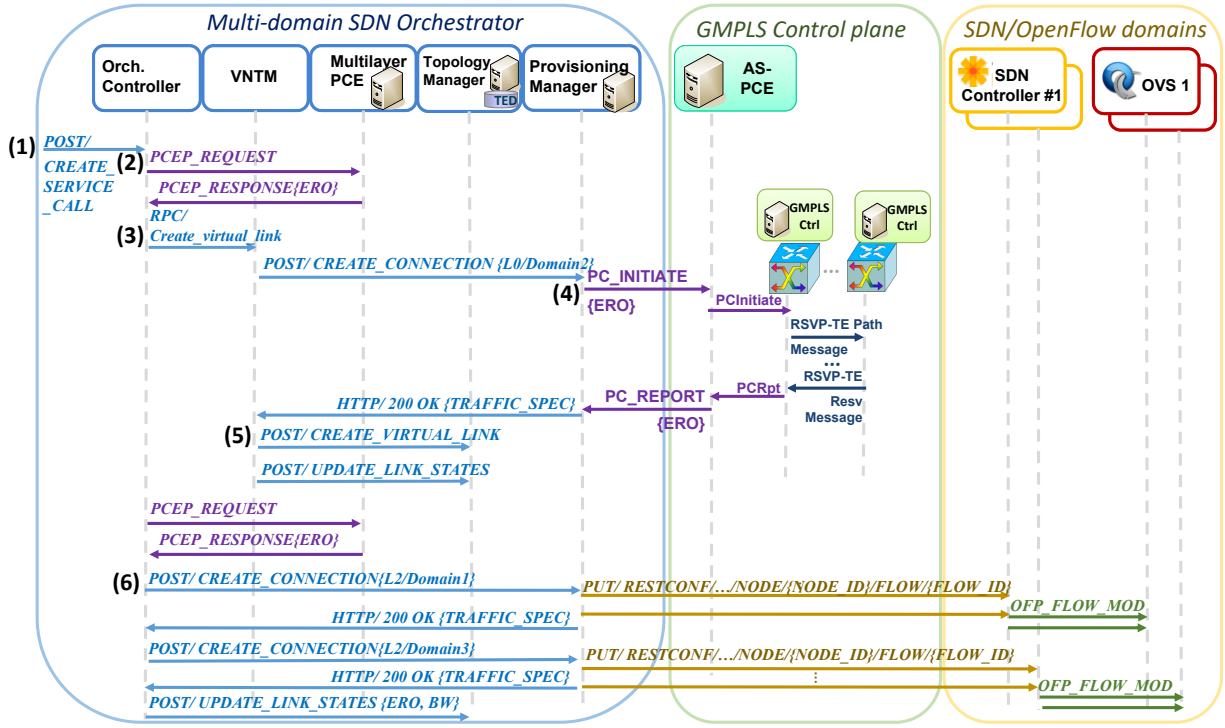


Figure 5.4: E2E provisioining workflow with ABNO orchestrtion achitecture.

The provisioning workflow (Figure 5.4) of an E2E service request between two Layer-2 service endpoints can be seen in Figure 5.4. First, the E2E service request is validated and processed by the OC, which firstly sends to the PCE a Path Computation Request (PCEP Request) through the PCEP interface (2). The depending on the nature of the PCEP Response, the calculated path could be uni-layer or multi-layer, in the latter the OC triggers the creation of a new virtual link (3) through the VNTM service. The internal VNTM workflow for the creation of a virtual link involves the creation of new Layer-0 connection or DWDM link through the PCEP provisioning plugin. (4) A PCEP Initiate message is sent to the AS-PCE to setting up the creation of a new LSP to interconnect the OF network domains. This request includes a pre-calculated path through the optical network and the border node's ports connected to the border links which interconnects the OF domains with the optical transport network. These Endpoints are represented as Unnumbered Interfaces composed by the router-ID and the Interface-ID encoded in 64 bits. After the creation of the virtual links, the topology is updated with a new Layer-2 link (5). Then, the OC request a new path computation over the updated topology, if the resulting path is multi-layer, the same workflow is repeated. Otherwise, the OC is responsible of generating per-domain connections connections, by partitioning the end-to-end calculated path, and sending them to the Provisioning Manager(6), which in turn, converts the received connection request into the corresponding protocol or API of the related domain controllers.

Figure 5.5 presents the experimental validation of the MSO architecture, the figure shows a complete traffic capture of the communication between the MSO and the different controllers for the provisioning of an end-to-end Ethernet service between two endpoints in the Adrenaline Testbed. The image includes the traffic captured in three points of the network: (1) the MSO server machine; (2-3) the servers on which the SDN controllers of domain 1 and 2 are running. The MSO implements the proprietary RESTful API of OpenDaylight Berylium SR4 release for the configuration of the two SDN controllers, which at once configure the switching infrastructure using Openflow 1.3. For the optical domain, the MSO implements the PCEP protocol including the LSP instantiation extensions to dynamically create and release LSPs in the GMPLS-based optical control plane (for more details

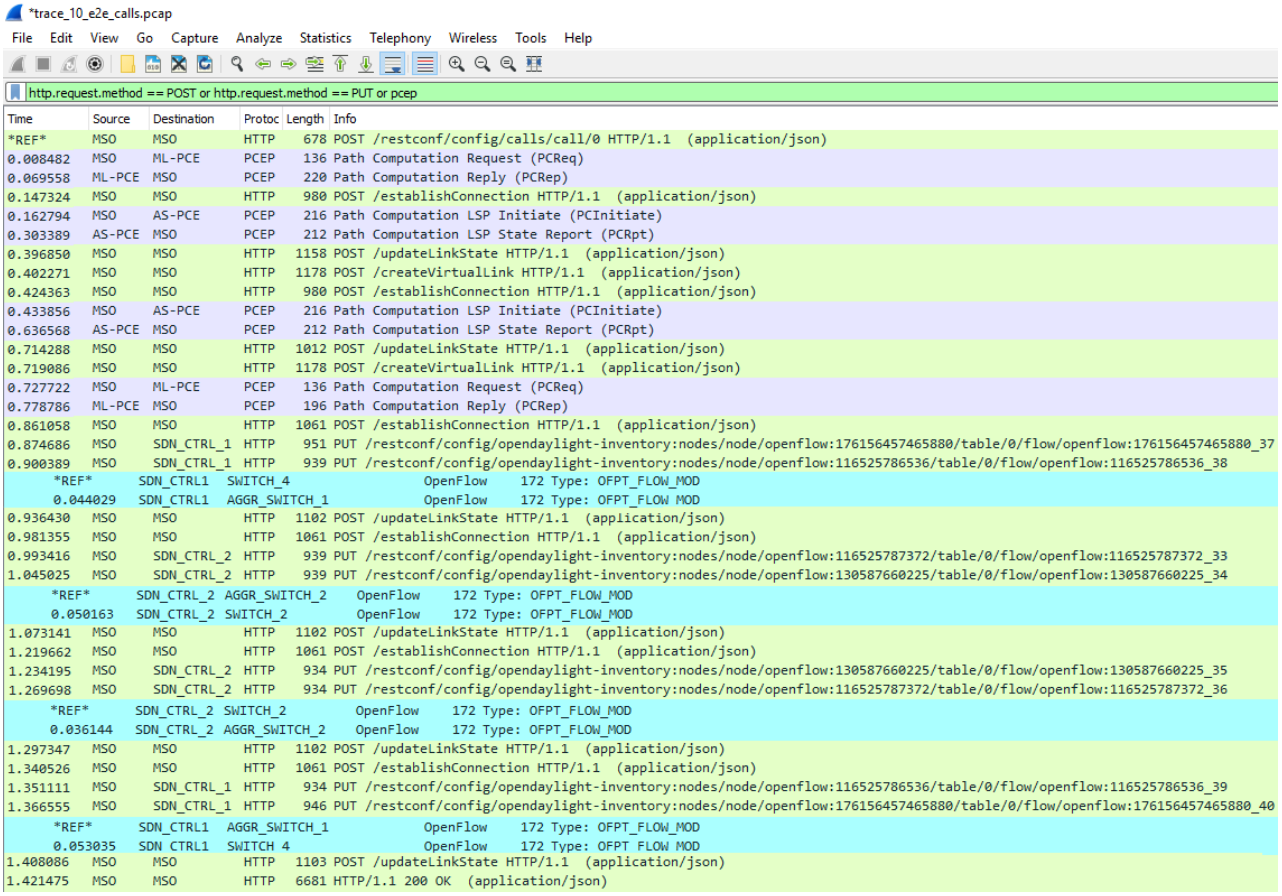


Figure 5.5: E2E provisioining workflow with ABNO orchestrtion achitecture.

consult Chapter 4).

Firstly, the capture validates the path computation operation between the MSO and the Multi-layer PCE. Secondly, the multi-layer orchestration is shown by the creation of a virtual link supported by a bidirectional optical LSP requested to an AS-PCE (PCEP Initiate and Report messages). Then a new path computation is performed once the multi-domain, multi-layer topology is updated in the TM through the internal REST API. Finally, the creation of the L2 connections are requested to the PM, which translates the information included in the connection structure into the specific flows sent to the SDN controllers. This process is shown in two steps, first the message exchange from the MSO to the SDN controllers and sequentially from the SDN controllers to the switches through the Openflow protocol.

5.4.1.1 Proactive enforcement of TE policies in OpenFlow-based SDN networks

In this subsection, it is introduced an experimental validation of the proactive enforcement of TE-policies in Openflow-based networks. As it was explained in 5.3.5.1, to guarantee the reservation of resources end-to-end, Openflow allows the creation of meter bands in the switches which can be associated to one or more flows.

The per-flow meter traffic limitation concept has been validated in the data plane by injecting different traffic flows through an Openflow v1.3 virtual switch using Lagopus implementation [37] which has been deployed in a Linux-based server Intel Xeon E5-2420 with 8 cores at 3.2 GHz and

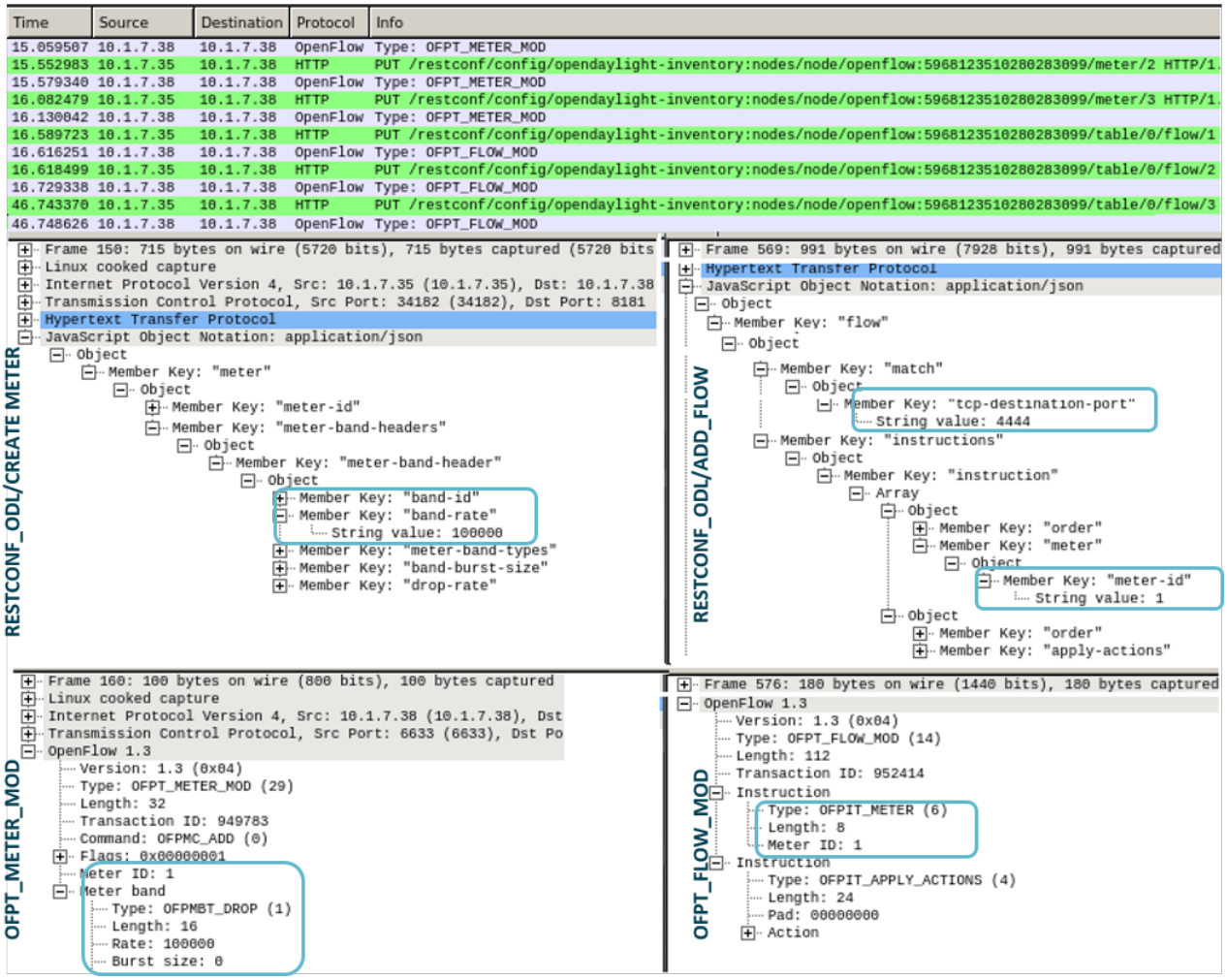


Figure 5.6: Per-flow bandwidth limitation Wireshark capture.

32GB RAM. Figure 5.6, shows the traffic capture of the control overhead of the creation of three different meter bands at 100, 300 and 600 Mbps. The RESTCONF messages, sent from the MSO to one of the OpenDaylight controllers are divided in two groups: (bottom-left) the creation of the METER_BANDs through the OpenDaylight REST API and the subsequent OFPT_METER_MOD messages to the switch; and (bottom-right) the subsequent flow creation request with the associated METER_ID.

In Figure 5.7 it is shown the experimental validation of the per-flow meter traffic limitation concept by showing a real traffic capture of the throughput reach by three different flows after traversing the OpenFlow switch, with and without attaching a meter bands to them (Figure 5.7.a, 5.7.b). The Iperf Bandwidth Measurement Tool has been employed to obtain the results placing a client into the machine connected to the input port and a server measuring the bandwidth achieved into a second machine, connected at the output port of the switch. The three meters were set to 100, 300 and 600 Mbps. It can be observed how after creating the flow-meters differentiated QoS levels to each flow has been successfully achieved.

5.4.2 Automatic Provisioning of Fixed and Mobile Services

In this section a different scenario is presented wherein both fixed and mobile client applications, running on top of the MSO, automatically instantiate connections within the aggregation network.

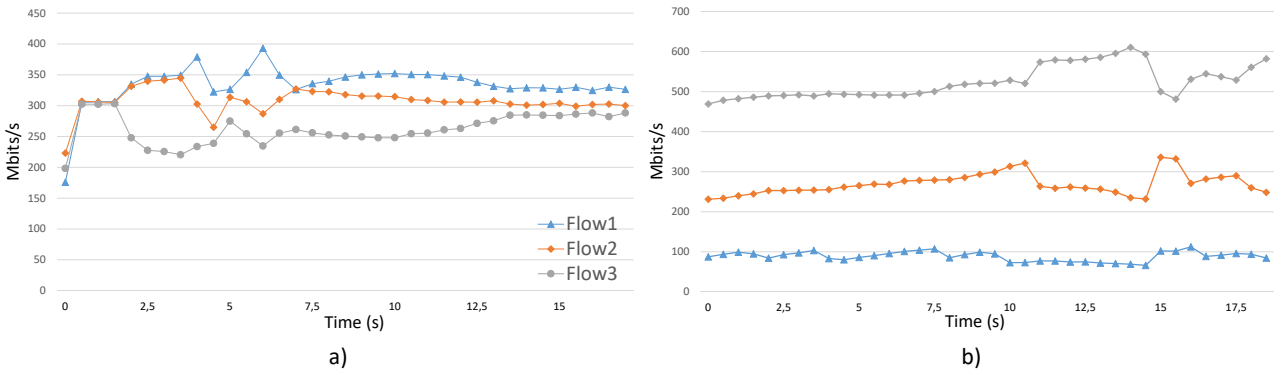


Figure 5.7: Per-flow bandwidth limitation: a) IO graph OF Switch output_port without meter limitation, b) IO graph OF Switch output_port throughput with 600, 300 and 100 Mbps meter limitation.

The experimental validation is carried out though the setting up of mobile LTE services (Evolve Packet System EPS- Bearers) between cell stations (eNBs) and the Evolved Packet Core (EPC), responsible for the Internet access for the mobile users. The validation is shown from a twofold perspective: i) at the control plane level, it is detailed the MSO building blocks interactions and exchanged heterogeneous control messages, including OpenFlow extensions; ii) at the data plane level, the traffic flow adaptation to actually transport EPS Bearers over the multi-layer (MPLS and optical) network infrastructure.

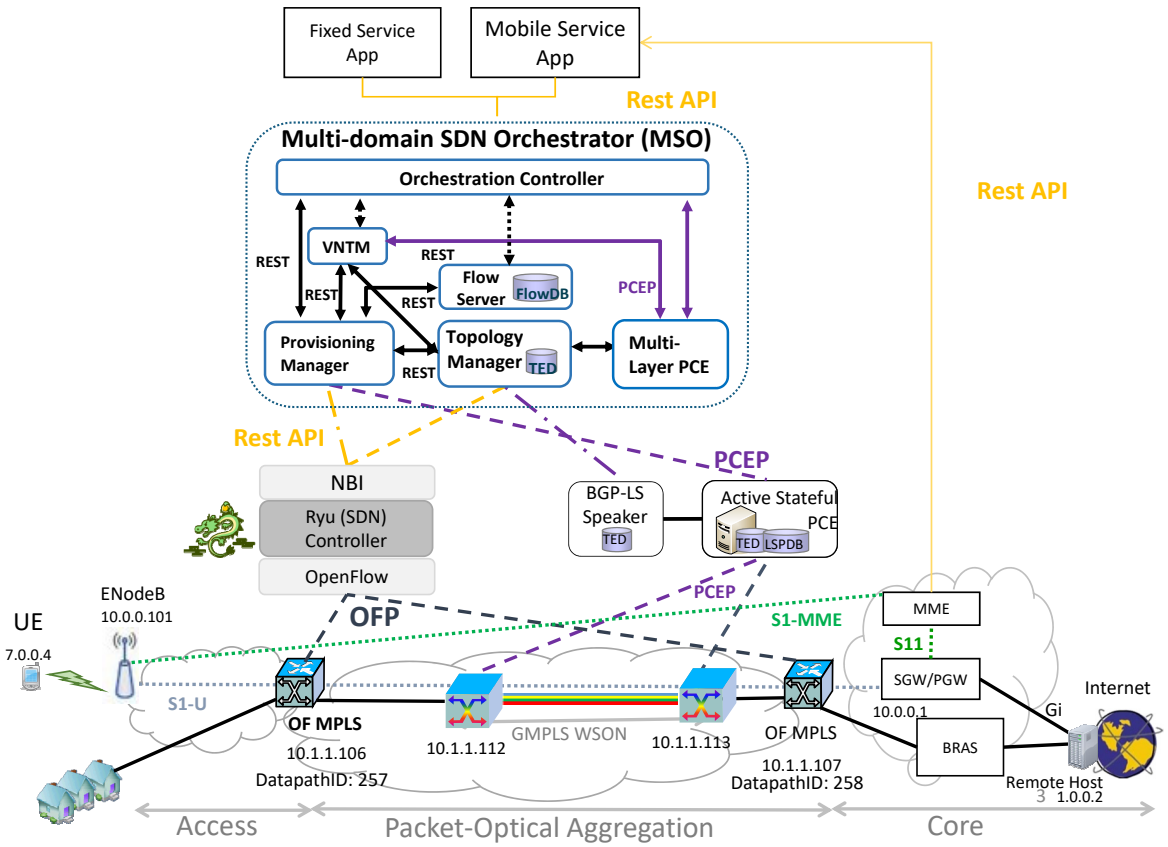


Figure 5.8: Multi-layer aggregation MSO for Fixed-Mobile convergence.

The considered SDN-based ABNO orchestrator is depicted in Figure 5.8. The Service Controller

handles incoming service requests from both fixed and mobile service applications. These applications trigger the service demands (via REST API) specifying the type of transport service (e.g., MPLS), endpoints (e.g., for mobile connections the eNBs and EPC Serving Gateway - SGW), the requested bandwidth and other QoS requirements (e.g., maximum latency). As shown in the Figure 5.8, in the aggregation network two heterogeneous control plane solutions (one per switching technology) are adopted: the control of MPLS switches is done by an SDN packet controller (Ryu); and the control of optical circuits is handled by an Active Stateful (AS) PCE. When a new bearer needs to be set up between one eNB and the EPC, a new MPLS service is requested to the MSO associated to the EPS Bearer. The control workflow in the MSO is equivalent to the one explained in the previous section.

The experimental setup is formed by three main elements: the LTE-EPC network provided by the LENA emulator [66] (including User Equipment and eNBs), the multi-layer OF-enabled MPLS - optical aggregation network deployed within the ADRENALINE testbed and the unified MSO discussed above. Without losing of generality, the experimental validation aims at transporting mobile services (EPS Bearers) between the eNBs and the EPC (SGW) via the multi-layer aggregation infrastructure. All the mechanisms and functions to do so are automatically coordinated by the MSO.

Focusing on the mobile services, once the EPS Bearer is negotiated between the eNB and the EPC Mobile Management Entity (MME) [67] through the (out-of-band) control S1-MME interface, the MME communicates with the Mobile Service App (running on top the ABNO) to request the transport of EPS Bearer data/user packets (i.e., S1-U interface). The interface between the MME and Mobile Service Apps is implemented using a REST API. Indeed, a new generic service call is defined to request from both Mobile and Fixed Service Apps, multi-layer transport connections handled by the MSO.

For the EPS Bearers, the service call specifies the endpoint IP addresses (i.e., eNB and SGW), the requested bandwidth (ReqBw) in Gb/s, and specific match attributes (such as mobile data packet attributes). The latter allows mapping EPS Bearers with specific MPLS flows within the aggregation network. Specifically, EPS Bearers use the GPRS Tunneling Protocol (GTP-U) to transport data packets between the eNB and the EPC. Each EPS Bearer flow (S1-U) has an individual Tunnel Endpoint Identifiers (TE_ID) which is carried into the GTP-U protocol. In this work we bind such TE_IDs with individual MPLS labels. That is, we apply a policy where every EPS Bearer (with its own TE_IDs) is transported over a different MPLS flow. Nevertheless, multiple MPLS flows may be aggregated into a unique optical tunnel.

Figure 5.9 (upper) depicts the messages exchanged among the MSO elements when a (mobile) service call is received. The service call includes the transport layer type (e.g., MPLS), the endpoints (eNB at 10.0.0.101; SGW 10.0.0.1), the TEIDs (set to 2 in the complete message sequence shown in the figure) and the requested bandwidth. On the other hand, in the SDN packet controller, the extended OpenFlow 1.3 OFPT_FLOW_MOD message with experimental matches configures the MPLS nodes according to the EPS Bearer attributes (S1-U interface). That is, for each EPS Bearer, the GTP-U packet is encapsulated and decapsulated over a MPLS tunnel. As shown in Figure 5.9 (bottom-left), the OFPT_FLOW_MOD carries a set of match rules and actions for the MPLS nodes that define the processing and treatment of the data packet (GTP-U) of each EPS Bearer. In the example, the match rules impose that: all the GTP-U packets received over an incoming port with the tuple formed by a determined pair of source and destination IP addresses (i.e., eNB 10.0.0.101 and SGW 10.0.0.1), UDP port set to 2152 and the TEID equals to 2, then a MPLS tag (1002) is pushed and the resulting packet is forwarded to the output port towards the optical domain. Similar operations but removing the MPLS tag are done when the EPS Bearer leaves the MPLS domain prior to be delivered to either eNB or SGW (downlink and uplink flows).

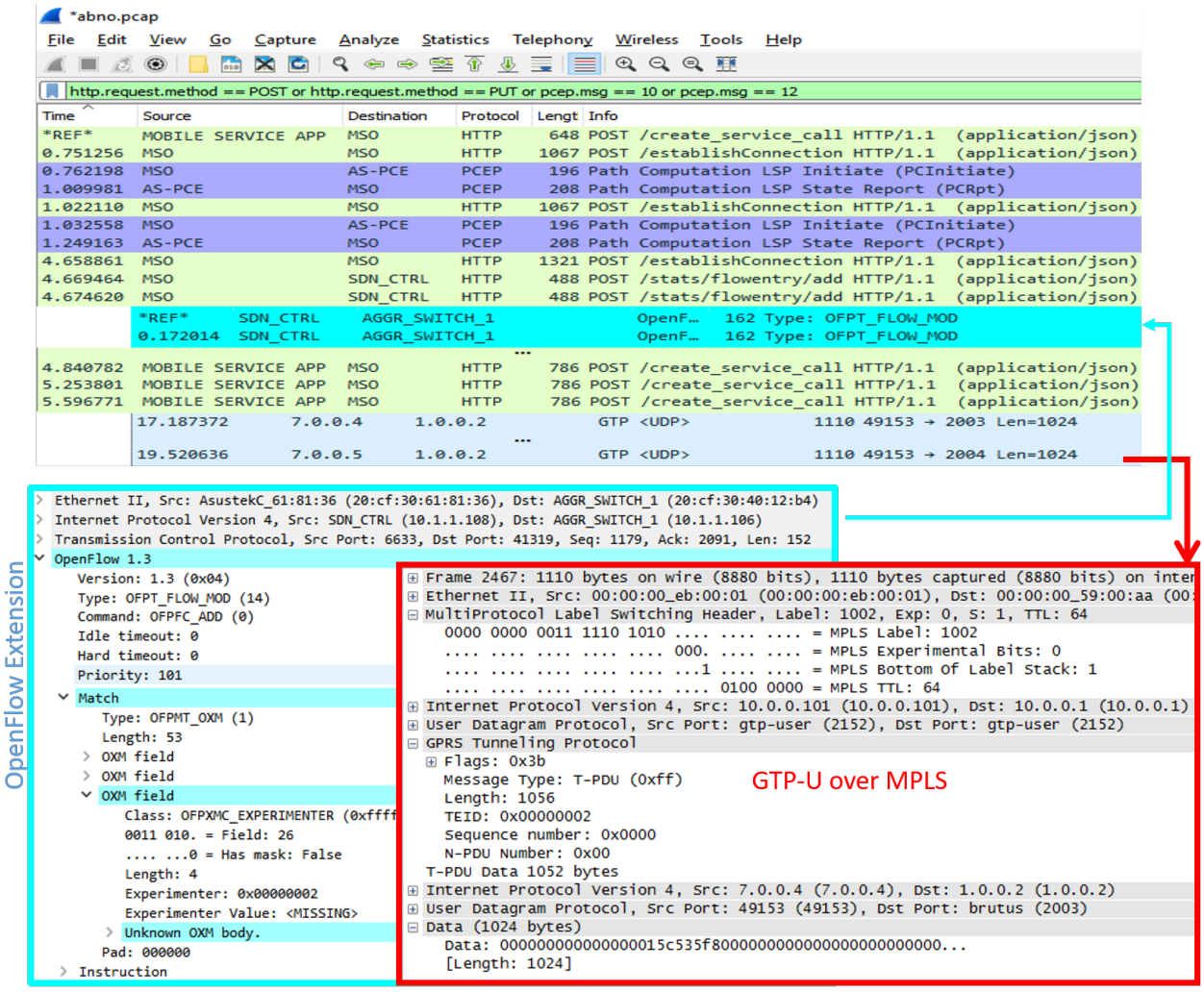


Figure 5.9: Creation of the mobile service via MSO, OFP Extension and GTP-U over MPLS.

5.5 Performance evaluation

In this section the previously proposed orchestration architecture is going to be examined measuring the time performance of the different orchestration operations realized on it. For all the experiments the network scenario is the Cloud Computing Platform and Transport Network of the ADRENALINE Testbed, described in Chapter 3.

5.5.1 Topology discovery and transfer analysis

The topology discovery and aggregation feature in the MSO implementation is analyzed to quantify the weight of mid-sized domain topologies and the time elapsed to retrieve them from the control plane. The aggregated multi-domain topology transfer throughput between MSO components is also analyzed. In order to analyse the throughput of the transfer/retrieval operations, the traffic between the TM and the SDN controllers has been captured with a Wireshark during the MSO initiation process. In Figure 5.10 the throughput of the topology retrieval form the network controllers and the MSO is shown. In the OF domains, each transfer is in the order of 60ms because it involves several independent HTTP requests and in the GMPLS the whole topology retrieved in a single request in less than 10ms. Finally, in the right side of the figure, the throughput of the aggregated topology

transfer between the TM and the PCE, with a total weight of 134.4 Kbytes in approximately 15ms, is graphically represented.

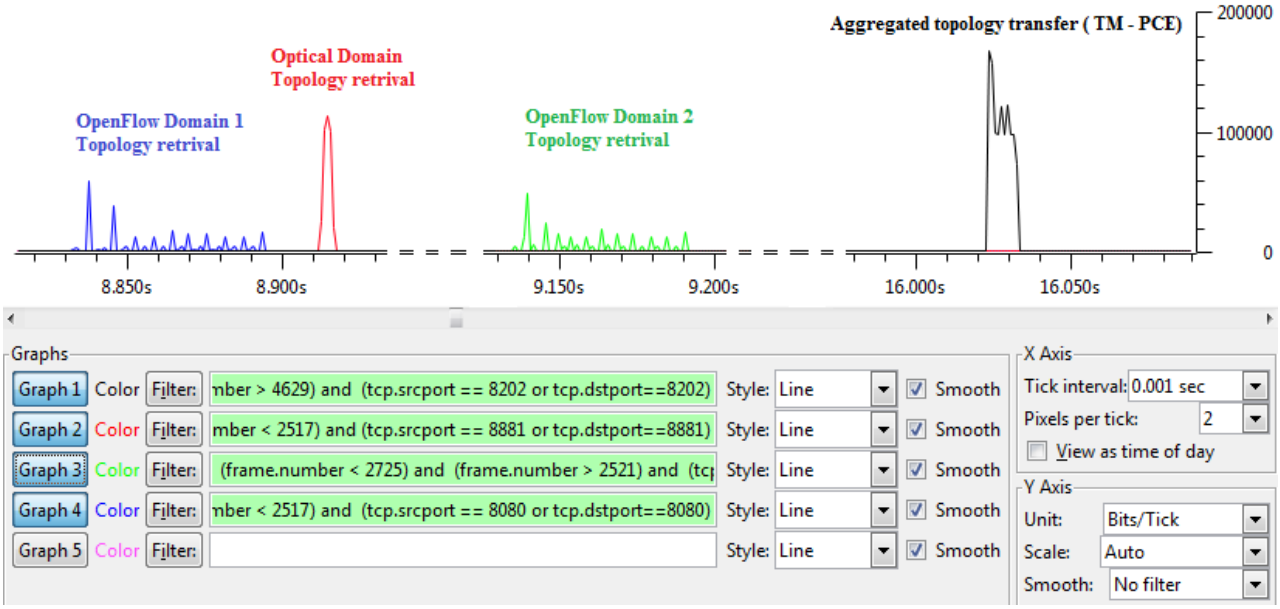
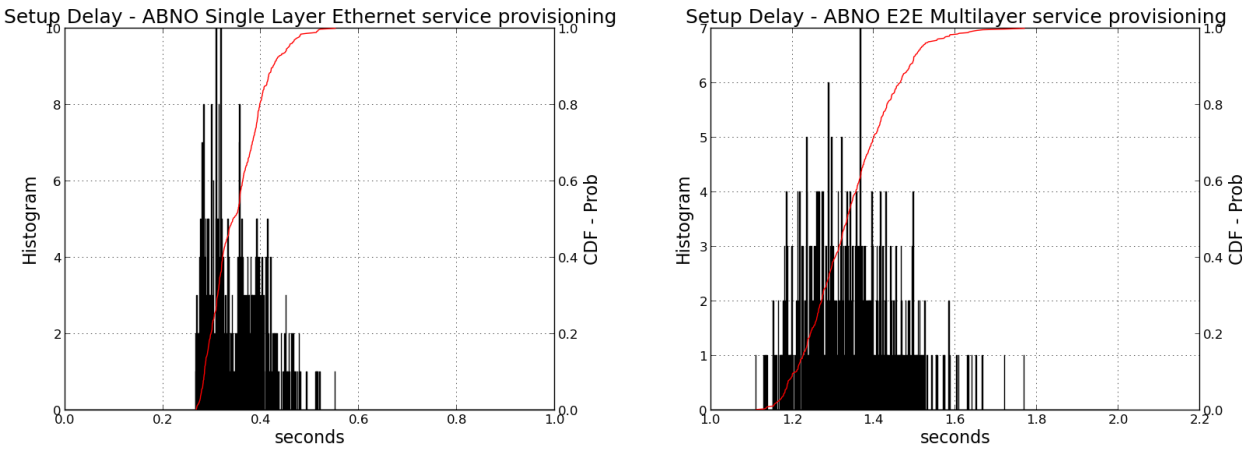


Figure 5.10: Topology retrieval/transfer throughput analysis.

5.5.2 Single layer and multi-layer E2E service provisioning performance evaluation.

In order to evaluate the time performance of our orchestration platform, two separated setup delay evaluations have been carried out: the orchestration of E2E service provisioning involving only single layer (L2/OpenFlow) flow provisioning, and the multi-layer orchestration involving sub-second integrated provisioning of OpenFlow and GMPLS orchestrated connections and the Vlink creation, supported by a bidirectional Label Switched Path (LSP) establishment.



(a) (b)

Figure 5.11: (a) Single Layer E2E and (b) Multilayer E2E service provisioning setup delay distribution.

In Figure 5.11a-5.11b, the histogram and the Cumulative Distribution Function (CDF) of the setup delay introduced by the MSO during the processing of every E2E service requests in a 1000

samples experiment are presented for the two cases exposed before. The E2E connections requests are uniformly distributed among the nodes within the network scenario presented in Figure 5.3. All the E2E requests start and finish at L2 endpoints and have the same TE characteristics (10 Mbps) and operate at the L2 transport layer (source and destination MAC address described the traffic flows introduced into the OF switches). Every E2E connection is released before the next iteration.

5.6 Conclusions

This chapter has been presented an in-depth discussion about the network orchestration problem in multi-domain networks comprising different transport technologies and control planes. The main challenges regarding the orchestration of multiple transport technologies have been presented as the introduction of the proposed solution in this PhD.Thesis, the Multi-domain SDN Orchestration (MSO) architecture.

The MSO architecture includes several well-defined building blocks to perform the main tasks involved in the control and management of multi-layer, multi-domain transport networks. The solution proposed features different southbound plugins for the orchestration of different control technologies by applying the main SDN concepts (control and data plane separation, definition of open and standard APIs).

Moreover, the MSO has been evaluated in the Cloud Computing Platform and Transport Network of the ADRENALINE Testbed for the demonstration of two different use cases: (i) SDN orchestration of TE-aware multi-domain, multi-layer networks, and (ii) Automatic Provisioning of Fixed and Mobile Services.

Finally, a performance evaluation of the MSO has been carried out with the E2E connection provisioning and the topology discovery and transfer features. The results presented have shown a performance of an average of 300ms setup delay for single-layer requests and of 1350ms for multi-layer E2E connections in lab-trial. This result led us to conclude that our MSO implementation is ready for future studies on online operations such as dynamic network restoration and online network planning optimization.

To continue the work presented in this chapter, the need of an standard interface between the orchestration and the control layers has been a recursive topic highlighted throughout the sections of this chapter. In the next chapter this topic is introduced and a novel protocol that abstracts the set of control plane functions used by the SDN Controller is proposed.

Chapter 6

The Control Orchestration Protocol (COP)

The need of offering end-to-end Ethernet service provisioning and orchestration across multiple domains with heterogeneous transport and control plane technologies was justified in the previous chapter. It was shown that in order to realize such end-to-end connectivity service provisioning, the SDN based service and network orchestration layer was required.

In this chapter, the Control Orchestration Protocol (COP) is proposed to simplify the orchestration procedures by unifying the northbound interface of the control layer, by abstracting a common set of control plane functions shared among the different implementations of the SDN controller.

The chapter is structured as follows: in section 6.1, the main requirements and design principles of the COP are introduced. Section 6.2 the actual COP protocol is formally described including the definition of its data model in YANG modeling language, and its interfaces in RESTCONF and SWAGGER APIs. Finally, in section 6.3 the COP is experimentally validated for the End-to-end service provisioning and recovery in OPS/OCS multi-domain networks.

A specific appendix A has been included at the end of the document where the COP definition is completed with the complete set of interfaces and information models.

6.1 Requirements identification, modeling and design

The Control Orchestration Protocol (COP) abstracts a common set of control plane functions used by various SDN controllers, allowing the interworking of heterogeneous control plane paradigms (i.e., OpenFlow, GMPLS/PCE).

The COP is aware of the existing background in network programmability and applies new SDN principles to enable cost reduction, innovation and reduced time to market of new services, while covering multi-domain and multi-technology path/packet networks.

This COP provides two main functionalities:

- Network-wide centralized orchestration. This high level, logically centralized entity exists on top of and across the different network domains and is able to drive the provisioning (and recovery) of connectivity across heterogeneous networks, dynamically and in real time.
- Abstraction of the particular control plane technology of a given domain. In this sense, the proposed architecture applies the same abstraction and generalization principles that OpenFlow/SDN have applied to data networks.

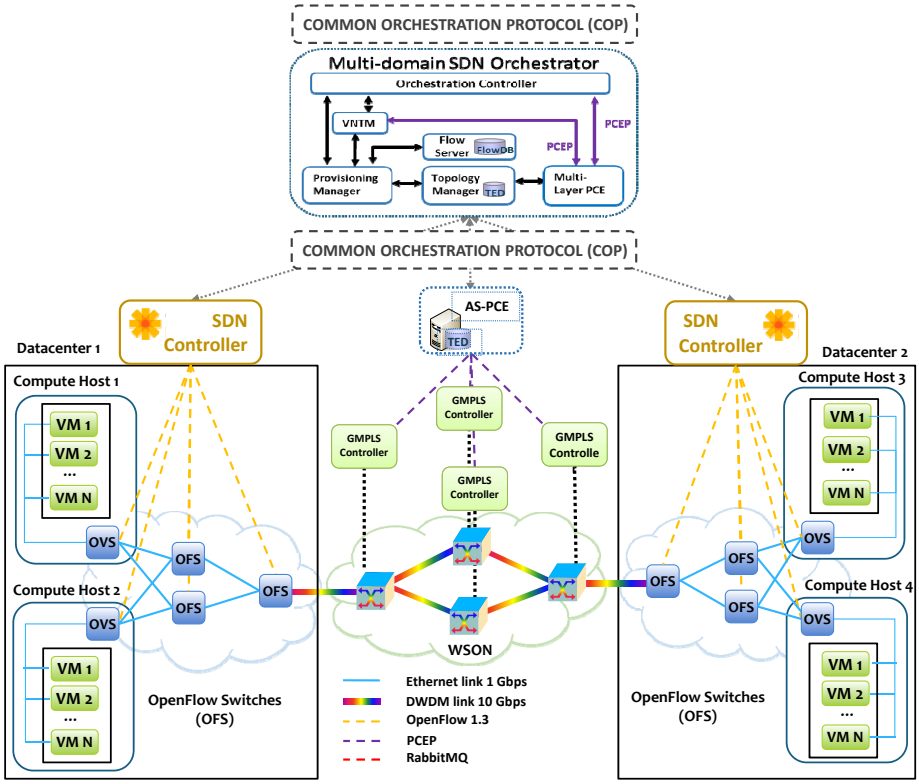


Figure 6.1: Multi-domain SDN Orchestration architecture with Control Orchestration Protocol (COP) as unified southbound (SBI) and northbound (NBI) interface

The Multi-domain SDN Orchestrator (MSO) (Figure 6.1) works under the assumption that each domain is composed of a data plane controlled by an instance of a given control plane technology, but transport and/or control plane technologies for each domain can be different. The main functionalities of the SDN orchestrator are abstract and not technology related. This control plane abstraction must enable the provisioning of data services using the underlying configuration technology.

The design of COP between the orchestration and control layers allows the simplification and optimization, in terms of scalability and compatibility between the different modules which compose the SDN architecture. COP unifies all the orchestration functionalities into a single protocol paradigm. The main reason for the need of this protocol is the heterogeneity of NorthBound Interfaces (NBI) of SDN controllers. Most of the current SDN controllers (e.g., OpenDayLight, ONOS) provide their own northbound API, which allows applications to directly program the underlying network resources, which are exposed by the SDN controller. The proposed COP provides a common NBI API so that SDN controllers can be orchestrated using a single common protocol.

From a Transport SDN perspective, it is needless to mention that there is a lack of specifications on how user applications should interact with the underlying network resources and how services should be requested. The proposed COP provides the necessary commands to bring the full benefits of programmable SDN transport networks to applications. The latest OIF/ONF Transport SDN API is in line with COP objectives. COP provides a research-oriented multi-layer approach using YANG/RESTCONF, while OIF/ONF Transport SDN API is focused on standardization efforts for orchestration of REST NBI for SDN controllers.

The COP definition is open for discussion and can be downloaded and contributed to at: `https://github.com/ict-strauss/COP`

6.2 Control Orchestration Protocol definition

In the following subsections the base definition for COP is presented. The usage of YANG models and RESTCONF protocol is also discussed.

6.2.1 COP data model definition based on YANG

After identifying the different functionalities a common orchestration interface must fulfill in order to provide useful orchestration mechanisms in a wide range of possible SDN scenarios, it is necessary to introduce the COP models in detail. For the formal definition of the models, YANG has been identified as the most suitable modeling language given its rich set of primitives for network management and data modeling.

6.2.1.1 Call Service

The COP service call is defined as the E2E provisioning interface. It is based on the concept of Call/Connection separation where a service is supported by a set of effective connections established in the network. The Call object formalize the intent of a service provisioning between two Endpoints, it defines the type of service that is requested or served (e.g., DWDM link, Ethernet Transport, MPLS), the Traffic Engineering (TE) parameters requested for the service (e.g., bandwidth, QoS class, latency) and may also include filtering parameters or matches (i.e., MAC, IP or TCP headers) which specifies the granularity of the services to be served.

The Call object also includes the list of effective Connections created in the data plane, to implement the E2E service. A Connection is single network domain scoped, however multiple connections may be established within a single domain. The Connection includes the path across the network topology that the data traverses, which may be fully described or abstracted depending on the orchestration/control schemes used.

The Call service provides the interface for E2E provisioning services but also includes monitoring capabilities for already deployed connections/calls. The COP allow an external entity to subscribe to the notification services of another control entity, through Websockets, to asynchronously receive information about the state of the calls controlled by the control entity.

In the Appendix A, the Figure A.1 shows the UML diagram corresponding to the Call service YANG model. The complete CALL yang file can be found at: `https://github.com/ict-strauss/COP/blob/master/yang-cop/service-call.yang`.

6.2.1.2 Topology Service

The COP allows to retrieve the topological information about the network, by providing a technology agnostic information model for the description of topology elements: Nodes, Edges and EdgeEnds.

The information model is arranged on a tree structure which base node is a Topology object which includes as leafs the list of the nodes and edges included in the related network. A Node must contain a list of ports or edgeEnds and their associated switching capabilities. An Edge object is defined as the connection link between two EdgeEnds. Due to the need of conforming to a common model among different transport network technologies, the definition of the three main objects described (Node, Edge, EdgeEnds) is extensible, featuring polymorphism programming concept, to define specific Node and Edge types. So far, the COP includes the definition of DWDM and Ethernet nodes and edges which defines the technology related switching capabilities from now for packet and wavelength switching.

In Appendix A, Figure A.2 shows the UML diagram corresponding to the Topology service YANG file. The complete Topology yang file can be found at: `https://github.com/ict-strauss/COP/blob/master/yang-cop/service-topology.yang`.

6.2.1.3 Path Computation Service

The Path Computation service provides an interface to request and return Path objects which contain the information about the route between two Endpoints. These operations are modeled as Remote Procedure Calls (RPCs) in the YANG definition and they are included into the operations subset of the RESTCONF API. Path computation is highly related to the previous group of resources. In the service Call, the Connection object has been designed to contain information about the traversed Path. The Path model is the same in both, the service Call and at the Path Computation. Furthermore each component in the Path object is represented as an Endpoint with TE information associated to it. Although basic service functionality has been modeled, there are some extensions proposed which are open for discussion, such a Backup Path request model, the Shared Risk Link Groups (SRLGs) or the Exclude Route Object (XROs).

The UML diagram corresponding to the Path Computation service YANG file (Figure A.3 shows) can be found in Appendix A. The complete Topology yang file can be found at: `https://github.com/ict-strauss/COP/blob/master/yang-cop/service-path-computation.yang`.

6.2.2 COP interface definition based on RESTCONF/SWAGGER

YANG/RESTCONF provides the suitable combination for COP in order to provide the necessary flexibility and usability.

The Section A.2 of Appendix A includes the complete COP RESTCONF definition including the set of HTTP interfaces and the JSON-encoded data models. For each COP service, two documentation

figures has been generated: the complete set of RESTCONF paths (HTTP urls) and the complete JSON data models.

In the next subsection it is described the Open source software resources developed for the translation from the YANG source COP models to the RESTCONF interfaces definition based using SWAGGER software tools, and finally the implementation of a baseline COP server stub.

6.2.2.1 Open source YANG tools for COP

In the scope of the STRAUSS project, where the COP was firstly presented, a set of OpenSource software tools, were developed to translate the COP definition in YANG, into a baseline implementation of the protocol.

This YANG Tools are open for use and contribution in `https://github.com/ict-strauss/COP`. The YANG tools are divided in two groups.

6.2.2.1.1 Pyang plugin for Swagger

Pyang is an extensible YANG validator and converter written in python.

It can be used to validate YANG modules for correctness, to transform YANG modules into other formats, and to generate code from the modules. We have written a pyang plugin to obtain the RESTCONF API from a yang model.

The contribution of the STRAUSS project was to develop a specific plugin automatically translates the YANG models into the Swagger Specification version 2.0 [68]. The RESTCONF API of the YANG model is interpreted with Swagger, which is a powerful framework for API description and implementation. By using the Swagger software it is possible to generate the complete RESTCONF API definition from the source YANG models.

The swagger pyang plugin file and the documentation of how to use it, can be found at: `https://github.com/ict-strauss/COP/tree/master/pyang_plugins`.

6.2.2.1.2 COP Server Generator for Python

The second group of YANG tools is a python code-generator base on the SWAGGER/RESTCONF protocol specification files. This second tool plays a very important role in the early stages of the COP evaluation in the multi-partner environment of the STRAUSS project. It allowed the different players involved in the project to automatically generate their own server stub in python to speed up the interoperability testing process.

However, this tool development was discontinued due to the adoption of the Swagger code-generator sofware which is a more complete and robust implementation and provide output code in multiple frameworks (Python, JAVA, Scala, JavaScript, etc.).

Besides, the STRAUSS COP Server Generator for Python is still available ins the github repository, altogether with the documentation of how to use it.

6.3 Experimental validation

6.3.1 Use case I: End-to-End service provisioning and recovery in OPS/OCS multi-domain networks

This section presents the experimental validation of End-to-End service provisioning and recovery in OPS/OCS multi-domain networks using COP as unified transport API.

Figure 6.2 shows the integrated cloud and heterogeneous network scenario. The proposed scenario includes a distributed datacenter (DC) infrastructure placed in two different locations and managed by different institutions (LIGHTNESS and CTTC). DCs are interconnected by and heterogeneous transport network, consisting on isolated administrative OPS/OCS domains. Each network domain is controlled either by an SDN controller, an Optical Network Hypervisor (ONH) or an Active Stateful PCE.

On top of the control plane the multi-domain SDN Orchestrator (MSO) is responsible of coordinate the different controllers to provide E2E network services. It integrates the COP as southbound interface to communicate with the different controllers. Each domain provides its abstracted topology (node abstraction) through the COP Topology Service. All topological information is gathered by the Topology Manager MSO's component which is responsible to compose the multi-domain abstracted topology. The inter-domain connectivity is pre-loaded into the MSO by static configuration files. Figure 6.3 shows the topology composed by the SDN orchestrator.

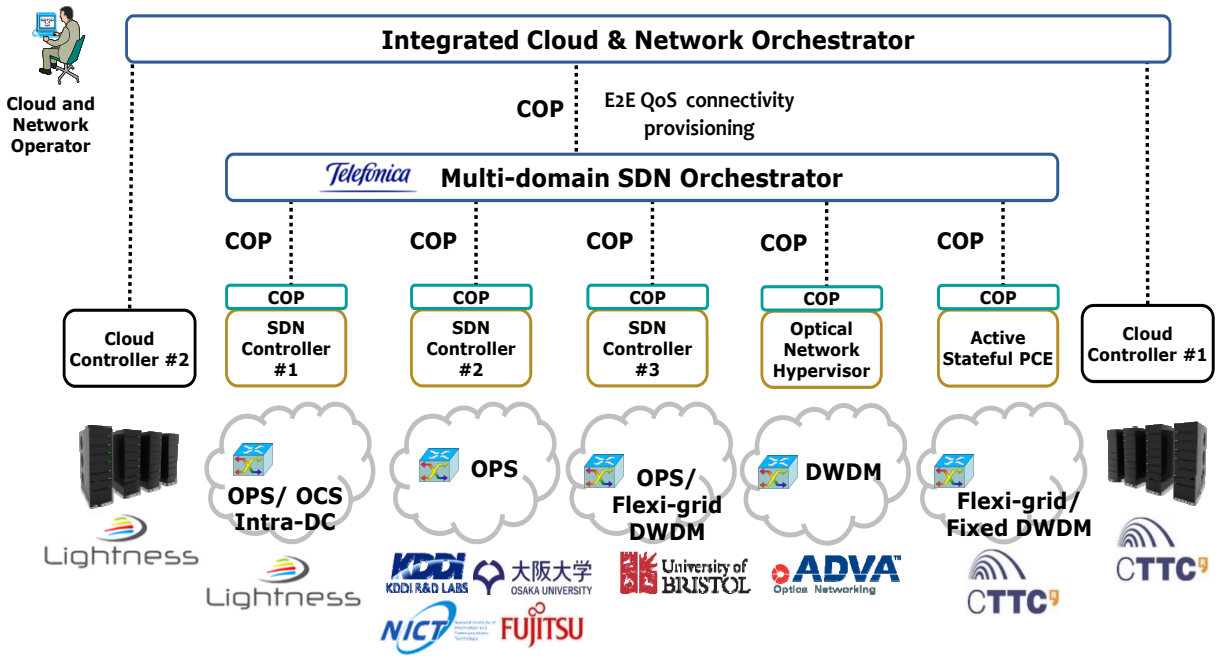


Figure 6.2: Multi-domain experimental multi-partner testbed scenario

E2E QoS capabilities are introduced by the definition of two QoS classes (Figure 6.1). Each QoS class defines a certain packet loss rate (PLR) for OPS domains, and a certain OSNR for OCS domains, for a given bandwidth request. The SDN orchestrator will convert the high level QoS classes into the corresponding traffic parameters (OSNR, PLR) based on the values (Figure 6.1) pre-installed in the

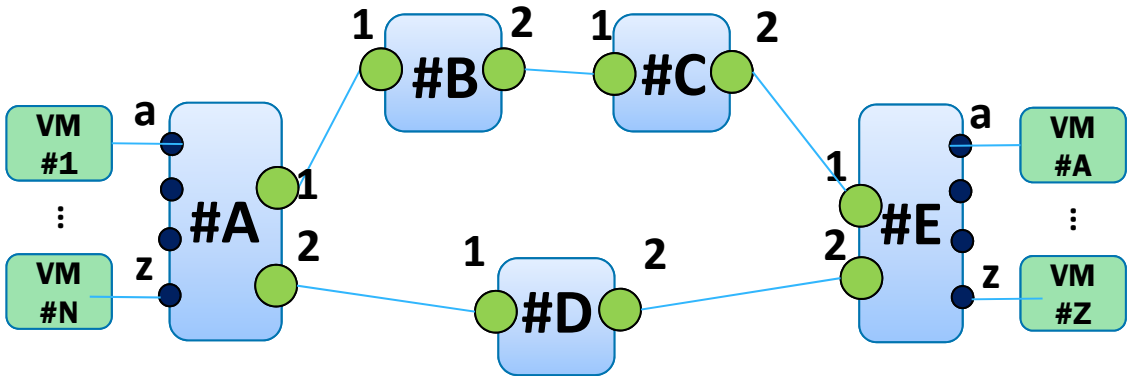


Figure 6.3: Abstracted topology seen by the MSO

	OCS Domains	OPS Domains
QoS Class	OSNR	PLR
Gold	< 23 dB (PM-16QAM, 224Gb/s)	< 0.1%
Silver	< 14 dB (PM-QPSK, 112 Gb/s)	< 4%

Table 6.1: QoS classes

SDN orchestrator. Those parameters are included in the *trafficParams* object within the Call request messages to be sent to the domain controllers.

6.3.1.1 E2E transport service deployment with QoS provisioning

Figure 6.4 shows the provisioning workflow which involves the orchestration of computing and network resources to offer an E2E system. The network service is requested to the MSO through the COP Call service by requesting the creation of a new Call (ID: 1) between the network Endpoints to which the VMs are attached, specifying the QoS class (Figure 6.7, trafficParams). The MSO computes the path across the different domains and issues a call creation request (IDs: 00001,00002,0000,00004) to the involved domain controllers. Those calls represent the abstract connections provisioned on each domain and , once they are established, they are included in the connections list of the E2E Call. Figure6.5 shows the Wireshark captures at the integrated cloud and network orchestrator and at the SDN orchestrator. VM (1 CPU, 20GB disk, 2GB RAM) creation process took around 15 seconds for each instance, the total E2E connectivity service in the current multi-domain scenario is 1.13 seconds.

6.3.1.2 Per-domain / E2E service recovery with QoS

Figure 6.6 shows three conducted experiments for QoS recovery: in an OPS domain (scenario A), in an OCS domain (scenario B) and finally E2E QoS recovery (scenario C).

The domain controllers are continuously monitoring the Packet Loss Ratio (PLR) and the Optical Signal Noise Ratio (OSNR) (OPS and OCS domains respectively). In the OPS domain when the PLR for a given flow arises over the specified QoS level (Figure 6.1), the packet congestion is detected by

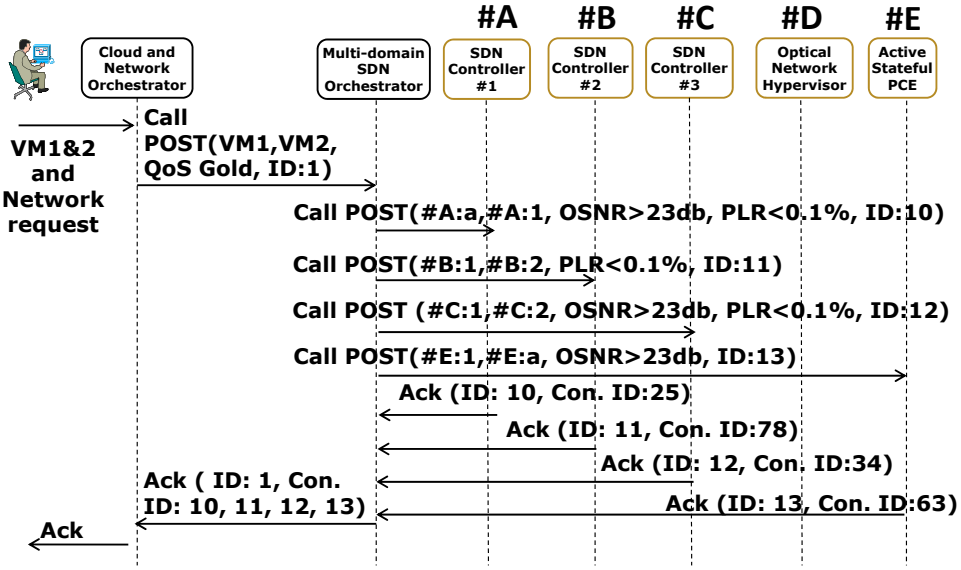


Figure 6.4: E2E QoS-aware service provisioning workflow in the proposed OPS/OCS multi-domain network scenario.

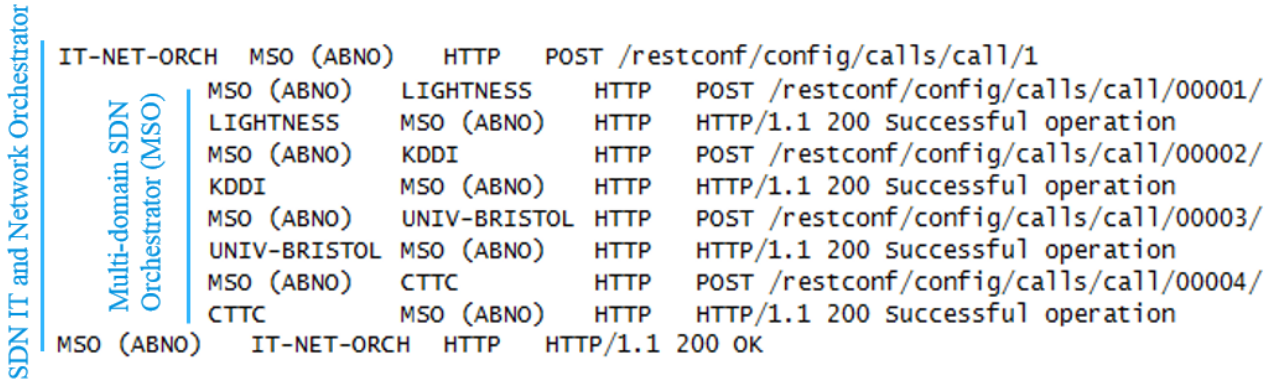


Figure 6.5: Experimental validation of COP call service for QoS-aware E2E connectivity service provisioning traffic capture

the local SDN controller and a local path restoration is triggered. If the local restoration succeeds, the route adaptation is announced to SDN orchestrator by an *updateServiceCall* notification to update the call service information with the new *connectionID* which refers to the new intra-domain connection created across the new path (6.6 Scenario A).

In the OCS domain, OSNR monitoring of a circuit flow can detect the OSNR degradation for optical links. The receiver-side error-vector-magnitude (EVM) based monitor provides in-band OSNR monitoring without deploying new hardware [69]. With these monitoring information, the COP can orchestrate multi-domain E2E service efficiently and reconfigure the network according to the traffic and link conditions to maintain QoS. The monitor notifies the SDN controller, when the OSNR is degraded up to a threshold specified by the multi-domain orchestrator during the provisioning. Then SDN controller reconfigures the link either to use another path or to adopt a lower order modulation format signal with a multi-format transceiver (scenario B).

When a transport domain is unable to recover itself from a failure, or QoS cannot be ensured, the

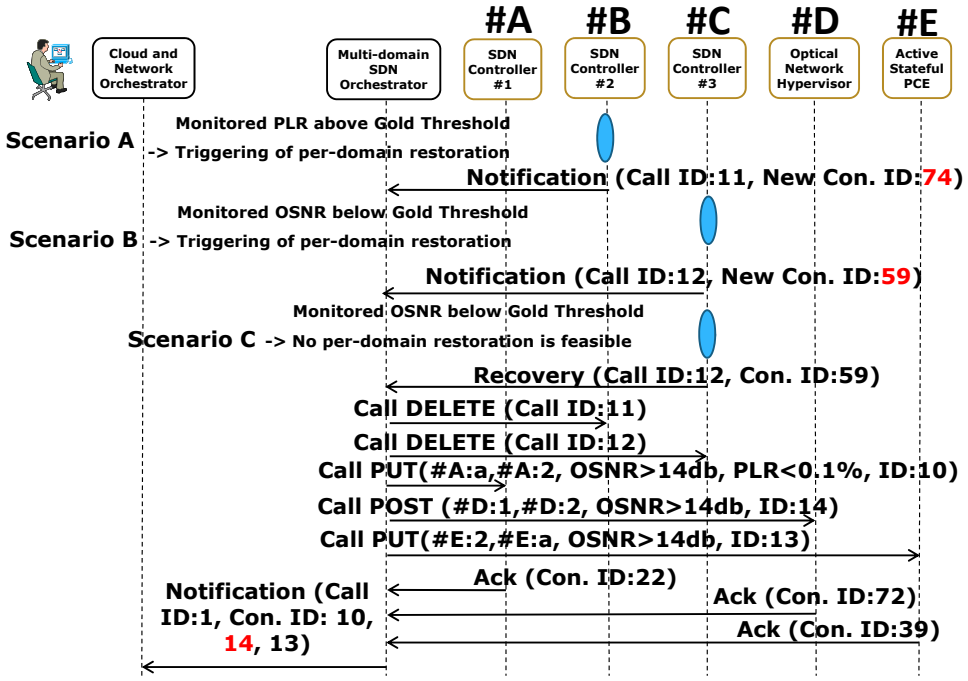


Figure 6.6: E2E service recovery with QoS workflow.

SDN orchestrator must be notified to perform an end-to-end QoS recovery (Fig.6.6, scenario D). This end-to-end recovery has a multi-domain scope, it involves the calculation of a new alternative route that satisfies the required QoS constrains, which avoids the network domain on which the failure has occurred.

Figure 6.7: Experimental validation of COP call service for QoS-aware E2E QoS transport service provisioning (Call Object).

Figure 6.8 shows the control messages between the orchestrator and the controller instances during the conducted experiment. First the IT and Network orchestrator subscribes the *updateCallService* and *removeCallService* services of the MSO through the corresponding URL (*/restconf/streams/serviceName*), the same process is performed by the MSO to each domain controller. When the MSO receives the notification from the *removeCallService* of Domain C, it calculate the new E2E path (excluding Domain C) which satisfies the required QoS (A-D-E) and it requests the necessary call deletions (IDs: 00002, 00003), modifications (IDs: 00001, 00004) and the establishment of a new call in Domain D (ID: 00005) to the corresponding SDN controllers. Once the E2E service is restored, it informs the cloud and network orchestrator about the E2E call modification (ID:1). The whole E2E recovery process in the current scenario took 0.624 seconds since the *removeCallService* notification was received by the orchestrator until the last Service Call response is received by the orchestrator too.

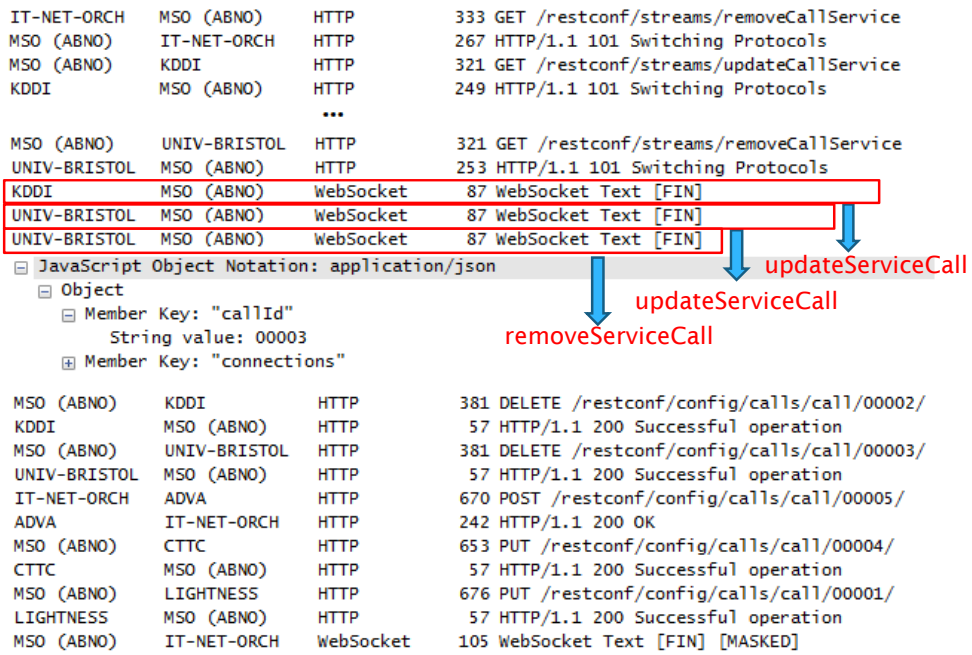


Figure 6.8: E2E service recovery with Traffic capture.

6.4 Conclusions

The Control Orchestration Protocol (COP) has been presented as a common protocol for the interworking of heterogenous control plane paradigms. The COP abstracts a set of control plane functions used by an SDN Controller, allowing the SDN orchestrator to uniformly interact with several domains, each controlled by a single SDN controller. COP has been defined using YANG model language and can be transported using RESTconf. The formal specification of the COP information model and its RESTCONF implementation is also included in the Appendix A of the present document.

To conclude this chapter, the COP was experimentally demonstrated in a multi-partner international control and data plane testbed, by two proof of concepts: the dynamic provisioning of E2E IT and network resources across the aforementioned network; and E2E QoS assurance, including per-domain and end-to-end QoS recovery based on data-plane QoT monitoring.

Chapter 7

The Hierarchical SDN Orchestration (H-ORCH) approach

In Chapter 5, SDN orchestration was proposed as a feasible solution to handle the heterogeneity of network domains, technologies and vendors. It focuses on network control and abstraction through several control domains, whilst using standard protocols and modules. A network domain is understood as a set of Network Elements (NE) under a logically centralized SDN Controller. A multi-domain SDN Orchestration (MSO) has been already analyzed in several contexts, such as pure OpenFlow (OF)-enabled networks, and heterogeneously-controlled networks (GMPLS/PCE and OF). Several initiatives in standardization bodies such as ONF or IETF advocate for the necessity of SDN orchestration.

In this chapter it is introduced a novel SDN orchestration architecture based on different levels of hierarchy, allowing the network resource abstraction and control. A level is understood as a stratum of hierarchical SDN abstraction. The need of hierarchical SDN orchestration has been previously justified in [70], to accomplish two main purposes: a) to improve the scalability and modularity of the actual SDN control architectures: each successively higher level has the potential for greater abstraction and broader scope, b) Security: each level may exist in a different trust domain. The level interface might be used as a standard reference point for inter-domain security enforcement. The benefits of hierarchical SDN orchestration become clear in the scope of the described future 5G networks with technology heterogeneity.

The remaining of the chapter is organized as follows: in section 7.1, the hierarchical orchestration approach is presented including an overall description of the network scenario to which applies to. In section 7.2, the extensions applied to the MSO architecture to support the proposed hierarchical approach are described. The experimental validation of the proposed architecture is included in section 7.3, and to conclude the chapter, in section 7.4 we characterize the performance of the H-ORCH approach in single and multi-domain scenarios.

7.1 Architecture overview

This section covers the description of the proposed Hierarchical SDN Orchestration (H-ORCH) architecture based on a hierarchy of MSO instances (parent/child MSOs). The architecture is displayed in Figure 7.1 over a sample 5G network consisting on multiple RANs and DC networks interconnected by an optical transport network. In the Radio Access Network (RAN) segment, we observe several SDN-enabled controllers for wireless networks, which tackle their complexities. In a transport network, the aggregation segments and core network are taken into account. SDN-enabled MPLS-TP can be used in the aggregation network, while a core network might use an Active Stateful PCE (AS-PCE) on top of a GMPLS-controlled optical network. Finally, several SDN-enabled controllers are responsible for intra-DC L2 networks.

Within the hierarchy, an SDN orchestrator may consider itself as the direct control entity of an information model instance that represents a suitable abstracted underlying network. It follows that, with the exception of network domain SDN controllers (which are directly related to NE), a given SDN orchestrator might provide an abstracted network view and be present at any hierarchy level and act as parent or child SDN orchestrator. At any level of the recursive hierarchy, a resource is understood to be subject to only one controlling entity.

In the proposed architecture, several child MSOs (cMSO) are considered. Each cMSO is responsible for a single network segment. A recursive hierarchy could be based on technological, SDN controller type, geographical/administrative domains or network segment basis. We introduce a parent MSO (pMSO), responsible for the provisioning of End-to-End (E2E) connections through different network segments.

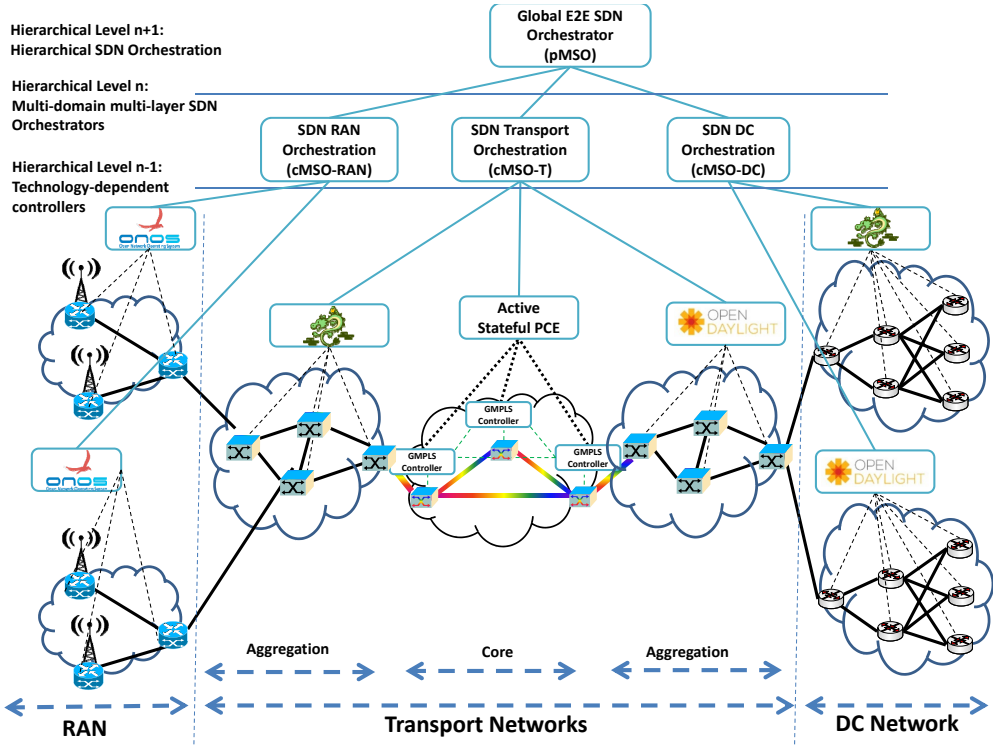


Figure 7.1: Proposed Hierarchical SDN Orchestration (H-ORCH) architecture in a multi-domain network scenario.

7.2 MSO extensions for H-ORCH: Abstraction Manager

For both the pMSO and the cMSO, the internal system architecture is identical and based on an extended version of the MSO architecture presented in chapter 5 (Figure 7.2).

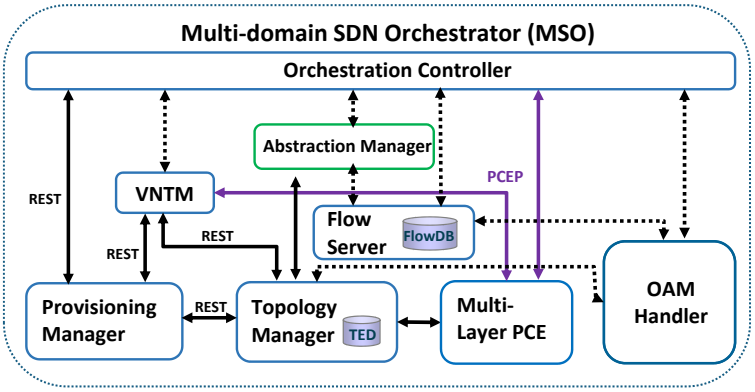


Figure 7.2: Extended MSO internal architecture

The Orchestration Controller is the component responsible for handling the workflow of all the processes involved (e.g., the provisioning of end-to-end connectivity services). It also exposes a NBI to offer its services to applications. For the cMSO, we have extended the NBI of the Orchestrator Controller (OC) to offer a REST based interface for virtual network creation. This new interface interacts directly with a new module named the Abstraction Manager (AM) to compose abstract representations of the underlying physical network topology. The proposed AM might be able to

provide several types of hierarchical level abstraction. The AM support two abstraction models for the creation of the virtual network: node and link abstractions. The node abstraction model offers to the higher level of hierarchy the underlying domains/segments as a single abstract node, with the input/output interfaces of the underlying domains/segments as interfaces of each abstracted node. On the other hand, a link abstraction represents a physical path between two interfaces of the underlying network as a virtual link between two abstract nodes. These two nodes must be mapped 1:1 to the physical nodes to which the physical interfaces corresponding to the source and the destination virtual edge ends are attached to. Additionally, in order to reduce the complexity of the models, the abstracted topologies being considered in this work are single layer. This implies that the transport layer of the service endpoints mapped to the abstracted representation must operate at the same layer and, in case of the link abstraction, the switching capabilities of the abstracted link must be the same to the ones of the physical service endpoints (packet-switching for L2/Ethernet or Lambda-switching for L0/DWDM optical SFP/XFP interfaces).

The creation of a virtual network is always associated with a client application or tenant. When a new virtual network is requested, first the AM creates the new abstracted topology based on the abstracted model (Node Abstraction or Link Abstraction) by requesting the creation of a new topology to the Topology Manager (TM). The Topology Manager is the component responsible for gathering the abstracted network topology from each control domain or underlying SDN orchestrators, and it builds the whole abstracted network topology which is stored in the Traffic Engineering Database (TED) and responds to the AM with the new created topology identification code (Topology Id) which will be associated to the tenant. After this initial setup phase is triggered the tenant will interacts with the MSO over the virtual network represented by the topology abstraction.

The second function of the AM is to correlate the requests based on the abstracted topology to the information of the underlying topology. To this aim, every time the MSO receives a connectivity service request (Call service request in the COP model), it request the Abstraction manager to convert the information of the service endpoints based on the abstracted network topology, into the corresponding physical ones. Once this translation occurs the OC performs a normal orchestration workflow over the physical network topology to create the necessary connections to serve the requested connectivity service. Once the connections are created, two records are stored in the Flow Server Database (Flow DB), one for the abstracted connectivity service request and one for the physical one. The correlation of this information is responsibility of the Abstraction Manager as well, and it will be used in the deletion process of the connectivity service.

The new introduced Virtual Network Service was added to COP as a new service (Figure A.4). The information model and RESTconf interface definitions are included in the Appendix A and the complete YANG files can be downloaded and contributed in the STRAUSS github repository: `https://github.com/ict-strauss/COP/blob/master/yang/yang-cop/service-virtual-network.yang`

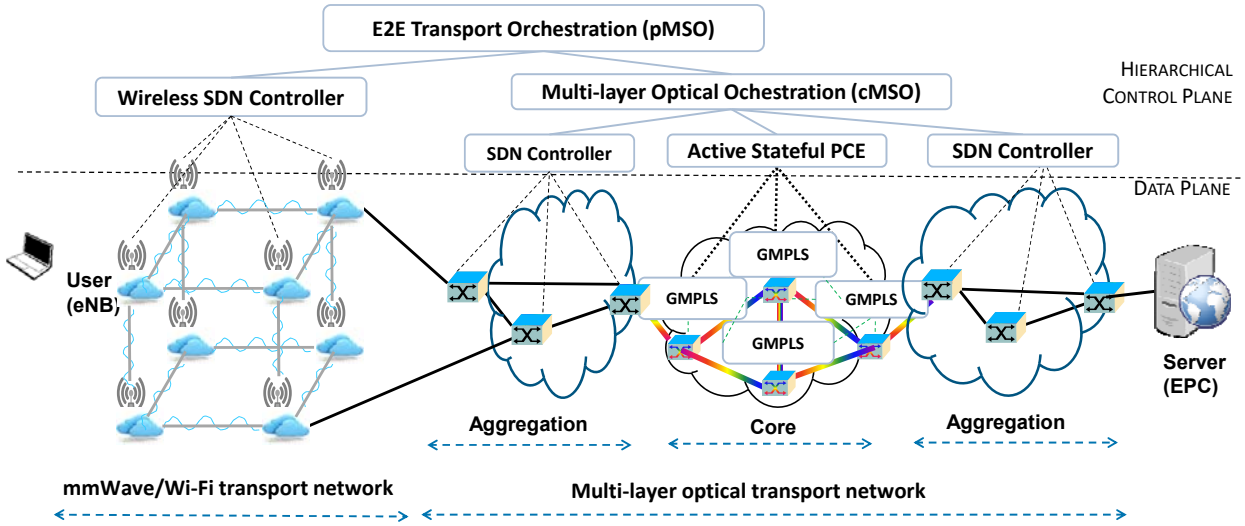


Figure 7.3: Hierarchical SDN orchestration architecture

7.3 Experimental assessment

In this section it is presented the experimental evaluation of the proposed H-ORCH for hierarchical SDN orchestration of wireless and optical transport domains, using a hierarchy of MSOs. In particular, it provides an experimental evaluation of E2E provisioning and E2E recovery procedures in a multi-domain network integrating a wireless network domain (CTTC EXTREME Testbed) and the Transport network of the ADRENALINE Testbed (Chapter 3).

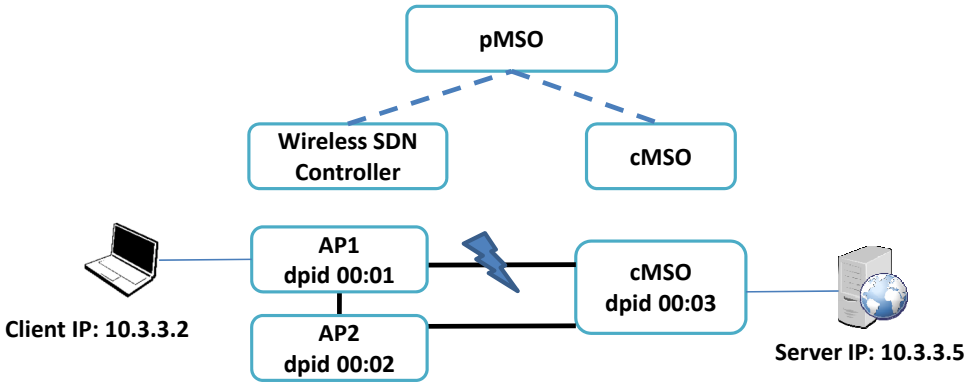


Figure 7.4: Network Topology view at the pMSO

The EXTREME SDN wireless testbed consists on 8 programmable backhaul nodes based on PC engines Intel Core i7 platform (6-Core 3.3Ghz x86 CPU and 32GB of RAM), where each node runs Ubuntu 15.04 as operating system. Currently, each node is equipped with three Compex WLE900VX IEEE 802.11ac/a/b/g/n cards, one integrated Intel IEEE 802.11ac/a/b/g/n card and two Gigabit Ethernet ports. Each node runs xDPd as software switch and the wireless SDN controller is based on Ryu. Two node are interconnected through 1 Gbps Ethernet links to two packet switches (OpenFlow-based too), providing two inter-domain links between the EXTREME and ADRENALINE testbeds

(Figure 7.3).

The pMSO implement the COP protocol for the orchestration of the cMSO and either COP or the propietary RESTful API of RYU controller for the Wireless SDN controller. The topology composition for this experimental validation leads in the topology vision displayed in Figure 7.4.

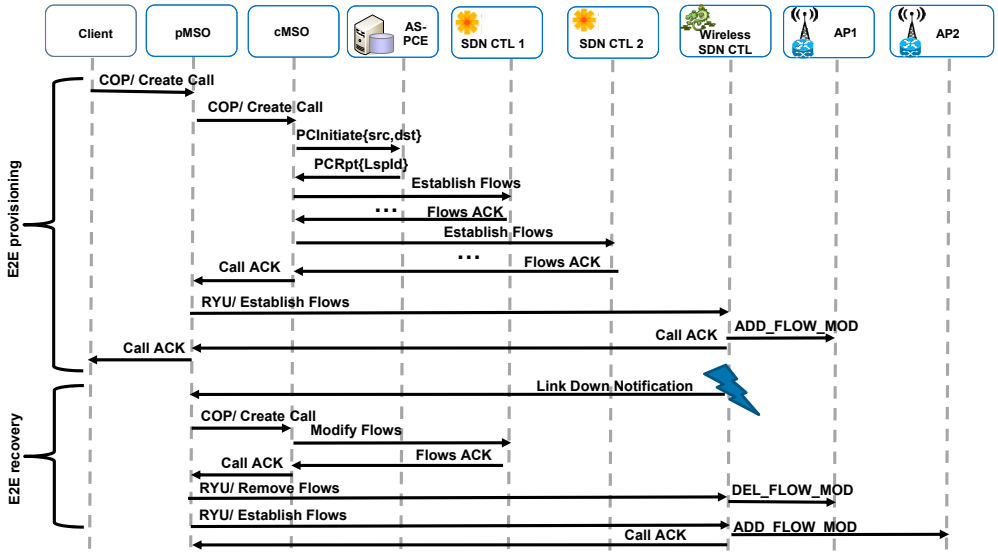


Figure 7.5: Message exchange workflow for E2E provisioning and recovery connectivity services.

Figure 7.5 shows the proposed message exchange between a pMSO and a wireless SDN controller/cMSO. Two different scenarios are presented: E2E provisioning and E2E recovery. In the first scenario, it can be observed that an E2E connection is requested (POST Call) to the pMSO. The pMSO computes the involved network controllers (Wireless SDN/cMSO) and requests the underlying connection. The request sent to the wireless SDN controller, which implements the propietary RYU REST interface, is directly translated into OF commands sent to the wireless SCs. Moreover, we can observe how the workflow follows inside a cMSO, which is responsible for another level of hierarchical SDN orchestration equivalent to the one presented in Chapter 5. The cMSO first requests an optical lightpath to the AS-PCE and, once the lightpath has been established, the different flows are established towards the underlying packet SDN controllers. In the second scenario, an inter-domain link failure is detected by the wireless SDN controller. The link failure is notified to the E2E SDN orchestrator using Websockets transport technology through a session which has been previously established between the Wireless SDN controller and the cMSO. When the cMSO receives the link down notification, it computes a new E2E path, and, following a break-before-make strategy, it first deletes or modifies the old connections and then establishes the new computed recovery connections.

Figure 7.6 shows the Wireshark captures of the exchanged messages at both the pMSO and cMSO. The bidirectional E2E service call request is received at the pMSO. The pMSO PCE is responsible for computing an E2E path. The pMSO VNTM decomposes the computed E2E path in order to request a call service to the cMSO. The cMSO is responsible for SDN orchestration towards the underlying network domains (SDN-CTL-1, AS-PCE, and SDN-CTL-2). After the cMSO provisions the requested call service, the pMSO requests the necessary flows to the wireless SDN controller. The setup delay for an E2E service call for this experiment was 3.06 seconds.

The E2E recovery message exchange is shown in Figure 7.7 The wireless SDN controller receives the OF port down message in an inter-domain link and notifies the pMSO of the link failure through

Time	Source	Destination	Protocol	Info
REF	pABNO	pABNO	HTTP	POST /restconf/config/calls/call/1 HTTP/1.1 (application/json)
0.009835	pABNO	pABNO-PCE	PCEP	PATH COMPUTATION REQUEST MESSAGE
0.017500	pABNO-PCE	pABNO	PCEP	PATH COMPUTATION REPLY MESSAGE
0.073936	pABNO	W-SDN-CTL	HTTP	POST /stats/flowentry/add HTTP/1.1 (application/json)
0.076621	W-SDN-CTL	pABNO	HTTP	HTTP/1.1 200 OK
0.104657	pABNO	cABNO	HTTP	POST /restconf/config/calls/call/00002 HTTP/1.1 (application/json)
REF	pABNO	cABNO	HTTP	POST /restconf/config/calls/call/00002 HTTP/1.1 (application/json)
0.017858	cABNO	cABNO-PCE	PCEP	Path Computation Request
0.089537	cABNO-PCE	cABNO	PCEP	Path Computation Reply
0.192055	cABNO	AS-PCE	PCEP	Initiate
0.413193	AS-PCE	cABNO	PCEP	Path Computation LSP State Report (PCRpt)
0.595911	cABNO	AS-PCE	PCEP	Initiate
0.818264	AS-PCE	cABNO	PCEP	Path Computation LSP State Report (PCRpt)
2.226984	cABNO	SDN-CTL-1	HTTP	PUT /controller/nb/v2/flowprogrammer/default/node/OF/00:00:00:1e:67
2.228431	cABNO	SDN-CTL-1	HTTP	PUT /controller/nb/v2/flowprogrammer/default/node/OF/00:00:00:1b:21
2.391473	cABNO	SDN-CTL-2	HTTP	PUT /restconf/config/opendaylight-inventory:nodes/node/openflow:116
2.398850	cABNO	SDN-CTL-2	HTTP	PUT /restconf/config/opendaylight-inventory:nodes/node/openflow:130
2.421527	cABNO	pABNO	HTTP	HTTP/1.1 200 OK (application/json)
2.526387	cABNO	pABNO	HTTP	HTTP/1.1 200 OK (application/json)
2.551317	pABNO	pABNO-PCE	PCEP	PATH COMPUTATION REQUEST MESSAGE
2.558513	pABNO-PCE	pABNO	PCEP	PATH COMPUTATION REPLY MESSAGE
2.617861	pABNO	cABNO	HTTP	POST /restconf/config/calls/call/00003 HTTP/1.1 (application/json)
3.024554	cABNO	pABNO	HTTP	HTTP/1.1 200 OK (application/json)
3.053772	pABNO	W-SDN-CTL	HTTP	POST /stats/flowentry/add HTTP/1.1 (application/json)
3.056336	W-SDN-CTL	pABNO	HTTP	HTTP/1.1 200 OK
3.065708	pABNO	pABNO	HTTP	HTTP/1.1 200 OK (application/json)

Figure 7.6: E2E provisioning wireshark captures at pMSO and cMSO.

WS: LINK Failure Notification

Time	Source	Destination	Protocol	Info
REF	W-SDN-CTL	pABNO	TCP	8080→49194 [PSH, ACK] Seq=1 Ack=1 Win=235 Len=173 TSval=259507532 TSecr=992188968
0.028197	pABNO	cABNO	HTTP	DELETE /restconf/config/calls/call/00003 HTTP/1.1 (application/json)
0.028396	pABNO	cABNO	HTTP	DELETE /restconf/config/calls/call/00002 HTTP/1.1 (application/json)
0.540708	pABNO	cABNO	HTTP	DELETE /stats/flowentry/clear/1 HTTP/1.1
0.550885	pABNO	cABNO	HTTP	POST /restconf/config/calls/call/4 HTTP/1.1 (application/json)
0.551236	pABNO	cABNO	HTTP	POST /restconf/config/calls/call/5 HTTP/1.1 (application/json)
3.094941	pABNO	W-SDN-CTL	HTTP	POST /stats/flowentry/add HTTP/1.1 (application/json)
3.101028	pABNO	W-SDN-CTL	HTTP	POST /stats/flowentry/add HTTP/1.1 (application/json)
3.107895	pABNO	W-SDN-CTL	HTTP	POST /stats/flowentry/add HTTP/1.1 (application/json)
3.113219	pABNO	W-SDN-CTL	HTTP	POST /stats/flowentry/add HTTP/1.1 (application/json)
3.495740	cABNO	pABNO	HTTP	HTTP/1.1 200 OK (application/json)

Figure 7.7: E2E recovery wireshark captures at the pMSO.

an open websocket. The pMSO computes the E2E recovery path and it first removes the previously provided connections. Then, it establishes the new connections between the wireless nodes.

Figure 7.8 shows the data plane connectivity between a User Equipment (UE) connected through a wireless SC and a server running at the aggregation network. ICMP ping messages (with a message interval of 10ms) are exchanged between both hosts. We emulate a link failure by disconnecting one of the inter-domain ports. When the E2E recovery is applied, the ICMP request messages are recovered and replied. The resulting E2E recovery time from the data plane perspective is around 3.7s.

Time	Source	Destination	Protocol	Info
REF	10.3.3.5	10.3.3.2	ICMP	Echo (ping) request id=0x2385, seq=3918/19983, ttl=64 (reply in 4009)
0.002444	10.3.3.2	10.3.3.5	ICMP	Echo (ping) reply id=0x2385, seq=3918/19983, ttl=64 (request in 4008)
0.002483	10.3.3.5	10.3.3.2	ICMP	Echo (ping) request id=0x2385, seq=3919/20239, ttl=64 (no response found!)
0.012550	10.3.3.5	10.3.3.2	ICMP	Echo (ping) request id=0x2385, seq=3920/20495, ttl=64 (no response found!)
...				
3.686449	10.3.3.5	10.3.3.2	ICMP	Echo (ping) request id=0x2385, seq=4285/48400, ttl=64 (no response found!)
3.696515	10.3.3.5	10.3.3.2	ICMP	Echo (ping) request id=0x2385, seq=4286/48656, ttl=64 (no response found!)
3.706581	10.3.3.5	10.3.3.2	ICMP	Echo (ping) request id=0x2385, seq=4287/48912, ttl=64 (reply in 4383)
3.716646	10.3.3.5	10.3.3.2	ICMP	Echo (ping) request id=0x2385, seq=4288/49168, ttl=64 (reply in 4385)
3.726712	10.3.3.5	10.3.3.2	ICMP	Echo (ping) request id=0x2385, seq=4289/49424, ttl=64 (reply in 4386)
3.736777	10.3.3.5	10.3.3.2	ICMP	Echo (ping) request id=0x2385, seq=4290/49680, ttl=64 (reply in 4388)
3.746842	10.3.3.5	10.3.3.2	ICMP	Echo (ping) request id=0x2385, seq=4291/49936, ttl=64 (reply in 4389)
3.749384	10.3.3.2	10.3.3.5	ICMP	Echo (ping) reply id=0x2385, seq=4287/48912, ttl=64 (request in 4378)

Figure 7.8: Data plane E2E recovery between UE and Server.

7.4 Performance evaluation

This section presents the results characterizing the hierarchical control plane: first study the behavior of the optical domain under the H-ORCH architecture and, after that, we present the results for the E2E service provisioning over the network scenario presented in the previous section, with two main differences. In this case four wireless SDN nodes conform the wireless network domain and secondly, the abstraction strategy assumed by the cMSO is based on the link abstraction, following the procedure detailed in section 7.2. As a result, the abstract topology exposed to the pMSO consist on four abstract nodes operating at the Layer 2 and conforming a ring topology. The complete topology observed by the pMSO is depicted in Figure 7.9.

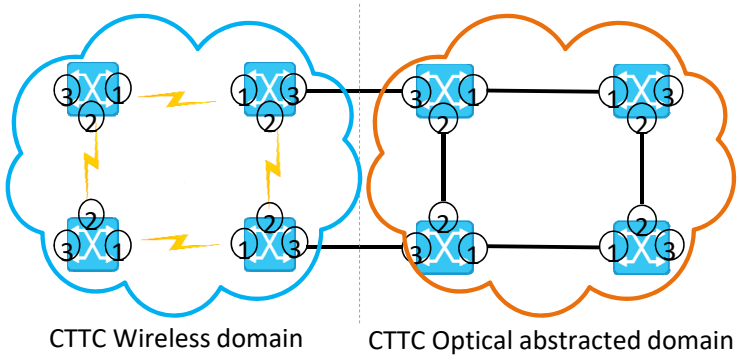


Figure 7.9: Network Topology view at the pMSO.

7.4.1 Single-domain characterization

This experiment consisted on 10000 requests arriving following a Poisson process with negative exponential holding times, fixing inter-arrival average time and increasing holding times depending on the offered traffic in Erlangs (e.g., one connection arriving on average every 3 seconds, and lasting on average 60 seconds for an offered traffic of 20 Erlangs). For the evaluation, we consider that requests are being uniformly distributed among the four optical nodes (optical cross connects, or OXCs) within the network scenario. Each link is characterized for having 8 bidirectional wavelengths supporting 10Gbps client rates. Requests go from one OXC to a destination OXC and do not involve client

transceivers due to hardware limitations that would constraint the results, since only a limited set of client transceivers are available, well below the theoretical maximum supported by the optical network.

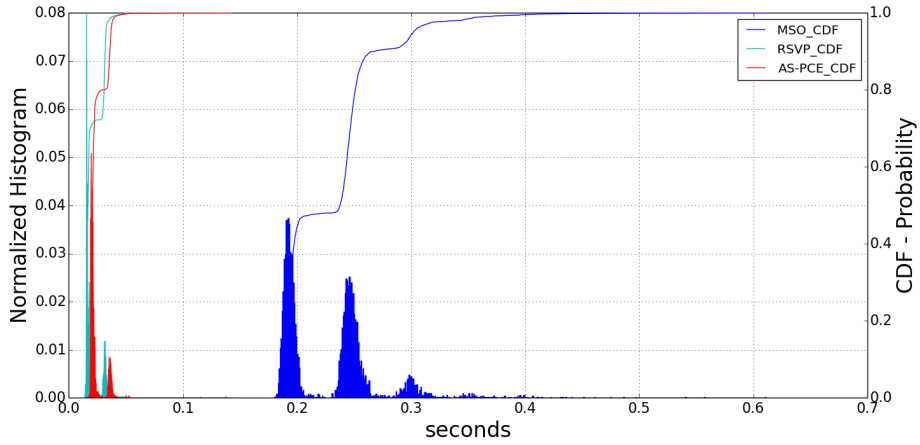


Figure 7.10: Setup delay histogram and CDF from the GMPLS/RSVP-TE controllers, from the AS-PCE, and from the cMSO.

The setup delay of the optical services' provisioning has been measured from multiple functional entities (i.e.,from the cMSO, from AS-PCE, from GMPLS ingress controller). In particular an histogram and the CDF of the different entities it is shown in Figure 7.10. Notice that since the aim of this section is to analyze the behavior of the system in the optical domain, just cMSO is taken into account. Path setup time is affected by the hardware configuration delay, which involves configuring the OXC and it is vendor dependent. We have performed different measurements with and without hardware configuration.

Without optical hardware configuration. To exclusively characterize the control plane behavior, we configure the hardware optical nodes in emulation mode, hence requiring a negligible configuration time. Figure 7.10 depicts the histogram and CDF of the setup delay seen at different reference points:

- From the GMPLS control plane: this means the setup delay considering the signaling process, from the RSVP-TE connection controller of the ingress node to that of the egress node, and roughly corresponding to the RTT of the signaling messages with forward Path and reverse Resv messages across the different transit nodes. We see that on average, this shows two peaks, roughly corresponding to whether 2 or 3 optical nodes have been involved in the provisioning.
- From the AS-PCE: the AS-PCE adds a small component to the setup delay, associated to the processing of requests from the child MSO and dispatching of requests to the corresponding head-end node.
- Finally, we see the setup time from child MSO, which adds an additional time due to the COP protocol and the use of text-based REST interfaces. Average setup delay seen from MSO is 250ms.

With optical hardware configuration. In this case, an additional set of 100 requests has been performed. The main finding is that the hardware configuration adds an additional 200ms to the provisioning time. As a summarizing guideline, the setup delay for the optical domain, seen from the cMSO, can be upper bounded to 500ms (sub-second provisioning delay in our considered scenario).

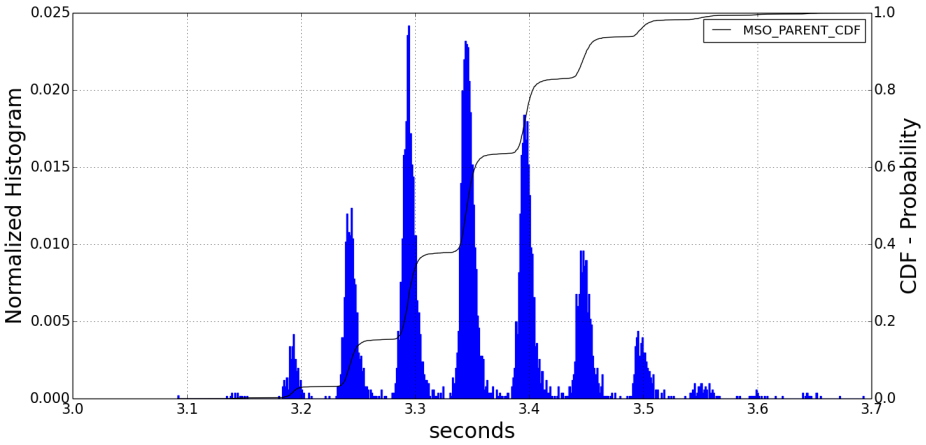


Figure 7.11: Setup delay and histogram at the parent MSO.

7.4.2 Multi-domain characterization

This experiment consisted of 10000 E2E service requests to the pMSO between randomly selected Service Endpoints of the abstract topology shown in Figure 7.9. Each service (COP Call request) is created in the network is removed before the next service request arrives, thus the blocking probability is theoretically zero and all requests have uniform network conditions.

The histogram and CDF of the E2E multi-domain, multi-layer services' setup delay as seen from the pMSO are presented in Figure 7.11. The multiple peaks in the histogram reflect an implementation artifact of the MSO (child and parent) due to the synchronous nature of the Python *threading* library's. Specifically the Event object contains an idle period of 50ms between every check of the internal flag which let the execution of the program continue if the operation has been concluded.

However, the most remarkable observation is the increase of the average setup delay from tens or hundreds of milliseconds in the single domain characterization to several seconds when the hierarchy of MSO is included. This increase in the setup delay is introduced by two main reasons: (1) the network abstraction and the communication between the parent and child MSOs introduce additional processing for message processing and data translation; (2) the multi-domain and multi-layer path computation occurring in within the cMSO controlled domains for the E2E service provisioning [1].

7.5 Conclusions

This chapter has proposed the hierarchical SDN Orchestration (H-ORCH) in order to handle SDN network orchestration scalability and security. The H-ORHC architecture has been proposed and experimentally validated for E2E provisioning and recovery use cases in a multi-domain optical/wireless network integrating EXTREME wireless and ADRENALINE optical testbeds. Experimental results suggest that certain levels of hierarchy can cope with the upcoming network heterogeneity from a multi-(technology, domain, and vendor) perspective.

[1]Note that the measurements of the setup delay included in Chapter 5 corresponds to the same operations done in this experiment by the child MSO and they were obtained in the same network scenario (Chapter 3).

Chapter 8

The Peer SDN Orchestration (P-ORCH) approach

Within the transport SDN community, it is commonly accepted that deploying a single, integrated controller for a large or complex network may present scalability issues, or may not be doable in practice. Two main reason are: a) Network size, in terms of controllable elements, which has a direct impact on the controller requirements (e.g. active and persistent TCP connections on top of which control sessions are established, memory requirements to store in memory e.g. a data structure representing the network graph that abstracts the network and CPU requirements for processing message exchange); and b) Security and robustness; a single SDN controller represents a single point of failure in the control architecture, thus network operators seek for most robust architectures which provide redundancy in the control plane layer.

To address such shortcomings, it is important to consider the deployment of multiple controllers, arranged in a specific setting, along with inter-controller protocols. Such network architectures apply both to heterogeneous and homogeneous control (different or same control plane and data plane technologies within the domain of responsibility of a given controller). Two approaches to controller interconnection are identified, which depend on the directivity of the interconnection model: hierarchical and peer.

In the previous chapter, the hierarchical orchestration model was presented, where a the orchestration entities are ranged in a topology which is, typically a tree, with a given root being the top-most controller. For a given hierarchy level, a parent SDN orchestrator handles the automation and it has a certain number of high level functions, while low level controllers (usually referred to as children) cover low-level, detail functions and operations.

In this chapter, we introduce the Peer Orchestration (P-ORCH) model where two orchestration entities manage two independent network administrative domains and cooperate to provide End-to-End (E2E) connectivity services which span across multiple domains. For the WEST-EAST interface

between the orchestrators, an extended version of the COP protocol is proposed to provide network topology abstraction, path computation and connectivity provisioning. The chapter is arrange as follows: in section 8.1 the proposed peer orchestration architecture is presented, including the orchestration workflow based on the recursion pattern and the COP extensions introduced to address the realization of the proposed model. In section 8.2, the P-ORCH architecture is validated in an international Europe/Japan multi-partner network scenario, involving several administrative domains with a per-domain dedicated SDN controller instance and a P-ORCH architecture on top consisting on two MSO components.

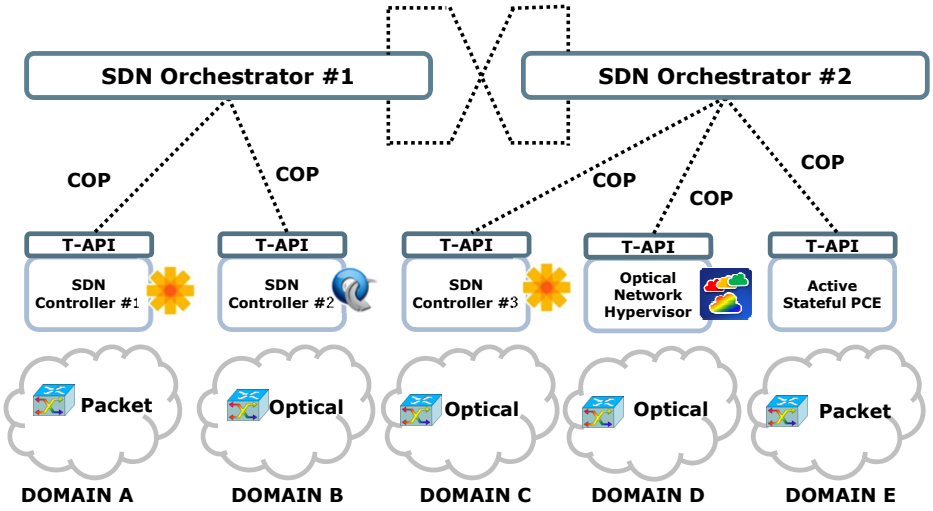


Figure 8.1: Proposed Peer Orchestration architecture network architecture

8.1 Peer Orchestration architecture

In the proposed architecture, Figure 8.1, each provider network is controlled though an SDN orchestrator (parent SDN controller, pSDN), which handles several child SDN controllers (cSDN). Each cSDN is responsible for a single network segment. A recursive hierarchy could be based on technological, vendor, SDN controller type, geographical domains or network segment basis. The COP, presented in Chapter 6, was demonstrated in Chapter 7 as a viable protocol for recursive hierarchical interconnection in between pSDN and cSDN entities. In this chapter, we propose to extend the COP to allow a pSDN from Network Provider A (SDN-O 1) being able to interact with its peer of Network Provider B (SDN-O 2).

Peer interconnection model corresponds to a set of controllers, interconnected in an arbitrary mesh, which cooperate to provision end-to-end services. In this setting, it is often assumed that the mesh is implicit by the actual (sub)domains connectivity; the controllers hide the internal control technology and synchronize state using East/West interfaces. The SDN controllers manage detailed information of their own, local topology and connection databases, as well as abstracted views of the external domains and the East/West interfaces should support functions such as network topology abstraction, path computation and connectivity provisioning.

Neighbor recursion has been proposed in [70] as the pattern in which SDN controllers peer to deliver services across multiple SDN control domains. All participants would be expected to expose

comparable levels of abstraction and services. Using neighbor recursion, any SDN controller can act as either client or server to its neighbors, depending on the requested service endpoints. It can be noted, that requested peer services will be understood in a call model, including service creation, service usage, and service release. Peer SDN orchestrators might use neighbor recursion to provision E2E services, such as DC interconnection. In all cases, the service endpoints must be coordinated and therefore recursively visible, while the internal details of the network are typically abstracted, and left for the immediate controller.

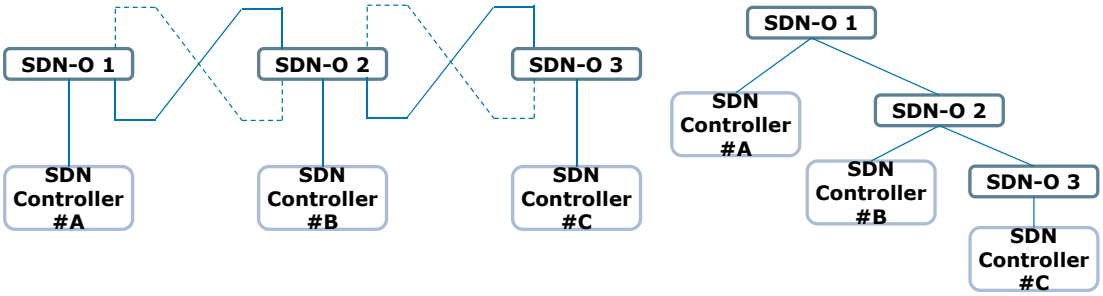


Figure 8.2: Neighbor recursion pattern.

Figure 8.2 shows an example for neighbor recursion. Figure 8.2 (left) shows three SDN orchestrators (pSDN), each responsible for a cSDN. For an E2E connectivity request starting in a service endpoint handled by SDN-O 1 and ending at service endpoint handled by SDN-O 3, Figure 8.2 (right) shows the neighbor recursion pattern, which results in a balanced hierarchy of SDN-Os. The proposed neighbor recursion pattern does not detail how inter-domain topology is obtained, as a mechanism to avoid topological loops shall be implemented. It is assumed that service end-point reachability is known.

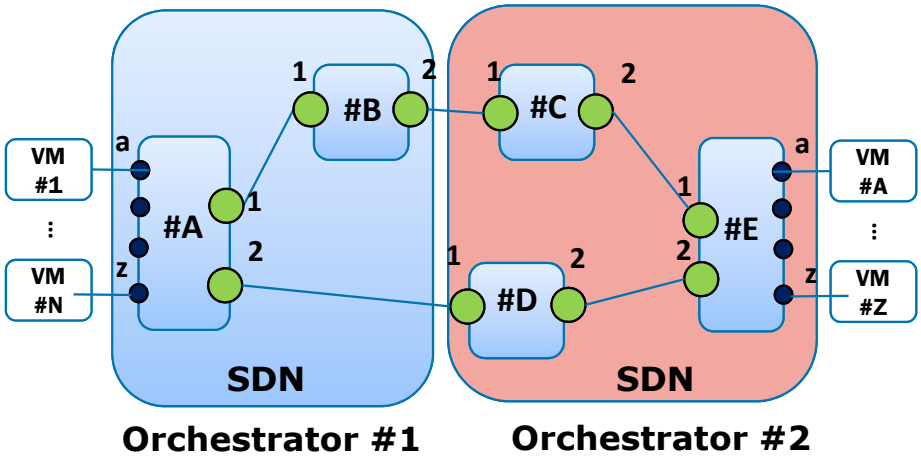


Figure 8.3: Topological views from SDN-OEU and SDN-O-JP.

Figure 8.3 provides the different overall topological views from the proposed scenario in Figure 8.1. The provided topological corresponds to the real scenario proposed for experimental validation

in section 8.2. SDN-O 1 is responsible for SDN controllers A and B; and SDN-O 2 is responsible for SDN controllers C, D, and E. Two inter-domain links are provided between #A:2-#D:1 and #B:2-#C:1. Moreover, COP has been extended in order to offer the context for a client, which includes the abstracted topology and the available service endpoints. In the proposed scenario, each SDN-O announces as service endpoints the different network endpoints on which virtual machines might be interconnected (SDN-O-JP service endpoints: az; SDN-O-EU service endpoints: A-Z).

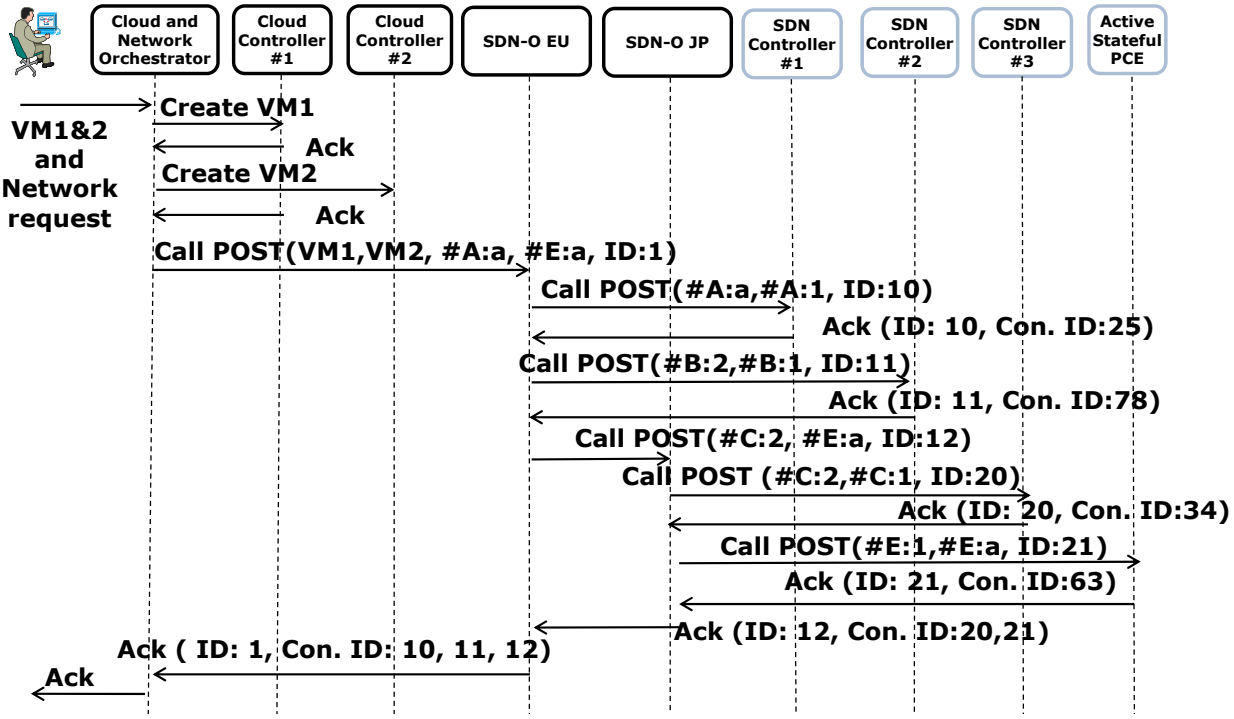


Figure 8.4: Message workflow for VM and Connectivity Service creation

Figure 8.4 shows the proposed workflow between a cloud and network orchestrator which requests two virtual machines (VM) (step 1) and requests their interconnection through a dedicated E2E connectivity service (step 2). The E2E connectivity request is sent to the SDN orchestrator responsible for the source service endpoint (SDN-O 1). The intra-domain connectivity is provisioned through SDN controller #A (step 3) and #B (step 4). Through neighbour recursion, the necessary connectivity is requested to SDN-O-EU (step 5). SDN-O 2 is responsible for the provisioning of the remaining intra-domain connectivity though SDN controllers #C (step 6) and #E (step 7). Once the necessary connections have been established the Cloud and Network Orchestrator is notified.

8.2 Experimental Assessment and evaluation

The experimental setup is shown in Figure 8.5, where Bristol, CTTC, and ADVA domains are controlled by an SDN orchestrator (SDN-O-EU), which is run by Telefónica, based on the ABNO architecture [1]. The controllers hide the internal setup of each domain. The MSO presented in 5 implementing the SDN-O-JP, is responsible for handling multiple technology SDN controllers from KDDI. Each SDN controller provides through COP the abstracted topology as a node. The multiple SDN orchestrators and controllers are interconnected through an OpenVPN over the public internet offering a control plane Testbed.

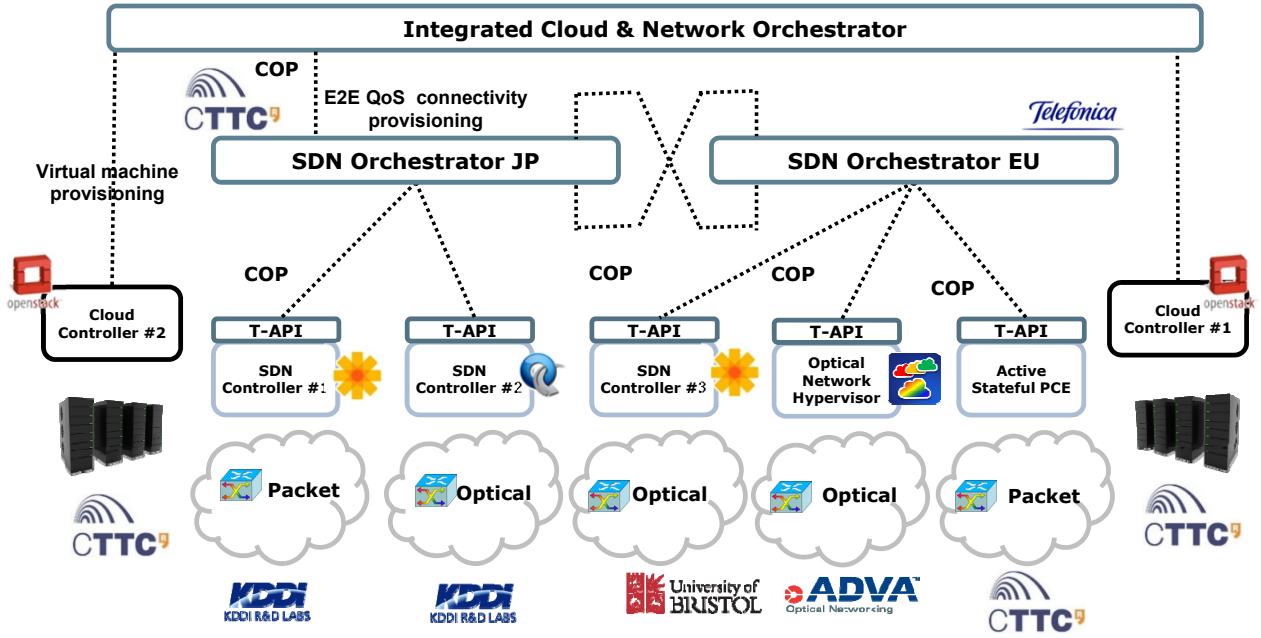


Figure 8.5: International Europe/Japan multi-partner network scenario for P-ORCH architecture demonstration using COP as unified orchestration interface.

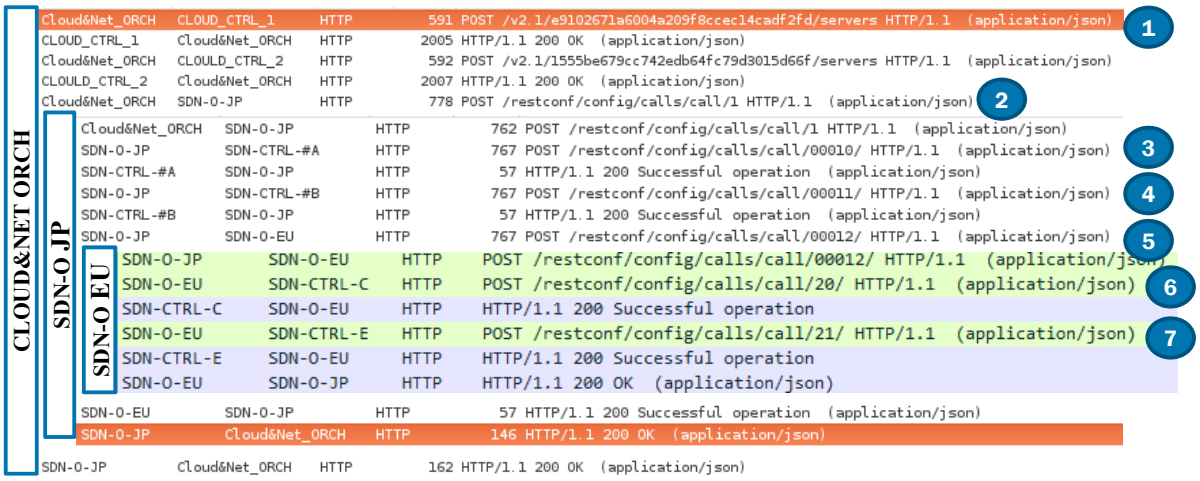


Figure 8.6: Wireshark captures from three viewpoints: Cloud and network orchestrator, SDN-O-JP, and SDN-O-EU

Figure 8.6 shows the messages exchanged from three different perspectives: a) the captured from the Cloud and Network Orchestrator; b) from the SDN-O-JP; and c) from the SDN-O-EU.

The objective of the experimental validation is to create two VMs in different DC, and provide a connectivity service between them. In step 1, two VMs are requested to each respective cloud controller. Later, in step 2 a connectivity service is requested to SDN-O-JP. This results in a HTTP POST command to create a Call object, which includes the necessary connection endpoints, as well as

the requested QoS, including bandwidth details. The SDN-O-JP processes the request, and triggers the connectivity service through SDN controllers A and B (steps 3 and 4).

In order to establish the requested service, SDN-O-JP requests the connectivity service to the peer SDN-O-EU (step 5). This service's request includes as a source endpoint the interdomain endpoint #C:1 and as destination endpoint the destination endpoint included in the E2E call. Once this request is processed in SDN-O-EU, a connectivity service is requested through the SDN controllers C and E (steps 6 and 7).

Finally, the requested E2E connectivity service has been provisioned in our setup with an average time of 1.6s; while the VM creation on each cloud controller is in the order of 40s.

8.3 Conclusions

In this Chapter it has been proposed a novel peer SDN Orchestration in order to handle SDN network orchestration in multiple administrative domains. Neighbour recursion and extensions to Control Orchestration Protocol have been proposed and experimentally validated in a control plane international testbed.

Part III

Integrated Orchestration of Cloud and Transport Network services

Chapter 9

Integrated IT and SDN Orchestration across geographically distributed Datacenters.

As it was introduced in chapter 1, one of the main objectives of network operators in the last years is to reduce the volume of their operations and capital expenditures (OpEx, CapEx) in order to face the stagnation of their revenues. In this context, Software Defined Networking (SDN) and Network Function Virtualization (NFV) have emerged as the key-enabler technologies to achieve more flexibility and automation in their network operations and an important reduction in the costs associated to the investment in new equipment.

In the previous chapters we already showed the benefits of SDN in the transport network segment, now we introduce NFV as a driven technology which benefits from SDN but also seeks for a closer relationship between the network and cloud management systems. A network service can be defined as a succession or chain of Network Functions (NFs) performed by physical or virtual appliances, which together conform a customized network function. In the NFV paradigm, these NFs are expected to be deployed in virtualized environments (VNFs) being instantiated in one or multiple NFV Infrastructure Points of Presence (NFVI-PoPs). The network service also includes the network connectivity services with Quality of Service (QoS) requirements (e.g., dedicated bandwidth, maximum supported latency) required to interconnect the VNFs.

In this chapter we present an integrated IT and network orchestration architecture for the joint deployment of Cloud and Transport network services across multiple Data Centers (DCs). The proposed

architecture targets the orchestration of multiple Cloud management systems which provide virtualized computing and storage services in geographically distributed DCs, together with the deployment of the required intra- and inter-DC connectivity services. The design principles for the architecture proposed in this chapter follow the SDN orchestration approach described in previous chapters, which has been presented and successfully validated in multiple scenarios, but now is extended to account for the requirements of virtualized computing and storage instances.

The chapter is organized as follows: in section 9.1 we introduce the scenario and the problem statement. In 9.2, we present the integrated SDN IT and Network orchestration architecture by explaining in detail its main building blocks and design principles. Section 9.3 includes the experimental validation of the proposed architecture. The assessment consist on two different use cases: (i) DC interconnection across a multi-domain, multi-layer network; (ii) Seamless Virtual Machine migration between geographically distributed DCs. Finally, section 9.4 outlines the main contributions of this chapter.

9.1 Distributed DC interconnection

Virtualization of compute, storage and networking resources in DCs is provided by private clouds through distributed cloud orchestrators that may be deployed over different Virtual Infrastructure Manager (VIM) software distributions (e.g. OpenStack, OpenNebula). Each VIM enables to segregate the DC into availability zones for different tenants and instantiate the creation/ migration/ deletion of Virtual Machine (VM) instances (computing service), storage of disk images (image service), and the management of the VMs network interfaces and the intra- DC network connectivity (networking service). The target solution should account for SDN-based network control of both, intra- and inter-DC networks, alongside with the aforementioned management of the virtualized computing infrastructure across geographically distributed DCs.

The intra-DC network denotes here the L2 network connecting the physical servers which conform the virtualized infrastructure for the deployment of VMs and also the software based virtual switches running inside the servers to which the VMs are attached. Each intra-DC network is controlled by a SDN controller instance which is responsible of discovering the network elements and hosts (IP and MAC addresses), including the topological associations between hosts and network ports. The target solution must support different types of VM network deployments, FLAT networking, VLAN or virtualized network types based on overlay technologies such VXLAN [71] or GRE[72], the detail description of these technologies is out of the scope of this Thesis. The target solution should be able to correlate information from both domains (Cloud and Transport network) to characterize the connectivity services through the matching forwarding rules installed in the network elements. As it was detailed in Chapters 4 and 6, both the proposed Control Orchestration Protocol (COP) and OpenFlow protocol, allow the definition of flow's matching rules as a combination of different packet headers (MAC, IP, TCP/UDP, VLAN-tags, MPLS) this flexibility will allow the Integrated Cloud and Network orchestration solution to define the connectivity services upon the different VM's network characteristics.

On the other hand, the inter-DC network control is also included in the scope of the proposed solution. As it was detailed in previous chapters, a hierarchical SDN orchestration architecture arranged upon the network scenario (e.g., depending on the number of domains/technologies the network consist of), is a suitable solution to guarantee end-to-end network transport services. This inter-DC network can be exposed explicitly, with full network topology visibility, or abstracted, to improve scalability and security of the overall solution,as detailed in Chapter 7. One of the requirements introduced by the NFV paradigm for the Transport Network infrastructure is the capability to slice or virtualize the

physical infrastructure to provide overlay networks to multiple tenants or Virtual Tenant Networks (VTNs). In chapter 11 this feature will be evaluated in detail but in brief, the proposed solution allows the creation of Virtual Private LAN Services (VPLSs) over MPLS between different NFVI/DC PoPs.

In brief, the objective is the design of an scalable orchestration architecture for both IT (Cloud) and network resources within and between geographically distributed DCs.

9.2 Integrated IT and Network Orchestration architecture

The solution proposed is a novel integrated IT and Network orchestration architecture (Figure 9.1), aligned with the NFV architectural framework proposed by the ETSI [73], which allows the control and management of the physical or virtual compute, storage and network (including the transport network) resources which in the referenced architecture is noted as the NFVI. Our proposal allows the coordination of the cloud management systems and the control systems of the transport network in order to provide the aforementioned required environment for the deployment of NFV-based network services. The architecture includes a hierarchical SDN orchestration architecture which accounts for multiple network scenarios and a distributed cloud computing infrastructure arranged in multiple DC locations.

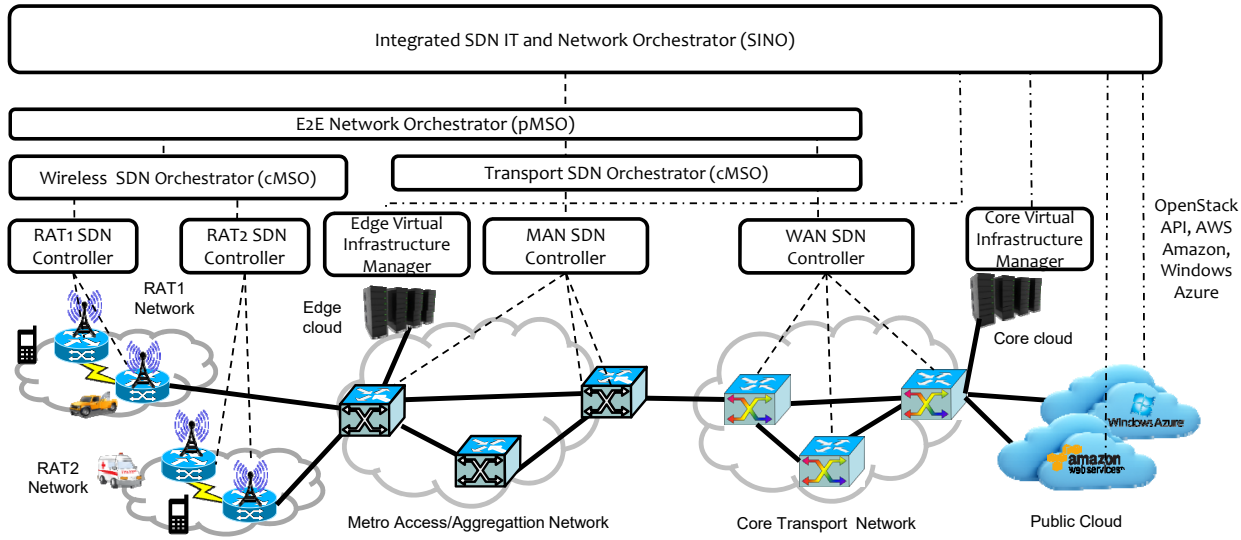


Figure 9.1: Integrated SDN IT and Network orchestration (SINO) architecture.

On the top of the architecture, the integrated SDN IT and Network Orchestrator (SINO) targets the global management of the virtual compute, storage and network resources provided by the E2E Network Orchestrator (MSO) and the distributed VIMs. It acts as a unified cloud and network operating system providing, dynamic and global provision, migration and deletion of VMs and the required connectivity between the distributed VIMs across the corresponding multi-layer, multi-domain transport network. A key enabler of such an integration is the COP (see Chapter 6), which is proposed interface between the SINO and the E2E Network Orchestrator, providing a common control primitives for seamless control of the transport network.

The SINO's northbound interface is based on HTTP RESTful to provide CRUD operations for VMs, L2/L3 networks, VTNs, Images, Connectivity Services (COP Calls) and a RPC based VM migration service (section 9.3.2). The VM CRUD mechanism allows, via a REST API, to create, read, update or delete a VM. A VM might be requested based on its availability zone, its hardware

resources (i.e., flavor), or the disk image to be loaded. A VM must be associated to one or more already created L2 networks which triggers the creation of a virtual network interface card (vNIC) in the VM which is attached to the virtualized network switch of the allocated compute node (Hypervisor). The L2 network CRUD mechanism allows to create, read, update or delete a L2 networks, which allow to specify the network type (FLAT, VLAN, overlay...).The NBI also includes a L3 CRUD to create IPv4, IPv6 subnets, from which an IP address is assigned to a VM virtual network interface card (vNIC).

The Connectivity Service CRUD permits the management of connectivity services through the underlying MSO. The SINO allows to define VMs as the connectivity endpoints and it is responsible of translating the VM-based request into a network context connectivity service between network endpoints (node interfaces). This information is obtained directly from the intra-DC SDN controllers through address tracking services based on the incoming packets or by correlating the Cloud Controller network ports information (VM's network attachment points) with the network topology discovered from the SDN controllers by the MSO.

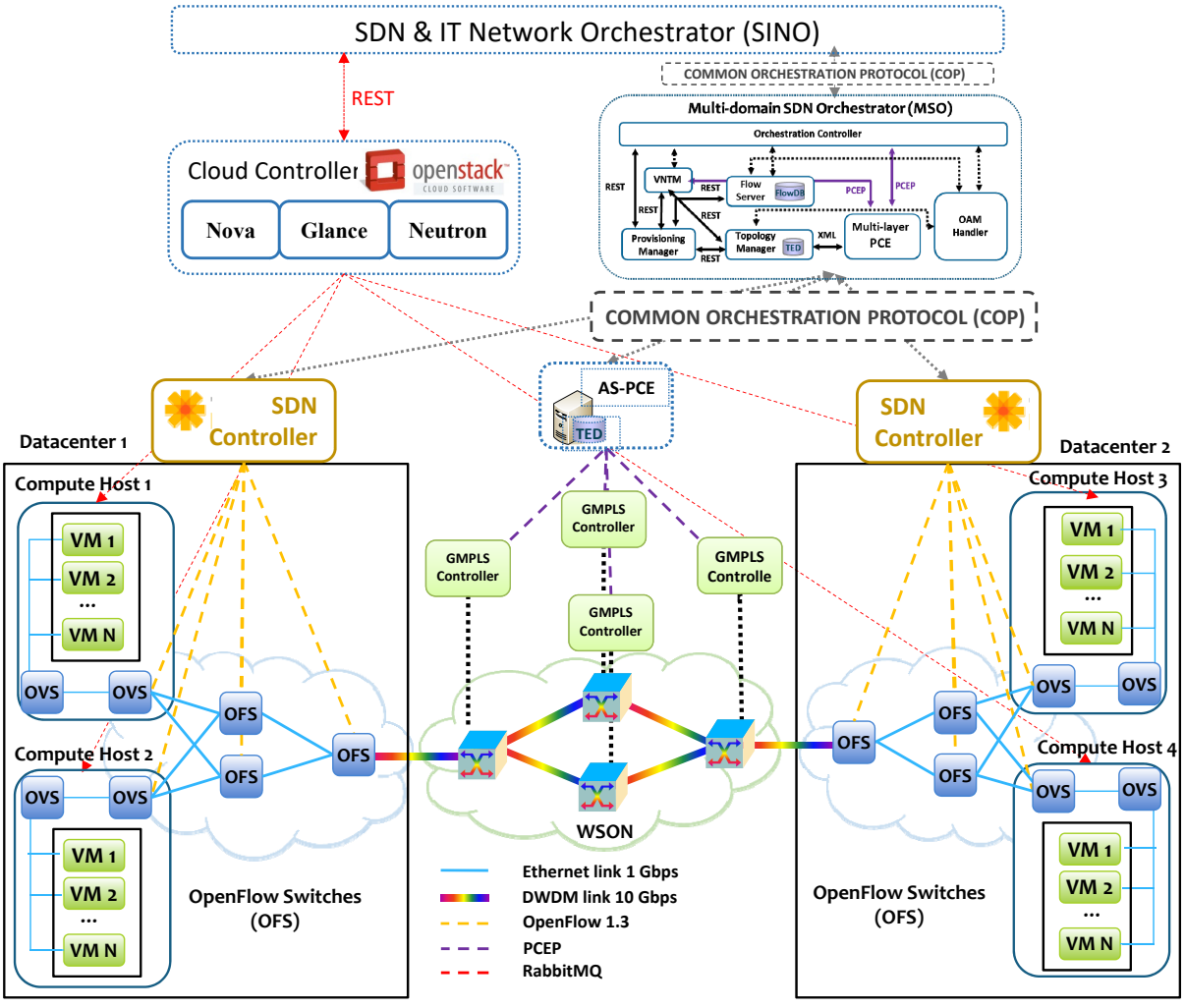


Figure 9.2: Experimental DC interconnection scenario.

9.3 Experimental validation

This section includes the experimental validation of the proposed architecture which cover the VM creation and migration operations in a geographically disperse multi-DC scenario. These experiments are based on FLAT network type for the VM interconnection, where the VM's connectivity services are provided end-to-end, i.e, from the first virtual switch attachment point of source VM to the virtual switch in the target host where the destination VM is instantiated.

9.3.1 Use case I: DC interconnection across a multi-domain, multi-layer network.

The proposed architecture has been validated in the Cloud Computing platform and Transport network of the ADRENALINE Testbed (Chapter 3). The COP has been employed as a transport API for the orchestration of: two SDN OpenDaylight Helium controllers responsible of controlling the Ethernet intra-DC domains via OpenFlow 1.3; and the optical transport network via an AS-PCE with instantiation capabilities as a single interfacing point for the GMPLS control plane. In the experimental validation, we have introduced COP agents on top of SDN controllers in order to translate the received COP commands to SDN controllers NBI. Figure 9.2 shows the multi-domain network scenario considered for this experiment, where two geographically distributed DCs are interconnected through the WSON.

Figure 9.3 presents the integrated IT/SDN orchestration workflow for the on-demand deployment of two VMs in the cloud (one on each DC location) and the E2E connectivity provisioning across the proposed scenario. The orchestration process consists of two different steps: the VM creation and network connectivity provisioning. Firstly, the SINO requests the creation of virtual instances (VMs) to the VIM implemented with OpenStack Liberty release, which, is responsible for the creation of the instances. The VIM is also responsible to attach the VMs to the virtual switch inside the corresponding compute node. When the VMs creation has finished, the VIM replies to the SINO with the VMs networking details (MAC address, IP address and physical computing node location) which will be used as a match filter within the E2E connectivity service provisioning.

For the network orchestration the interface between the SINO and the MSO is based on the COP and also between the MSO and the per-domain controllers. For this experimental validation, a bidirectional CALL_SERVICE is requested by the SINO to provide an E2E connectivity to the previously deployed VMs. The MSO firstly requests the creation of a virtual link in the upper layer topology (L2) which is translated internally by the VNTM MSO module into two unidirectional L0 CALL_SERVICEs sent to the AS-PCE through the Provisioning Manager. They trigger, in the AS-PCE, the creation of the corresponding GMPLS connections (Label Switched Paths (LSPs)). Afterwards the provisioning of the E2E service in the upper layer is requested to the SDN controllers, by two new unidirectional CALL_SERVICESs to each domain.

The traffic capture showed in Figure 9.4 validates the use of the COP as common interface for the configuration of the transport network forwarding plane. Firstly, we can observe the request for Virtual Machine (VM) creation from the SINO towards the Cloud Controller (which is running on the same server). The creation time for a single VM is of 15 seconds, which include the necessary time to boot up the VM. Secondly, in Figure 9.5 we can observe the JSON encoded body of the Call request (Call Id: 1) sent from the SINO to the multi-domain SDN orchestrator which includes several traffic parameters (such as requested bandwidth), the requested transport layer and the MAC addresses of the interconnected VMs. The MSO computes the necessary domain Call requests and sends them towards the AS-PCE for the optical domain (Call Id: 00002, 00005), the SDN Controller 1 (Call Id: 00001, 00006) and the SDN Controller 2 (Call Id: 00003, 00004). The multi-domain call service set-up delay is of 2.52 seconds.

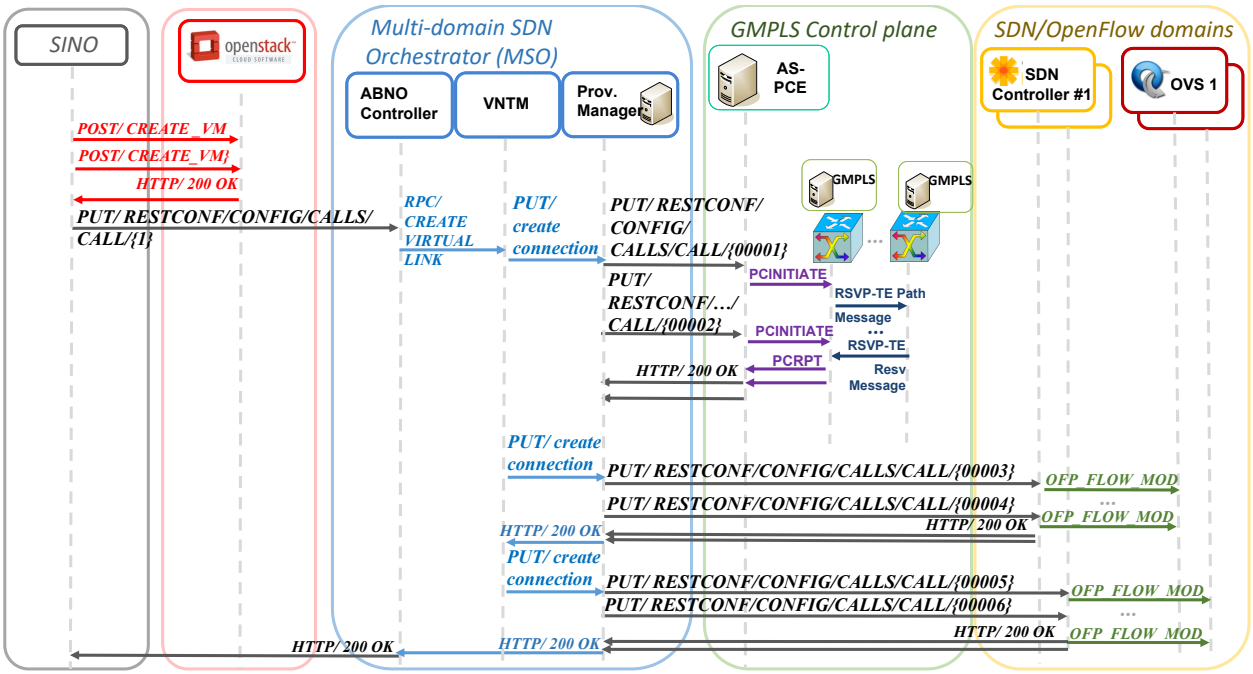


Figure 9.3: Integrated SDN/IT Orchestration workflow.

Time	Source	Destination	Protocol	Info
REF	SINO-ABNO	SINO-ABNO	HTTP	POST /create_vm HTTP/1.1 (application/json)
16.178267	SINO-ABNO	SINO-ABNO	HTTP	HTTP/1.1 200 OK (text/html)
16.181848	SINO-ABNO	SINO-ABNO	HTTP	POST /create_vm HTTP/1.1 (application/json)
30.914099	SINO-ABNO	SINO-ABNO	HTTP	HTTP/1.1 200 OK (text/html)
REF	SINO-ABNO	SINO-ABNO	HTTP	POST /restconf/config/calls/call/1 HTTP/1.1 (ap
0.123129	SINO-ABNO	SDN-CTL-1	HTTP	POST /restconf/config/calls/call/00001 HTTP/1.1
0.287926	SDN-CTL-1	SINO-ABNO	HTTP	HTTP/1.1 200 OK (application/json)
0.329396	SINO-ABNO	AS-PCE	HTTP	POST /restconf/config/calls/call/00002 HTTP/1.1
1.439163	AS-PCE	SINO-ABNO	HTTP	HTTP/1.1 200 OK (application/json)
1.493375	SINO-ABNO	SDN-CTL-2	HTTP	POST /restconf/config/calls/call/00003 HTTP/1.1
1.527963	SDN-CTL-2	SINO-ABNO	HTTP	HTTP/1.1 200 OK (application/json)
1.628074	SINO-ABNO	SDN-CTL-2	HTTP	POST /restconf/config/calls/call/00004 HTTP/1.1
1.660056	SDN-CTL-2	SINO-ABNO	HTTP	HTTP/1.1 200 OK (application/json)
1.699451	SINO-ABNO	AS-PCE	HTTP	POST /restconf/config/calls/call/00005 HTTP/1.1
2.308023	AS-PCE	SINO-ABNO	HTTP	HTTP/1.1 200 OK (application/json)
2.358390	SINO-ABNO	SDN-CTL-1	HTTP	POST /restconf/config/calls/call/00006 HTTP/1.1
2.502622	SDN-CTL-1	SINO-ABNO	HTTP	HTTP/1.1 200 OK (application/json)
2.519650	SINO-ABNO	SINO-ABNO	HTTP	HTTP/1.1 200 OK (application/json)

Figure 9.4: Control traffic capture - IT and Network orchestration workflow based on COP.

9.3.2 Use case II: Seamless Virtual Machine migration between geographically distributed datacenters

The second experiment on which we are going to base our evaluation of the proposed architecture consist on the seamless VM migration between two DCs across a multi-layer, multi-domain transport network. VMs do not run isolated, typically a VM is connected with other VMs in the same network to offer a joint service. If one of the VMs from a network is migrated, its connection state must be maintained, which is known as a VM seamless migration [74]. The migration of a virtual instance

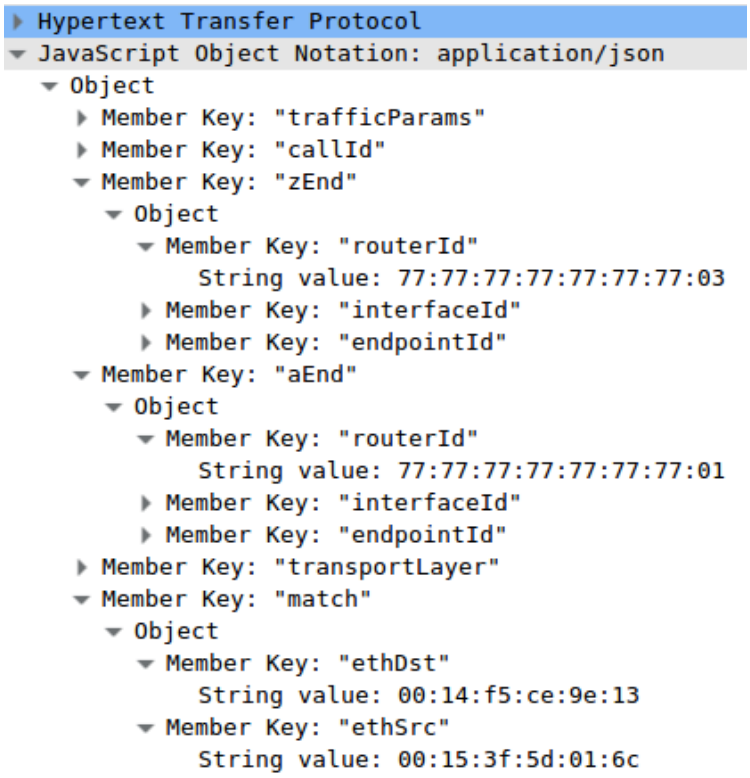


Figure 9.5: Control traffic capture - COP Call request detail.

must be transparent for all the hosts connected to the virtual machine migrated (i.e., after a disruption time the connectivity between them will be recovered).

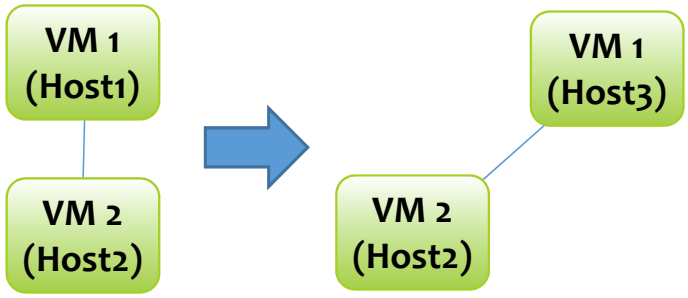


Figure 9.6: VM Migration scenario.

In the proposed scenario (Figure 9.2) the VM1 is connected to the VM2 both are running into the DC-1. We validate that the proposed architecture is able to provide seamless migration of VM1 to DC-2. The experiment assesses a block migration using the inter-DC network (Figure 9.6) and measures the disruption time in which VM2 cannot access to the VM1. The process involves six steps, detailed in Figure 9.7.

When the SINO receives a VM1 migration request through HTTP request, firstly all the flows established in the network in which the target VM is involved must be removed before start the migration. The MSO looks into the Flow Server database for all flows involving VM1, the provisioning

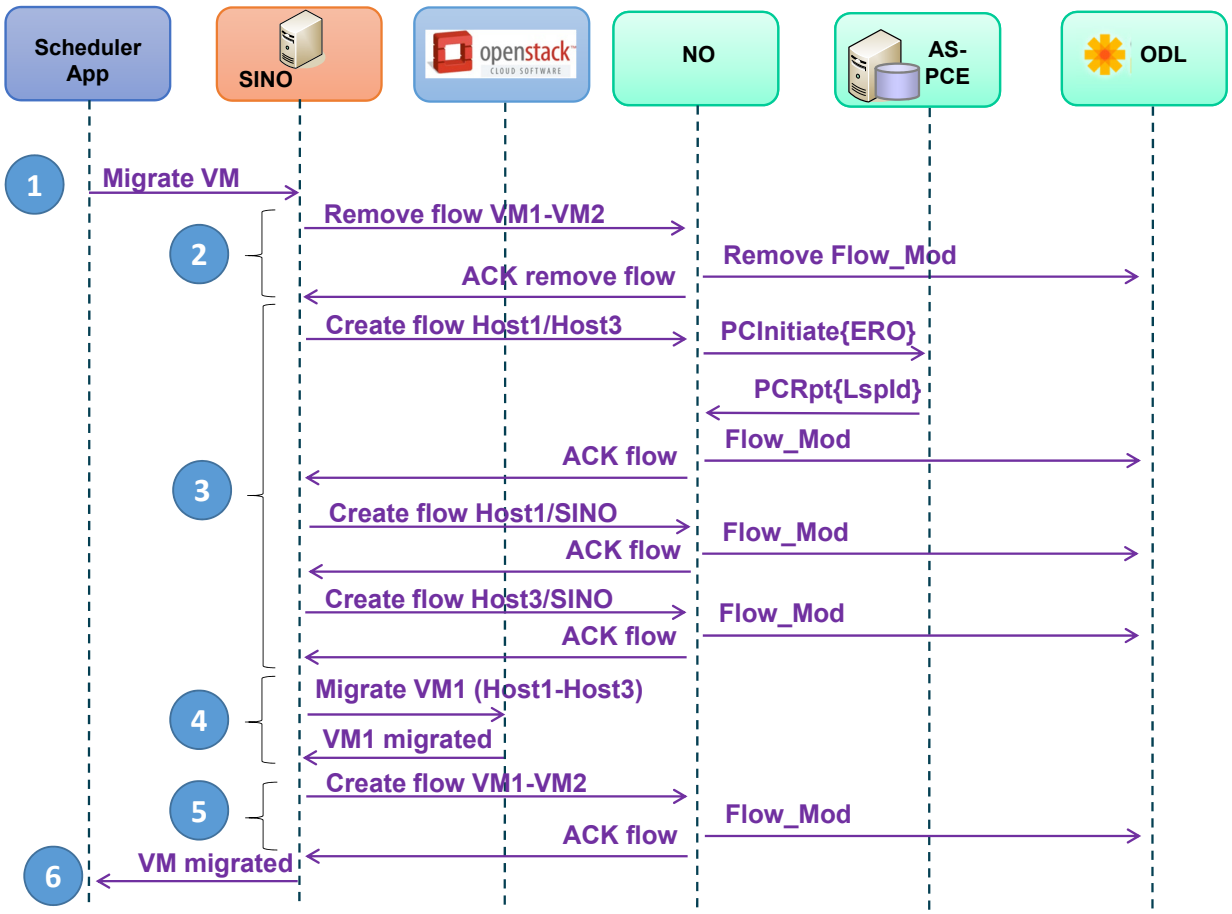


Figure 9.7: VM migration flow diagram.

manager deletes all of them by sending the related HTTP requests to the corresponding OpenDaylight controller which in turn will send the OFPT_FLOW_REMOVED messages to the OFSs. Then, the physical servers involved in the migration (the controller node, the source and destination compute nodes) must be connected during all the migration process in order to exchange the VM images data. The SINO adds each of the three hosts (Controller, Compute1 and Compute3) as external hosts into the MSO available endpoints, and request an end-to-end connectivity service between each node pair. The inter-DC connectivity service triggers the creation of an optical connection within the WSON domain through the AS-PCE and several flows in the two packet domains through each corresponding SDN controller.

The VM migration process of the computing instance start by stopping VM1. Then a snapshot image of VM1 is taken in and stored in the Cloud Controller image database (Openstack Glance image service). Once the VM1 snapshot is ready, the VM1 instance is removed and a new instance based in the recently created snapshot is deployed into the destination compute node. To conclude, the connections are restored between the migrated VM and all the rest of VMs to which was connected. In the proposed scenario, the new path between VM1 and VM2 is calculated by the NO PCE, the LSP between DCs is reused, and it is only necessary to establish the flows of the OFSs. Finally, the acknowledgment of virtual machine successful migration is sent by the SINO to the client application.

Figure 9.8 shows the traffic capture taken at the SINO's host which includes the orchestration messages sent to the Cloud controller and the MSO and, the traffic captured in the control interface between the MSO, the SDN controllers and the AS-PCE. We can obtain this detailed capture as we

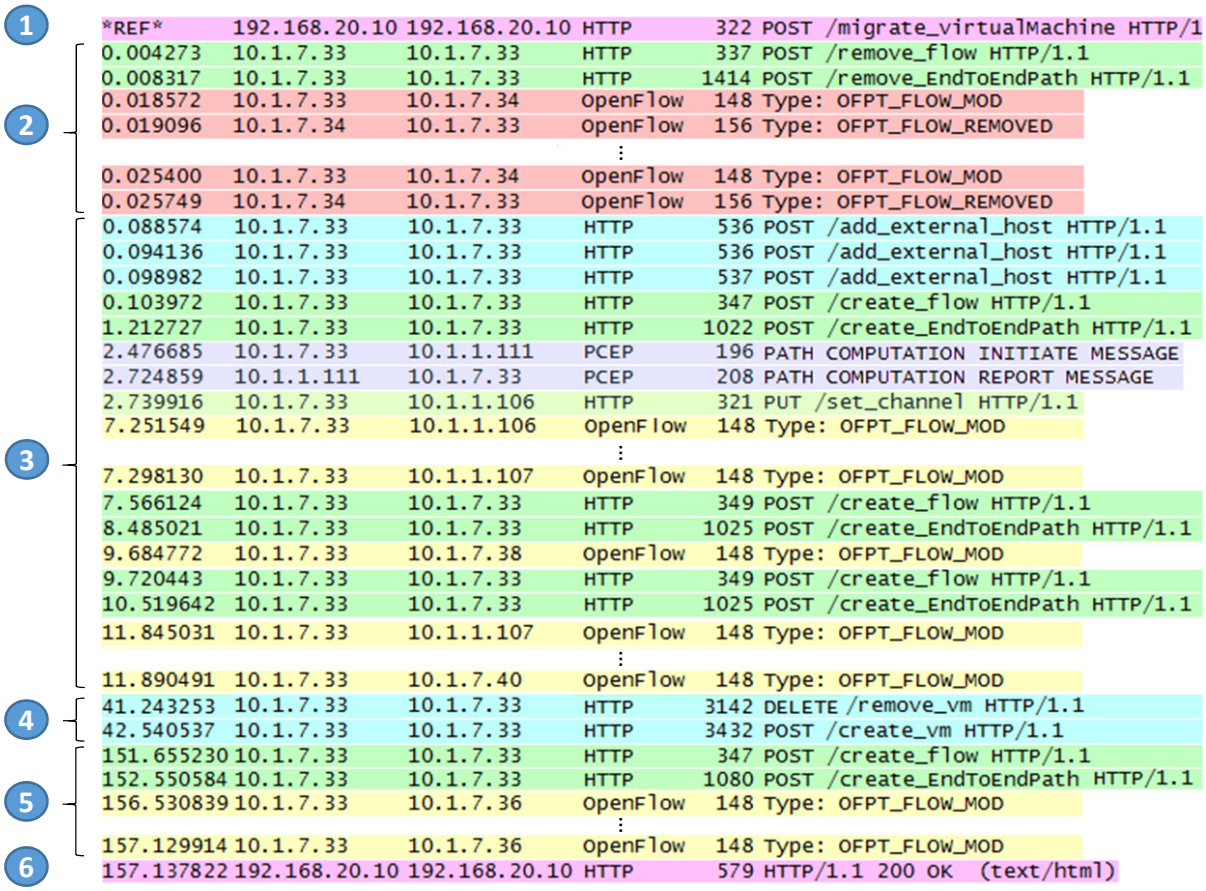


Figure 9.8: Wireshark capture of SINO commands;

are running on the same machine the SINO, the Cloud Controller Node and the MSO. In Figure 9.9, the incoming traffic to DC-2 is shown by sampling each 5 seconds the packets received in the border Openflow Switch of DC-2. A significant increase of the number of packets occurs after the trigger of creation of a VM1 into Compute Node 3 due to the download of the VM1 image from Cloud Controller Node (which is physically located in DC-1). Finally, a traffic capture of the dialog between VM2 and VM1 is shown in Figure 9.10 which provides the total disruption time occurred due to the seamless migration process. We can observe that after a downtime of 158s the connectivity between VM1 and VM2 is recovered. It can be observed that the MAC address of VM1 has changed due to the block migration, the final L2 flows connecting VM1 and VM2 reflect this change thanks to the orchestration of the SINO.

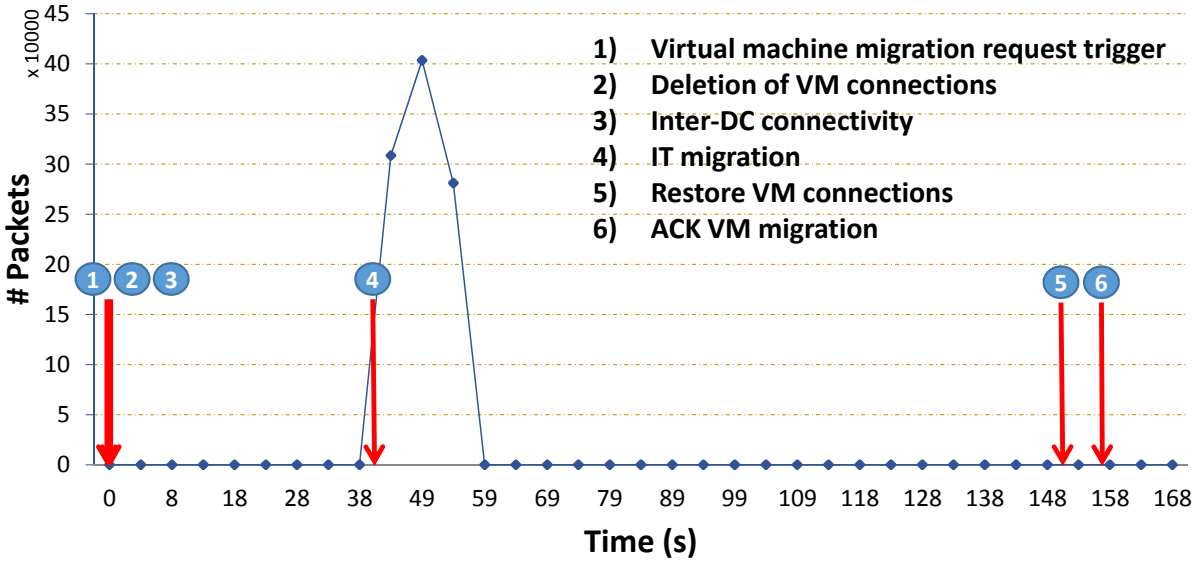


Figure 9.9: VM migration traffic (packets/s) received in DC2 over time

REF	VM2	VM1	ICMP	98	Echo (ping) request
0.000801	VM1	VM2	ICMP	98	Echo (ping) reply
Ethernet II, Src: fa:16:3e:e0:07:92 (fa:16:3e:e0:07:92), Dst: fa:16:3e:64:63:06					
Internet Protocol Version 4, Src: VM2 (10.10.0.15), Dst: VM1 (10.10.0.12)					

158.164090	VM2	VM1	ICMP	98	Echo (ping) request
159.172043	VM1	VM2	ICMP	98	Echo (ping) reply
Ethernet II, Src: fa:16:3e:e0:07:92 (fa:16:3e:e0:07:92), Dst: fa:16:3e:cd:d9:75					
Internet Protocol Version 4, Src: VM2 (10.10.0.15), Dst: VM1 (10.10.0.12)					

Figure 9.10: ICMP traffic capture between VM1 and VM2 during VM1 seamless migration.

9.4 Conclusions

In this chapter has been presented an integrated IT and Network Orchestration architecture, which is aligned with the NFV architectural framework proposed by the ETSI, that assesses the need of holistic management of storage, compute and networking resources. The proposed architecture includes a novel software component named SDN IT and Network Orchestrator (SINO) which provides E2E orchestration of IT (compute, storage) and network resources over distributed cloud management systems and SDN-based heterogeneous network segments. The proposed architecture benefits from the usage of open standard interfaces (COP, OpenStack API) to provide the holistic control of heterogeneous resources which can be deployed in a coordinated manner conforming and E2E service.

The context of application of this work is a distributed Data Center (DC) scenario where computing pools of resources are geographically distributed across Wide Area Networks (WANs). The Multi-domain SDN Orchestration (MSO) component is a key enabler for inter-DC network control and allows efficient and flexible allocation of network resources in such a dynamic scenario where virtual machines need to be created, deleted and migrated, while maintaining network services with other instances. These concepts have been successfully demonstrated in the Adrenaline Testbed 3, including the VM creation and migration across distributed cloud sites.

The dynamic management of interconnected computing resources (VMs) lays the foundation for

the next chapters, where the deployment of virtual network services consisting on interconnected Virtual Network Functions (VNFs), will be evaluated. NFV and SDN are the key enablers for the upcoming 5G, which will be also a central topic to be discussed in the following chapters.

Chapter 10

IT and Network resource allocation and orchestration

The new 5G paradigm seeks for a scalable architecture able to efficiently manage the increasing volume of traffic generated by smart devices to be processed in a distributed cloud infrastructure. To this end, a coordinated management of the network and the cloud resources forming an end-to-end system, is of great importance. Software Defined Networking (SDN) and Network Function Virtualization (NFV) architectures are the key enablers to integrate both network and cloud resources, enabling cross-optimization in both sides. This optimization requires efficient resource allocation algorithms which take into account both computing and network resources.

In Chapter 9, an end-to-end orchestration architecture for distributed cloud and network resources aligned with the ETSI NFV architectural framework was presented. In this chapter, the Virtual Infrastructure Manager and Planner (VIMaP) component it is introduced to enable dynamic resource allocation for interconnected virtual instances in distributed cloud locations. The VIMAP extends the SINO architecture presented in the previous chapter to introduce two main new features: (a) the inclusion of a resource planner component responsible of run resource allocation algorithms for the optimal allocation of computing, storage and network resources for incoming infrastructure deployment requests; and (b) the management of multiple tenants to perform context-aware IT and Network orchestration.

This chapter formally defines and models the resource allocation problem of dynamically provisioning of Infrastructure-as-a-Service (IaaS), which we refer here as Virtual Machine Graphs (VMGs) placement. In order to solve this problem, an heuristic baseline solution based on greedy approach for the selection of DCs and First Fit (FF) for the VM allocation is proposed. The objective is to provide a baseline implementation of a resource allocation algorithm for the latter experimental validation of the whole architecture (with a special focus on the VIMaP component and the VMG allocation). However, to verify its validity, the proposed algorithm performance is evaluated and compared with a Random Fit based solution in a controlled simulation scenario.

The chapter is organized as follows: section 10.1 includes the proposed architecture and the novelties of the VIMaP respect to the one presented in the previous chapter. In section 10.2, the VMG allocation problem and the proposed heuristic are formally presented. Finally, in section 10.3, the proposed heuristic algorithm is evaluated including description of the simulation environment the simulation results.

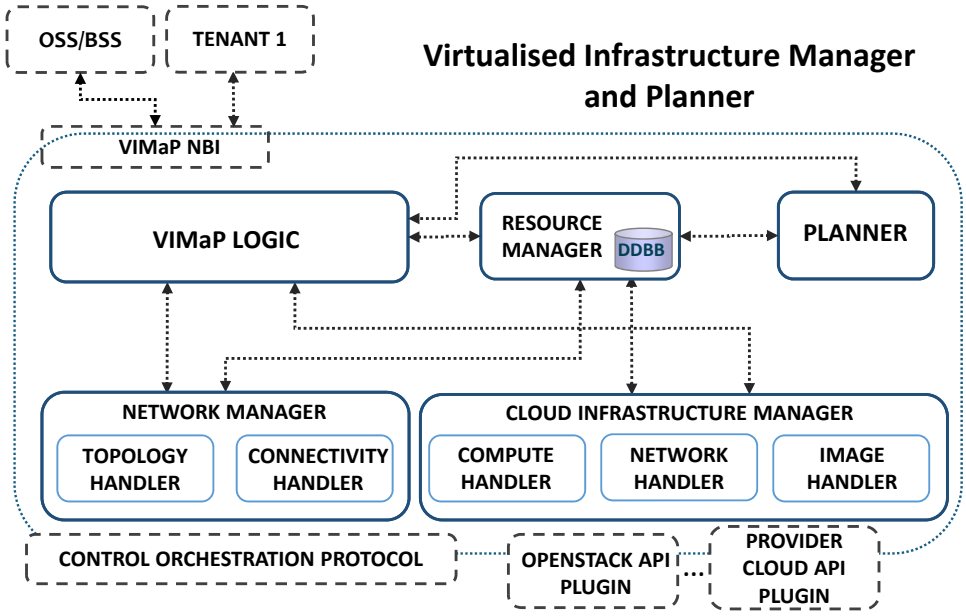


Figure 10.1: VIMaP internal architecture, building blocks.

10.1 Virtual Infrastructure Manager and Planner (VIMAP) architecture

In this section we present the VIMaP architecture including the description of its main building blocks, which are shown in Figure 10.1. The VIMaP has been designed to provide coordinated orchestration of network and cloud resources distributed among different cloud providers and locations. The VIMaP provides per-tenant programmability of its own dedicated resources, it performs the partitioning of the underlying infrastructure exposing an abstracted view of virtual infrastructure slices to each tenant.

Initially, the VIMaP is requested to provide a virtual infrastructure slice to a dedicated tenant. This request includes a set of virtual instances interconnected forming a Virtual Machine Graph (VMG). The VIMaP architecture includes a Planner component dedicated to perform resource planning optimization. Different resource optimization policies may be applied depending on the tenant and provider requirements. In [75], the authors assessed the problem of VMGs resource allocation in

distributed DC scenarios by finding the minimum diameter graph (in terms of distance) to minimize the latency between VMs. Authors in [76] proposes a resource allocation approach taking into account the distance between DC and the network load to select the connection path. The VIMaP architecture allows the VIMaP LOGIC component to select the preferred algorithm depending on the desired resource allocation policy. The algorithm receives the resource allocation requests from the VIMaP logic and it obtains all the substrate infrastructure information from the Resource Manager component which maintain up-to-date information of both the cloud and the network underlying infrastructure.

The VIMaP includes a dedicated configuration interface for slice provisioning which is exposed to OSS/NMS management systems through a RESTful API. The VIMaP LOGIC component is the responsible of orchestrate the workflows among the different architectural components in order to provision the cloud and network resources for an upcoming request. It is the responsible for performing context-aware orchestration, exposing to each tenant only those resources allocated to the tenant by means of virtual representation. It includes a northbound interface (NBI) which exposes the custom set of VIMaP programmable resources to each tenant.

The Resource Manager is responsible for storing and maintaining up-to-date state of all virtual and physical sources controlled by the VIMaP. It is also responsible for maintaining the resource allocation relationship between the requested virtual resources and the allocated physical resources.

Network Manager functions are two-fold: first it provides the southbound interface towards network infrastructure controllers including the necessary application programmable interfaces (API) or protocols implementations. As it was presented in Chapter 6, the COP is the protocol chosen to unify the network orchestration interface towards different SDN controllers. Secondly, the Network Manager is responsible for managing the virtual network resources of each tenant. The network manager correlates the VTN representation with the dedicated SDN controller slice, there is a 1:1 relation between a VTN and a SDN Controller Slice.

Cloud Infrastructure Manager is responsible for distributed cloud orchestration. Differently to the Network Manager, it is responsible of the partitioning and aggregation of cloud resources which might be distributed across different clouds (private, public). Once the selected DCs are allocated for a given tenant, it is responsible of creating a tenant session on each child cloud system and mapping all these client sessions to the corresponding VIMaP *TenantID*. Once this initial abstraction is performed, it is responsible for aggregating all the resources distributed among different clouds into a single unified view accessible by the tenant through the VIMaP NBI. This is performed populating the Resource Manager database with virtual representation of the resources deployed in the underlying infrastructure, these resources are segmented by its corresponding VIMaP global *TenantID*.

10.2 Virtual Machine Graphs (VMG) resource allocation

In this section it is firstly described the proposed Virtual Machine Graph (VMG) allocation problem. Then, a reduction of the problem is presented based on constructing the aggregated VMG solution graph, where the objective is to find groups of VMs to be allocated together in the same substrate hosting nodes. This reduction is modeled based on a constrained mapping function. Finally, a heuristic algorithm solution to this problem is proposed and simulation results for the algorithm behavior are provided.

10.2.1 Problem definition

Substrate infrastructure. We model the substrate infrastructure as a directed graph and denote it by $G^S = (N^S, H^S, L^S)$, where N^S is the set of substrate switching nodes, H^S is the set of substrate hosting nodes (DCs) and L^S denotes the set of substrate links $l^s = (u, v), l^s \in L^S, \forall u, v \in N^S \cup H^S$.

Virtual machine graph request. We denote by a directed graph $G^S = (H^V, L^V)$ the VMGP request. H^V denotes the set of virtual hosts (VMs) and L^V denotes the set of links between virtual hosts.

Now a set of capacity functions are defined for the substrate and virtual resources. Each host (physical or virtual) $h^x \in H^x, x \in \{S, V\}$ is attributed with a set of A attributes whose capacities are denoted as $c_a(h^x), a \in A, h^x \in H^X, A \in \{CPU, MEM, STO\}$ (only CPU, memory and storage have been considered as host attributes). Moreover, each link $l^x \in L^X$ is associated with a bandwidth capacity $bw(l^x)$. We also denote P^S as the set of free loop paths in the substrate network between hosting nodes.

The objective is to find a mapping function for all virtual hosts and links to the substrate infrastructure as:

$$M : (H^V, L^V) \mapsto (H^S, P^S)$$

In the next subsection, a reduction of the problem is proposed and the constrains in terms of capacities for hosts and links are introduced.

10.2.2 VMG mapping problem

To solve the above described problem, we propose a first reduction which consist in: a) finding a VM allocation among the substrate hosting nodes and, b) find an feasible allocation solution for the links connecting VM in different hosting nodes. It is assumed that several virtual hosts can be placed in the same substrate hosting node if enough computing resources are available in the substrate node for the aggregated capacity of the virtual hosts allocated to it.

We model the aggregated VMG solution graph as $G' = (H', L')$, where each $h' \in H'$ denotes a subset $h' \subseteq H^V$ of virtual hosts. Given the powerset of all possible subsets of H^V denoted as $\mathcal{P}(H^V)$, the subsets included in a hosting allocation solution $H' \subset \mathcal{P}(H^V)$, must be complementary and disjoint, i.e., that satisfies both $\bigcup_{H'} h' = H^V$ and $\bigcap_{H'} h' = \emptyset$.

On the other hand, L' denotes the set of links between virtual hosts in different aggregated subsets $l' = (u, v), \forall l' \in L', u \in h'_i, v \in h'_j$ and $h'_i \neq h'_j$.

Once G' has been described, we can define the mapping function between the VMG solution graph and the substrate infrastructure as:

$$M : (H', L') \mapsto (H^{S'}, P^{S'})$$

where $H^{S'} \subseteq H^S, P^{S'} \subseteq P^S$. The mapping function can be split hosting and link mapping as:

- **Hosting mapping function**:

  $$M^H : (H') \mapsto (H^{S'})$$

  which satisfies:

  $$\forall h' \in H', \forall h^{s'} \in H^{S'}, \quad \sum_{\forall h^v \in h'} c_a(h^v) \leq c_a(h^{s'}) \tag{10.1}$$

  In order to compare the sizes of the hosts (physical or virtual) in relative terms, we define the function *weight*, as the weighted sum of the individual computing capacities, we use the constants α, β, γ to weight up the CPU, Memory and Storage capacities respectively:

  $$weight(h^x) = \alpha c_{CPU}(h^x) + \beta c_{MEM}(h^x) + \gamma c_{STO}(h^x) \tag{10.2}$$

- **Link mapping function**:

$$M^L : (L') \mapsto (P^{S'})$$

which satisfies:

$$\forall l' \in L', \forall p^{s'} \in P^{S'}, \quad bw(l') \leq BW(p^{s'}) \tag{10.3}$$

$$where, BW(p^s) = \min_{\forall l^s \in p^{s'}} bw(l^s)$$

10.2.3 Baseline VMG embedding algorithm

The problem has been reduced to find a feasible allocation for the solution graph G' which satisfies (10.1) and (10.3).

We asses the problem in two steps:

- Step 1: Following a Greedy procedure, we select the minimum number of substrate hosting nodes with enough capacity to allocate all the virtual hosts in H^V, which are embedded sequentially following a First Fit approach (see Algorithm 1).
- Step 2: Based on the selected $H^{s\prime} \subseteq H^S$ we employ the Constrained Shortest Path First (CSPF) algorithm to find a feasible path in the substrate network, for each s-t pair allocated in different substrate hosting nodes (see Algorithm 2).

Algorithm 1 first computes the Greedy and the First Fit (FF) host mapping procedure to find the minimum cluster with enough capacity to allocate virtual hosts within the VMG request. Firstly, it sorts the substrate host set in decreasing order by weight and it sequentially allocates the virtual hosts into the substrate hosting nodes with higher capacities. As a result this function returns the solution subset with minimum size $H^{S'} \subseteq H^S$.

Algorithm 2 receives the host solution subset and both substrate and virtual links of the VMG request. Based on the host mapping solution, for each virtual link $l'(u, v)$, a feasible path p' between nodes allocated to different $h'_u, h'_v, i \neq j$ is calculated. We use the CSPF algorithm with the $bw(u, v)$ as a constrain parameter. If there is a feasible path for each $l' \in L'$, the mapping solution is returned : $(H', L') \mapsto (H^{S'}, P')$.

Algorithm 1 GreedyFFHostMapping(H^S, H^V)

Input: H^S: Substrate hosting nodes, H^V: Virtual hosts.
Output: $H', H^{S'}$: host solution set
  Sort $H^S = h_1^s, h_2^s, ..., h_n^s$ in decreasing order by its weight.
  $H^{S\prime} \leftarrow \emptyset$
  $H' \leftarrow \emptyset$
  $H^{v\prime} \leftarrow H^V$
  while $\sum_{\forall h^{s\prime} \in H^{S\prime}}(c_a(h^{s\prime})) < \sum_{\forall h^v \in H^V}(c_a(h^v))$,
    $\forall a \in A$ **do**
    $h^s \leftarrow H^S.pop()$
    $currentC_a \leftarrow c_a(h^s), \forall a \in A$
    $current_s \leftarrow \emptyset$
    for v **in** $H^{v\prime}$ **do**
      if oneOf $currentC_a < c_a(v), \forall a \in A$ **then**
        $H^{v\prime} \leftarrow H^{v\prime} - current_s$
        break
      else
        $current_s \leftarrow current_s \cup \{v\}$
        $currentC_a \leftarrow currentC_a - c_a(v), \forall a \in A$
      end if
    end for
    $H' \leftarrow H' \cup current_s$
    $H^{s\prime} \leftarrow H^{s\prime} \cup h^s$
  end while
  return $M^H : H' \mapsto H^{S\prime}$

Algorithm 2 CSPF Link Mapping ($H', H^{S'}, L^S, L^V$)

Input: $H', H^{S'}$: substrate host solution set,
  L^S: Input substrate links,
  L^V: Input links request
Output: $M : (H', H^{S'}), (L', P^{S'})$: Mapping solution from $G^V \mapsto G^S$
  for (u, v) **in** L^V **do**
    if $h'_u \neq h'_v$ **then**
      $L' \leftarrow L' \cup (u, v)$
    end if
  end for
  for $l'(u, v)$ **in** L' **do**
    $p^{s\prime} \leftarrow CSPF(G^S, h_u^{s\prime}, h_v^{s\prime}, bw(l'))$
  end for
  return $M : (H', L') \mapsto (H^{S'}, P^{S'})$

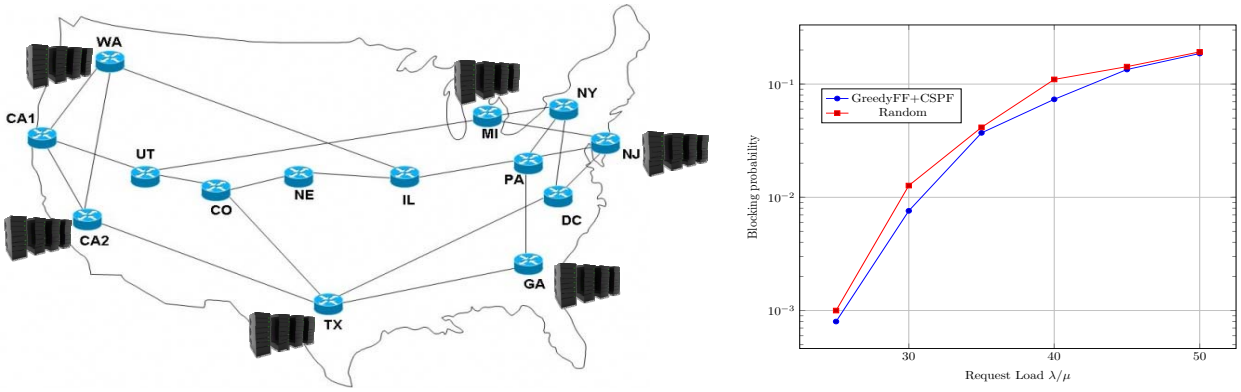
10.3 VMG allocation results

Parameter	Values
H^S CPU values	[100, 200, 400]
H^S Memory values	[200, 400, 800]
H^S Storage values	[10000, 20000, 40000]
L^S Bandwidth	100 Gbps
H^V CPU values	[1, 2, 4, 8]
H^V Memory values	[2, 4, 8, 16]
H^V Storage values	[20, 40, 80, 160]
L^V Bandwidth	(0.1:1) Gbps

Table 10.1: Experiments parameter configuration

In this section the evaluation of the proposed heuristic baseline solution is presented and compared with a Random Fit based algorithm. The random solution differs on the DC selection strategy but keeps CSPF to assure path feasibility in the virtual link selection stage.

The substrate infrastructure scenario employed for the experiments is an extended version of the NSFNet of 14 nodes and 42 unidirectional links and 6 DCs (Figure 10.2a). For simplicity, the DCs are co-located within the same network node locations and the connectivity between DC's and its corresponding network nodes is modeled to have infinite bandwidth. The substrate infrastructure is initially configured with a pre-defined capacities which are maintained along all the experiments. The values of the capacities of each DC are uniformly distributed among the values included in each range depicted in Table 10.1.



(a) (b)

Figure 10.2: (a) NSF Network of 14 nodes with 6 DC; (b) VMG request blocking orobability of GreedyFF+CSPF and Random Fit+CSPF algorithms.

In the VMG requests, the number of virtual nodes is randomly determined by a uniform distribution between 2 and 20. Each pair of nodes are randomly connected with probability 0.5, in total we will have $n(n-1)/4$ links in average. The capacities of the virtual hosts and the virtual links are also selected randomly following an uniform distribution along the values depicted in Table 10.1.

The VMG requests arrives to the VIMaP following a Poisson process on which the arrival rate is varying. The holding time of the VMG requests in the system follows an exponential distribution

with 10 time windows on average. We run all the simulations for 10000 requests for each instance of the simulation.

The results of the simulation for different loads can be seen in Figure 10.2b. The results show a slightly better performance of the GreedyFF approach compared with the Random. This result is explained by the fact that the Greedy approach minimizes the number of DC selected in the first stage, minimizing as well the number of connections between DCs and thus decreasing network utilization. The differences obtained are minimal, showing that the dominant factor for the blocking probability in this experiment is not the exhaustion of network resources but the cloud. In this paper, the target is to present the problem of VMG allocation and the baseline solution for the proposed VIMaP architecture. It is intended for future work the evaluation of more complex algorithms and its comparison within the VIMaP.

10.4 Conclusions

In this chapter we have presented the Virtual Infrastructure Manager and Planner (VIMaP) component which extends the SINO architecture introduced in the previous chapter to to introduce two main new features: (a) the inclusion of a resource planner component responsible of run resource allocation algorithms for the optimal allocation of computing, storage and network resources for incoming infrastructure deployment requests; and (b) the management of multiple tenants to perform context-aware IT and Network orchestration.

The VMG placement problem has been mathematically defined and a baseline resource allocation heuristic algorithm, based on a Greedy approach for the selection of DCs and First Fit (FF) for the VM allocation, was also defined. The algorithm has been evaluated in a simulation environment based on the NSFNet14 reference scenario and the results compared with a simple RANDOM FIT heuristic.

The main contribution of this work is the introduction of a resource optimization engine in the architecture, for the deployment of NFVi slices to dedicated tenants. The objective is to provide the architectural basis for the provisioning of 5G network slices which will be the central topic of the next part of this PhD Thesis. The VMGP problem can easily extended for the reservation of pools of resources in multiple DCs and aggregated pipes of bandwidth capacity in the transport network to interconnect those pools conforming a L2 VTN.

The next part which consist on Chapter 11 and Chapter 12 will be focused on the network slicing concept. The architecture proposed is the base for the 5G Network Slicing architecture and the VIMaP and the MSO its main building blocks.

Part IV

5G Network Slicing

Chapter 11

Multi-tenant 5G Network slicing

As it was previously described in the introduction (see section 1.5.2), the 5G architecture shall accommodate a wide range of use cases derived from the new needs of vertical industries, customers and enterprises. This new requirements impose the need of offering different combination network performance metrics (bandwidth, latency, availability) in a customized way for the different 5G use cases.

In order to face this challenge, the '5G network slicing' concept introduced by the NGNM, is presented and experimentally assessed through novel multi-tenant 5G network slicing architecture that dynamically provides 5G network slices addressing specific tenant requirements, including virtualized SDN/NFV control and management instances for a full tenant control of allocated resources.

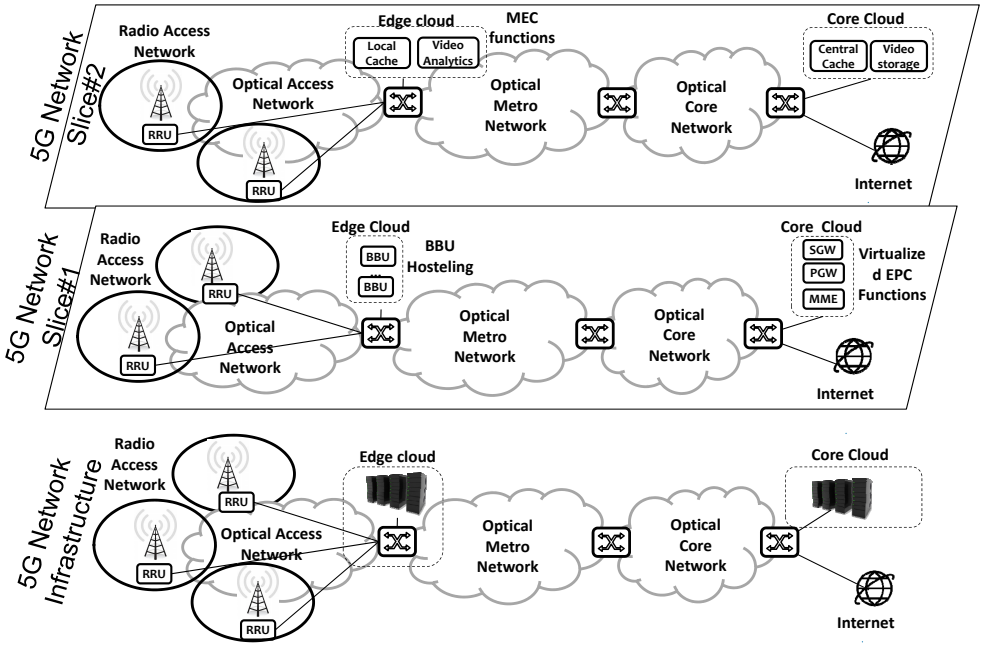


Figure 11.1: 5G network slicing

11.1 Multi-tenant 5G Network slicing architecture

The proposed 5G slicing architecture aims at providing multiple, highly flexible, end-to-end network and cloud infrastructure slices operated in parallel over the same physical resources, to fulfill vertical-specific and mobile broadband services' requirements. A 5G slice is composed of tailored virtualized network and cloud resources, as well as the required Virtualized Network Functions (VNFs) that can be controlled by third-parties (we will refer here generally as tenants) through their own management and orchestration (MANO) instance, consisting on a SDN controller, a cloud orchestrator and a NFV orchestrator (NFVO), via a suitable open APIs. Bearing this in mind, the proposed 5G architecture for network slicing is depicted in Figure 11.2. It is composed of three main building blocks or layers: the Virtualized Infrastructure Manager (VIM), the Network Function Virtualization Orchestrator (NFVO), and the Multi-tenant Slicing Manager (MTSM).

The VIM acts as a unified cloud and network operating system providing the allocation and orchestration of IT and network resources across multiple distributed clouds and multi-domain multi-technology networks [77]. The VIM comprises an End-to-End (E2E) SDN Orchestrator component consisting of: (i) the Multi-domain SDN Orchestrator (MSO), which is responsible of the orchestration of multiple network domains with heterogeneous transport (wireless/optical) and control technologies (SDN/GMPLS) following the same procedures described in chapters 5 and 7; and (ii) an E2E Network Hypervisor (ENH) component which is responsible of dynamic creation and control, at a higher and abstracted level, of end-to-end virtual networks for multiple tenants [78]. The main task of the ENH is to provide an abstraction layer exposing virtualized network resources as they were real to customized tenant SDN controllers.

On the other hand, the VIM must provide multi-cloud orchestration, enabling the management of multiple cloud systems on distributed data center (DC) sites (edge, core, and public cloud). It allows to create administrative tenants across multiple cloud sites, exposing an unified view of the computing, storage and intra-DC network resources allocated to the customized tenant VIM instance. It also provides the API to instantiate the creation/deletion of virtual machine (VM) instances, disk

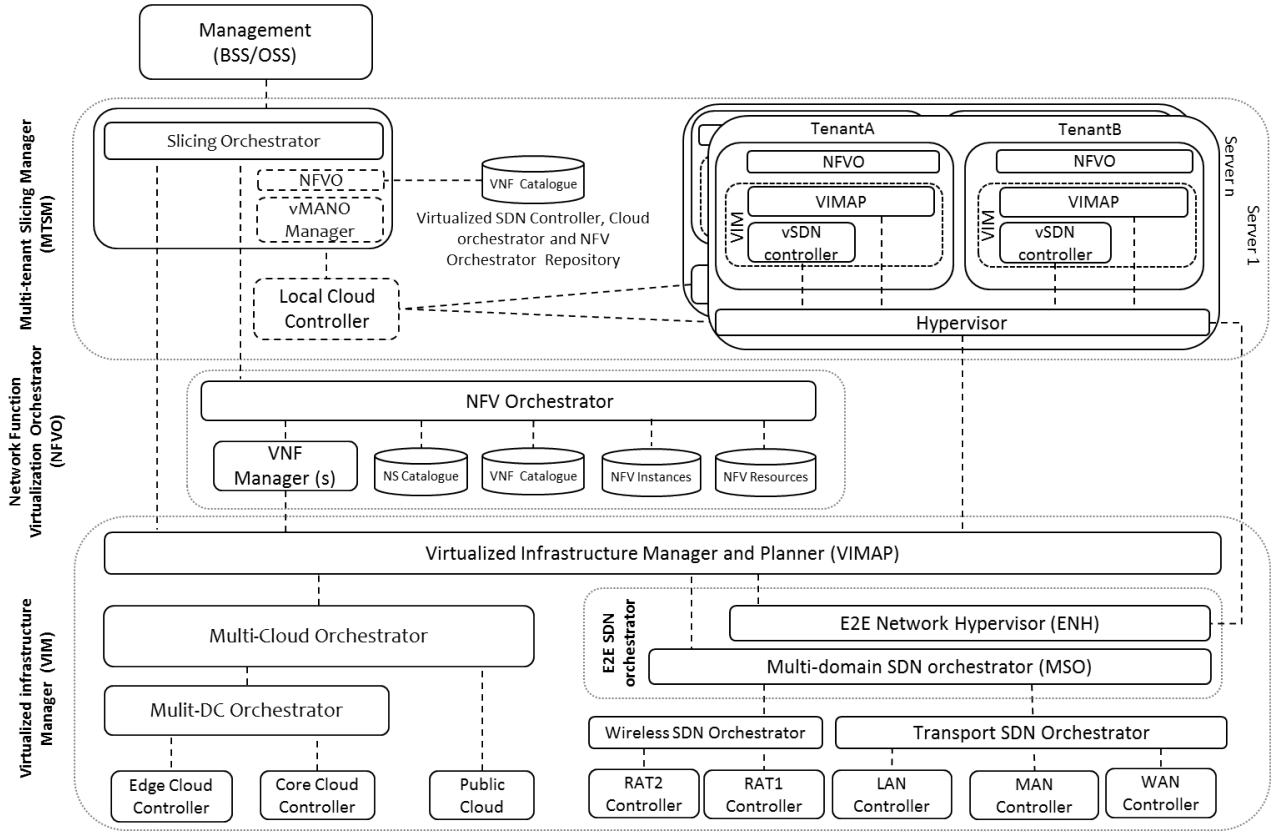


Figure 11.2: Proposed 5G slicing architecture

images, and overlay networks between VMs. The VIM includes the VIMAP component presented in Chapter 10 as unified IT and network orchestrator, acting as a resource broker which maps the virtual resources pertaining to the tenant with the actual physical resources allocated to it. It also exposes a network slicing API which allows to customize the tenant infrastructure slice with the definition of the Virtual Tenant Network (VTN) topology, the quotas of computing and storage resources for each DC exposed to the tenant and the creation of VNF templates (disk images) to be available into vMANO instance provided to the tenant.

The Operator NFVO is responsible of the deployment of Network Services (NS), VNF forwarding graphs (VNF-FGs) and VNF packages on top of a 5G Network Slice. The VNFs run in virtual instances that can be allocated in the most appropriate NFV Infrastructure Point-of-Presence (NFVI-PoP) (DC site). These VNFs shall be interconnected with each other and with the user service end-points (SEPs) conforming a forwarding graph, to perform the desired overall end-to-end functionality or NS. The NFVO can request to the VIM the creation of: (1) the VNFs consisting on a VNF image template running into a VM with a custom networking configuration (i.e., IP address, VLAN ports, IPTables configuration); (2) the provisioning of the connectivity services between the VNFs; and (3), to the VNF Managers, the life-cycle management of each of the individual VNFs which form the VNF-FG or NS, including the bootstrap configuration for the NS, the VNF are part of, the monitoring of the VNFs performance and the modification (e.g., scaling, migration) or release of the reserved resources in case the network service changes or is set to be terminated.

The MTSM is responsible for the dynamic life-cycle management (provisioning, modification and deletion) of not only the requested infrastructure slices but also of the virtualized MANO instance associated to each tenant which conform the 5G Network Slice. Virtualized MANO instances enable tenants to have full SDN/NFV control of all virtual resources (network, cloud, and VNFs) assigned to

its slice as if they were real. The MTSM layer consist on a Slice Orchestration application, a vMANO manger which is in charge of the lifecycle management of the vMANO instances and a local cloud which is used by the operator to deploy the tenant vMANO instances.

11.2 Dynamic deployment, operation and management of 5G network slices

The network slicing operation is a complex task which involves the orchestration of cloud and network resources but it also adds the management of dedicated control VNFs deployed on a per-tenant basis. The main idea beneath is to provide a completely autonomous infrastructure slice to each tenant which includes the virtual VIM and the vMANO instances which control a portion of the provider resources dedicated to that tenant.

In this section we are introducing the operational workflows involved in the deployment of a 5G network slicing under the proposed architecture.

11.2.1 Virtualization of the Transport Network infrastructure.

We consider the term transport network virtualization as the partitioning (slicing) of the physical infrastructure to create multiple co-existing and independent VTNs on top of it. For the VTN slice composition, the abstracted topology view shall be defined by the tenant, including a set of physical network interfaces connected following a determined graph and the characterization of the network elements (nodes, interfaces) at the data plane level (i.e., Layer 2-3). On the other hand, the virtualization is carried out by the ENH, which upon the VTN composition request, performs the mapping between the logical and physical network elements (VTN topology abstraction) and request to the MSO the required data plane connectivity services within the physical network interfaces to provide the desired level of data plane abstraction.

In Chapter 7, two abstraction models were proposed for virtual network composition: single virtual node and abstract link model. The former represents either completely or partially network topology as a single virtual node. By doing so, the internal domain topology is hidden by the abstraction function. On the other hand, the abstract link model provides a summarized view of the internal topology of the network domain. For instance, the abstract view on this model may present the different network domains as network nodes interconnected by virtual links, which are internally mapped to physical routes.

At the data plane level, the network virtualization can be performed at different layers. Layer 0 or Optical Virtual Networks (VON) can be realized by means of logically split the available wavelengths in a fiber which are reserved to different VONs which are controlled by a virtualized, dedicated control instance (i.e., GMPLS [79]). Layer 2 virtual private networks (L2VPN) [80] provides a suitable reference framework for Layer 2 network virtualization. Specifically, point-to-point Virtual Private Wire Services (VPWS) model can be directly applied for the proposed virtual link abstraction, and Virtual Private LAN Services (VPLS) can enable a logical L2 switch through the creation of multiple point-to-multipoint connections among the logical interfaces composing the virtual node (Node Abstraction model). Provider Edge (PE) network elements can be connected by MPLS Label Switched Paths (LSPs) or Generic Routing Encapsulation protocol (GRE) tunnels, providing data isolation across WANs [81]. The connection between the Customer Edge (CE) and the PE, i.e., the attachment circuit (AC), can be a dedicated physical interface (i.e., Ethernet port) or a tagged interface i.e., a VLANs, which can be mapped to a logical port/interface in the VTN topology. At the Layer

3, the composition of overlay networks through tunneling mechanisms i.e., VXLAN or GRE, severely increases the number of coexistent Layer 2 collision domains (VXLAN support up to 16 million logical networks) that can be transported over the same physical network. L3VPNs can also be implemented using MPLS or GRE tunneling mechanisms together with Virtual Routing Forwarding tables (VRF) in the PE network elements.

Back to the SDN context, the OpenFlow protocol (OF) allows to implement the two key virtualization mechanisms used to implement L2VPN and L3VPN networks: (i) VRF concept is implemented through the multiple table pipeline processing system inherently supported by most OpenFlow Virtual Switches (OVS) implementations since OpenFlow v1.1 specification. (ii) MPLS and GRE tunneling allows traffic encapsulation, which can be filtered based on any combination of L2-L4 packed headers, enabling that logical SEPs in the VTN can be mapped to physical Ethernet ports, VLANs or Layer 3 overlay IP interfaces. Both protocols (MPLS, GRE) are supported since OpenFlow v1.3.

11.2.2 Virtualization of SDN controller instance.

The ENH, proposed in [78], is the element responsible for receiving VTN requests, processing them and allocating physical resources. Moreover, the ENH is responsible for the mapping between the allocated physical resources and the abstracted resources that are offered to the Customer SDN Controllers (CSCs), and the control of such abstract networks, acting as a proxy for the OF protocol between a CSC and the underlying Provider SDN Controller (PSC). The partitioning of the resources is performed by the ENH, and to this aim, the proposed system architecture relies on the MSO, which provides a generic network abstraction mechanism for the different transport infrastructure resources (e.g., OPS, flex-grid).

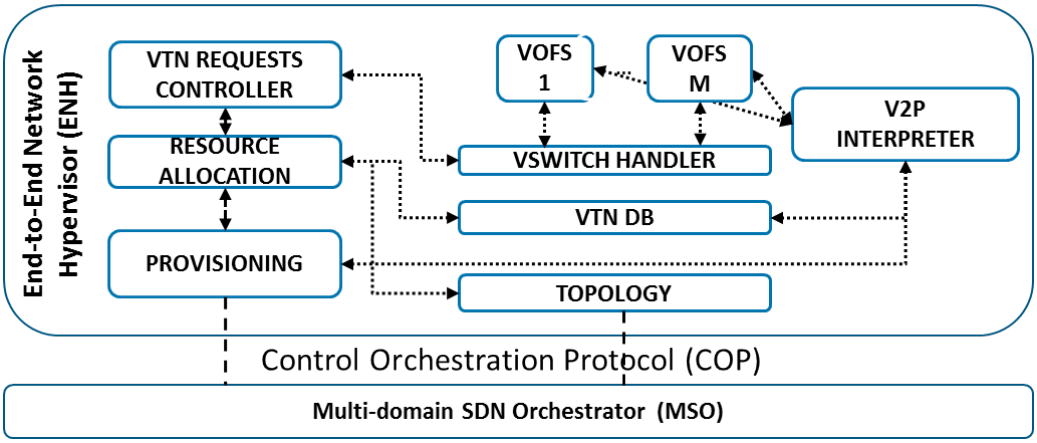


Figure 11.3: 5G Slice creation workflow

Once the VTN has been successfully created, the CSC acts as a standard SDN controller where the controlled VTN is an abstracted slice of the different allocated physical resources, which are managed by the MSO. The ENH architecture (Figure 11.3) is as follows. The VTN Controller is the component that is responsible for providing the ENH interface to request virtual switches and virtual links to deploy a VTN. To do so, also the IP address of the CSC is necessary, so that the Virtual Switch Handler is able to provide an abstract network view of the allocated VTN to the CSC. A virtual switch request includes the related physical domains (abstracted as nodes by the MSO) and a number of virtual Ethernet ports. A virtual link request includes the source and destination virtual switches. The Resource Allocation (RA) component is responsible for the allocation of the physical ports of

the physical domains to the virtual switches and to request to the MSO (through the provisioning component) the necessary multi-domain connections to interconnect the requested virtual switches, which are related to physical domains. Once the connections have been established, the RA allocates the virtual port identifiers, to which the connections are related.

For each VTN, the Virtual Switch Handler establishes the necessary OF datapaths with the provided IP address of the corresponding CSC. Each OF datapath is provided by an emulated OF virtual switch. The different emulated OF virtual switches are interconnected with virtual links, so when the CSC triggers the LLDP to the emulated virtual switches, it is able to recover the VN topology. The emulated virtual OF switches are connected to the Virtual to Physical (V2P) Interpreter, which is the responsible to translate the received OF command (e.g., FLOW MOD) from the CSC using the abstract VTN topological view, to the allocated physical resources. To this end, it consults the VTN Database for the allocated physical ports and the established LSPs. The processed requests are sent to the provisioning module, which is the responsible to request the provisioning of the physical resources to the MSO. The connectivity provisioning interface between the ENH components and the MSO, is the COP (Chapter 6).

11.2.3 Virtualization of Management and Orchestration (MANO) instances.

The mode of operation of the proposed architecture for the virtualization of the MANO instances is the following: the vMANO instance provisioning is requested from Slicing Orchestrator to the NFVO which, in sequentially, requests the activation of a vMANO Manager which triggers the creation of a vMANO instance in the MTSM's local cloud. This VNF is generated by a customized vMANO image template with a Linux OS and a SDN controller (e.g. OpenDaylight), a VIMAP instance, and a NFVO (e.g., OpenMano) software appliances with the specific configuration of the target slice.

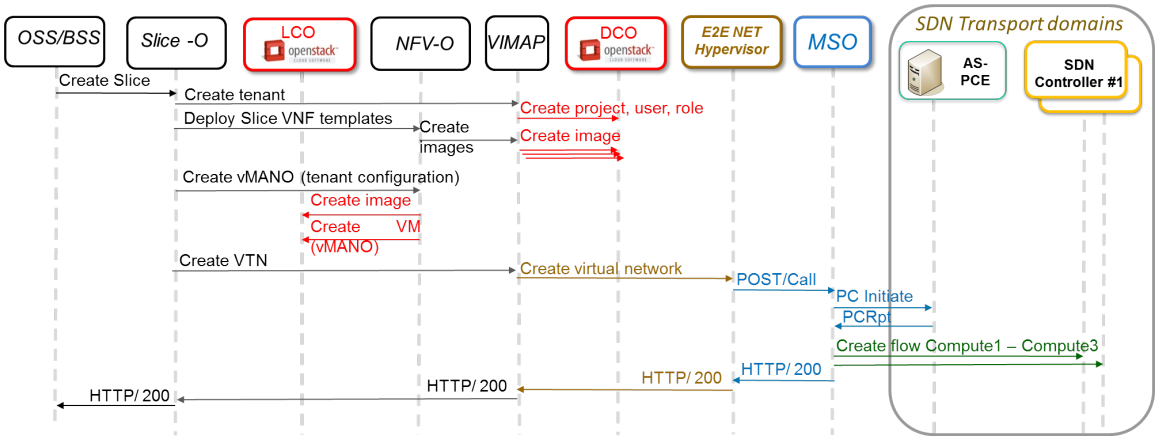


Figure 11.4: 5G Slice creation workflow

In Figure 11.4 is shown the complete 5G network slice creation workflow following the proposed architecture. First, when the Slicing Orchestrator receives a 5G network slice provisioning request from the operator management systems (BSS/OSS), it requests the creation of a new tenant instance to be deployed in the distributed cloud orchestrator (DCO) through the VIMAP. As it was detailed in the previous chapter, this operation requires that the VIMAP provides a mapping between a single tenant view of their allocated resources and the multiple tenant sessions created on each Cloud controller managed by the VIM. Second, the set of VNF templates (disk images), which will be available for the tenant vMANO instance, are requested to be deployed in the cloud tenant slice through the first the NFVO (who manage the VNF templates) and then to the VIM (who manage the infrastructure).

Afterwards, the Slice orchestrator request the instantiation of a vMANO instance to the NFVO. The NFVO then instantiates a vMANO manager which is in charge of the life-cycle management of the vMANO instance. The VNF configuration requires the information about the cloud tenant slice, i.e., the VIMAP network address and the user credentials. Finally, when the vMANO instance is created, the VTN creation is requested to the VIMAP with the tenant SDN controller information, required to configure the tenant OF datapaths generated by the ENH.

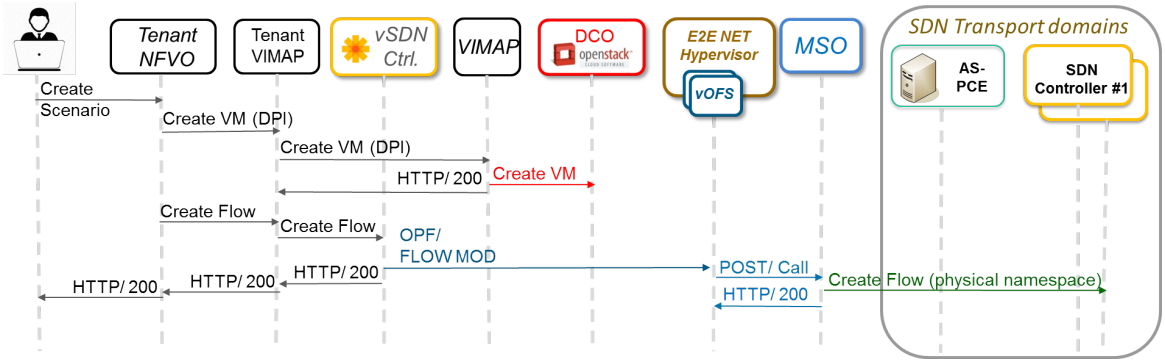


Figure 11.5: DPI VNF instance and network services provisioning operations done by Tenant MANO instance.

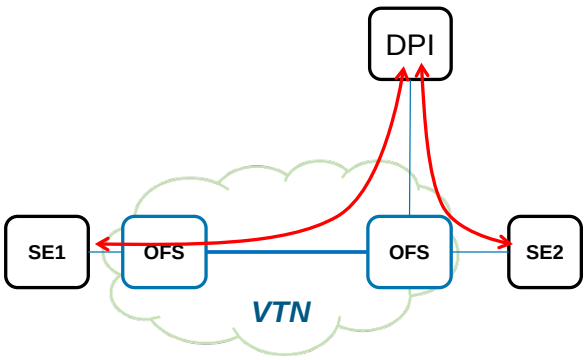


Figure 11.6: DPI forwarding graph.

11.3 Experimental validation and results

The proposed architecture has been validated in the cloud-computing platform and transport network of the ADRENALINE Testbed. The IT infrastructure consists of a local cloud (LCO) for vMANO tenant instances deployment and a distributed cloud (DCO) provider infrastructure, both controlled with Openstack Liberty cloud computing software. The DCO is geographically distributed in two locations across a SDN multi-layer transport network composed by three domains: two intra-DC networks, consisting on OpenFlow switches deployed on COTS hardware and using OpenVSwitch (OVS) and an optical WSON with 2 ROADMs and 2 OXCs based on GMPLS distributed control. Per-domain SDN controllers control the electrical and optical domains are orchestrated by a Multi-domain SDN Orchestrator based on ABNO architecture. E2E network virtualization is done by the ENH component as detailed in section 11.2.2.

11.3.1 Use case I: Creation and operation of a 5G Network Slice.

Firstly, we experimentally validate the deployment of a dedicated tenant 5G slice by provisioning a new tenant session, within the DCO, which includes the VNF images requested and the quota of IT resources allocated to the tenant. A vMANO, with a customized catalog of VNFs, is instantiated within the LCO, including a vSDN controller (OpenDaylight) for a dedicated VTN interconnecting a defined set of SEPs (user defined physical Ethernet switch ports and the DCO points-of-presence (PoP)). The VTN is composed by a virtual node (vNode) for every physical switch on which a SEPs is attached. Each vNode pair is connected by a virtual link supported by a MPLS tunnel connecting the peer nodes in the physical infrastructure, which encapsulates all the vNodes outgoing traffic. When a new VNF is deployed within the VTN, a new virtual port is attached to the corresponding vNode by the ENH.

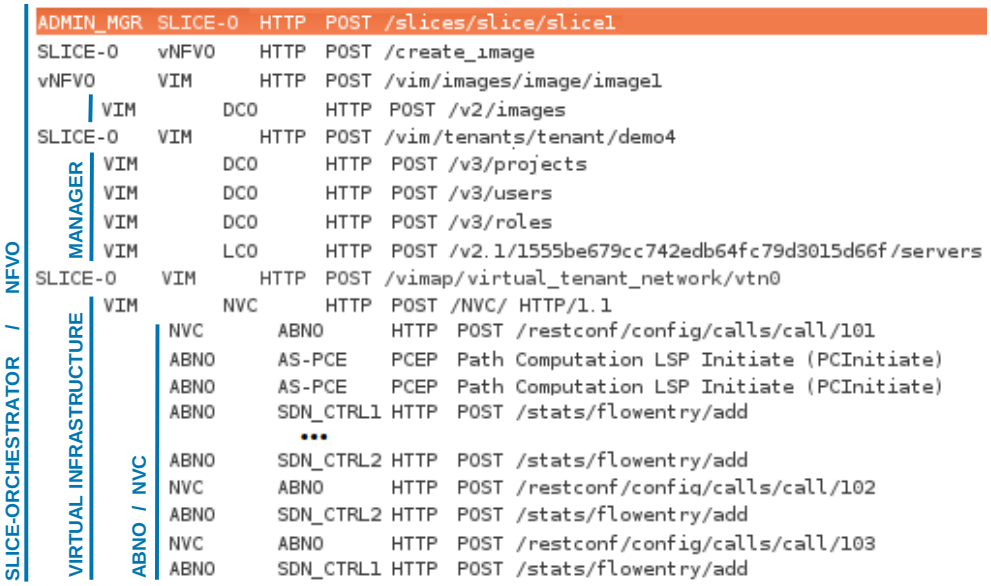


Figure 11.7: 5G slice provisioning traffic capture

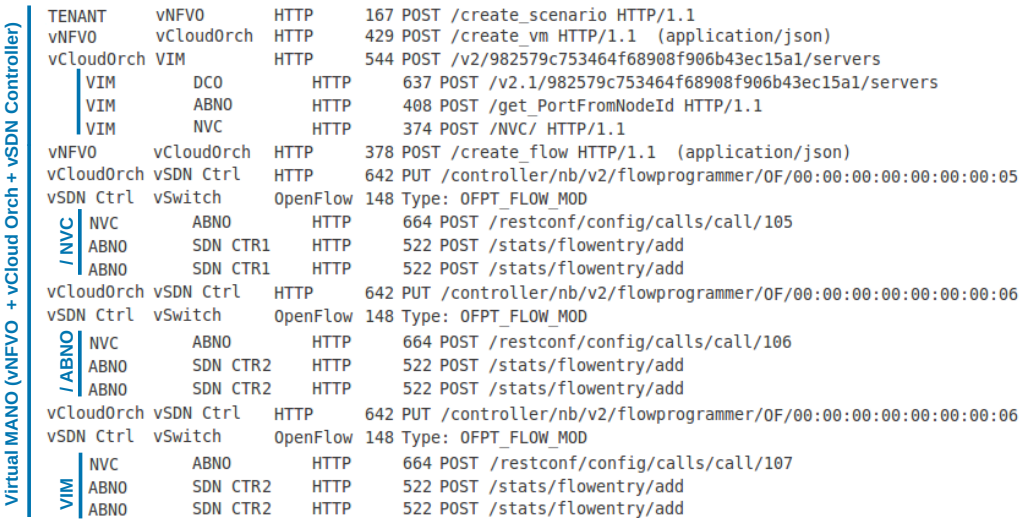


Figure 11.8: DPI VNF deployment traffic capture.

Secondly, we will validate the operation of the created 5G Network Slice, through the deployment of a network service based on a Deep Packet Inspector (DPI) VNF, deployed in the tenant slice, to inspect the traffic between the user-defined SEPs (11.5). To this aim, the vMANO orchestrates the creation of the DPI-VNF and the VNF forwarding graph through its vSDN instance, to forward the traffic between the SEs across the DPI (Figure 11.6).

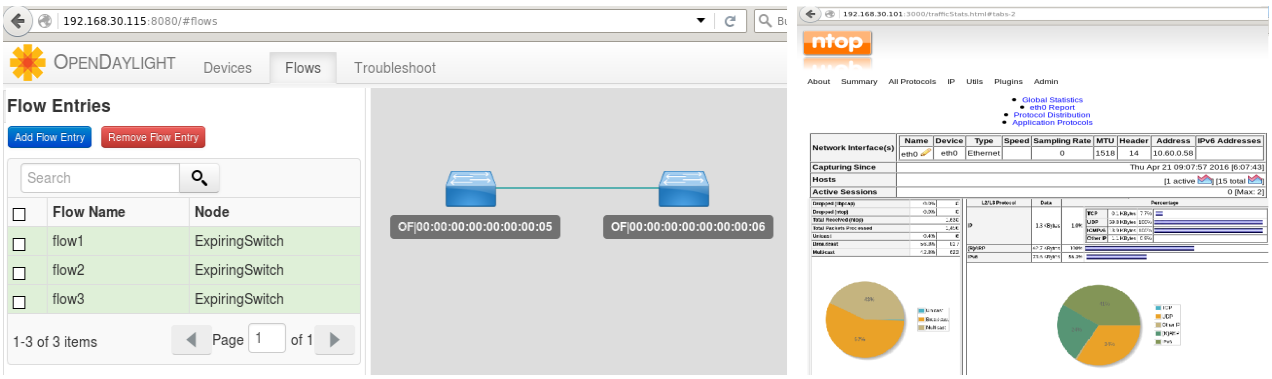


Figure 11.9: a) vSDN controller view, b) DPI statistics.

Figures 11.7 and 11.8 shows the network traffic capture for both validations (Slice provisioning and DPI use case). In Figure 11.9-a, it is shown the vSDN controller view (GUI) of the VTN and, in Figure 11.9-b, it is shown the traffic statistics gathered by the DPI. Finally, Table 11.1 recap the setup delay for the different experimental validations.

	Setup Delay (s)
Create Slice	22,075
Tenant session	0,969
vMANO VM instance	16,191
VTN Setup	5,034
Deploy VNF (DPI)	17,333
VNF Forwarding	3,196

Table 11.1: Experimental setup delays

11.3.2 Use case II: Deployment of virtual Mobile Network Operator (vMNO)

The Virtual Operator use case can be defined as the ability of partitioning the physical network infrastructure for the deployment of multi-tenant, application specific and customized virtual infrastructures (VIs). Each VI will have its own control plane to enable custom provisioning of the network services. This use case foreseen a scenario where a provider operator, which owns a physical infrastructure, is willing to resell it to other client operators (virtual operators).

The virtual operator use case fits with the new challenges faced by mobile network operators (MNOs) due to the drastic growth of mobile traffic. This traffic growth is pushing MNOs to invest in their backhaul network to cope with such demands. Typically MNOs deploy new dedicated network appliances for both control and data planes which are generally over-dimensioned, resulting into non-efficient cost-efficient strategy in terms of both CapEx and OpEx. Virtualization of network functions (NFV) and infrastructure (SDN) are appealing to be more scalable, cost-efficient and flexible for MNO deployment, in particular, in the backhaul infrastructure.

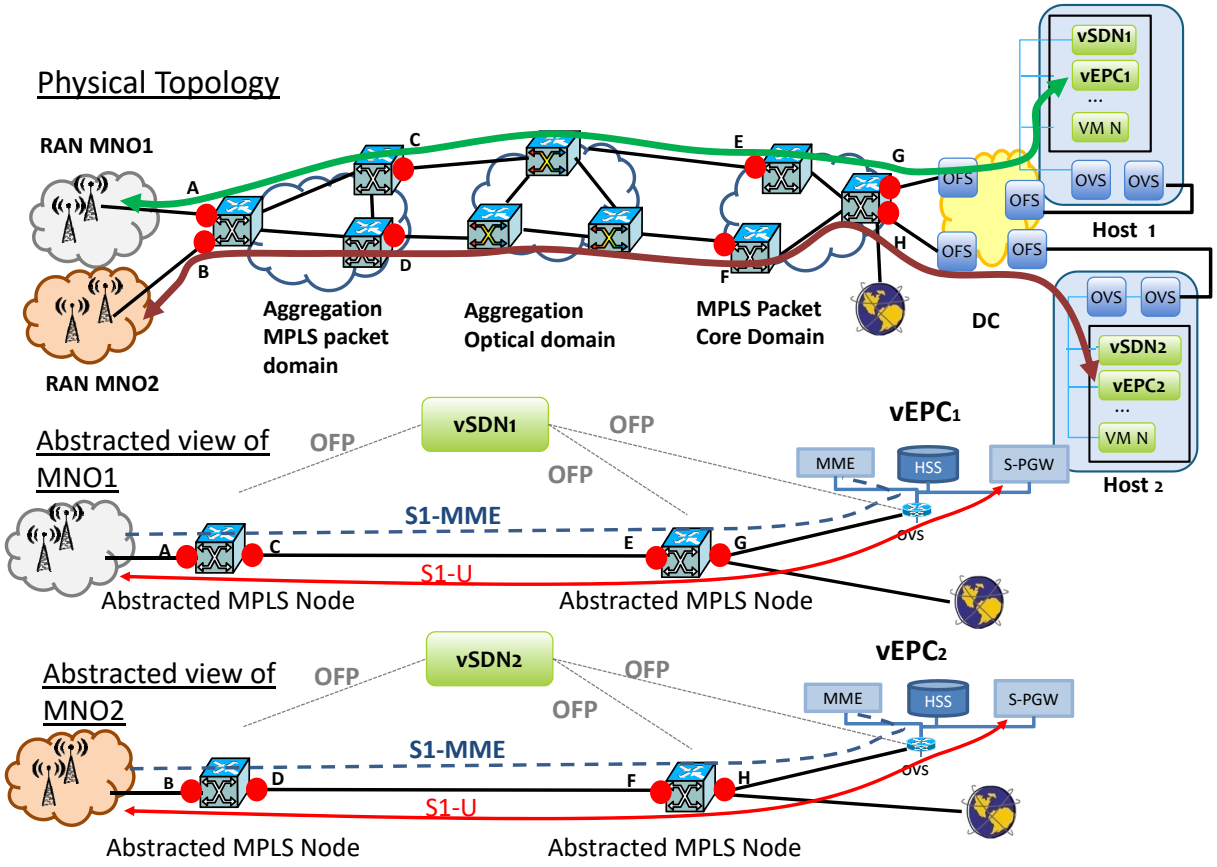


Figure 11.10: Physical multi-layer aggregation network connecting RANs and DCs and abstracted view of the backhaul networks per MNO.

We assume a scenario (Figure 11.10) where that a number of MNOs owning their radio area network (RAN) are connected to a common physical multi-layer (packet and optical) aggregation infrastructure. This common and physical infrastructure is partitioned to compose individual virtual backhaul tenants on top of it. Furthermore, the MNO Evolve Packet Core (EPC) functions are as well virtualized into the cloud connected to the aggregation network.

The physical multi-layer (packet and optical) aggregation infrastructure, as it has been extensively discussed, may consist on multiple domains and technologies, thus the proposed solution in STRAUSS for multi-domain network virtualization is based on the MSO (chapter 5) and MNH entities. The MSO provides E2E network orchestration while the MNH, placed on top of the MSO architecture (Figure 11.11), provides abstracted VIs that can be controlled through SDN controllers by individual customers. The SDN controller instance can be offered as a Virtualized Network Function (VNF) in the cloud. In the proposed architecture the NFV orchestrator is responsible of the deployment of VNFs on top of a common cloud and network platform (NFV infrastructure, NFVI).

The Cloud and Network Orchestrator (VIM) handles the coordination and management of cloud resources (virtual machines, VM) and network resources in the multi-layer aggregation infrastructure. Hence, it provides a common framework for a cloud and network operating system towards deploying the MNO virtual backhaul and vEPC function. For the backhaul service the MSO dynamically set up packet MPLS tunnels for backhauling upcoming mobile LTE signaling and data bearers (i.e., S1-MME and S1-U interfaces) between the RAN and vEPC. The interfaces for E2E network orchestration

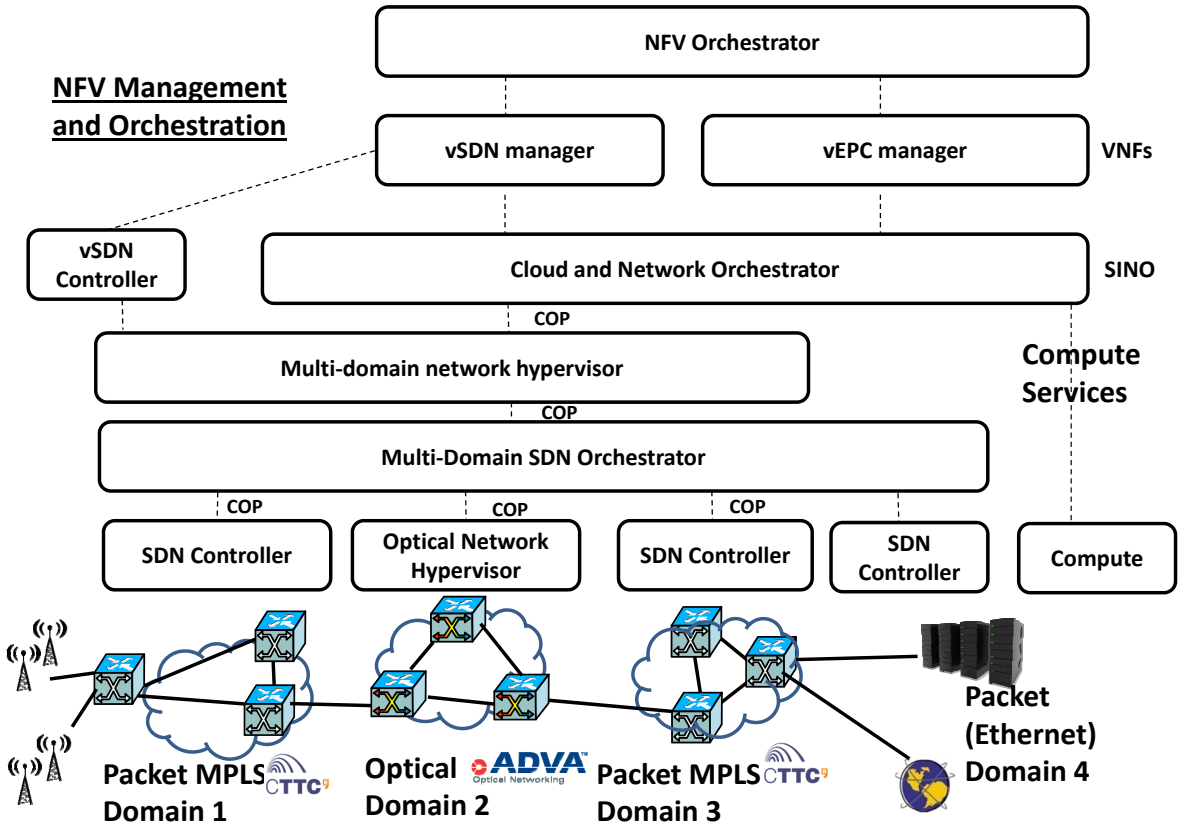


Figure 11.11: SDN/NFV orchestration architecture providing MNO backhaul virtual networks.

between SINO, MNH and MSO is implemented using COP.

Figure 11.12 shows the workflow between the involved functional blocks of the SDN/NFV orchestrator to manage the creation of an SDN-controlled virtual backhaul and the corresponding vEPC.

In Step 1, the NFV orchestrator to request the provisioning of the vSDN controller (for the virtual backhaul) and the vEPC. This is handled by the corresponding VNF managers sending requests to the Compute controller of VMs with the respective implementation (image) of the VNFs (vSDN and vEPC). Next, in step 2, the MNO virtual backhaul and the connectivity for the created vSDN controller to configure such an infrastructure are deployed. To do that, the MNH receives the request and it sends a COP/Call request to the MSO to provide an MPLS E2E service between the MNO RAN and the vEPC. Then the MSO orchestrate a multi-layer service provisioning generates a bidirectional L0 optical connection to support the MPLS tunnel which transport the backhaul service.

Once the virtual backhaul connectivity is ready, this is notified to the NFV orchestrator, and at that time, the vSDN has a view of the virtual packet backhaul used to transport LTE bearers between the RAN and the vEPC. The complete capture of the experimental control traffic for setting up the VNFs and the virtual backhaul network is depicted in Figure 11.13.

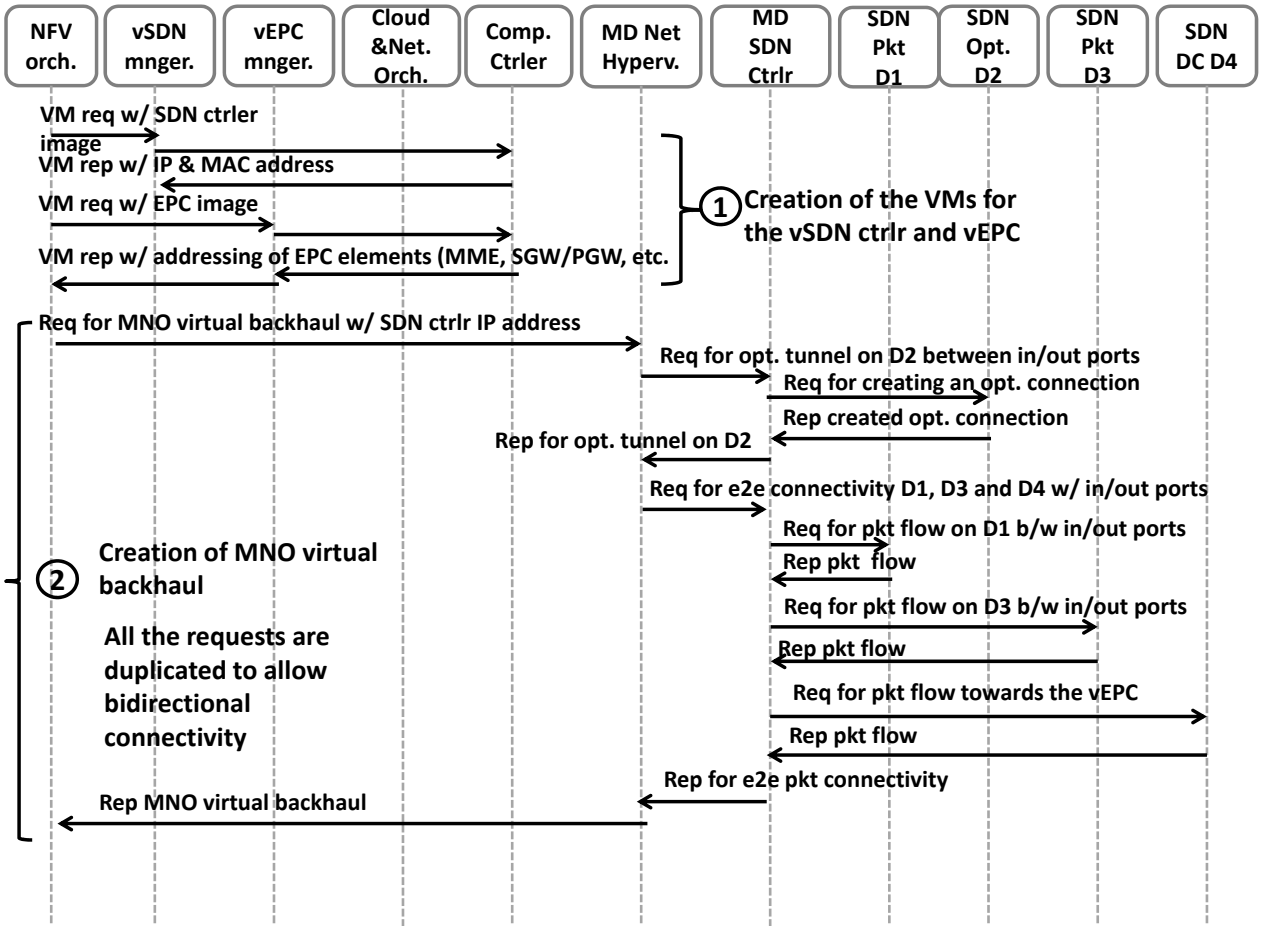


Figure 11.12: Workflow for provisioning MNO virtual backhaul network and VNFs.

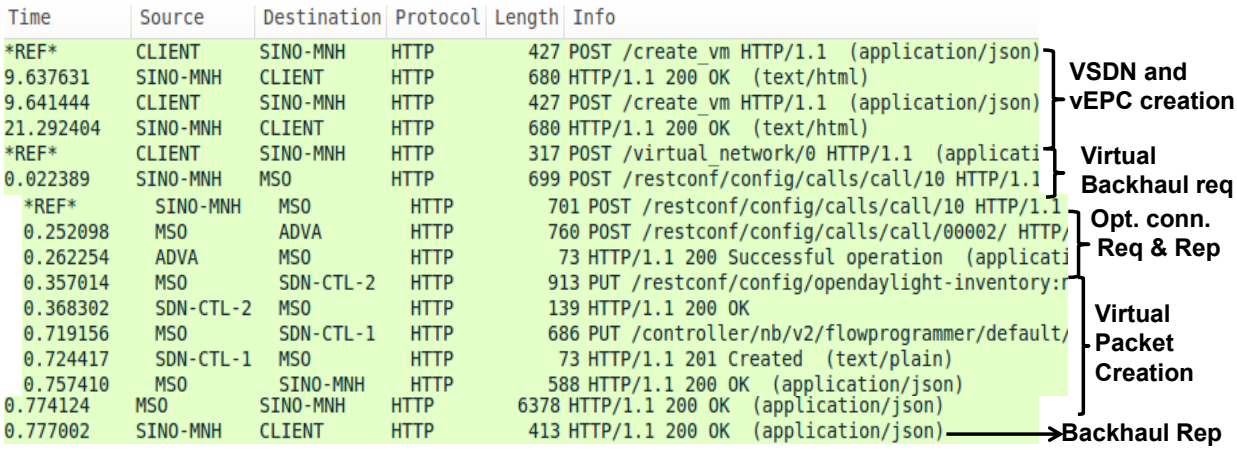


Figure 11.13: Capture of the experimental control messages for setting up the VNFs and virtual backhaul network.

11.4 Conclusions

This chapter presented the design and experimental validation of the proposed 5G network slicing architecture for 5G infrastructures with distributed cloud and multi-domain networks.

The network slicing concept has been introduced in detail, describing the different virtualization operations needed to provided a fully operational 5G Network Slice for a dedicated tenant over geographically DCs interconnected through a multi-domain, multi-layer WAN.

Finally, to validate the architecture proposed, we have presented the results of two different use cases realized through our proposed 5G network slicing architecture over the CTTC ADRENALINE Testbed: the creation and operation of a 5G Network Slice together with the deployment of a Deep Packet Inspection network service; and the deployment of a virtual Mobile Network Operator (vMNO) use case.

This chapter describe the result of many years of research on SDN and NFV technologies which have led into the different components of which the 5G network slicing solution proposed relays on. As next steps, the evaluation of the scalability of the solution along with the introduction of security requirements will be two key drivers to be analyzed in order to push the proposed solution into real network deployments.

Chapter 12

Cascading of tenant SDN and cloud controllers for 5G network slicing

The main contribution of this last chapter was extending the VIMaP architecture to enable the cascading of tenants SDN orchestrators (Chapter 5), cloud and VIMAP controllers (Chapters 9 and 10) and MANO instances (Chapter 11) in a multi-tenant scenario. Moreover, we have also extended the Network Slicing architecture presented in Chapter 11 in order to support the proposed cascading of resources. We have extended the VIMaP in order to support Transport API [82] and Openstack APIs [50] [83] as the NBI to enable the cascading of tenants SDN and cloud orchestrators, which allows to provide per-tenant network slices. As depicted in Figure 12.1, the proposed architecture can be applied recursively, and to enable that tenants can offer part of the virtual network and cloud resources allocated to their slice to other tenants, by cascading the VIM components.

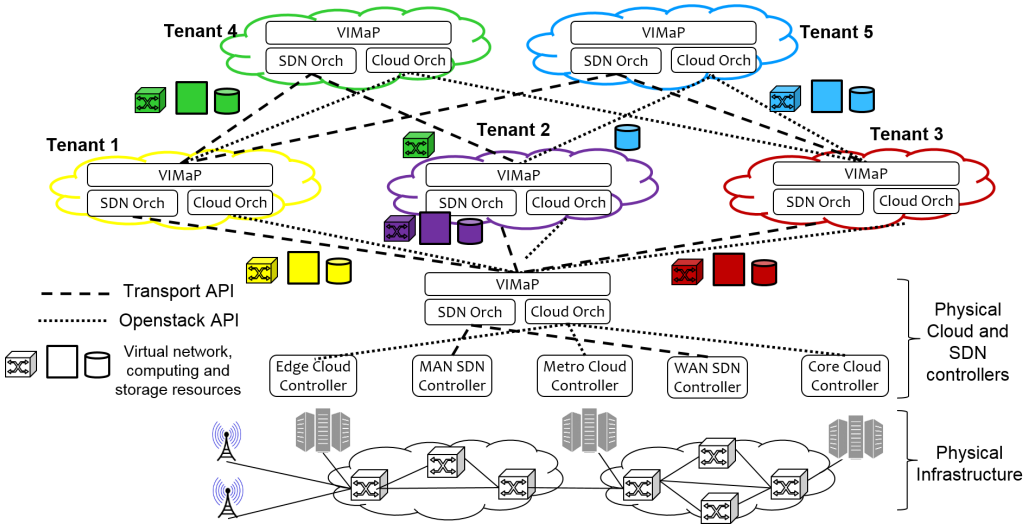


Figure 12.1: Cascading of SDN orchestrators, cloud orchestrators and VIMs for multi-tenant network slicing

12.1 Cloud and Network Cascading architecture for 5G Network Slicing

Virtual resources are logical partition and aggregation of the physical infrastructure. These logical representations can be exposed through well-designed APIs in the same manner that are physical resources, allowing the cascading of control components. This in one of the key concepts intended to be exploited by the proposed architecture. Therefore, the use of standard APIs plays a crucial role to achieve the proposed concept. The Transport API provides a standard interface for the orchestration of heterogeneous networks. On the other hand, the OpenStack APIs have become the open-source de-facto standard for cloud computing. By using the same interfaces as a Northbound Interface (NBI) and Southbound Interface (SBI) of the components of the proposed architecture, they can be stacked recursively to create dedicated slices for different tenants.

Bearing this in mind, the proposed architecture (Figure 12.2) is supported by the following main building blocks: the SDN orchestrator, the cloud orchestrator, the VIMaP which conform the VIM; and the network slicing service orchestrator (SLICE-O) (Figure 12.2).

The SDN Orchestrator is responsible for providing network discovery, programmability and traffic engineering functionalities over the underlying heterogeneous network resources. The Cloud Orchestrator provides computing, storage and networking functions over datacenters, which may be located in different sites and provided by independent cloud controllers. The VIMaP (Figure 12.3) coordinates the Cloud and Network by means of correlating the resources information from both domains and performing efficient allocation policies based on the resources' status. Finally, the SLICE-O is responsible for the dynamic lifecycle management (provisioning, modification and deletion) of not only the requested network slices but also of the virtualized MANO (vMANO) instance associated to each tenant. Virtualized MANO instances enable tenants to have full SDN/NFV control of all virtual resources (network, cloud, and VNFs) assigned to its slice as if they were real. Each VIMaP can slice, upon request of the SLICE-O, its underlying resources by providing dedicated VTN and virtual cloud resources to another tenant VIMaP. In order to provide these resources, the VIMaP is able to handle two contexts: provider and virtual. The provider context provides control and management for the underlying network resources, including network and cloud infrastructure managers, which are able to handle underlying heterogeneous resources (virtualized or not) using Transport API and OpenStack

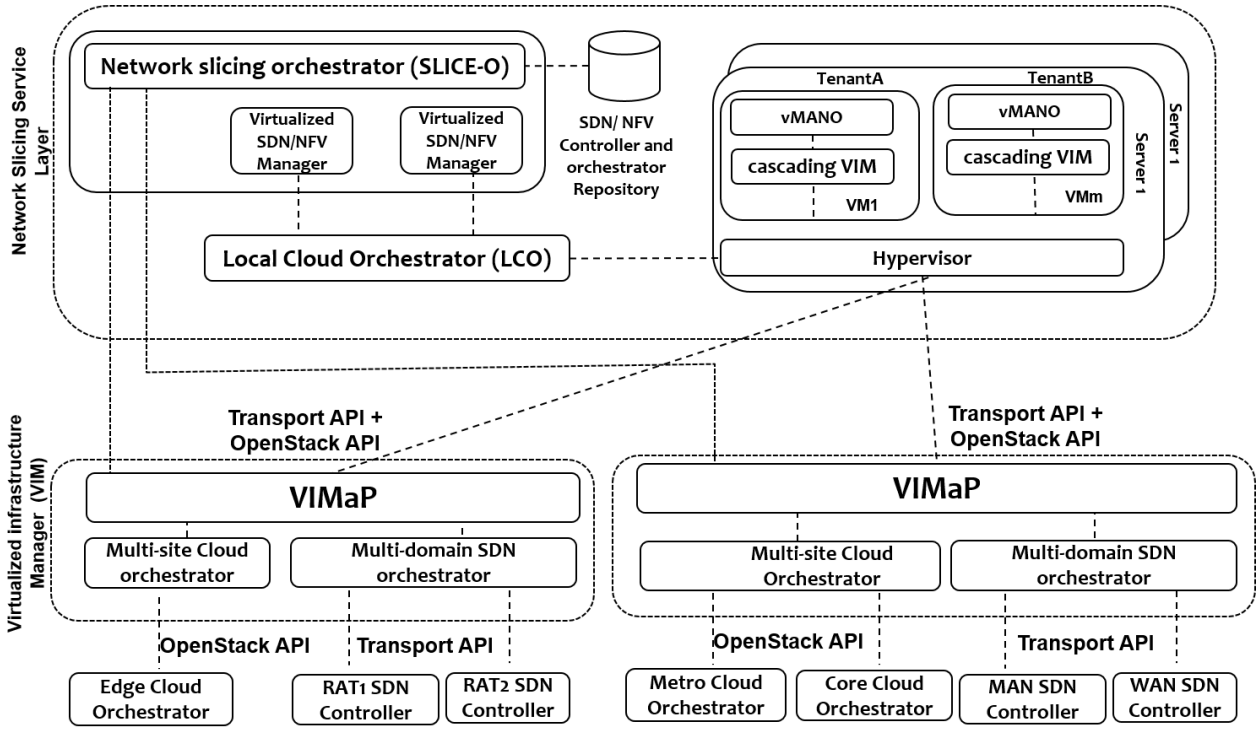


Figure 12.2: Network Slicing architecture.

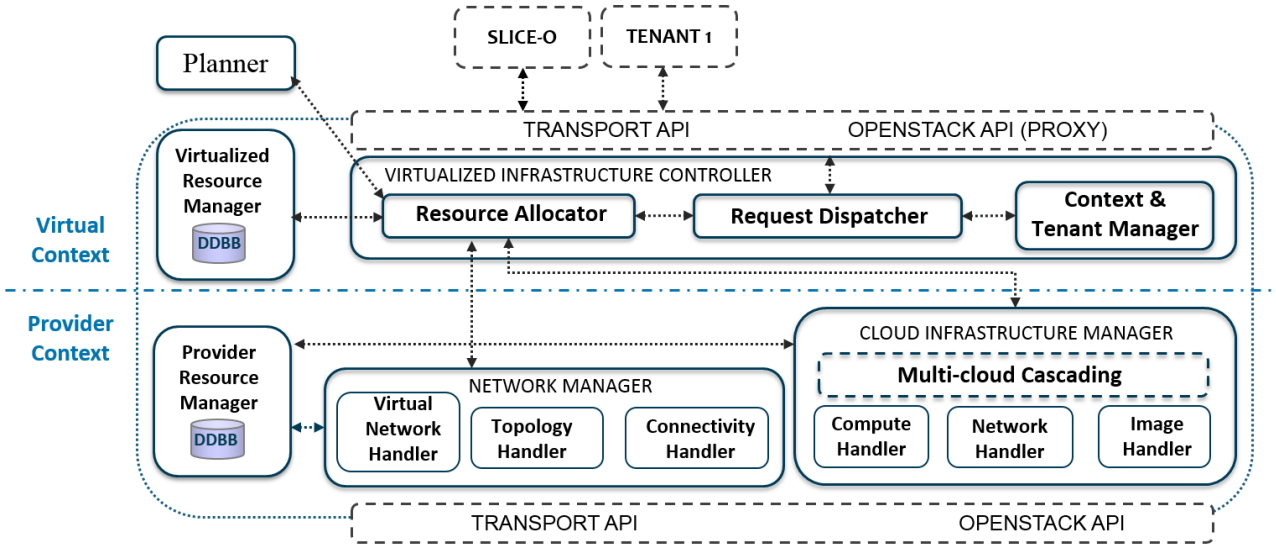


Figure 12.3: VIMaP extended architecture.

API. The virtualized infrastructure controller is responsible for mapping the requested virtual context upon the provider one. It is also responsible for providing the allocated virtual resources to the cascading vVIM or vMANO, depending on the composed cascade. To do so, it exposes to each tenant its virtual context resources through the Transport API and OpenStack API

12.2 Experimental validation and results

The proposed architecture has been validated in the cloud-computing platform and transport network of the CTTC ADRENALINE Testbed. The IT infrastructure consists of a local cloud (LCO) for the deployment of vMANO and vVIM, for network slicing, and different cloud sites as provider infrastructure, all controlled with Openstack Liberty cloud computing software. Two independent cloud sites (OpenStack) are geographically distributed across a SDN multilayer transport network composed by three domains: two intra-DC networks, consisting on OpenFlow switches deployed on COTS hardware and using OpenVSwitch (OVS), and a metro-core optical WSON with 2 ROADMs and 2 OXCs based on GMPLS distributed control. Per-domain SDN controllers control the electrical and optical domains are orchestrated by a Multi-domain SDN Orchestrator. The multi-domain SDN orchestrator (MSO) and VIMaP entities have been mostly implemented in Python. For the current implementation the Control Orchestration Protocol has been employed. Following, we describe the message exchange workflows for provisioning and operating a network slice and its validation in our Testbed.

12.2.1 Network Slice provisioning

The SLICE-O is responsible for creating the network slice upon demand. The workflow (Figure 12.4) is as follows, firstly, it requests the creation of a new tenant towards the VIMaP which subsequently duplicates the tenant creation on each of the cloud sites through its corresponding cloud orchestrator. Then, it creates the necessary vMANO and vVIM instances in a customized VNF instantiated in the LCO. Due to resources constrains, we employed one of the two cloud sites orchestrator controlled by the provider VIMaP as LCO. Finally, it provisions a dedicated VTN interconnecting the two cloud sites intra-DC networks points of presence (PoPs), through MPLS tunnels, across the metro-core segment. Figure 12.5) shows the traffic capture of the network slice provisioning workflow.

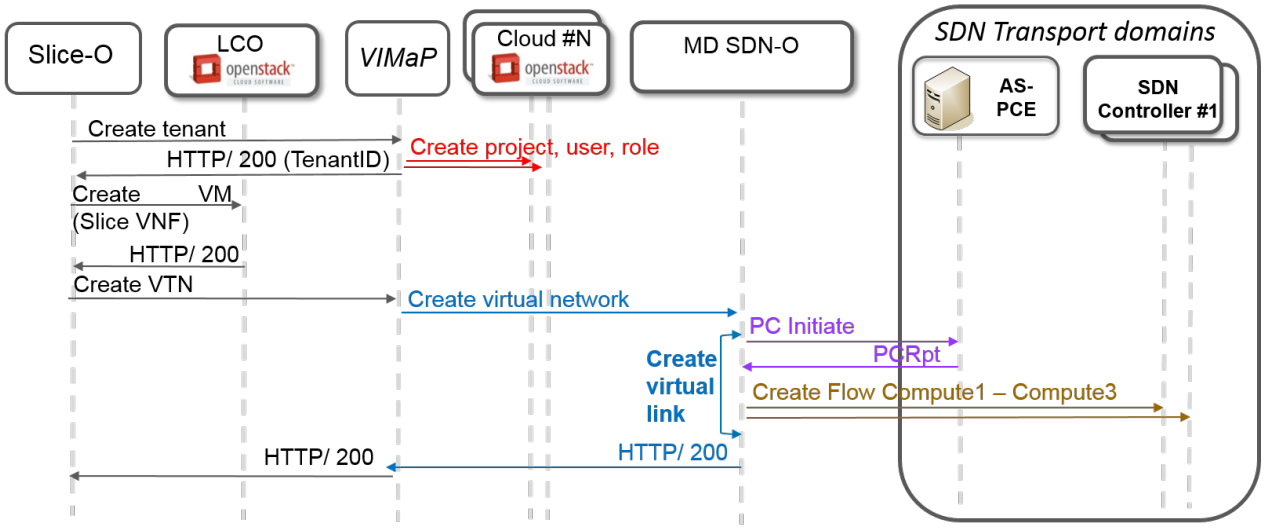


Figure 12.4: 5G Slice provisioning workflow

12.2.2 Network Slice operation

Once created, the tenant vMANO is able to allocate network services on top of the assigned resources. Requests for networks, compute and connectivity resources are cascaded, resulting in the necessary resource allocation. To validate this operation, the vMANO request of the creation of a new subnet

```
SLICE-O  VIMAP          POST /vim/slices/slice/slice1 HTTP/1.1
VIMAP    CLOUD SITE 1   POST /v3/auth/tokens HTTP/1.1
VIMAP    CLOUD SITE 1   POST /v3/projects HTTP/1.1
VIMAP    CLOUD SITE 1   POST /v3/users HTTP/1.1
VIMAP    CLOUD SITE 1   POST /v3/roles HTTP/1.1
VIMAP    CLOUD SITE 2   POST /v3/auth/tokens HTTP/1.
VIMAP    CLOUD SITE 2   POST /v3/projects HTTP/1.1
VIMAP    CLOUD SITE 2   POST /v3/users HTTP/1.1
VIMAP    CLOUD SITE 2   POST /v3/roles HTTP/1.1
VIMAP    CLOUD SITE 2   POST /v2.1/8e17 ... 9aa54660/servers
VIMAP    CLOUD SITE 2   POST /v2.1/8e17... 9aa54660/servers/actic
VIMAP    MD SDN-O       POST /restconf/config/virtual_networks
    MD SDN-O    SDN CTRL2   POST /stats/flowentry/add
       ...
    MD SDN-O    SDN CTRL1   POST /stats/flowentry/add
    MD SDN-O    AS-PCE      Unknown Message (12). PCEP_INITIATE
    AS-PCE      MD SDN-O    Unknown Message (10).
    MD SDN-O    AS-PCE      Unknown Message (12).
    AS-PCE      MD SDN-O    Unknown Message (10). PCEP_RPT
    MD SDN-O    SDN CTRL1   POST /stats/flowentry/add
       ...
    MD SDN-O    SDN CTRL2   POST /stats/flowentry/add
```

Figure 12.5: Network Slice provisioning traffic caputure.

(cloud domain) and two VMs, for further deployment of VNFs, towards the vVIM. It results in a cascaded network creation operation which triggers a duplicated network creation in the two provider cloud sites. Instead, the cascaded VM creation request, triggers resource allocation selection in the VIMaP among the provider cloud sites (12.6). Availability zones segmentation is used by VIMaP to select the appropriate cloud site to which forward the VM creation request. Please, notice the VM is allocated just once, in a single DC but a new logical representation is created in the Slice vVIM. To conclude, the cascading vVIM request a L3 end-to-end connectivity service (Call) between the newly created VMs based on its IP addresses. Figure 12.7) shows the traffic capture of the proposed workflow.

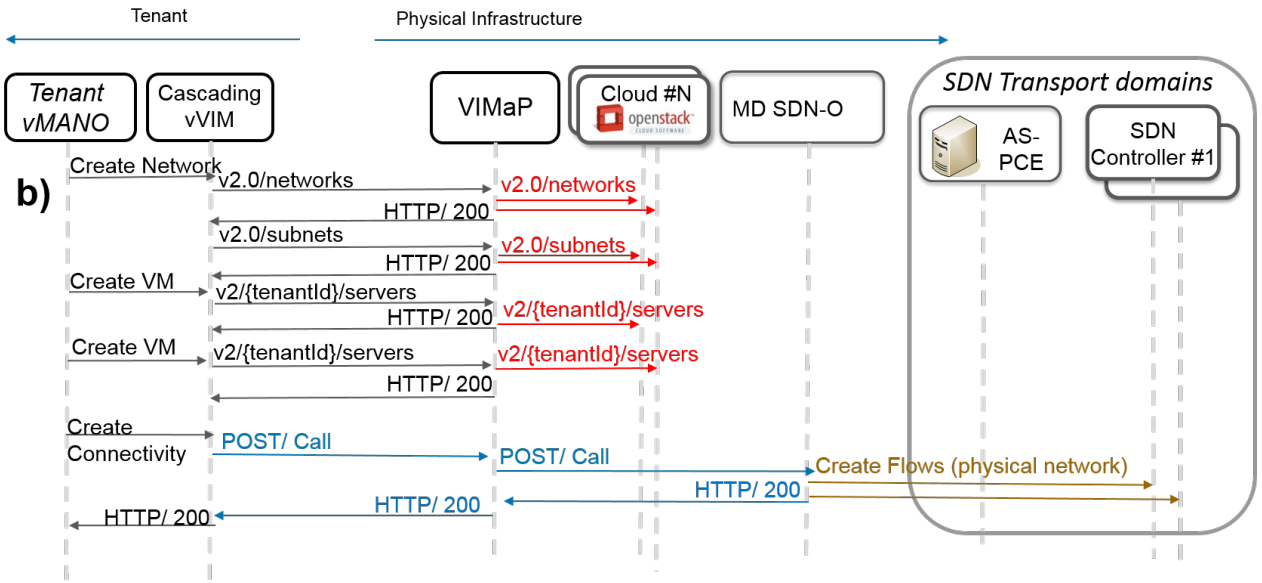


Figure 12.6: 5G Slice provisioning workflow

```
Tenant cVIM   VIMaP           508 POST /v2.0/networks HTTP/1.1  (application/json)
VIMAP         Cloud Site 1    426 POST /v2.0/networks HTTP/1.1  (application/json)
VIMAP         Cloud Site 2    426 POST /v2.0/networks HTTP/1.1  (application/json)
Tenant cVIM   VIMAP           391 GET /v2.0/networks HTTP/1.1
Tenant cVIM   VIMAP           581 POST /v2.0/subnets HTTP/1.1  (application/json)
VIMAP         Cloud Site 1    309 GET /v2.0/networks HTTP/1.1
VIMAP         Cloud Site 1    499 POST /v2.0/subnets HTTP/1.1  (application/json)
VIMAP         Cloud Site 2    309 GET /v2.0/networks HTTP/1.1
VIMAP         Cloud Site 2    499 POST /v2.0/subnets HTTP/1.1  (application/json)
Tenant cVIM   VIMAP           391 GET /v2.0/networks HTTP/1.1
Tenant cVIM   VIMAP           670 POST /v2.1/418364f6c4e741839e0c07a89dd127dc/servers
VIMAP         Cloud Site 1    588 POST /v2.1/4f8769fbc0e04483910700a89edd24db/servers
Tenant cVIM   VIMAP           670 POST /v2.1/418364f6c4e741839e0c07a89dd127dc/servers
VIMAP         Cloud Site 2    588 POST /v2.1/8e17e7b14d3242898de7cca19aa54660/servers
Tenant cVIM   VIMAP           261 GET /restconf/config/topologies HTTP/1.1
VIMAP         Tenant cVIM     352 HTTP/1.1 200 OK  (application/json)
Tenant cVIM   VIMAP           743 POST /restconf/config/calls/call/00002 HTTP/1.1
     VIMAP         MD SDN-O       760 POST /restconf/config/calls/call/1 HTTP/1.1
     VIMAP         MD SDN-O       760 POST /restconf/config/calls/call/2 HTTP/1.1
VIMAP         Tenant cVIM     352 HTTP/1.1 200 OK  (application/json)
```

Figure 12.7: Network Slice operation traffic caputure.

12.3 Conclusions

In this chapter it has been presented the design and experimental validation of the concept of resource cascading applied to both network and compute resources as an enabler for a 5G network slicing architecture for 5G infrastructures with distributed cloud and multi-domain networks.

Part V

Dissemination and Exploitation Results

Chapter 13

Scientific publications

This chapter presents the scientific publications published, accepted or submitted. Publications are classified in journals, international conferences, and collaborations, in inverse chronological order.

13.1 Journals

1. A. Mayoral, R. Vilalta, R. Muñoz, R. Casellas, R. Martínez, "SDN orchestration architectures and their integration with Cloud Computing Applications", in Special Issue on Advances in Path Computation Element, Optical Switching and Networking (OSN), Elsevier, February 2016.

2. R. Vilalta, A. Mayoral, R. Muñoz, R. Casellas, R. Martínez, "Multi-Tenant Transport Networks with SDN/NFV" , IEEE/OSA Journal of Lightwave Technology, Vol. 34, No. 8, January 2016.

3. A. Mayoral, R. Vilalta, R. Muñoz, R. Martínez, R. Casellas, V. López, "Need for a Transport API in 5G for Global Orchestration of Cloud and Networks Through a Virtualized Infrastructure Manager and Planner" , Journal of Optical Communications and Networking, Vol. 9, No. 1, pp. A55-A62, January 2017.

4. A. Mayoral, R. Vilalta, R. Muñoz, R. Casellas, R. Martínez, M. Svaluto Moreolo, J. M. Fabrega, A. Aguado, S. Yan, D. Simeonidou, J. M. Gran, V. López, P. Kaczmarek, R. Szwedowski, T. Szyrkowiec, A. Autenrieth, N. Yoshikane, T. Tsuritani, I. Morita, M. Shiraiwa, N. Wada, M. Nishihara, T. Tanaka, T. Takahara, J. C. Rasmussen, Y. Yoshida, K. Kitayama, "Control Orchestration Protocol: unified transport API for distributed cloud and network orchestration" , Journal of Optical Communications and Networking, Vol. 9, No. 2, pp. A216-A222, February 2017,

5. R. Martínez, A. Mayoral, R. Vilalta, R. Casellas, R. Muñoz, S. Pachnicke, T. Szyrkowiec, A. Autenrieth, "Integrated SDN/NFV Orchestration for the Dynamic Deployment of Mobile Virtual Backhaul Networks Over Multilayer (Packet/Optical) Aggregation Infrastructure" , Journal of Optical Communications and Networks, Vol. 9, No. 2, pp. A135 - A142, February 2017.

13.2 Conference papers

1. A. Mayoral, R. Vilalta, R. Muñoz, R. Casellas, R. Martínez, "Experimental validation of automatic lightpath establishment integrating OpenDaylight SDN controller and Active Stateful PCE within the ADRENALINE Testbed",in International Conference on Transparent Optical Networks (ICTON) 6-10 July 2014, Graz (Austria).

2. A. Mayoral, R. Vilalta, R. Muñoz, R. Casellas, R. Martínez, "Integrated IT and Network Orchestration Using OpenStack, OpenDaylight and Active Stateful PCE for Intra and Inter Data Center Connectivity", 40th European Conference on Optical Communication (ECOC 2014).

3. A. Mayoral, R. Vilalta, R. Muñoz, R. Casellas, R. Martínez, "Experimental Seamless Virtual Machine Migration Using an Integrated SDN IT and Network Orchestrator" , in Proc. of Optical Fiber Communication Conference (OFC), 2015.

4. A. Mayoral, R. Vilalta, R. Muñoz, R. Casellas, R. Martínez, "Performance analysis of SDN orchestration in the Cloud Computing Platform and Transport Network of the ADRENALINE Testbed",in International Conference on Transparent Optical Networks (ICTON) 5-9 July 2015, Budapest (Hungary).

5. A. Mayoral, R. Vilalta, R. Muñoz, R. Casellas, R. Martínez, "Traffic Engineering enforcement in multi-domain SDN orchestration of Multi-Layer (packet/optical) networks", 41th European Conference on Optical Communication (ECOC 2015).

6. R. Vilalta, A. Mayoral, R. Muñoz, R. Casellas, R. Martínez, "Hierarchical SDN Orchestration for Multi-technology Multi-domain Networks with Hierarchical ABNO" , in Proceedings of 41st European Conference on Optical Communication (ECOC 2015), 27 September-1 October 2015, Valencia (Spain).

7. R. Vilalta, A. Mayoral, R. Muñoz, R. Casellas, R. Martínez, The SDN/NFV Cloud Computing Platform and Transport Network of the ADRENALINE Testbed , in Proceedings of 1st IEEE Conference of Network Softwarization, 13-17 April 2015, London (UK).

8. R. Vilalta, V. López, A. Mayoral, N. Yoshikane, M. Ruffini, D. Siracusa, R. Martínez, T. Szyrkowiec, A. Autenrieth, S. Peng, R. Casellas, R. Nejabati, D. Simeonidou, X. Cao ,T. Tsuritani, I. Morita, J. P. Fernández-Palacios, and R. Muñoz, "The Need for a Control Orchestration Protocol in Research Projects on Optical Networking", in European Conference on Networks and Communications (EuCNC), Paris, France, June 29/July 2, 2015.

9. A. Mayoral, R. Vilalta, R. Muñoz, R. Casellas, R. Martínez, M. Svaluto Moreolo, J. M. Fabrega, S. Yan, A. Aguado, E. Hugues-Salas, S. Peng, F. Meng, Y. Shu, G. Zervas, R. Nejabati, D. Simeonidou, J. Manuel Gran, V. López, O. González de Dios, J.P Fernández-Palacios, P. Kaczmarek, R. Szwedowski, T. Szyrkowiec, A. Autenrieth, N. Yoshikane, X. Cao, T. Tsuritani, I. Morita, M. Shiraiwa, N. Wada, M. Nichihara, T. Tanaka, T. Takahara, J. C. Rasmussen, Y. Yoshida, K. Kitayama, "First experimental demonstration of a distributed cloud and heterogeneous network orchestration with a common Transport API for E2E services with QoS" , in Proceedings of the Optical Fiber Communication Conference and Exhibition (OFC), 20-24 March 2016, Anaheim, California (USA).

10. R. Muñoz, A. Mayoral, R. Vilalta, R. Casellas, R. Martínez, V. López, "The Need for a Transport API in 5G networks: the Control Orchestration Protocol" , in Proceedings of the Optical Fiber Communication Conference and Exhibition (OFC), 20-24 March 2016, Anaheim, California (USA).

11. A. Mayoral, R. Vilalta, R. Casellas, R. Martínez, R. Muñoz, "Multi-tenant 5G Network Slicing Architecture with Dynamic Deployment of Virtualized Tenant Management and Orchestration (MANO) Instances" , in Proceedings of 42nd European Conference and Exhibition on Optical Communications, 18-22 September 2016, Düsseldorf, Germany.

12. R. Vilalta, A. Mayoral, V. López, V. Uceda, R. Casellas, R. Martínez, R. Muñoz, A. Aguado, J. Marhuenda, R. Nejabati, D. Simeonidou, N. Yoshikane, T. Tsuritani, I. Morita, T. Szyrkowiec, A. Autenrieth, "Peer SDN Orchestration: End-to-End Connectivity Service Provisioning Through Multiple Administrative Domains" , in Proceedings of 42nd European Conference and Exhibition on Optical Communications (ECOC 2016), 18-22 September 2016 Düsseldorf (Germany).

13. A. Mayoral, R. Vilalta, R. Muñoz, R. Casellas, R. Martínez, V. López, "Cascading of tenant SDN and cloud controllers for 5G network slicing using Transport API and Openstack API" , in Proceedings of International Conference on Optical Fiber Communications (OFC), 19-23 March 2017, Los Angeles (USA).

13.3 Collaborations

1. R. Vilalta, R. Muñoz, A. Mayoral, R. Casellas, R. Martínez, D. López, V. López, Transport Network Function Virtualization , Journal of Lightwave Technology, Vol. 33, No. 5, pp. 1-8, April 2015.

2. R. Martínez, R. Vilalta, A. Mayoral, R. Casellas, R. Muñoz, "Experimental Validation of a SDN Orchestrator for the Automatic Provisioning of Fixed and Mobile Services" , in Proceedings of the 41st European Conference on Optical Communication (ECOC 2015), 27 September-1 October 2015, Valencia (Spain).

3. R. Vilalta, A. Mayoral, D. Pubill, R. Casellas, R. Martínez, J. Serra, C. Verikoukis, R. Muñoz, "End-to-End SDN Orchestration of IoT Services Using an SDN/NFV-enabled Edge Node" , in Proceedings of the Optical Fiber Communication Conference and Exhibition (OFC), 20-24 March 2016, Anaheim, California (USA).

4. J. M. Fabrega, M. Svaluto Moreolo, A. Mayoral, R. Vilalta, R. Casellas, R. Martínez, R. Muñoz, Y. Yoshida, K. I. Kitayama, Y. Kai, M. Nishihara, R. Okabe, T. Tanaka, T. Takahara, J. C. Rasmussen, N. Yoshikane, X. Cao, T. Tsuritani, I. Morita, K. Habel, R. Freund, V. López, A. Aguado, S. Yan, D. Simeonidou, T. Szyrkowiec, A. Autenrieth, M. Shiraiwa, Y. Awaji, N. Wada, "Demonstration of Adaptive SDN Orchestration: A Real-time Congestion-aware Services Provisioning over OFDM-based 400G OPS and Flexi-WDM OCS" , Journal of Lightwave Technology, Vol. 35, No. 3, pp. 1-7, February 2017.

5. V. López, I. Maor, K. Sethuraman, A. Mayoral, L. Ong, K. Mrówka, F. Marques, A. Sharma, F. Bosisio, O. González de Dios, O. Gerstel, F. Druesedau, R. Vilalta, H. Silva, A. Autenrieth, N. Borges, C. Liou, G. Cazzaniga, J.P. Fernández-Palacios, "E2E Transport API demonstration in hierarchical scenarios" , in Proceedings of SDN & NFV Demo Zone at International Conference on Optical Fiber Communications (OFC), 19-23 March 2017, Los Angeles (USA).

6. J. Mangues, J. Núñez, R. Casellas, A. Mayoral, J. Baranda, J. Xavier Salvat, A. García-Saavedra, R. Vilalta, I. Pascual, X. Li, R. Martínez, R. Muñoz, "Experimental Evaluation of Hierarchical Control over Multi-Domain Wireless/Optical Networks" , in Proceedings of the 26th edition of European Conference on Networks and Communications (EUCNC'17), 12-15 June 2017, Oulu (Finland).

Chapter 14

International, European and national R&D projects and standardization activities

Some experimental results obtained in this PhD thesis are used in several National, European and International R&D projects in optical networking. This chapter introduces these R&D projects and how this PhD thesis is related with them.

14.1 International R&D projects

14.1.1 STRAUSS - Scalable and efficient orchestration of Ethernet services using software-defined and flexible optical networks

14.1.1.1 Project abstract

The STRAUSS project aims to define a highly efficient and global (multi-domain) optical infrastructure for Ethernet transport, covering heterogeneous transport and network control plane technologies,

enabling an Ethernet ecosystem. It will design, implement and evaluate, via large-scale demonstrations, an advanced optical Ethernet transport architecture. The proposed architecture leverages on software defined networking principles, on optical network virtualization as well as on flexible optical circuit and packet switching technologies beyond 100 Gbps.

In particular, the STRAUSS project focuses on the integration and development of a) cost/energy efficient and extremely fast-performing switching nodes, based on variable-capacity and fixed-length optical packet switching technology for access and aggregation networks, and on flexi-grid DWDM optical circuit switching technology for long haul transport; b) highly integrated and scalable software defined optical transceivers supporting bandwidth variable multi-flows for flexible Ethernet transmission; c) a virtualization layer for dynamic and on-demand partitioning of the optical infrastructure offering virtual optical Ethernet transport networks (slices); d) legacy (e.g. GMPLS) and new (e.g. OpenFlow based) control plane approaches for control and management of virtual slices and finally e) a service and network orchestration layer for the interworking and coordination of heterogeneous control plane and transport technologies to offer end-to-end Ethernet transport services.

Outcomes of this project will be experimentally validated by means of demonstrations on large scale testbeds in EU & Japan. STRAUSS will provide technological roadmaps, technical approaches and deployment strategies aiming at shortening innovation and exploitation cycles in the area of future optical Ethernet transport networks for both academia and industry in EU and Japan.

14.1.1.2 PhD Thesis relationship

The work performed in this PhD Thesis has resulted in the design of the Multi-domain SDN Orchestration (MSO) architecture (Chapter 5) which addresses one of the main goals of STRAUSS project which was the E2E transport of Ethernet service over heterogeneous transport and network control plane technologies.

On Chapter 6 one of the main outcomes of STRAUSS project has been presented, the Control Orchestration Protocol (COP). In STRAUSS, me and my thesis coordinators we have lead the first experimental validations of the COP over a multi-partner international Testbed involving Japan and EU partners.

Finally, the Integrated IT and Network orchestration architecture and experiments presented in Chapter 9 contributed significantly to WP3 - Network Virtualization, Control Plane and Service Orchestration and WP4 - Integration and Demonstration, STRAUSS work packages.

14.2 European R&D projects

14.2.1 COMBO - COnvergence of fixed and Mobile BrOadband access/aggregation networks

14.2.1.1 Project abstract

COMBO project (http://www.ict-combo.eu/) proposes new integrated approaches for Fixed / Mobile Converged (FMC) broadband access / aggregation networks for different scenarios (dense urban, urban, rural). COMBO architectures are based on joint optimization of fixed and mobile access / aggregation networks around the innovative concept of Next Generation Point of Presence (NG-POP).

The main objective of COMBO will be to define, develop and technically assess network scenarios organized around the concept of Next Generation Point of Presence (NG-POP) and which embody the most promising directions for FMC at network level. So as to show experimentally the high potential of FMC for future networks, COMBO will develop proof of concept demonstrations including both a unified access / aggregation network and a Universal Access Gateway (UAG), the key elements to implement NG-POP concept.

The overall project targets of COMBO are:

- Define and develop FMC architectures for future networks, which will be technically assessed with respect to FMC use cases defined by the project;
- Demonstrate experimentally key FMC network features to show the feasibility of proposed architectures;
- Influence standardization bodies with respect to FMC architectures to push COMBO concepts.

14.2.1.2 PhD Thesis relationship

The contribution of this PhD Thesis includes the proposed Multi-domain SDN Orchestration (MSO) architecture (Chapter 5) which was employed for the automatic provisioning of Fixed and Mobile Services over the ADRENALINE Testbed within the COMBO projects activities. A novel SDN-based tunnelling approach based on MPLS tags was proposed to provide the transport network data plane for LTE communication between the eNodeBs and the Evolved Packet Core (EPC) located in the core of the network.

Moreover, the Network Slicing architecture presented in Chapter 11 was also employed for the validation of the virtual Mobile Network Operator (vMNO) use case.

14.2.2 5G-CROSSHAUL - The 5G Integrated fronthaul/backhaul

14.2.2.1 Project abstract

The 5G-Crosshaul project (http://5g-crosshaul.eu/) aims at developing a 5G integrated backhaul and fronthaul transport network enabling a flexible and software-defined reconfiguration of all networking elements in a multi-tenant and service-oriented unified management environment. The 5G-Crosshaul transport network envisioned will consist of high-capacity switches and heterogeneous transmission links (e.g., fibre or wireless optics, high-capacity copper, mmWave) interconnecting Remote Radio Heads, 5GPoAs (e.g., macro and small cells), cloud-processing units (mini data centers), and points-of-presence of the core networks of one or multiple service providers. This transport network will flexibly interconnect distributed 5G radio access and core network functions, hosted on in-network cloud nodes, through the implementation of: (i) a control infrastructure using a unified, abstract network model for control plane integration (Crosshaul Control Infrastructure, XCI); (ii) a unified data plane encompassing innovative high-capacity transmission technologies and novel deterministic-latency switch architectures (Crosshaul Packet Forwarding Element, XFE).

Demonstration and validation of the 5G-Crosshaul technology components developed will be integrated into a software-defined flexible and re-configurable 5G Test-bed in Berlin. Mobility-related 5G-Crosshaul experiments will be performed using Taiwans high- speed trains. 5G-Crosshaul KPI targets evaluated will include among others a 20% network capacity increase, latencies <1 ms and 30% TCO reduction.

14.2.2.2 PhD Thesis relationship

The contribution of this PhD Thesis to the 5G-Crosshaul project was two fold: on one hand the Hierarchical Multi-domain SDN Orchestration architecture included in Chapter 7 was proposed for the integration of different Radio Access Network (RAN) technologies such mmWave or IEEE 802.11ac, with optical DWDM core networks. The proposed architecture was validated over the integration of ADRENALINE and EXTREME CTTC Testbeds by employing a hierarchy of SDN control entities over which the MSO plays a crucial role on the E2E service orchestration.

On the other hand, the Virtual Infrastructure Manager and Planner (VIMaP) component presented in Chapter 10 for optimization of cloud and network resource allocation, has been developed in the scope of WP3 - Xhaul Control and Data planes activities of 5G-Crosshaul project.

14.3 Standardization activities

This section highlights the main contributions of this PhD. Thesis to the standardization bodies in the field of Optical Communications, SDN and NFV.

14.3.1 ONF Transport API

Ricard Vilalta co-director of this PhD.Thesis as Research Associated of the ONF invited Arturo Mayoral (the author) to participate in the Optical Transport Working Group (WG) for the development of the Transport SDN API initiative [82]. The motivation for this collaboration was the close relationship between the Control Orchestration Protocol (COP) developed during STRAUSS project as a research oriented activity, with the emergent standard proposed by the ONF.

The Open Transport project address the SDN control capabilities of transport networks of different types, including optical and wireless. The work carried out by the ONF in this project includes identifying and addressing different use cases, defining the application of SDN architecture and information modelling to transport networks, and defining standard SDN interfaces for transport networks, including transport controller APIs.

Since November 2015, Ricard Vilalta and Arturo Mayoral have been collaborating in different ONF Open Transport sub-projects (Snowmass-ONFOpenTransport, EAGLE-Open-Model-Profile-and-Tools and Englewood) where the experience acquire during the design and testing of the COP within STRAUSS was applied to the Transport API. In particular, the COP YANG-modeling tools and automatic code generation based on SWAGGER specification [68], were shared with the community as open source code in a public github repository within the EAGLE-Open-Model-Profile-and-Tools (`https://github.com/OpenNetworkingFoundation/EAGLE-Open-Model-Profile-and-Tools`).

14.3.2 Optical Internetworking Forum - OIF

In September 2016 a inter operability event was launched by the OIF to test the implementation of the Transport API between several optical vendors. Arturo Mayoral and Ricard Vilalta participated in this event as external technical experts to help in the coordination of the testing activities in Telefonica labs in Madrid. The results obtained in this activity were presented in the Optical Fibre Conference (OFC) 2017 as an interactive demo [84].

Part VI

Conclusions and Future Work

Chapter 15

Conclusions and future work

15.1 Conclusions

This PhD Thesis has discussed about a novel holistic approach for end-to-end Network and Cloud Computing resources orchestration applying Software Defined Networking concepts. The two main shortcommings transport networks suffer, are that they remain managed partitioned by technology and also by the equipment supplier, and, the control of these networks is tightly coupled with the switching hardware, implying a highly complex and extremely slow network operations. These two realities are the main motivations of introducing SDN as the technology enabler to achieve higher degrees of automation and control to multi-layer networks.

The contiguous objective being assessed during the work done in this Thesis, has been the design and demonstration of a multi-layer, multi-technology and multi-domain, cloud and transport network management architecture capable of offering end-to-end (E2E) service management and control and virtualization of network resources. The orchestration of network and cloud resources has been demonstrated critical to achieve the higher degrees of dynamicity and resource's utilization required by the emergence of the 5G paradigm.

The SDN orchestration architecture solution proposed, based on a clear separation between the data, control and management planes, and the abstraction of network technologies and control layers has been developed along the Chapters 4-8 corresponding to the second part of this Thesis (first part introduced the background and motivation and also the objectives of the current work). Firstly, in Chapter 4, the multi-layer network control was proposed to be solved by a single SDN Controller instance (based on open source distribution OpenDaylight) by integrating OpenFlow protocol for the IP layer and the PCEP protocol for the intent-based dynamic provisioning of Label Switched Paths (LSPs) in a distributed GMPLS control plane over the WDM network.

In chapter 5, a deeper discussion about the network orchestration problem in multi-domain networks was introduced. The proposed solution, the Multi-domain SDN Orchestration (MSO) architecture features different southbound plugins for the orchestration of different control technologies by applying the main SDN concepts (control and data plane separation, definition of open and standard APIs). The MSO has been evaluated in the Cloud Computing Platform and Transport Network of the ADRENALINE Testbed for the demonstration of two different use cases: (i) SDN orchestration of TE-aware multi-domain, multi-layer networks, and (ii) Automatic Provisioning of Fixed and Mobile Services. Moreover, a performance evaluation of the MSO has been carried out to measure the E2E connection provisioning and the topology discovery and transfer features. The results presented have shown a performance of an average of 300ms setup delay for single-layer requests and of 1350ms for multi-layer E2E connections. These results, although improvable through the use of more efficient core programming languages for the development of the MSO implementation, highlights the fact that service provisioning in multi-layer, multi-domain networks can be reduced from hours or days to seconds or milliseconds by the introduction of the proposed MSO.

In chapter 6, the Control Orchestration Protocol (COP) has been presented as a common and technology agnostic information model for the development of management interfaces, enabling the inter working of heterogeneous control plane paradigms. The COP abstracts a common set of control plane functions shared by different technologies and transport layers, such topology visualization, service and connection provisioning and path computation. The COP was experimentally demonstrated in a multi-partner international control and data plane testbed, by two proof of concepts: the dynamic provisioning of E2E network resources across the aforementioned network; and E2E QoS assurance, including per-domain and E2E QoS recovery based on data-plane monitoring. The value of the COP, as innovation enabler in the network management research field, has been demonstrated by the influence of the COP on the development of the ONF Transport API information models, which has become of one of the most mature standard information models employed in production multi-layer transport networks.

To conclude this part, the chapters 7 and 8, discussed the hierarchical vs peer approaches on the SDN Orchestration of multiple domains. The experimental results obtained suggests that certain levels of hierarchy can cope with the upcoming network heterogeneity from a multi-(technology, domain, and vendor) perspective. On the other hand, the peer approach, fostering the neighbor recursion and extensions introduced to the COP, can alleviate the scalability problems introduced in network orchestration scenarios with multiple domains.

The second objective of the PhD Thesis was to the integration of the management of IT/Cloud computing resources within the SDN Orchestration architecture. In chapter 9, an integrated IT and Network Orchestration architecture aligned with the NFV architectural framework proposed by the ETSI was presented. The context of application of this work is a distributed Data Center (DC) scenario, where computing pools of resources are geographically distributed across Wide Area Networks (WANs), introduces the need of holistic management of storage, compute and networking resources. The proposed solution is based on a novel software component named SDN IT and Network Orchestrator (SINO) and the benefits of introducing open standard interfaces (COP, OpenStack API) between the different components of the the architecture (SINO, MSO and OpenStack cloud controllers). The solution has been experimentally demonstrated in the Adrenaline Testbed 3, including the VM creation and migration across distributed cloud sites.

In chapter 10, the Virtual Infrastructure Manager and Planner (VIMaP) architecture which extends the SINO architecture introduced in the previous chapter with an resource deployment algorithm engine was presented. The main enhancements introduced by this new architecture: (a) the inclusion of a resource planner component responsible of run resource allocation algorithms for the optimal allocation of computing, storage and network resources for incoming infrastructure deployment requests;

and (b) the management of multiple tenants to perform context-aware IT and Network orchestration. The Virtual Machine Graphs (VMG) placement problem was mathematically defined and a baseline resource allocation heuristic algorithm, based on a Greedy approach for the selection of DCs and First Fit (FF) for the VM allocation, was also defined. The algorithm has been evaluated in a simulation environment based on the NSFNet14 reference scenario and the results compared with a simple RANDOM FIT heuristic.

The final part of this PhD Thesis which consist on Chapter 11 and Chapter 12 were focused on the network slicing concept. The VIMAP architecture proposed in chapter 9 is the base solution for the 5G Network Slicing architecture and the VIMaP and the MSO its main building blocks.

The third major contribution is the result of the previous two. With a converged cloud and network infrastructure controlled and operated jointly, the holistic view of the network allows the on-demand provisioning of network slices consisting of dedicated network and cloud resources over a distributed DC infrastructure interconnected by an optical transport network. The last chapters of this thesis discuss the management and orchestration of 5G network slices based over the control and management components designed and developed in the previous chapters. Thus, the design of one of the first network slicing architectures and the deployment of a fully operational 5G network slice in a real Testbed conforms one of the major contributions of this thesis, and completes a circle where the different architectures, software components and interfaces were demonstrated key building blocks for the forthcoming 5G era.

15.2 Future work

Although the main proposed objectives have been dealt with, this PhD Thesis is a mirror of the continuous work performed by research institutions in optical network virtualization. As it is an on-going research, future results are expected in the proposed topics.

In order to continue enhancing network management and control in multi-domain, multi-layer and multi-technology transport networks one of the topics which is getting a lot of attraction is the development of open and standard information models to be used in the Southbound Interface (SBI) of the SDN architectures discussed in this Thesis. Open initiatives such OpenROADM for the WDM and OTN transport segments, and OpenConfig also including IP, are interesting works which need to be closely followed in the following years as they might be easily introduced in the SDN Orchestration architectures proposed in this work. Specifically, the introduction of the so-called optical disaggregation, where the different components of an optical WDM network can be independently managed by a centralized controller following the SDN approach is getting a lot of attention by Telecom Operators as it brings new opportunities of opening a market which has been traditionally managed by few big equipment suppliers.

On the other hand, the advent of Network Function Virtualization (NFV) is driving the need of improving current management and orchestration (MANO) techniques to bring NFV to real deployments. The evolution of this field, leaded by the ETSI, is also an interesting opportunity for research. Also novel, network slicing approaches are yet to come, where traditional data plane virtualization technologies will play an important role for the resource partitioning and isolation between network tenants.

Furthermore, higher levels of network security, for such shared environments, is foreseen to be the key challenge the industry will need to assessed in order to bring 5G into a reality. In this line, recent research topics such Quantum Key Distribution (QKD) based Networks can also be of special relevance for 5G.

Bibliography

[1] D. King and A. Farrel, "A PCE-based architecture for application-based network operations," 2015.

[2] N. F. V. N. A. Framework, "Etsi gs nfv 002 v1.2.1 (2014-12)," 2014.

[3] H. Zimmermann, "Osi reference model–the iso model of architecture for open systems interconnection," *IEEE Transactions on communications*, vol. 28, no. 4, pp. 425–432, 1980.

[4] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, *et al.*, "B4: Experience with a globally-deployed software defined WAN," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 3–14, 2013.

[5] G. Recommendation, "694.1. Spectral grids for WDM applications: DWDM frequency grid," tech. rep., International Telecommunications Union, Tech. Rep, 2012.

[6] P. Roorda and B. Collings, "Evolution to colorless and directionless ROADM architectures," in *Optical Fiber communication/National Fiber Optic Engineers Conference, 2008. OFC/NFOEC 2008. Conference on*, pp. 1–3, IEEE, 2008.

[7] D. M. Marom, D. T. Neilson, D. S. Greywall, C.-S. Pai, N. R. Basavanhally, V. A. Aksyuk, D. O. López, F. Pardo, M. E. Simon, Y. Low, *et al.*, "Wavelength-selective 1 x k switches using free-space optics and mems micromirrors: theory, design, and implementation," *Journal of Lightwave Technology*, vol. 23, no. 4, p. 1620, 2005.

[8] S. Frisken, G. Baxter, D. Abakoumov, H. Zhou, I. Clarke, and S. Poole, "Flexible and grid-less wavelength selective switch using lcos technology," in *Optical Fiber Communication Conference and Exposition (OFC/NFOEC), 2011 and the National Fiber Optic Engineers Conference*, pp. 1–3, IEEE, 2011.

[9] I. Rec, "G. 872: Architecture of optical transport networks," *International Telecommunication Union, ITU-T*, 2001.

[10] M. Jinno, H. Takara, B. Kozicki, Y. Tsukishima, Y. Sone, and S. Matsuoka, "Spectrum-efficient and scalable elastic optical path network: architecture, benefits, and enabling technologies," *IEEE Communications Magazine*, vol. 47, no. 11, 2009.

[11] O. Gerstel, M. Jinno, A. Lord, and S. B. Yoo, "Elastic optical networking: A new dawn for the optical layer?," *IEEE Communications Magazine*, vol. 50, no. 2, 2012.

[12] G. Zhang, M. De Leenheer, A. Morea, and B. Mukherjee, "A survey on OFDM-based elastic core optical networking," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 65–87, 2013.

[13] G. Bosco, A. Carena, V. Curri, P. Poggiolini, and F. Forghieri, "Performance Limits of Nyquist-WDM and CO-OFDM in High-Speed PM-QPSK Systems," *IEEE Photonics Technology Letters*, vol. 22, pp. 1129–1131, Aug 2010.

[14] L. Velasco, M. Klinkowski, M. Ruiz, and J. Comellas, "Modeling the routing and spectrum allocation problem for flexgrid optical networks," *Photonic Network Communications*, vol. 24, no. 3, pp. 177–186, 2012.

[15] A. Mayoral, V. López, O. G. de Dios, and J. P. Fernandez-Palacios, "Migration steps toward flexi-grid networks," *Journal of Optical Communications and Networking*, vol. 6, no. 11, pp. 988–996, 2014.

[16] "Guidelines for creation, selection, and registration of an Autonomous System (AS)." RFC 1930, Mar. 1996.

[17] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, "Requirements for traffic engineering over mpls," tech. rep., 1999.

[18] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao, "Overview and principles of internet traffic engineering," tech. rep., 2002.

[19] I. Rec, "G. 805: Generic functional architecture of transport networks," *International Telecommunication Union, ITU-T*, 2000.

[20] A. Farrel and I. Bryskin, *GMPLS: architecture and applications*. Academic Press, 2005.

[21] A. Ayyangar, A. Farrel, E. Oki, A. Atlas, A. Dolganow, Y. Ikejiri, K. Kumaki, J. Vasseur, and J.-L. L. Roux, "Path Computation Element (PCE) Communication Protocol (PCEP)." RFC 5440, Mar. 2009.

[22] E. Crabbe, I. Minei, J. Medved, and R. Varga, "PCEP extensions for stateful PCE," tech. rep., draft-ietf-pce-stateful-pce-09 (work in progress), 2014.

[23] E. Crabbe, S. Sivabalan, I. Minei, and R. Varga, "PCEP Extensions for PCE-initiated LSP Setup in a Stateful PCE Model," tech. rep., 2013.

[24] R. Casellas, R. Martinez, R. Muñoz, L. Liu, T. Tsuritani, and I. Morita, "Dynamic provisioning via a stateful PCE with instantiation capabilities in GMPLS-controlled flexi-grid DWDM networks," in *Optical Communication (ECOC 2013), 39th European Conference and Exhibition on*, pp. 1–3, IET, 2013.

[25] T. Benson, A. Akella, and D. Maltz, "Unraveling the Complexity of Network Management," in *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'09, (Berkeley, CA, USA), pp. 335–348, USENIX Association, 2009.

[26] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[27] B. Pfaff, B. Lantz, B. Heller, *et al.*, "Openflow switch specification, version 1.3.0," tech. rep., 2012.

[28] S. Das, G. Parulkar, and N. McKeown, "Why OpenFlow/SDN can succeed where GMPLS failed," in *European Conference and Exhibition on Optical Communication*, pp. Tu–1, Optical Society of America, 2012.

[29] M. Channegowda, R. Nejabati, and D. Simeonidou, "Software-defined optical networks technology and infrastructure: Enabling software-defined optical network operations [invited]," *Journal of Optical Communications and Networking*, vol. 5, no. 10, pp. A274–A282, 2013.

[30] M. Bjorklund, "Special Report: OpenFlow and SDN State of the Union," tech. rep., SDNCentral LLC., 2016.

[31] L. Ong *et al.*, "Optical Transport Protocol Extensions," tech. rep., March 2015.

[32] S. Das, G. Parulkar, N. McKeown, P. Singh, D. Getachew, and L. Ong, "Packet and circuit network convergence with OpenFlow," in *Optical Fiber Communication Conference*, p. OTuG1, Optical Society of America, 2010.

[33] "OpenDaylight: An Open Source Community and Meritocracy for Software-Defined Networking (White Paper)," tech. rep., April 2014.

[34] R. T. Fielding, *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine, 2000.

[35] B. Pfaff, J. Pettit, T. Koponen, E. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, *et al.*, "The design and implementation of open vswitch," in *12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15)*, pp. 117–130, 2015.

[36] "Light switch - thin switching for bare metal physical and virtual switches." Available online at `http://go.bigswitch.com/rs/974-WXR-561/images/Switch%20Light%20Overview.pdf`.

[37] Y. Nakajima, W. Ishida, T. Fujita, T. Hirokazu, T. Hibi, H. Matsutahi, and K. Shimano, "High-performance vSwitch of the user by the user for the user," in *DPDK Summit*, 2014.

[38] Y. Yiakoumis, J. Schulz-Zander, and J. Zhu, "Pantou: Openflow 1.0 for openwrt (2011)."

[39] T. Bray, "The javascript object notation (json) data interchange format," tech. rep., 2017.

[40] T. Bray, J. Paoli, C. Sperberg-McQueen, Y. Mailer, and F. Yergeau, "Extensible markup language (xml) 1.0 5th edition, w3c recommendation, november 2008."

[41] R. Enns, M. Bjorklund, A. Bierman, and J. Schönwälder, "Network Configuration Protocol (NETCONF)." RFC 6241, June 2011.

[42] A. Bierman, M. Björklund, and K. Watsen, "RESTCONF Protocol." RFC 8040, Jan. 2017.

[43] M. Bjorklund, "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)." RFC 6020, Oct. 2010.

[44] P. Mell, T. Grance, *et al.*, "The nist definition of cloud computing," 2011.

[45] "VMware ESXi." Available online at `https://www.vmware.com/products/esxi-and-esx.html`.

[46] "linux KVM." Available online at `https://www.linux-kvm.org/page/Main_Page`.

[47] R. Kumar, K. Jain, H. Maharwal, N. Jain, and A. Dadhich, "Apache cloudstack: Open source infrastructure as a service cloud computing platform," *Proceedings of the International Journal of advancement in Engineering technology, Management and Applied Science*, pp. 111–116, 2014.

[48] A. Vogel, D. Griebler, C. A. Maron, C. Schepke, and L. G. Fernandes, "Private iaas clouds: a comparative analysis of opennebula, cloudstack and openstack," in *2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, pp. 672–679, IEEE, 2016.

[49] D. Milojičić, I. M. Llorente, and R. S. Montero, "Opennebula: A cloud management tool," *IEEE Internet Computing*, vol. 15, no. 2, pp. 11–14, 2011.

[50] O. Sefraoui, M. Aissaoui, and M. Eleuldj, "Openstack: toward an open-source solution for cloud computing," *International Journal of Computer Applications*, vol. 55, no. 3, pp. 38–42, 2012.

[51] Cisco, "Cisco Global Cloud Index: Forecast and Methodology, 20152020 (White Paper)," tech. rep., November 2016.

[52] R. Buyya, R. Ranjan, and R. Calheiros, "Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services," *Algorithms and architectures for parallel processing*, pp. 13–31, 2010.

[53] R. Nejabati, S. Peng, and D. Simeonidou, "Role of optical network infrastructure virtualization in data center connectivity and cloud computing," in *Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference (OFC/NFOEC), 2013*, pp. 1–3, IEEE, 2013.

[54] 5G PPP, "5G Vision: The 5G Infrastructure Public Private Partnership: the next generation of communication networks and services.," March 2015.

[55] 5G PPP, "5G empowering vertical industries.," February 2016.

[56] N. F. V. NFV and U. Cases, "Etsi gs nfv 001 v1. 1.1 (2013-10)," 2013.

[57] N. Alliance, "5G white paper," 2015.

[58] N. Alliance, "Description of network slicing concept," *NGMN 5G P*, vol. 1, 2016.

[59] E. Mannie, "Generalized Multi-Protocol Label Switching (GMPLS) Architecture." RFC 3945, Nov. 2004.

[60] R. Casellas, R. Muñoz, R. Martínez, R. Vilalta, L. Liu, T. Tsuritani, I. Morita, V. López, O. G. de Dios, and J. P. Fernández-Palacios, "SDN Orchestration of OpenFlow and GMPLS Flexi-Grid Networks With a Stateful Hierarchical PCE [Invited]," *J. Opt. Commun. Netw.*, vol. 7, pp. A106–A117, Jan 2015.

[61] Y.Yoshida, A. Maruta, K. Kitayama, M. Nishihara, T. Tanaka, T. Takahara, J. C. Rasmussen, N. Yoshikane, T. Tsuritani, I. Morita, S. Yan, Y. Shu, M. Channegowda, Y. Yan, B. Rofoee, E. Hugues-Salas, G. Saridis, G. Zervas, R. Nejabati, D. Simeonidou, R. Vilalta, R. Muñoz, R. Casellas, R. Martínez, M. S. Moreolo, J. M. Fabrega, A. Aguado, V. López, J. Marhuenda, O. G. de Dios, and J. P. Fernández-Palacios, "First international SDN-based Network Orchestration of Variable capacity OPS over Programmable Flexi-grid EON," in *PDP Optical Fiber Conference (OFC)*, Mar. 2014.

[62] R. Casellas, R. Martínez, R. Muñoz, L. Liu, T. Tsuritani, I. Morita, and M. Tsurusawa, "Dynamic virtual link mesh topology aggregation in multi-domain translucent WSON with hierarchical-PCE," *Optics express*, vol. 19, no. 26, pp. B611–B620, 2011.

[63] A. Farrel and G. Ash, "A Path Computation Element (PCE)-Based Architecture." RFC 4655, Aug. 2006.

[64] R. Martínez, R. Casellas, R. Muñoz, and R. Vilalta, "Experimental evaluation of delay-sensitive traffic routing in multi-layer (packet-optical) aggregation networks for fixed mobile convergence," in *39th European Conference and Exhibition on Optical Communication (ECOC 2013)*, pp. 1–3, Sept 2013.

[65] "Ryu OpenFlow controller." Available online at `http://osrg.github.io/ryu/`.

[66] N. Baldo, R. Martinez, P. Dini, R. Vilalta, M. Miozzo, R. Casellas, and R. Munoz, "A Testbed for Fixed Mobile Convergence Experimentation: ADRENALINE-LENA Integration," in *European Wireless 2014; 20th European Wireless Conference*, pp. 1–6, May 2014.

[67] 3GPP, "E-UTRA and E-UTRAN; Overall description - TS 36600," March 2012.

[68] S. Software, "OpenAPI Specification v2.0," 2014.

[69] R. Schmogrow, B. Nebendahl, M. Winter, A. Josten, D. Hillerkuss, S. Koenig, J. Meyer, M. Dreschmann, M.Huebner, C. Koos, J. Becker, W. Freude, and J. Leuthold, "Error Vector Magnitude as a Performance Measure for Advanced Modulation Formats," *IEEE Photonics Technology Letters*, vol. 24, pp. 61–63, 2012.

[70] "SDN Architecture," tech. rep., April 2016.

[71] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks." RFC 7348, Aug. 2014.

[72] D. T. Li, D. Farinacci, S. P. Hanks, D. Meyer, and P. S. Traina, "Generic Routing Encapsulation (GRE)." RFC 2784, Mar. 2000.

[73] ETSI Group Specification, "Network Function Virtualization (NFV): Architectural Framework," *ETSI GS NFV 002 v.1.1.1,*, 2013.

[74] F. Travostino, P. Daspit, L. Gommans, C. Jog, C. De Laat, J. Mambretti, I. Monga, B. Van Oudenaarde, S. Raghunath, and P. Y. Wang, "Seamless live migration of virtual machines over the MAN/WAN," *Future Generation Computer Systems*, vol. 22, no. 8, pp. 901–907, 2006.

[75] M. Alicherry and T. Lakshman, "Network aware resource allocation in distributed clouds," in *INFOCOM, 2012 Proceedings IEEE*, pp. 963–971, IEEE, 2012.

[76] B. Martini, M. Gharbaoui, and P. Castoldi, "Cross-Functional resource orchestration in optical telco clouds," in *2015 17th International Conference on Transparent Optical Networks (ICTON)*, pp. 1–5, July 2015.

[77] R. M. noz, A. Mayoral, R. Vilalta, R. Casellas, R. Martínez, and V. Lopez, "The Need for a Transport API in 5G networks: the Control Orchestration Protocol," in *Optical Fiber Communication Conference*, p. Th3K.4, Optical Society of America, 2016.

[78] R. Vilalta, R. M. noz, R. Casellas, R. Martínez, F. Francois, S. Peng, R. Nejabati, D. E. Simeonidou, N. Yoshikane, T. Tsuritani, I. Morita, V. López, T. Szyrkowiec, and A. Autenrieth, "Network Virtualization Controller for Abstraction and Control of OpenFlow-enabled Multi-tenant Multi-technology Transport Networks," in *Optical Fiber Communication Conference*, p. Th3J.6, Optical Society of America, 2015.

[79] R. Vilalta, R. M. noz, R. Casellas, and R. Martinez, "Dynamic virtual GMPLS-controlled WSON using a Resource Broker with a VNT Manager on the ADRENALINE testbed," *Opt. Express*, vol. 20, pp. 29149–29154, Dec 2012.

[80] E. C. Rosen and L. Andersson, "Framework for Layer 2 Virtual Private Networks (L2VPNs)." RFC 4664, Sept. 2006.

[81] J. Drake, W. Henderickx, A. Sajassi, R. Aggarwal, D. N. N. Bitar, A. Isaac, and J. Uttaro, "BGP MPLS-Based Ethernet VPN." RFC 7432, Feb. 2015.

[82] "Functional Requirements for Transport API ," tech. rep., June 2016.

[83] T. O. Foundation, "Openstack api documentation."

[84] V. Lopez, I. Maor, K. Sethuraman, A. M. L. de Lerma, L. Y. Ong, R. Szwedowski, F. Marques, A. Sharma, F. Bosisio, O. G. de dios, O. Gerstel, F. Druesedau, R. Vilalta, H. Silva, A. Autenrieth, N. Borges, C. Liou, G. Cazzaniga, and J. P. Fernandez-Palacios, "E2E Transport API demonstration in hierarchical scenarios," in *Optical Fiber Communication Conference*, p. Tu3L.4, Optical Society of America, 2017.

Appendix A

Control Orchestration Protocol specification

A.1 COP data model definition based on YANG

A.1.1 Call Service

```
module service-call {

    yang-version 1;

    namespace "http://ict-strauss.eu/cop/service-call";

    prefix s-call;
    organization "CTTC";

    description  "YANG version of the Common Orchestration Protocol (COP).";
```

```
    revision "2015-05-30" {
        description
        "Service call module for STRAUSS";
    }

    typedef qos_class_enumeration {
        type enumeration {
            enum gold {
                description "Gold QoS Class";
                value 1;
            }
            enum silver {
                description "Silver QoS Class";
                value 2;
            }
        }
    }

    typedef transport_layer_type_enumeration {
        type enumeration {
            enum dwdm_link {
                description "Setup a dwdm link";
                value 1;
            }
            enum ethernet {
                description "Setup an end to end ethernet path";
                value 2;
            }
            enum ethernet_broadcast {
                description "Setup an ethernet Flood";
                value 3;
            }
            enum mpls {
                description "Setup an MPLS path";
                value 4;
            }
        }
    }

    typedef direction_type_enumeration{
        type enumeration{
            enum unidir{
                description "Unidirectional service";
                value 1;
            }
            enum bidir{
                description "Bidirectional service";
                value 2;
            }
        }
    }

    grouping transport_layer-type{
        description "";
        leaf layer {
            type transport_layer_type_enumeration;
        }
        leaf layer_id {
            type string;
        }
        leaf direction{
```

```
            type direction_type_enumeration;
        }
    }

    typedef protocol-type{
        type enumeration{
            enum TCP;
            enum UDP;
            enum ARP;
            enum DHCP;
        }
    }

    typedef address-type{
        type enumeration{
            enum IPv4 {
                value 0;
            }
            enum IPv6 {
                value 1;
            }
            enum DatapathID {
                value 2;
            }
            enum MAC {
                value 3;
            }
        }
    }

    grouping label{
        description "";
        leaf label-type{
            type enumeration{
                enum GMPLS_FIXED {
                    value 0;
                }
                enum GMPLS_FLEXI {
                    value 1;
                }
            }
        }

        leaf label-value{
            type int32;
        }
    }

    grouping path-type{
        description "The Path represents the individual routes of a Connection.";
        leaf id{
            type string;
        }
        list topo_components{
            key "endpoint_id";
            uses endpoint;
        }
        leaf no_path{
            type boolean;
        }
        leaf multi_layer{
            type boolean;
```

```
        }
        container label{
            uses label;
        }
    }

    grouping match-rules{
        description "Match rules for call - OF match rules + extentions";
        leaf in_port{
            //type inet:port-number; // Switch input port.
            type string;
        }
        leaf in_phy_port{
            //type inet:port-number; // Switch physical input port.
            type string;
        }
        leaf metadata{
            type string; // Metadata passed between tables.
        }
        leaf eth_src{
            //type yang:mac-address; // Ethernet source address.
            type string;
        }
        leaf eth_dst{
            //type yang:mac-address; // Ethernet destination address.
            type string;
        }
        leaf eth_type{
            type int32; // Ethernet frame type.
        }
        leaf vlan_vid{
            type int32; // VLAN id.
        }
        leaf vlan_pcp{
            type int32; // VLAN priority.
        }
        leaf ip_dscp{
            type int32; // IP DSCP (6 bits in ToS field).
        }
        leaf ip_ecn{
            type int32; // IP ECN (2 bits in ToS field).
        }
        leaf ip_proto{
            type int32; // IP protocol.
        }
        leaf ipv4_src{
            //type inet:ip-address; // IPv4 source address.
            type string;
        }
        leaf ipv4_dst{
            //type inet:ip-address; // IPv4 destination address.
            type string;
        }
        leaf tcp_src{
            type int32; // TCP source port.
        }
        leaf tcp_dst{
            type int32; // TCP destination port.
        }
        leaf udp_src{
            type int32; // UDP source port.
        }
```

```
leaf udp_dst{
    type int32; // UDP destination port.
}
leaf sctp_src{
    type int32; // SCTP source port.
}
leaf sctp_dst{
    type int32; // SCTP destination port.
}
leaf icmpv4_type{
    type int32; // ICMP type.
}
leaf icmpv4_code{
    type int32; // ICMP code.
}
leaf arp_op{
    type int32; // ARP opcode.
}
leaf arp_spa{
    type int32; // ARP source IPv4 address.
}
leaf arp_tpa{
    type int32; // ARP target IPv4 address.
}
leaf arp_sha{
    type int32; // ARP source hardware address.
}
leaf arp_tha{
    type int32; // ARP target hardware address.
}
leaf ipv6_src{
    //type inet:ipv6-address; // IPv6 source address.
    type string;
}
leaf ipv6_dst{
    //type inet:ipv6-address; // IPv6 destination address.
    type string;
}
leaf ipv6_flabel{
    //type inet:ipv6-flow-label; // IPv6 Flow Label
    type string;
}
leaf icmpv6_type{
    type int32; // ICMPv6 type.
}
leaf icmpv6_code{
    type int32; // ICMPv6 code.
}
leaf ipv6_nd_target{
    type int32; // Target address for ND.
}
leaf ipv6_nd_sll{
    type int32; // Source link-layer for ND.
}
leaf ipv6_nd_tll{
    type int32; // Target link-layer for
}
leaf mpls_label{
    type int32; // MPLS label. */
}
leaf mpls_tc{
    type int32; // MPLS TC. */
```

```
        }
        leaf mpls_bos{
            type int32; // MPLS BoS bit. */
        }
        leaf pbb_isid{
            type int32; // PBB I-SID. */
        }
        leaf tunnel_id{
            type int32; // Logical Port Metadata. */
        }
        leaf ipv6_exthdr{
            type int32; // IPv6 Extension Header pseudo-field */
        }
        leaf experimental_gmpls_wson_label{
            type int32; // GMPLS-TE Label
        }
        leaf experimental_teid{
            type int64;// EXPERIMENTAL MATCH
        }
    }


    grouping endpoint {
        description "The End-Point represents the access to the forwarding (as
        Connection-End-Point) and/or adjacency (as Link-End-Point) function";

        leaf endpoint_id{
            type string;
            description "Name of the endpoint, for example host1-port1";
        }
        leaf router_id{
            type string;
        }
        leaf interface_id{
            type string;
        }
        /*leaf topology-component{
            type topo:end-ref;
        }*/
    }

    grouping traffic_params{
        description "Basic Traffic Parameters to be offered within a Call";
        leaf reserved_bandwidth {
            description "Reserved Bandwidth measured in Mb/s i.e. 10, 100, 1000 Mb/s";
            type int32;
        }
                leaf latency {
                        description "Connection latency measured in ms";
            type int32;
        }

        leaf estimated_PLR {
            description "Estimated packet loss ratio in %";
            type decimal64{
                fraction-digits 2;
            }
        }
        leaf OSNR {
            description "Optical Signal-to-noise ratio";
            type decimal64{
                fraction-digits 2;
```

```
        }
    }
    leaf qos_class {
        type qos_class_enumeration;
    }
}

grouping connection {
    description "The Connection represents an enabled potential for forwarding
    (including all circuit and packet forms) between two or more endpoints";

    leaf connection_id{
        type string;
    }
    container aEnd{
        uses endpoint;
    }
    container zEnd{
        uses endpoint;
    }
    container path{
        uses path-type;
    }
    container match {
        uses match-rules;
    }
    container traffic_params {
        uses traffic_params;
    }
    leaf controller_domain_id{
         type string;
    }
    container transport_layer{
        uses transport_layer-type;
    }
    leaf operStatus {
        description "Running status";
        config false;
        type enumeration {
            enum down {
                value 0;
                description "down";
            }
            enum up {
                value 1;
                description "up";
            }
        }
    }
}

grouping call {
    description "A call represents an intent-request for connectivity within a
    Forwarding-Domain between the endpoints. Call is distinct from the Connection
    that realizes the Call.";

    leaf call_id{
        type string;
    }
    container aEnd{
        uses endpoint;
    }
```

```
        container zEnd{
            uses endpoint;
        }
        container transport_layer{
            uses transport_layer-type;
        }
        leaf operStatus {
            description "Running status";
            type enumeration {
                enum down {
                    value 0;
                    description "down";
                }
                enum up {
                    value 1;
                    description "up";
                }
            }
        }
        container match{
            uses match-rules;
        }
        container traffic_params {
            uses traffic_params;
        }
        list connections{
            config false;
            key "connection_id";
            uses connection;
        }
    }

    notification update_call{
        uses call;
    }

    notification remove_call{
        uses call;
    }

    container calls{
        list call {
            key "call_id";
            uses call;
        }
    }
    container connections{
        list connection {
            key "connection_id";
            uses connection;
        }
    }

}
```

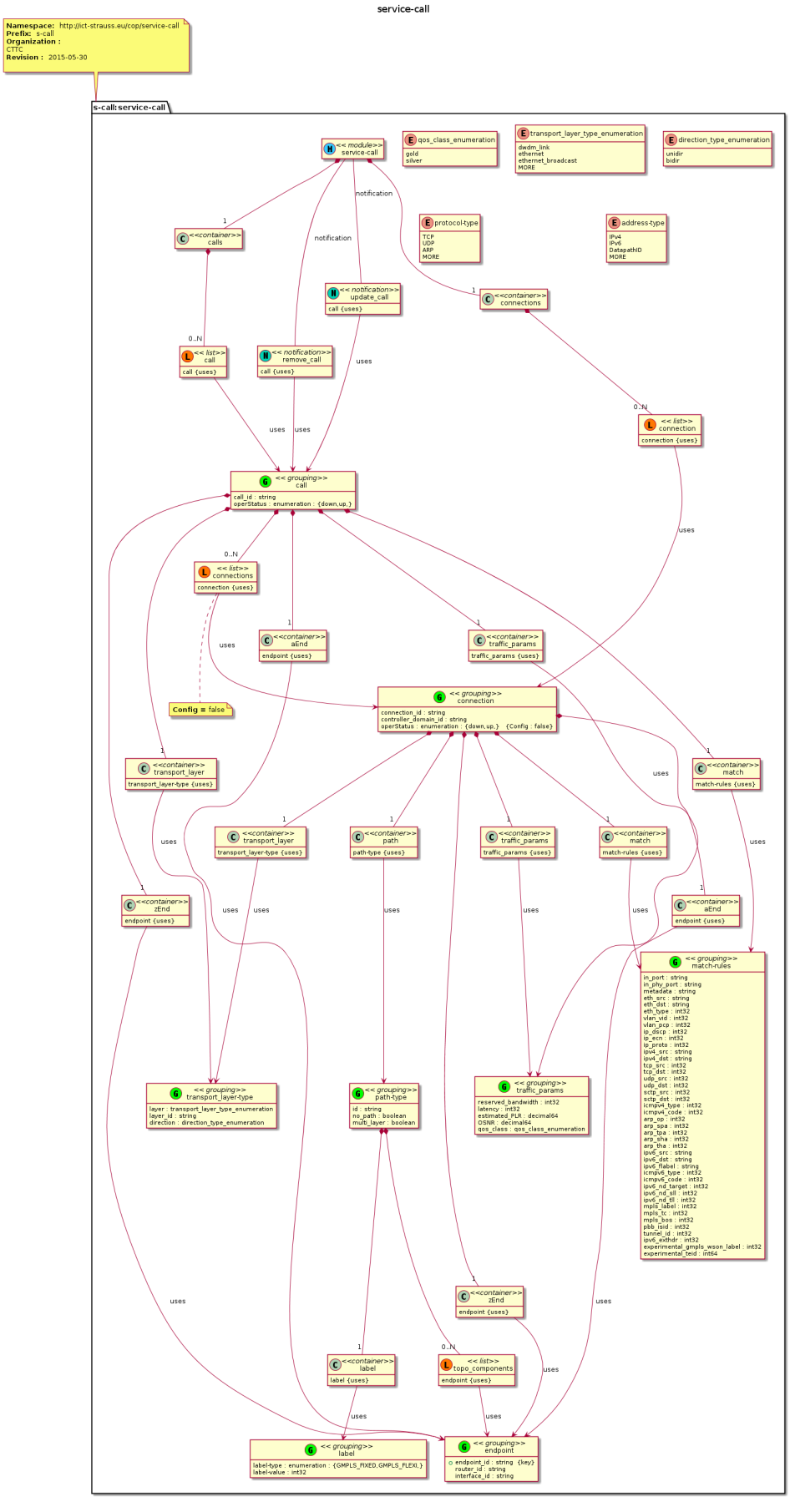


Figure A.1: COP Call service UML diagram

A.1.2 Topology Service

```
module service-topology {

    yang-version 1;

    namespace "http://ict-strauss.eu/cop/service-topology";

    prefix s-topology;

    organization "CTTC";

    description "YANG version of the Common Orchestration Protocol (COP).";

    revision "2015-05-30" {
        description
        "Service Topology module for STRAUSS";
    }

    typedef topology-ref {
        type leafref {
            path "/topologies/topology/topology_id";
        }
        description
            "A type for an absolute reference a topology instance.";
    }

    typedef edge_type_enumeration {
        type enumeration {
            enum dwdm_edge {
                description "DWDM link";
                value 1;
            }
            enum eth_edge {
                description "Ethernet link";
                value 2;
            }
        }
    }

    typedef switching_cap_enumeration {
        type enumeration {
            enum lsc {
                description "lsc sw_cap";
                value 1;
            }
            enum psc {
                description "psc sw_cap";
                value 2;
            }
        }
    }

    grouping topology{
        leaf topology_id{
            type string;
        }
        leaf-list underlay-topology {
            description "List of topologies from which this depends on.";
            type topology-ref;
```

```
            }
        list nodes{
            key "node_id";
            uses node;
        }
        list edges{
            key "edge_id";
            uses edge;
        }
    }

    grouping edge_end{
        leaf edge_end_id{
            type string;
        }
        leaf switching_cap{
            type switching_cap_enumeration;
        }
        leaf name{
            type string;
        }
        leaf peer_node_id{
            type string;
        }
    }

    grouping node{
        leaf node_id{
            type string;
        }
        leaf-list underlay-abstract-topology{
            description "List of topology_ids which are represented by this node.";
            type topology-ref;
        }
        leaf name{
            type string;
        }
        leaf domain{
            type string;
        }
        leaf nodetype{
            type string;
        }
        list edge_end{
            key "edge_end_id";
            uses edge_end;
        }
    }

    grouping edge{
        leaf edge_type{
            mandatory true;
                type edge_type_enumeration;
            }
        leaf edge_id{
            type string;
        }
        leaf name{
            type string;
        }
        leaf switching_cap{
            type string;
```

```
        }
        leaf metric{
            type string;
        }
        leaf max_resv_bw{
            type string;
        }
        leaf unreserv_bw{
            type string;
        }
        container source{
            config false;
            uses node;
        }
        container target{
            config false;
            uses node;
        }
        container local_ifid{
            config false;
            uses edge_end;
        }
        container remote_ifid{
            config false;
            uses edge_end;
        }
    }

    grouping dwdm_channel{
        leaf g694_id{
            type int32;
        }
        leaf state{
            type int32;
        }
    }

    grouping bitmap{
        leaf numChannels{
            type int16;
        }
        leaf-list arrayBits{
            type int16;
        }
    }

    grouping dwdm_edge{
        container bitmap{
            uses bitmap;
        }
        list channels{
            key "g694_id";
            uses dwdm_channel;
        }
        uses edge;
    }

    grouping eth_edge{
        uses edge;
    }

    container topologies{
```

```
        config false;
        list topology{
            key "topology_id";
            uses topology;
        }
    }

}
```

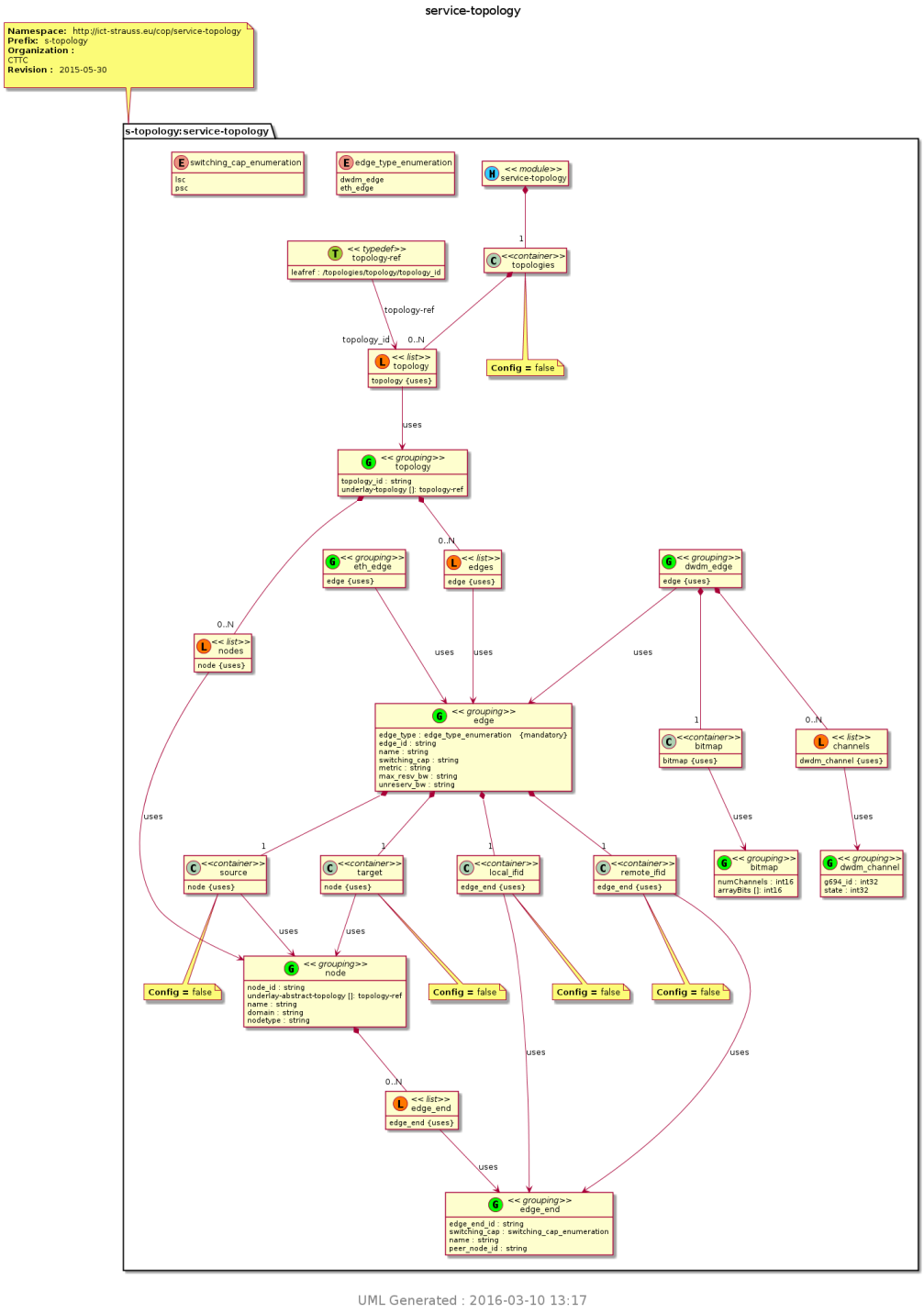


Figure A.2: COP Topology service UML diagram

A.1.3 Path Computation Service

```
module service-path-computation {

    yang-version 1;

    namespace "http://ict-strauss.eu/cop/service-path-computation";

    prefix s-path;

    import service-call {
                prefix sc;
        }

    organization "CTTC";

    revision "2015-05-30" {
        description
        "Service Path Computation module for STRAUSS";
    }



    grouping path_request{
        description "";

        leaf request-id{
            type string;
        }
        container aEnd{
            uses sc:endpoint;
        }
        container zEnd{
            uses sc:endpoint;
        }
        container traffic_parameters{
            description "TrafficParam covers service specific traffic attributes";
            uses sc:traffic_params;
        }

    }

    grouping path_reply{
        description "";
        leaf reply-id{
            type string;
        }
        container path{
            uses sc:path-type;
        }
    }

    rpc path_computation{
        input {
            uses path_request;
        }
        output{
            uses path_reply;
        }
```

```
    }
}
```

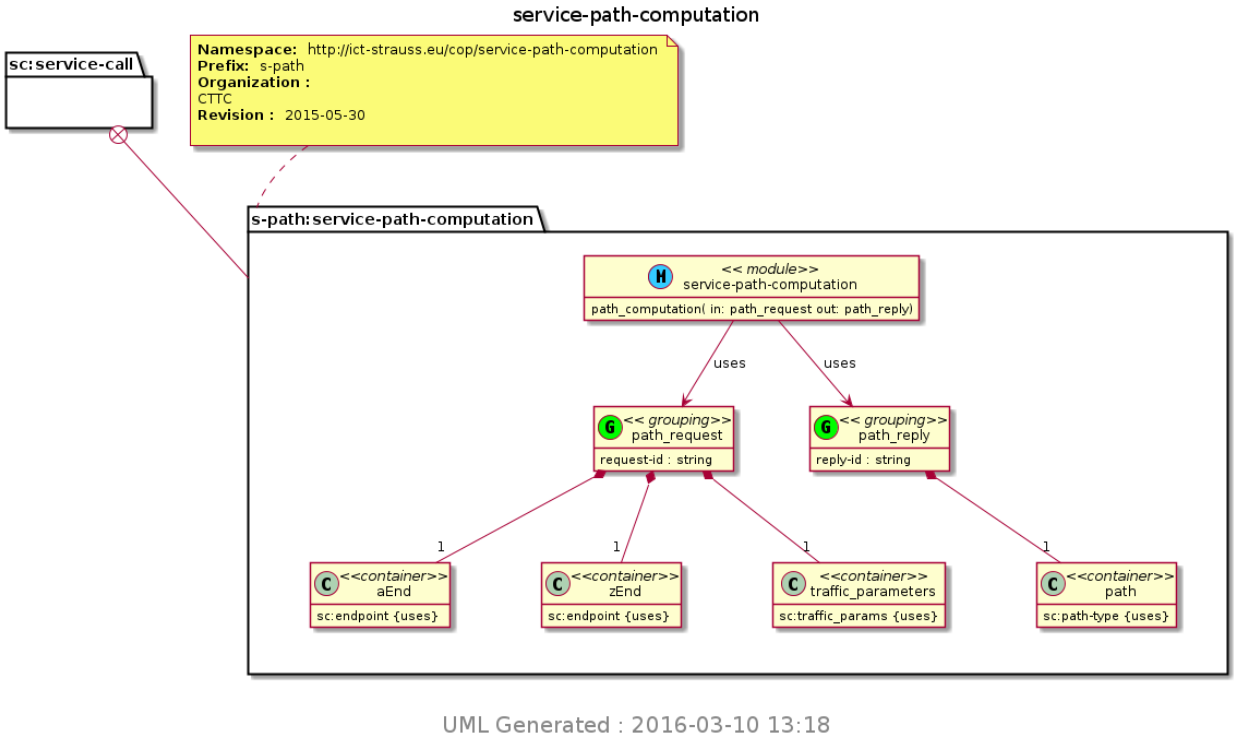


Figure A.3: COP Path Computation service UML diagram

A.1.4 Virtual Network Service

```
module service-virtual-network {

    yang-version 1;

    namespace "http://ict-strauss.eu/cop/service-virtual-network";

    prefix s-vnet;

    import service-topology{
        prefix s-topo;
    }

    organization "CTTC";

    description "YANG version of the Common Orchestration Protocol (COP).";

    revision "2015-06-23" {
        description
        "Service Virtual Network module for STRAUSS";
    }

    typedef virtual-network-ref {
        type leafref {
            path "/virtual_networks/virtual_network/virtual_network_id";
        }
        description
```

```
            "A type for a virtual network instance.";
    }


    grouping virtual_network{
        leaf virtual_network_id{
            type string;
        }
        leaf tenant_ip{
            type string;
        }
        list vNodes{
            config false;
            key "vNode_id";
            uses vNode;
        }
        list vEdges{
            config false;
            key "vEdge_id";
            uses vEdge;
        }
    }

    grouping vEdge_end{
        leaf vEdge_end_id{
            type string;
        }
        container pNode{
            uses s-topo:node;
        }
        container pEdgeEnd{
            uses s-topo:edge_end;
        }
    }

    grouping vNode{
        leaf vNode_id{
            type string;
        }
        list vEdge_ends{
            key "vEdge_end_id";
            uses vEdge_end;
        }
    }

    grouping vEdge{
        leaf vEdge_id{
            type string;
        }

        container source{
            config false;
            uses vNode;
        }
        container target{
            config false;
            uses vNode;
        }
        container local_ifid{
            config false;
            uses vEdge_end;
        }
```

```
        container remote_ifid{
            config false;
            uses vEdge_end;
        }
    }

    container virtual_networks{
        list virtual_network{
            key "virtual_network_id";
            uses virtual_network;
        }
    }

}
```

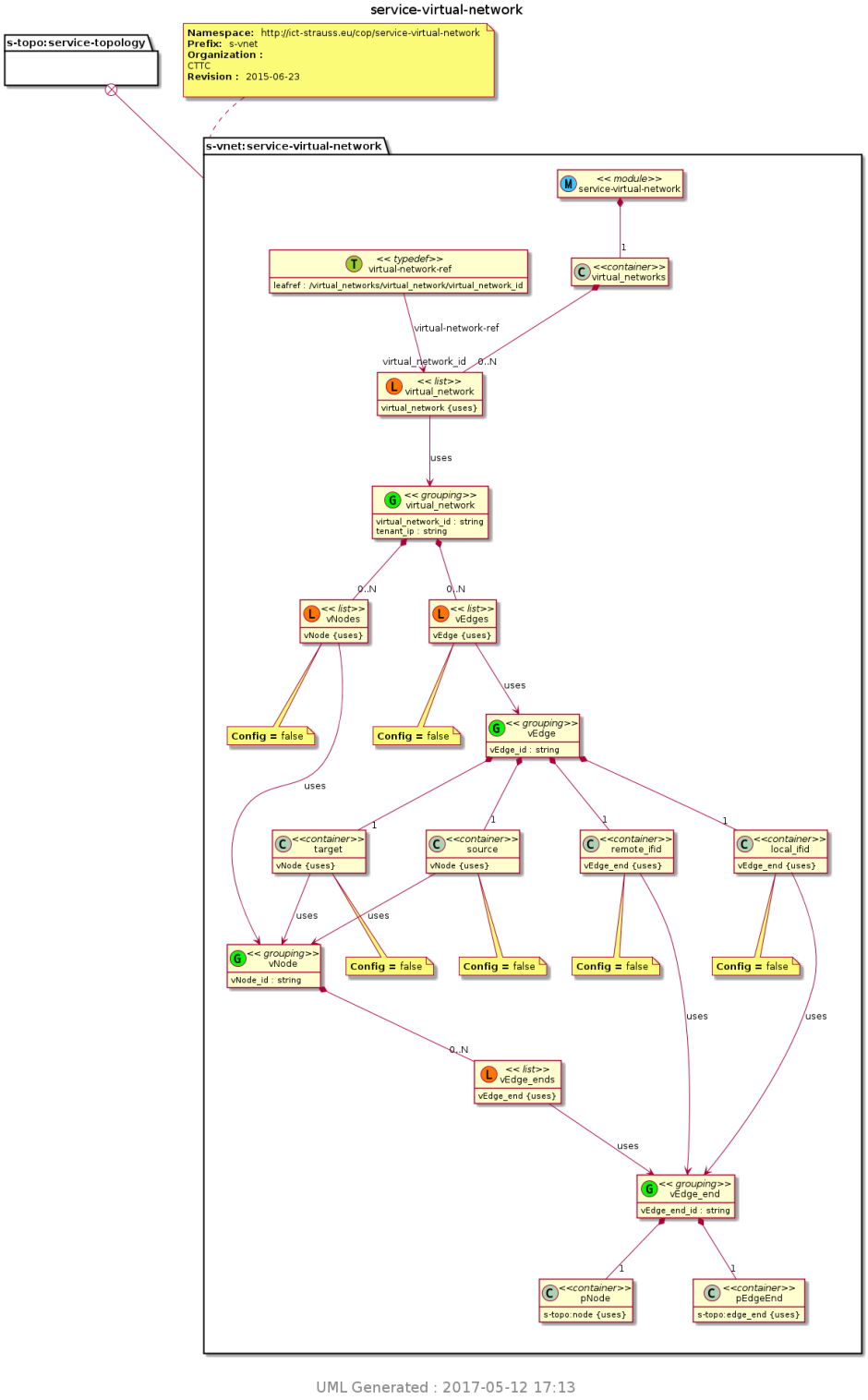


Figure A.4: COP Virtual Network service UML diagram

A.2 COP interface definition based on RESTCONF/SWAGGER

Following, a detailed set of examples of the COP configuration data store, operations and notification following the RESTCONF standard are presented in the next subsections. The examples show the HTTP Request commands required to excite the COP RESTCONF interface including examples of

the JSON-encoded body messages included in the HTTP Requests.

A.2.1 Call Service

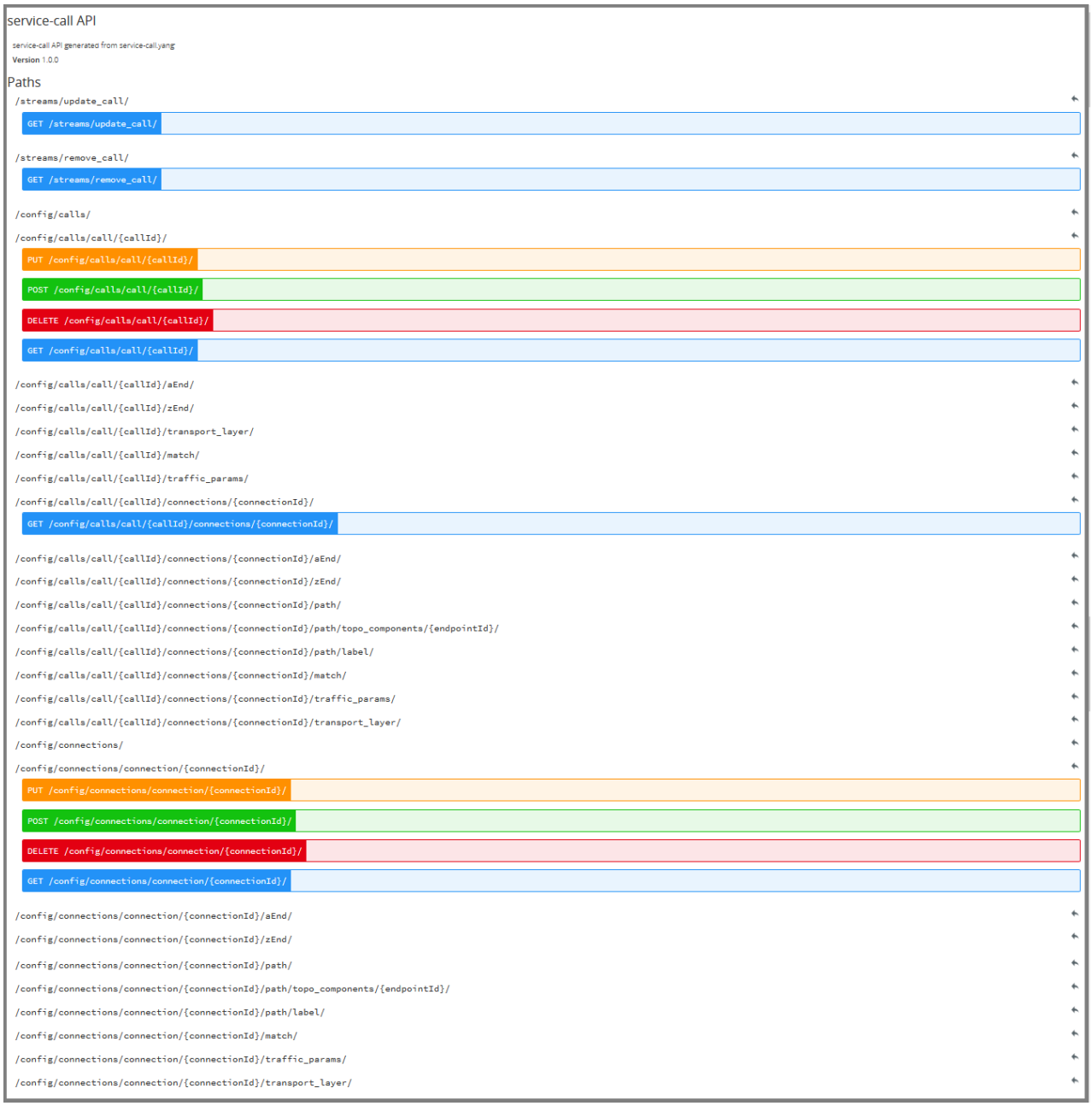


Figure A.5: COP Call service RESTCONF interface.

Figure A.6: COP Call service RESTCONF interface.

A.2.2 Topology Service

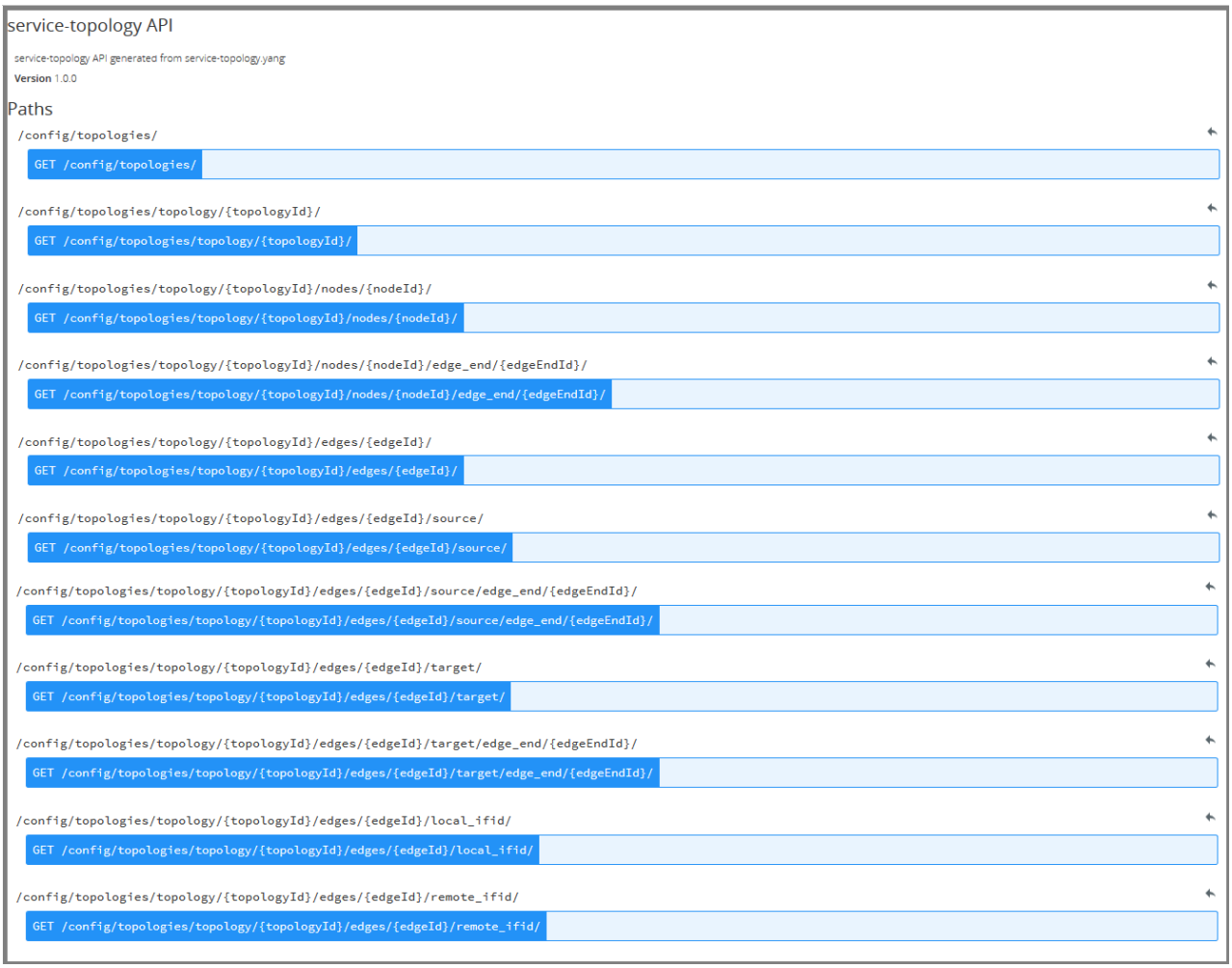


Figure A.7: COP Topology service RESTCONF interface paths SWAGGER editor display.

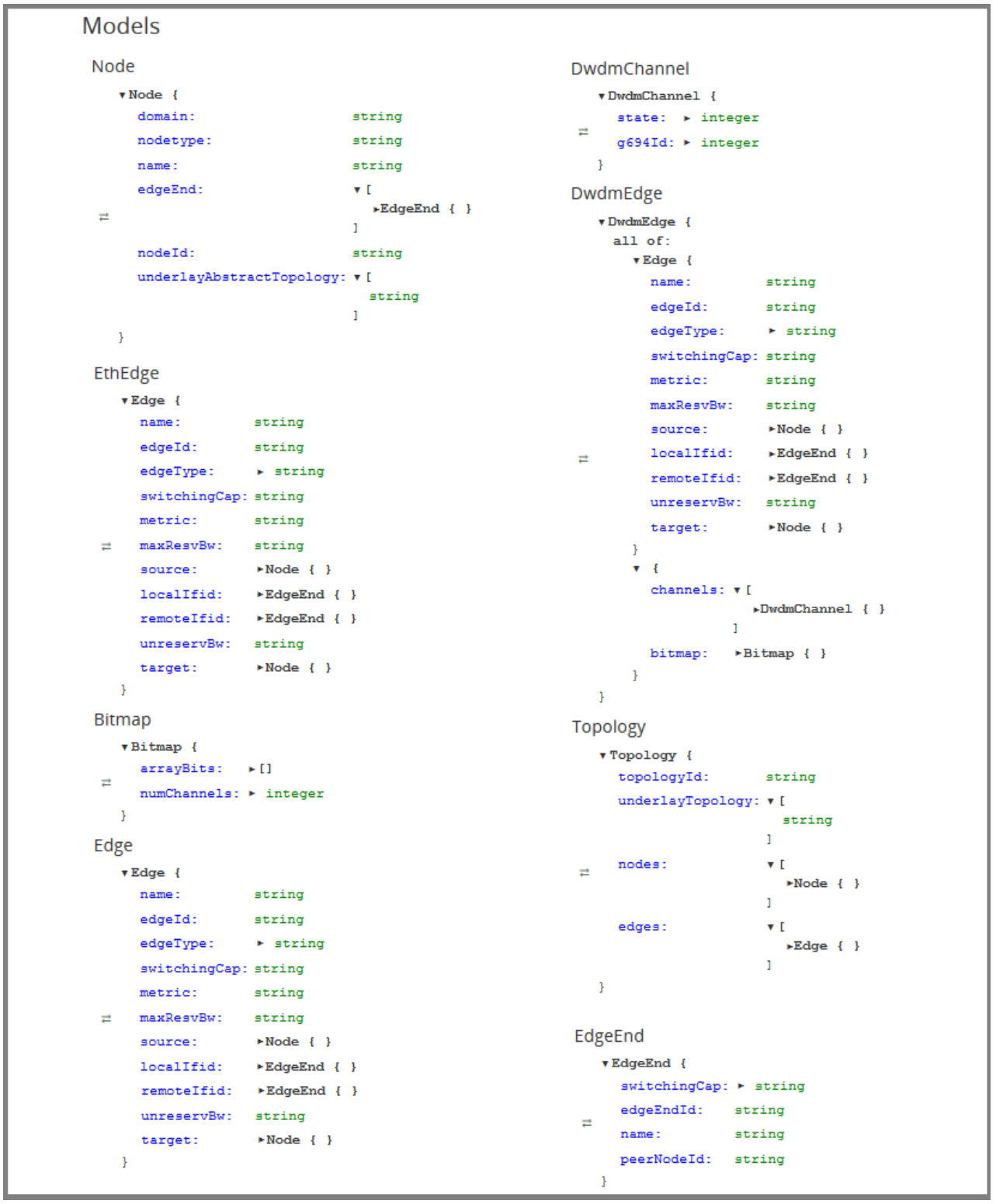


Figure A.8: COP Topology service JSON Data models SWAGGER editor display.

A.2.3 Path Computation Service

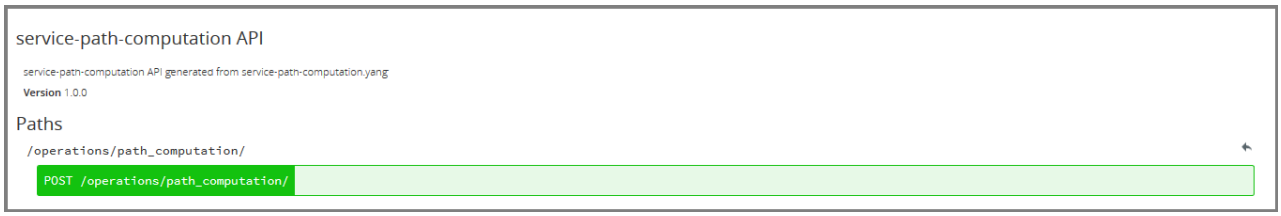


Figure A.9: COP Path Computation service RESTCONF interface paths SWAGGER editor display.

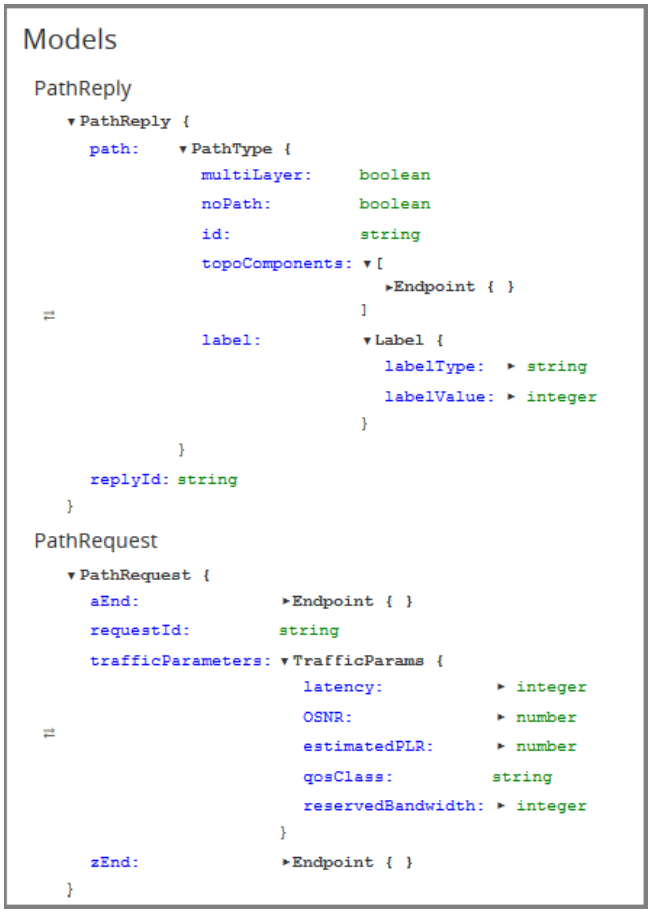


Figure A.10: COP Path Computation service JSON Data models SWAGGER editor display.

A.2.4 Virtual Network Service

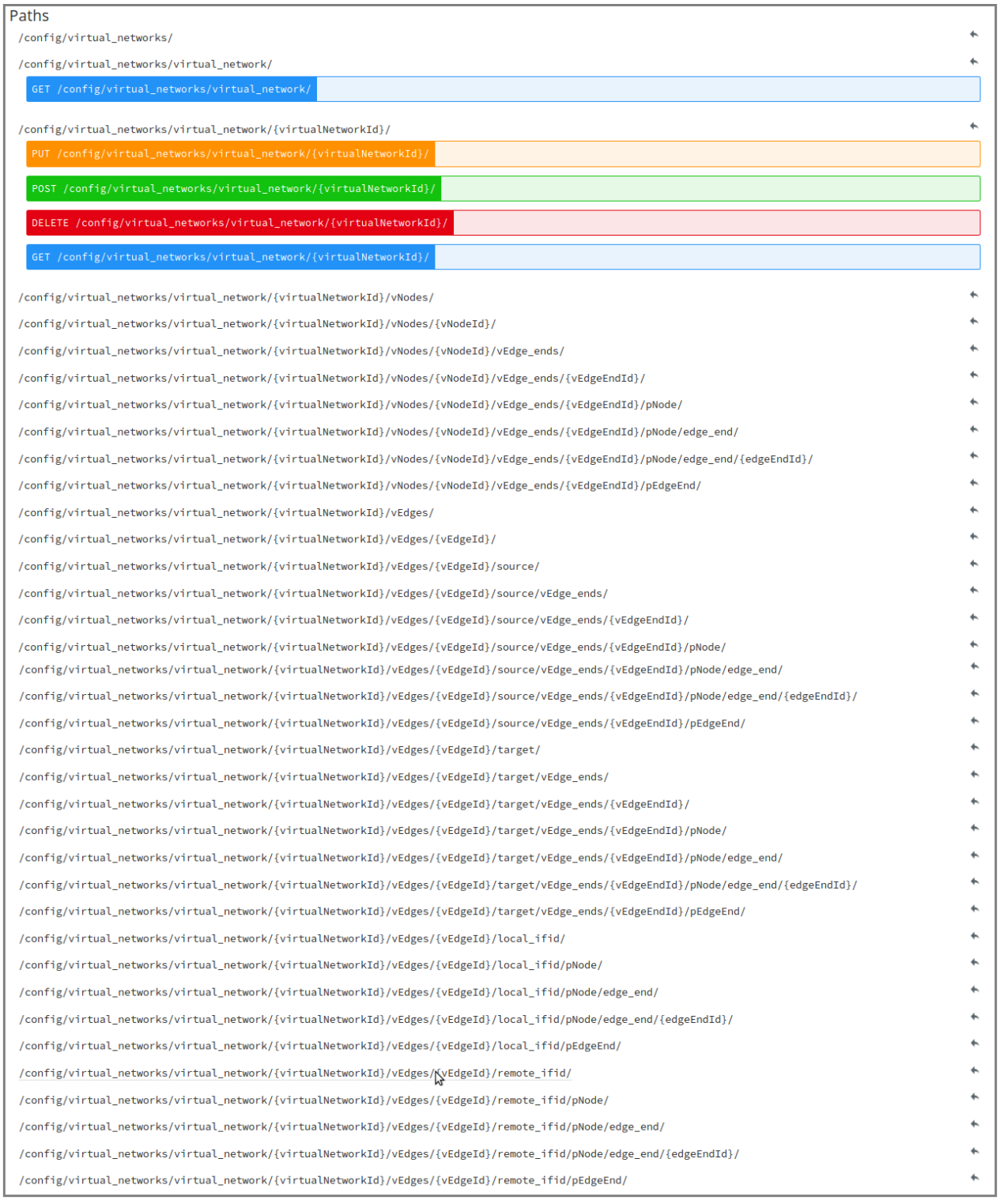


Figure A.11: COP Virtual Network service RESTCONF interface paths SWAGGER editor display.

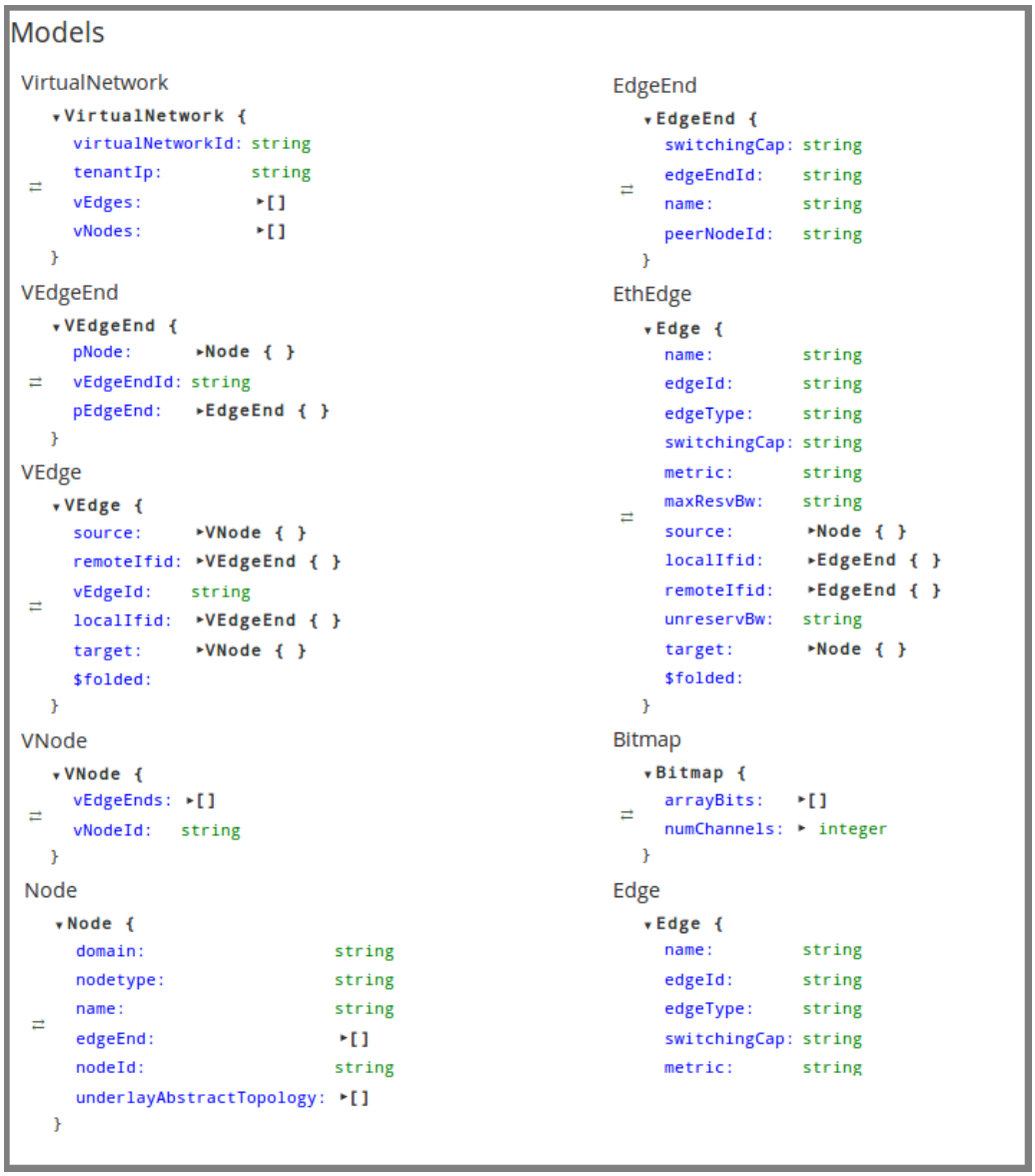


Figure A.12: COP Virtual Network service JSON Data models SWAGGER editor display.