Universitat Politècnica de Catalunya
Escola de Doctorat
Programa de Doctorat:
Teoria del Senyal i Comunicacions

# Quality of Service, Security and Trustworthiness for Network Slices

PhD. Thesis

***Proposer:***
Pol Alemany Prats

***Director(s):***
Ph.D. Ricard Vilalta Cañellas - Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA)
Ph.D. Raül Muñoz González- Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA)

***Tutor:***
Prof. Gabriel Junyent Giralt - Universitat Politècnica de Catalunya (UPC)


# Summary

The telecommunications' systems are becoming much more intelligent and dynamic due to the expansion of the multiple network types (i.e., wired, wireless, Internet of Things (IoT) and cloud-based networks). Due to this network variety, the old model of designing a specific network for a single purpose and so, the coexistence of different and multiple control systems is evolving towards a new model in which the use of a more unified control system is able to offer a wide range of services for multiple purposes with different requirements and characteristics. To achieve this situation, the networks have become more digital and virtual thanks to the creation of the Software-Defined Networking (SDN) and the Network Function Virtualization (NFV).

Network Slicing takes the strengths from these two technologies and allows the network control systems to improve their performance as the services may be deployed and their interconnection configured through multiple-transport domains by using NFV/SDN tools such as NFV-Orchestrators (NFV-O) and SDN Controllers.

This thesis has the main objective to contribute to the state of the art of Network Slicing, with a special focus on security aspects towards the architectures and processes to deploy, monitor and enforce secured and trusted resources to compose network slices. Finally, this document is structured in eight chapters.

Chapter 1 provides the motivation and objectives of this thesis which describes to where this thesis contributes and what it was expected to study, evaluate and research.

Chapter 2 presents the background necessary to understand the following chapters. This chapter presents a state of the art with three clear sections. First, the key technologies necessary to create network slices when using multiple domains and different technologies to improve the management

performance of each element within a network slice. Secondly, an overview about the relationship between Service Level Agreements (SLAs) and network slices is presented together with a more specific view on Security Service Level Agreements (SSLAs). Finally, it presents the literature related about distributed architectures and systems and the use of abstraction models to generate trust, security, and avoid management centralization.

Chapter 3 introduces the research done associated to Network Slicing. First it describes the creation of network slices using resources placed in different and multiple computing and transport domains and how the control systems interact among them to ensure the correct deployment of the selected resources. Then, this chapter also illustrates how the use of multiple virtualization technologies allows to have more efficient network slices deployments and where each technology fits better to accomplish the performance improvements. As a final remark, the work done in this chapter served as the starting point for the research about the Quality of Service (QoS) awareness, security and trust aspects described in the following chapters.

Chapter 4 presents the research done about the management of network slices and the definition of SLAs and SSLAs to define the service and security requirements to accomplish the expected QoS and the right security level. To do so, the chapter first focuses on the use of SLAs and how to monitor them to react in case there is a violation. Based on this experience, the chapter moves towards the specific need to define and deploy the necessary elements to monitor and enforce the desired security on the deployed network slices.

Chapter 5 studies the possibility to change at certain level the trend to centralise the control and management architectures towards a distributed design. To do so, the chapter explores the use of a Distributed Ledger Technology (DLT) such as Blockchain as the key element to create and implement cooperative multi-domain infrastructures to deploy End-to-End (E2E) network slices. First, the chapter describes the new designed element to let a computing domain to become part of a Blockchain system and to participate in the cooperative deployment procedure to instantiate computing resources across a single transport domain. Secondly, the chapter focuses in a scenario with multiple transport domains and how together, the they collaborate to configure the interconnection to link one extreme with another through the use of a new Blockchain-based component.

Chapter 6 follows the work started in the previous chapter and focuses on the generation of trust among service resources providers. It is equally important to have secured deployment procedures and deployed resources

but also, that the resources offered and used for those deployments may be properly trusted. This chapter first describes how the concept of trust, which depend on feelings, is mapped into an analytical system and then, how the trust management among providers and clients is done in a transparent and fair way.

Chapter 7 is devoted to the dissemination results and presents the set of scientific publications produced in the format of journals, international conferences or collaborations. Moreover, it is also described how the work performed in this thesis is related to a set of international research projects. In terms of numbers, 5 Journal articles were published or submitted at the time of writing this PhD thesis, 9 conferences and 4 collaborations were presented and finally, the results obtained were used to contribute in 4 international projects.

Chapter 8 concludes the work and outcomes previously presented and presents possible future research.

**Keywords:** NFV, SDN, Network Slicing, Security, Trust, SLA, SSLA, Blockchain

# Resum

Els sistemes de telecomunicacions s'estan tornant molt més intel·ligents i dinàmics degut a l'expansió de les múltiples classes de xarxes (i.e., xarxes amb i sense fils, Internet of Things (IoT) i xarxes basades al núvol). Tenint en consideració aquesta varietat d'escenaris, el model antic de disseny d'una xarxa enfocada a una única finalitat i, per tant, la una coexistència de varis i diferents sistemes de control està evolucionant cap a un nou model en el qual es busca unificar el control cap a un sistema més unificat capaç d'oferir una amplia gama de serveis amb diferents finalitats, requeriments i característiques. Per assolir aquesta nova situació, les xarxes han hagut de canviar i convertir-se en un element més digitalitzat i virtualitzat degut a la creació de xarxes definides per software i la virtualització de les funcions de xarxa (amb anglès Software-Defined Networking (SDN) i Network Function Virtualization (NFV), respectivament).

Network Slicing fa ús dels punts forts de les dues tecnologies anteriors (SDN i NFV) i permet als sistemes de control de xarxes millorar el seu rendiment ja que els serveis poden ser desaplegats i la seva interconnexió a través de múltiples dominis de transport configurada fent servir eines NFV/SDN com per exemple orquestradors NFV (NFV-O) i controladors SDN.

Aquesta tesi té com a objectiu principal, contribuir en diferents aspectes a la literatura actual al voltant de les network slices. Més concretament, el focus és en aspectes de seguretat de cara a les arquitectures i processos necessaris per desplegar, monitoritzar i aplicar recursos segurs i fiables per generar network slices. Finalment, el document es divideix en 8 capítols.

El Capítol 1 correspon a la introducció de la temàtica principal, la motivació per estudiar-la i els objectius plantejats a l'inici dels estudis de doctorat.

El Capítol 2 presenta un recull d'elements i exemples en la literatura actual per presentar els conceptes bàsics i necessaris en relació a les tecnologies NFV, SDN i Network Slicing.

El Capítol 3 introdueix el lector a les tasques i resultats obtinguts per l'estudiant respecte l'ús de network slices enfocades en escenaris amb múltiples dominis de transport i posteriorment en la creació i gestió de network slices Híbrides que utilitzen diferents tecnologies de virtualització.

El Capítol 4 s'enfoca en l'ús d'eines de monitorització tant en avaluar i assegurar que es compleixen els nivells esperats de qualitat del servei i sobretot de qualitat de seguretat de les network slices desplegades. Per fer-ho s'estudia l'ús de contractes de servei i de seguretat, en anglès: Service Level Agreements i Security Service Level Agreements.

El Capítol 5 estudia la possibilitat de canviar el model d'arquitectura per tal de no seguir centralitzant la gestió de tots els dominis en un únic element, aquest capítol presenta la feina feta en l'ús del Blockchain com a eina per canviar el model de gestió de recursos de múltiples dominis cap a un punt de vista cooperatiu i transparent entre dominis.

El Capítol 6 segueix el camí iniciat en el capítol anterior i presenta un escenari en el qual a part de tenir múltiples dominis, també tenim múltiples proveïdors oferint un mateix servei (multi-stakeholder). En aquest cas, l'objectiu del Blockchain passa a ser la generació, gestió i distribució de paràmetres de reputació que defineixin un nivell de fiabilitat associat a cada proveïdor. De manera que, quan un client vulgui demanar un servei, pugui veure quins proveïdors són més fiables i en quins aspectes tenen millor reputació.

El Capítol 7 descriu els articles de conferència i revista publicats per l'estudiant tant com a primer autor com a col·laborador. A més a més, també descriu els projectes nacionals i internacionals en els quals ha participat durant el període de recerca de doctorant.

El Capítol 8 finalitza la tesi amb les conclusions finals, en l'anàlisi d'assoliment dels objectius presentats al primer capítol i possibles feines de futur.

**Paraules Clau:** NFV, SDN, Network Slicing, Security, Trust, SLA, SSLA, Blockchain

# Acknowledgment

Now that I am at the end of this long journey and after all the troubles solved and the effort done, I just have few words of gratitude.

First to my family; to my mother and brothers who taught me that there is a wider world outside my laptop and sometimes it is better to leave aside what you are doing, to keep advancing; to my father, from whom I inherit the love for the telco world and have learned most of what I know in life, specially the bad jokes that nobody else understands.

To my friends; who despite I had to tell them "no" to stay at home some times in the last three years, they still kept proposing me to have dinner together with them and insisted me to finish this work.

To my thesis directors; since 2017 when I started in CTTC, they have always treated me like one of them and made me see that even though research is quite hard, it is also something you can have fun and, of course, learn a lot. Without their guidance, this work would not have been possible.

To all the researchers and people that I met and who helped me and contributed in any of the works I presented here.

Finally, to my wife: while all the previous people were key players in different aspects, she was the key. She was happy for me when I started the PhD adventure, she was patient when I had to work during weekends and nights, she was who suffered when I was nervous and foremost, she is the one that gave me support and pushed me when I had doubts to follow this adventure. To her.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## Contents

This chapter introduces the reader into the context of this thesis through its three sections. First, the motivation and main topic of this thesis is described. Then, the main objectives where this thesis contributes are described, including the specific objectives regarding the different elements studied in the following chapters. Finally, the thesis structure is presented.

## 1.1 Motivation

The evolution of the network communications is in constant discussion and investigation, two examples of this are the fact that the European Union has 5G networks as one of its key issues [1] within its plans and the network operators which, due to their business point of view (i.e., costs reduction), are constantly looking for new ways to improve their deployed network resources control and management procedures and operations. In the last years, the number of mobile and optical communications (i.e., Long Term Evolution (LTE), 5G, fiber to the 'x') users has increased [2] together with the services offered in the internet such as data streaming, video-conferences or online gaming. To support all these services, the infrastructure has been growing during the last two decades.

Despite the growth of the infrastructures, new vertical sectors/services [3] (e.g., automotive, manufacturing, public safety and others) are being studied and developed. On the one hand, it is expected that they will lead to an increment of users and bring massive benefits to the society but, on the other hand, they will demand much higher requirements than the current mobile or video-conferences services and generate an increment of traffic over the current infrastructures. Moreover, all these different services will have to coexist within the same network infrastructure, keeping their independence, not disturbing the other services while performing as expected. The easiest option to reach this scenario would be to keep increasing the current network and computing resources to dedicate a set of them to each specific service but, in a long-term context, their management would lead towards a waste of resources. Whether it is from an energy, a climate or a business point of view, Network operators want to avoid wasting resources and reducing their costs as much as possible. To do so, one of the possibilities is to make the current network infrastructures much more efficient.

To achieve this objective, in the last 10 to 15 years, network infrastructures undergone an evolution from static-role network elements (i.e., one

specific hardware and tied to a specific software and performing a specific task) to more dynamic-role network elements in which software rules over a generic hardware. To do so, the Software Defined Networking (SDN) concept [4] was defined. SDN allows to configure generic hardware nodes to act as specific networking elements depending on the needs that a network manager may have and so the use of specific hardware is avoided. Moreover, the use of software (i.e., SDN Controllers) allows the network owner to monitor and control its network in a more efficient and dynamic way while centralizing all these actions in a single element avoiding the need to go and configure all the elements one by one.

While SDN was the first step in the network evolution, the virtualization technology became the second one. Due to its flexibility, the use of virtualized elements offers a new way to manage multiple services in parallel, fulfilling their specific requirements and without the need of a constant investment on new networking resources. To do so, Network Function Virtualization (NFV) was defined [5] and later standardized [6]. NFV allows the removal of networking functions like firewalls, load balancers and other functionalities from dedicated hardware to convert them into software elements that are virtualized and deployed over cloud computing domains. The first objective is to decrease the number of elements constantly deployed in the network and make them available to be used only when required. Moreover, NFV brings other benefits such as scalability and robustness as in case of need (i.e., security threats, need for more resources, etc.) the virtualized functions may be replicated and/or migrated as desired.

Joining SDN and NFV is the key step that gives the complete dynamism to the new network management model as SDN allows to configure the necessary connectivity services to interconnect the different virtualized (i.e., NFV) elements deployed in multiple cloud computing nodes (i.e., Data Centers (DCs)). The main benefits obtained by using these two technologies (i.e., NFV/SDN) are: a) the flexibility to deploy and configure the most suitable elements depending on the needs, b) a centralised management of the elements, c) the automation of network deployments and configurations and d) minimize cost due to the reduction of operational expenses or the need of specialized network hardware. The use of NFV/SDN allowed one more step in the evolution towards the near-future and next generation networks: Network Slicing [7]. This technology permits the deployment of logical networks (called network slices) using a set of deployed computing and networking resources focused on a specific and dedicated service over a common and

shared physical infrastructure. Each network slice is defined based on the specific characteristics and deployed with the required resources that fulfill them. Despite sharing the physical infrastructure, the final user believes the network infrastructure is absolutely dedicated to its requested service due to the virtual independence among the different network slices.

Thanks to the use of NFV/SDN and Network Slicing technologies, it becomes possible to disassociate hardware and software and so, a new scenario appears in which multiple providers for a specific resource may co-exist and be requested at any moment in parallel. Having a wide range of providers to select resources implies a reduction of costs (i.e., more competitive options to choose) but also an increment of trust-related risks (i.e., new providers appearance, historical behaviour), a trade off that needs to be properly managed. For this reason, it is necessary to find a way to define how trustworthy are the available providers within the dynamic and virtualised scenario offered by the use of the NFV/SDN and Network Slicing. Finding the right way to apply trust within network control and management systems is an interesting task due to the need to make an subjective concept (i.e., a perception of an entity with respect to another) to become an objective parameter that can be use in analytical systems.

Finally, based on the previously introduced concepts and looking at two of the issues described in [1] (i.e., networks and security), this thesis focuses on security aspects around Network Slicing. The implementation of all the previous elements (i.e., SDN, NFV, Network Slicing) implies new possibilities but also new security threats. The use of network slices means that multiple and different services will share the same infrastructure and it is crucial to ensure that the security of each individual network slice is properly respected. To do so, two main points of view need to be studied: before and during the service deployment. Before the deployment it is important to validate that all the elements provided and who provides them may be trusted. As SDN/NFV and Network Slicing open the door towards the scenario in which multiple providers and operators may interact all together, trust among them must be assured. On the other hand, once deployed, it is of the most absolute importance to ensure that the expected Quality of Service (QoS) and Quality of Security (QoSec) are achieved and, in case there is an anomaly, to react and solve it with the best and fastest solution as possible.

## 1.2 Objectives

Based on the motivation introduced in the previous section, this PhD thesis investigates multiple security features around the Network Slicing. The focus will be presented in both points of view: the need of having reputable and trustworthy players and to ensure that a deployed network slice does not have its security compromised due to an external threat. This section presents the two set of objectives planned in this PhD thesis: a list of generic objectives and, a list of more specific (technical) objectives focused on the key topics studied.

### 1.2.1 Main objective

The main objective of this PhD thesis is to contribute to the state of the art on Network Slicing security aspects from two different points of view:

- The security enforcement of the deployed network slices and the services composing them based on the security requirements defined in a Security Service Level Agreement (SSLA) by the service requester. To achieve this, the security architecture and its components, the processes and interactions among these components and a set of experimental results are presented.

- The security needs to be applied prior to the deployment of any network slice. For this reason, this thesis also looks at how to distribute the responsibility of the deployments to avoid potential central point of failure in multi-domain scenarios and to generate trust not only on the Network Slicing resources offered but also on the providers that offer those resources.

### 1.2.2 Technical objectives

To achieve the main objective, a set of technical objectives were defined as the tasks to work on step by step:

- To reinforce the current experience and to acquire new knowledge about NFV/SDN and Network Slicing systems. To generate new knowledge, it is essential to have a clear image of all the steps about how to define and design network slices, how they are deployed through a Network

Slice Manager and the NFV/SDN elements, the technologies around it and how they can be used. To do so, this PhD thesis presents its contribution with new possibilities and conceptual ideas (i.e., functions, architectures, etc.) related to Network Slicing. Moreover, it is key to demonstrate the feasibility of the designed new ideas with the development of use cases and the obtained results from experimental implementations.

- Using SSLAs to define and deploy security elements as part of network slices and increase their security. To achieve this objective, the knowledge acquired in the previous objective is used to study the use of SLAs to enforce the service performance and, later, a re-definition of the SLA concept is done but specifically focused on security. By doing so, the integration of SSLAs and network slices is studied and presented with the designed architecture and modules required to manage the secured network slices and their SSLAs.

- To investigate how Distributed Ledger Technologies (e.g., Blockchain) may be used to design distributed architectures for a cooperative and collaborative Network Slicing management and trust procedures. To do so, a research on reputation and trust is a key task because in the near future networks, the scenario will pass from a close group of providers and a single operator scenario to a new one with multiple providers and operators interacting with each other (i.e., multi-stakeholder scenario) to create End-to-End (E2E) network slices. For this reason, it became interesting to study a possibility about how reputation and trust may be used to evaluate the different providers within the deployment of a network slice.

Finally, and despite it is expected within the PhD process, to acquire new skills to improve the research career. As a PhD student, it is expected to reinforce the current skills and knowledge while acquiring new ones. Step by step, the student should take more responsibilities in terms of thinking, planning, developing and validating the multiple ideas that may come during the realization of this PhD thesis. Through the peer-review process implemented in international conferences and in journal/magazine articles, the student has to learn and improve the writing and oral communicative abilities.

## 1.3 Thesis structure

This PhD Thesis is structured in:

- Chapter 2 presents the state of the art related to Network Slicing, Security and Trust.
- Chapter 3 extends the concepts around the network slices focusing on multi-domain and hybrid network slices deployments.
- Chapter 4 describes the relationship between Network Slicing and Security Service Level Agreements.
- Chapter 5 presents the use of Blockchain towards a distributed management of Connectivity Services and Network Slicing resources.
- Chapter 6 presents the management of trust related to the network slices service providers.
- Chapter 7 presents the impact of the results with the list of journal and conference articles produced and the international projects where the student has participated during the thesis.
- Chapter 8 concludes and assesses the technical objectives and proposes further research work.

# Chapter 2

# State of the art

## Contents

Based on the objectives described previously, it is a key task to check and analyze the current aspects being researched about the selected topics in this work. First, a list with the main concepts and their descriptions is given to assist the reader. Then, a set of three different state-of-the-art sections are presented, each one focusing on a specific topic: network resources and services management, security and SLAs management and, trust and security aspects in distributed architectures.

## 2.1 Main concepts and definitions

The following list is presented in order to assist the reader with the content described in the following chapters. The definitions were obtained from different ETSI documentation in order to ensure the most standard description. The references are given at the end of each definition

- Software Defined Networking (SDN): The means to dynamically control the network and the provisioning of networks as a service. [4]
- Network Function Virtualisation (NFV): The principle of separating network functions from the hardware they run on by using virtual hardware abstraction. [8]
- Virtual Network Function (VNF): An implementation of a network function that can be deployed on a Network Function Virtualisation Infrastructure. [8]
- Network Function Virtualisation Infrastructure (NFVI): The totality of all hardware and software components that build up the environment in which any VNF is deployed. [8]
- Network Service (NS): A composition of Network Function(s) and/or Network Service(s), defined by its functional and behavioural specification. [8]
- Connectivity Service (CS): An implementation of the connectivity aspects (i.e., ports, bandwidth, spectrum, etc.) between one or more points across one or more transport domains. [9]
- Network slice: A logical network that provides specific network capabilities and characteristics, supporting various service properties. [10]

- E2E Network Slicing: A set of management and orchestration activities that allow the deployment and operation of network slices across multiple management domains. [10]
- Network slice subnet (slice-subnet): A representation of a set of network functions and the associated resources (e.g., compute, storage and networking resources) supporting network slice. This concept can also be found in the literature as "sub-slice". [10]
- Service Level Agreement (SLA): A set of negotiated agreements between two or more parties, recording a common understanding about the service and/or service behaviour (e.g., availability, performance, service continuity, responsiveness to anomalies, security, serviceability, operation) offered by one party to another, and the measurable target values characterizing the level of services. [8]
- Distributed Ledger Technologies (DLT): A technology that enables the operation and use of distributed ledgers. In there, a ledger is shared across a set of DLT nodes and synchronized between the DLT nodes using a consensus mechanism. [11]
- Smart Contract (SC): A computer program stored in a distributed ledger system, wherein the outcome of any execution of the program is recorded on the distributed ledger. [12]

## 2.2 Introduction to Network Slicing

As previously introduced, telecommunications systems are in constant growth and evolution in order to accomplish the requirements and objectives defined with the appearance of new services. Some examples of this continuous growth may be found on mobile communications systems evolution from the first generation (1G) in the 80's to the current 5G, and also on non-mobile communications with the use of optical fiber technology in other domains aside from transport domains with backbone functionality. In both cases, a common aspect that is continuously looking for improvements is the control of these infrastructures and their resources management based on the services offered. As the same service may have different requirements in each of the multiple known scenarios and verticals possibilities (i.e., 5G verticals

[3]), it is necessary for the management of infrastructures and resources to be as flexible, responsive and dynamical as possible. To this end, the use of SDN, NFV and Network Slicing has become key player.

### 2.2.1 Software-Defined Networking technology

As previously commented, the appearance of SDN allowed to evolve [13] the control and management of network elements using a piece of software in a centralized and much more dynamic method instead of having to configure them one-by-one meaning a slower and complex process. As widely known, its main strength is that it clearly differentiates the control and the data planes, it allows to provide an abstracted view of the infrastructure to identify strong dependencies on proprietary technologies and protocols which leads towards an opening of the networks and the avoidance of monopolies by giving the possibility of new vendors to provide new tools. Despite these advantages, different aspects are being researched such as tools for the transition [14] from Ethernet towards SDN or in terms of security; the possible threats around the SDN architecture [15] or how it can be used to block certain attacks [16].

### 2.2.2 Network Function Virtualization technology

The implementation of NFV [5] technologies give the possibility to deploy ad-hoc network functions (i.e., Virtual Network Functions (VNFs)) and Network Services (NSs), a chain of VNFs, over neutral pieces of hardware without the need of modifying the physical infrastructure. By doing so, dedicated hardware is avoided in favor of generic purpose hardware able to contain multiple and different functions such as firewalls, load balancers and other elements. Among its main advantages, there are flexibility, scalability and a very important improvement in terms of operational and capital costs due to the possibility of having different elements within the same generic box. Even though NFV is widely used and its implementation extended, different aspects are being investigated with some examples focused on the NFV resource allocation challenge [17] or security use cases in the Internet of Things (IoT) [18] and smart manufacturing [19] scenarios.

NFV bases its processes and actions on two virtualization technologies. On the one hand, the Kernel-based Virtual Machine (KVM) and, on the other hand, the use of containers. Both of them allow to create virtual Information Technologies (IT) elements such as servers over a single physical node

isolating each one of them from the others. The main and most important difference [20] between these two technologies is how the virtual resources are created and managed. KVM uses the physical resources of the host Operating System (OS) to deploy virtual servers each with its own guest OS and the necessary applications. The division of host and guest OSs and the management of the different virtual elements is done through an Hypervisor. On the other hand, containers share the resources from the host OS, which makes them lighter and more dynamic but their resources assignment is less stable compared to KVM elements. OpenStack [21] is the most known KVM-based solution within the NFV scenarios, while Docker [22] and Kubernetes [23] are two examples of container technologies used in the design and management of the future networks.

### 2.2.3 Joining SDN and NFV: Network Slicing

Network Slicing takes the strengths of both technologies to deploy multiple and different services with a variety of requirements in parallel over a common infrastructure making an efficient use of the resources available. Based on the demand, a set of virtualized network functions is deployed and configured to compose a logical network. Then, once their are not necessary anymore, the used resources will be released and made available again for a future (an equal or different) service request. Despite the biggest effort to study and define aspects about Network Slicing happened with the latest 5G-related research programs, the first definition and key concepts were proposed by the Next Generation Mobile Networks Alliance (NGNM) organization [7] and later, the first architecture design was proposed by the 3rd Generation Partnership Project (3GPP) [24]. Both documents were accepted and taken by the European Telecommunications Standards Institute (ETSI) as the source documentation to identify how Network Slicing could become part of the NFV architecture [25]. To do so, the Network Slicing functionalities were placed within the Operation Support Services and Business Support Services (OSS/BSS) module as illustrated in [26], next to the Management and Orchestration (MANO) architecture [27].

The use of SDN/NFV and Network Slicing architectures improves the overall network infrastructure resources performance as it allows to organize the resources from different domains and make them interact among them as a single unit while keeping each management domain functions [28] as illustrated in Figure 2.1. A network slice may be composed using either NSs

or recursive network slices with resources that are managed by the different domains. For this reason, Network Slicing may be implemented and used either from a single domain (i.e., edge, transport, core) or from a multi-domain (i.e., E2E) point of view. While it is not mandatory, the most common way of deploying network slices in single domain scenarios is by composing them using only NSs resources. Instead, from an E2E scenario, it is common for the E2E network slice to be composed using network slice resources available in the multiple domains involved. This aspect is important as it allows to keep the individual characteristics of each domain while they work together as a single entity under the control of an E2E NFV MANO with the Network Slicing (i.e., Network Slice Manager) feature and the capability to interact with SDN Controllers.

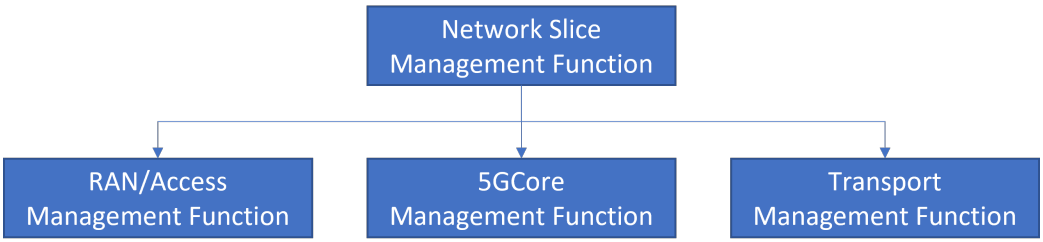


Figure 2.1: 3GPP management functions architecture for Network Slicing.

In addition to how a network slice may be composed (i.e., NSs or recursive network slices), one more aspect to take into account is the way to stitch [29] (i.e., interconnect) the deployed Network Slicing resources in order to have the appropriate data flow within the network slice. In the case of networking resources, the multi-domain scenario is the most common scenario as different transport domains may be placed between the edge and core domains. For this reason, the Network Slice Manager and the NFV MANO below need to interact somehow with a domain SDN Controller or an E2E SDN Controller like the solution presented by R.Vilalta et al. [30] with a single element on top with the knowledge of the E2E network infrastructure.

Because SDN, NFV and Network Slicing are considered the key players[31] within the networks management and control field in the present and near future, there are multiple projects aiming to develop either an NFV Orchestrator (NFV-O) or an SDN Controller. For example, SONATA-NFV [32] is an NFV-O that was originally started in the SONATA project [32] and later evolved into the 5GTANGO project [33] under the supervision of

the European Commission (EC) and the 5G Infrastructure Public Private Partnership (5GPPP). The SONATA-NFV is composed by a set of tools aiming to design, develop, test and validate VNFs/Cloud-native Network Functions(CNFs), NSs and network slices. Then, once completely validated, the SONATA-NFV is able to deploy and manage them in production environments as illustrated in [34]. While the SONATA-NFV was a time-limited project, more well-known (and bigger) projects focusing on a NFV-O development are: Open Network Operation Platform (ONAP) [35] and Open Source MANO (OSM) [36]. Both of them are two open-source projects under the guidance of The Linux Foundation and ETSI, respectively. In both cases, the objective is to create an NFV-O accessible to any actor involved around the network virtualization world.

Regarding SDN, some of the most known projects are the OpenDayLight (ODL) [37] platform and the Open Network Operating System (ONOS) [38]. Both SDN controllers were designed to be adapted to different requirement scenarios, but ODL especially aims to support and join legacy -e.g., Border Gateway Protocol (BGP), Simple Network Management Protocol (SNMP), etc.- and Next Generation Networks (NGN) (e.g., OpenFlow) SDN platforms, while ONOS focuses on performance and clustering aspects in terms of availability and scalability Carriers. At last, an innovative project is the one being proposed and developed in the TeraFlow SDN [39] which has been adopted by the ETSI into a new open-source group. The novelty of the TeraFlow SDN is that it has been designed with a cloud-native architecture based on micro-services [40].

Similarly to SDN or NFV and despite Network Slicing already being a backbone element [41] in the current and near future networks management, there is a variety of aspects being studied and evaluated as there are multiple challenges at different layers [42] and with the appearance of multiple and different players together with the defined verticals [43]. Some literature examples are the works illustrated by M.Vincenzi et al. [44] who present a network slice auctioneer as the central element for the different stakeholders to interact with each other in 5G networks. M.Richart et al. [45] used network slices in wireless networks environments, Y.Cui et al. [46] focused on a Network Slicing architecture for automotive communications and F.Ansah et al. [47] studied network slices on industrial networks communications.

### 2.2.4 Transport network slices

Up until this section, when the network slice concept has been referred, the type of resources involved were both networking and computing resources. But, there is the possibility that only networking resources are used. In this case we can refer to the deployed element as transport network slice.

The complexity in the management of transport network slices recalls in the algorithms to have the most efficient use of the available transport resources to deploy Connectivity Services (CSs) in a transport domain (i.e., a stable amount of bandwidth per CS) and in case of a scenario with multiple transport domains working together, the synchronization of resources (i.e., the same bandwidth or spectrum from one extreme to the other of the CS). Moreover, one more complexity needs to be added when managing transport resources: the dynamic configuration of CSs due to multiple reasons (e.g., re-deployment of NSs, re-configuration of existing CSs, etc.).

It is when transport network slices are necessary that the SDN technologies appear. As described in [27], while the computing resources are deployed using a Virtual Infrastructure Manager (VIM), the transport resources are managed through a Wide Area network (WAN) Infrastructure Manager (WIM). Some existing solutions selected as WIM are the previously introduced ONOS, the ODL and the TeraFlow SDN Controller. This last one has been used in a use case to deploy transport network slices with SLAs [48] or for E2E inter-domain connection management [49].

### 2.2.5 Beyond SoTA

Current implemented and near-future networks look towards lively infrastructures based on the implementation of all the previous technologies (i.e., SDN, NFV and Network Slicing) together with the desagregation concept (i.e., to separate the dependency between hardware and software elements). By doing so, it will allow to have multiple providers able to deploy their services over neutral hardware devices. In order to complement all the references presented about NFV, SDN and Network Slicing, Chapter 3 presents the work and studies done related to the management of network slices on multi-domain networks across transport domains. Furthermore, this chapter also describes how the use of multiple technologies helps on having an efficient resources management and adapted to the characteristics of each domain involved in a network slice.

## 2.3 Network slices, SLAs and security

As it happens with any new enabler/technology, the security around it is a key aspect that needs to be studied by both research and industry actors in order to keep evolving and improving its usage. Despite, as described by D.Sattar et al. [50], Network Slicing may help on improving the deployed services security by mitigating Distributed Denial of Service (DDoS) attacks, the security aspects around it need to be further studied. Especially since the GSMA [51] put emphasis on isolation, low latency communications or the use of SLAs.

Regarding low latency communications, C.Djabir et al. [52] illustrated the use of a Network Slicing framework for a low latency vehicle scenario while regarding isolation, A.Marotta et al. [53] introduced how to make use of isolation to enforce the deployment of reliable network slices and in their later work [54], isolation is described to have multiple shapes, from having a specific optical fiber dedicated to each network slice to sharing an optical fiber with multiple virtual links associated to different network slices. While dedicating a single optical fiber per network slice is the most secured option, it is also not easily feasible in terms of transport resources management because it implies an optical fiber per network slice. Instead, the second option (i.e., virtual links) improves the management of transport domain resources but forces the different network slices to coexist using virtual links. Because of this coexistence, it is necessary to complement the use of isolation techniques with the design and implementation of network slices with an associated SLA.

### 2.3.1 SLA and network slices

As it is known, an SLA is an agreement in terms of service performance between the service requester and the service provider to select a set of Key Performance Indicators (KPIs) that defines the expected QoS. In case these KPIs are not respected, the service client may complain to the service provider and request the corresponding compensation. For this reason, the main objective of a service provider is to reach the expected QoS. To do so, the service provider must constantly check and evaluate (i.e., monitor) multiple service performance values and resolve any possible violation affecting them. Despite the relationship between network slices and SLAs is quite recent, there is already some literature about it. For example, A.Papageorgiou et al. [55] describes an SLA manager on top of an architecture defining how the network

slices must be. K.D.R.Assis et al. [56] presented a new methodology that efficiently designs (transport) network slices based on the SLA demands focusing especially on the route and spectrum allocation requirements. X.Xue et al. [57] demonstrated with interesting results the reconfiguration of Optical Data Centers participating in a network slice, based on QoS requirements. C.Chen et al. [58] used spectral network layer together with Network Slicing resource allocation to deploy the requested service with the expected QoS under an Orthogonal Frequency Division Multiple Access (OFDMA) environment.

When using SLAs and monitoring systems, it is not an easy task to identify where the source of the violation is the origin of a problem may be related either to the amount of deployed service resources or to a security-related issue. For example, a Denial of Service (DoS) attack may decrease the throughput of a service and, in this case, the source of the problem is not related to the service performance (i.e., service needs more bandwidth for video quality) but to the appearance of external factors (i.e., evil transmitters). Therefore, the corresponding solution (i.e., block transmitters flows) will differ from the one taken on the service performance (i.e., increase bandwidth). For this reason, differentiating between different types of quality is necessary. Clear examples of this differentiation are the QoS and QoSec. They focus on the service performance itself and the security around the service respectively.

### 2.3.2 Security SLAs

The term SSLA was first proposed by Henning R. [59] and, similarly to SLA, it defines the requirements that a provider had to fulfill towards its clients from a security point of view.

As described by [60], the use of SSLA implies a set of challenges that needs to be investigated. These are:

- Comprehensive and measurable: How to describe the security features in a way that clients and providers understand it the same way. Moreover, these features need to be properly evaluated (i.e., measured, compared, etc.) with the use of Service Level Objectives (SLOs) focused on security.

- Automated resources management: Not only the requested service has to be properly deployed but also the necessary elements to ensure the

security around it as specified in the SSLA with the right amount of resources.

- Security Enforcement: Until the service is not terminated, it is a must to have a continuous evaluation and (if necessary) enforcement (i.e., monitoring) of the security requirements.

Different standardization organizations or public agencies published their view about the status and use of SSLAs. For example, on the European side, European Network and Information Security Agency (ENISA) [61] presented a study that describes how security parameters were used to define cloud SLAs, and their main conclusions were: a) despite providers were concerned about security aspects, they were mostly offering service performance-related SLAs and, b) the clients could not validate properly if their requested security requirements were achieved. Later in 2012, ENISA presented the work in [62] to guide the clients on how to monitor their requested cloud services. On the American side, the National Institute of Standards and Technology (NIST) [63] described their security framework that focuses on different areas such as mobile and cloud computing, resiliency of information systems and more. Based on all these references, different projects in the last years had the definition of elements around SSLAs within their objective. Two examples are the Secure Provisioning of Cloud Services based on SLA management (SPECS) [64] and the MUlti-cloud Secure Applications (MUSA) [65] projects funded by the EC.

All these previous reports and projects are based on specific works such as the one presented by N.Kaaniche et al. [66] who worked on a specific extension of the common SLA objects by evolving the rSLA language towards security requirements, the research done by K.W.Ullah et al, [67] who presented an SSLA framework focused on cloud services or, finally, the articles from A.De Benedictis et al. [68] and V.Casola et al. [60]. While in [68] they focus on having a chained set of cloud services with an associated per-service SSLA, in [60] the objective is the description of an SSLA model that is used on a security-driven planning process to obtain the best security deployment possible.

### 2.3.3 Beyond SoTA

An interesting gap was detected regarding the relationship between Network Slicing and SSLAs. Among the literature presented, no direct relationship

between network slices and SSLAs has been studied. To do so, learning about the concepts around QoS and QoSec is fundamental in order to research possible enforcement methodologies and elements linked to the network slices deployment. Due to their longer existence, the use of SLAs on network slices is studied, specially on the following aspects: (i) the NFV MANO architecture extensions able to manage network slices and the associated vertical/application domain KPIs, (ii) the KPIs run-time monitoring and QoS enforcement in the deployed network slices, and (iii) the capability to provide QoS guarantees across all the involved layers: from the network slice and its NSs to the final deployed connectivity and computing services. Based on this initial SLA research, SSLAs will be introduced and studied in order to design a NFV MANO architecture extension focused on the use of SSLAs and the enforcement of the service-related security. Our work aims to innovate with the use of the ETSI Zero-touch network and Service Management (ZSM) [69], which aims to provide a new architecture design focused on having an E2E solution with all the services and networks management actions being implemented as automated as possible without the need of the human participation. ZSM was selected to become the security and service management architecture model to deploy the services and enforce the security described in SSLAs.

## 2.4 Security and trust on distributed architectures

Telecommunication systems are continuously facing challenges such as the increase in the number of users and their needs, the appearance of new services and their requirements and the increment of players and the roles that they might have (i.e., multi-stakeholder scenario).

To solve the first challenge, the most basic solution being implemented is to add new resources to the current infrastructures but, in a long term view, this action will lead towards a set of resources being underused.

For this reason and because of the second challenge, multiple network architecture solutions based on new technologies or the distribution of resources across different domains will have to coexist to achieve the final objective to transmit information from one point to another. A clear example already introduced is the use of SDN and NFV standards together with a Mobile

Edge Computing (MEC) and cloud computing concept in the edge and core networks respectively, as they allow a more flexible, dynamic and efficient configuration of the networking and computing resources, or the use of elements to ensure a trusted relationship between resource owners.

Finally, the multi-stakeholder challenge will oblige the current infrastructures to offer the possibility to the service requester to have multiple trusted service providers to choose among them instead of just a small group. For example, a scenario with multiple packet and/or optical transport networks in which the overall infrastructure might be organized in segments, each owned by a different operator (i.e., multi-operator architectures). As each operator has its own interests, having a centralized solution to manage E2E service deployments might be difficult to apply. To solve it, distributed and secure solutions are necessary to allow all the operators to keep their independence while collaborating among them.

The key aspect in all the previous cases is the need of having a trustworthy system accepted by all the players (i.e., users, owners, provider, etc.). Up until now, the common way to have it is to implement a hierarchical model as presented by R.Muñoz et al. [70] or D.Gkounis et al. [71], with a component on top of the infrastructure (Figure 2.2-A) which centralizes the management and control of the whole network under its domain as it is done when using SDN/NFV technologies. In this case, as the component on top is known, all the components below trust it. Despite hierarchical infrastructure are extensively used, they still have weak points. The first one is the "central point of failure": essentially if the component on top fails, the whole control may get blocked, of course there is a solution which implies duplicating that component leading towards the underuse of control resources. The second weak point is the fact that hierarchical models might lead towards a never ending set of layers (i.e., like an onion) which will increase the complexity of the whole management infrastructure. The third and last weak point is that they do not support multi-operator environments.

A possible solution is the use of a collaborative system model in which the network owners collaborate and share their resources in a trustworthy and reliable way, whilst being able of offering different services across the network and keeping the privacy of each service consumer. In a mesh model (Figure 2.2-B) all nodes are equal and they work in a collaborative way, creating what is called a peer-to-peer (P2P) network. P2P networks have been used in multiple applications such as files exchange systems [72] [73] or Voice on Demand services [74] among other possibilities.

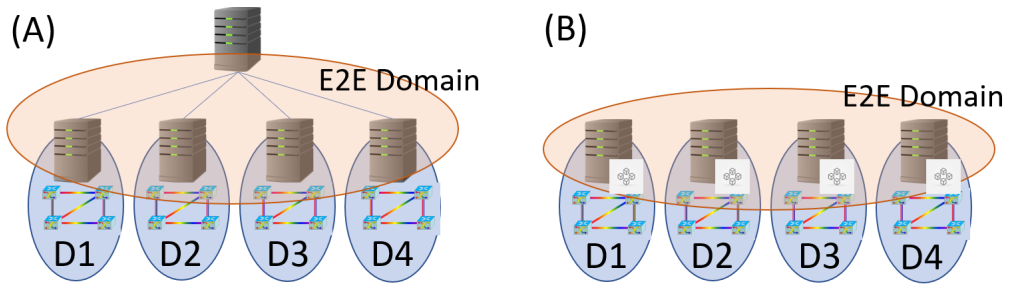


Figure 2.2: Hierarchical (A) and P2P/Blockchain (B) architectures.

### 2.4.1 P2P systems and Blockchain

One of the main problems of P2P networks such as the Emule example is the trust between peers. While in centralised architectures trust may be given and evaluated due to the existence of a central authority (i.e., component on top) like the cloud model presented by F.Di Cerbo et al. in [75], in distributed architectures like P2P networks this is not possible. So, any received data may be corrupted so that once opened and executed, it could harm the devices. A very interesting solution appeared few years ago with Distributed Ledger Technologies (DLTs) like Blockchain [76], specially since the work presented by S.Nakamoto [77] with Bitcoin. Since then, Blockchain has evolved in multiple ways allowing the possibility of using P2P networks for different services, from financial exchanges (e.g., Bitcoin) to new applications using the Ethereum presented by G.Wood et al. [78] or the Hyperledger [79], which is promoted by The Linux Foundation.

A DLT (i.e., Blockchain) is a digital system that records asset transactions (i.e., some information such as money, votes, resources, etc.) by saving the transactions and their detail in different places at the same moment. It might be understood as a geographically distributed DataBase (DB) with a set of nodes (i.e., peers) keeping the same information that all together create a P2P network. Blockchain allows to update the information in an iterative and secure way and, as there is no central authority, all the peers have the same rights to add or modify the data in the DB while maintaining it stable and safe using a consensus mechanism as described by W.Wang et al. [80].

Blockchain is tamper-proof for two reasons: a) due to the implementation of a consensus mechanism and b) the way the data is stored. The consensus mechanism is a procedure that follows a set of rules publicly known by all

the peers. So, if one of them tries to act in a malicious way, the others will notice and block its fraudulent action. For this reason, if there are any changes, they have to be applied to the majority of the nodes. About how the data is stored, it follows a block structure (e.g., a block is a set of data transactions between peers) and each new block is linked to the previous one. So, in order to modify a piece of data already stored, it is necessary to modify all the blocks after that specific data and again (in the majority of all the peers). The main characteristics of Blockchain are:

- Distributed: As the data is distributed and there is no central authority, the system is robust against hacks.
- Secure: All information in the DB is encrypted using private and public keys.
- Public: The system is more transparent as there is no central authority that tracks and validates all the information, but all peers do it.

Finally, one last capability regarding Blockchain is the fact that it is not just a geographically distributed data base anymore, some of the technologies available like Ethereum or Hyperledger added a feature capable of executing small pieces of code (i.e., Smart Contracts (SCs)) to carry out specific actions allowing the peers to work together without the need of a central element commanding any action. A SC only runs under certain conditions and it is used to make procedures more automatic, especially when a transaction is requested.

Blockchain has been already used on the management of computing and optical network resources as described by S.Kou et al. [81] and H.Yang et al.[82] or on security aspects such as the secured management of data flows presented by S.Boukria et al.[83] or the recovery of SND nodes after a failure described by S.Misra et al. [84]. Despite these multiple paths, most of the literature focuses on the demonstration of possible applications to manage SDN/NFV networks. For example, R.V.Rosa et al. [85] presents the idea of using Blockchain in a multi-domain environment with three possible scenarios. S.Fichera at al. [86] use Blockchain to keep track of SLAs events over a disaggregated network. Furthermore, H.Yang et al. [87] uses Blockchain to allow the sharing of information among a set of optical switches to calculate the best path possible across them and Y.Liang et al. [88] describes an algorithm using Blockchain to quickly configure switches to be controlled by the most optimal master when their initial master goes down or becomes evil.

In the previous papers and most of the network infrastructures and Blockchain literature, the focus is on the management of physical resources -e.g optical path calculation, traffic SLA fulfillment and switches management-, but there is few research looking into higher layer elements such as Network Services or network slices. Some examples are the works presented by J.Backman et al. [89] and B.Nour et al. [90] which focus on the use of Blockchain to create E2E network slices in a hierarchical architecture with a single Network Slice Manager on the top over different network domains.

One of the main Blockchain weaknesses related to the scalability is the speed to distribute and process it before it is accepted by the majority of peers and stored it in all of them. This aspect becomes very important in scenarios using SDN and NFV resources, especially in those dealing with multiple transport domains which belong to different operators. One example is the DC Interconnection (DCI) scenario which allows a new architectural model by giving the possibility of having third party cloud domains connected among them across flexi-grid Wavelength Division Multiplexing (WDM) networks of one or several telecommunications operators, avoiding packet-based aggregation networks. Using next generation plugabble coherent optics on their router devices, cloud operators can request a CS between DCs across multiple transport domains with a specific spectrum slot (e.g., 50GHz) instead of an amount of bandwidth (e.g., 100 or 200 GB/s). For this reason, taking into account that transport domains may contain a massive amount of information, it becomes interesting to study how a transport resources market may be implemented using Blockchain. To solve it, abstraction models for transport network domains were studied.

### 2.4.2 Abstraction models on Blockchain

Abstraction models [91] allow the reduction of the amount of information that an element in a control plane layer sends to the element on the upper layer as done in SDN control infrastructures. By doing so, the element on top has an overview of the domain network topology below instead of all the details, while keeping the capability of requesting CSs over the physical infrastructure.

The use of abstraction models allows the improvement on the optical network resource management in relation to the following issues: a) the complexity of having an efficient resources usage; b) the transport domains massive amount of information (e.g., nodes, links, node ports); and c) the

fulfillment of optical requirements such as spectrum continuity. Multiple works have studied to use either Blockchain, abstraction models or both of them to solve the previous issues.

For example, N.Đerić et al. [92] use two abstraction models called transparent and Virtual Node (VNode), the second one also referred to as Big Switch, to evaluate how the amount of information used in these different abstraction models may affect the available hardware resources (e.g., Central Processing Unit (CPU)) in an SDN hypervisor. M.Licciardello et al. [93] compares the Big Switch and the weighted Virtual Link (VLink) abstraction models in a DC Interconnection network scenario. While these two works presented interesting results, in [92] only two abstraction models were used omitting the VLink model and in [93], the authors assumed that all CSs deployed had wavelength continuity. R.Casellas et al. [94] used the capability to abstract network resources to manage disaggregated optical networks resources. The work done in [94] has been used as a reference to carry out the development of the infrastructure and of the results later described.

Regarding the use of Blockchain with abstraction models, S.Ding et al. [95] presented an algorithm to trade spectrum resources between Elastic Virtual Optical Networks (EVONs) using Blockchain. While their results are quite interesting and present a way to have a fair spectrum trade between EVONs, they do not describe the cost of distributing the network resources information in the Blockchain and its usage. H.Yang et al. [96] proposed a distributed Blockchain-based trusted multi-domain collaboration for mobile edge computing called BlockTC. In there, all SDN controllers share their topology information with the other domains, so they can verify if the routes selected are legitimate. In their work, they only use the transparent abstraction model (i.e., the complete topology), and so no comparison with other possible abstracted models is done to validate if the routes generated might still be legitimate while using less information. A.Derhab et al. [97] described a security architecture that integrates SDN and Blockchain technologies in order to improve the security among intercontroller communications using a reputation mechanism that classified the controllers using two possible historical-based actions strategies. Similarly, P.Gorla et al. [98] present an architecture based on Blockchain specifically dedicated to be used on edge domains as it allows different Mobile Network Operators to share their spectrum in order to achieve the most efficient use of their resources. While in [97] no optical aspects are checked and in [98] their focus is on spectrum resources in edge domains, our work fulfills both aspects: the use

of a Blockchain-based architecture to manage spectrum resources on optical transport domains. P.Fernando et al. [99] designed an architecture similar to the one presented in this work focused on security aspects such as the messages integration between controllers, the detection of malicious hosts and the SLA enforcement managing bandwidth resources. In their work no optical aspects are taken into account as bandwidth is used instead of spectrum to create the E2E CSs between domains. Finally, S.Fichera et al. [100] designed and developed a Blockchain architecture for optical multi-domain scenarios similar to the work later described, but with the objective of keeping the data record associated to multiple SDN events to validate their accomplishment and, in case of failure, to register which component is the responsible one. These last works did not check how the amount of information affects the Blockchain and it is studied in our work by using the different abstraction models.

The previous analysis was based on a multi-domain/operator scenario in which all of them participate under the same rules. But, in the near-future, the scenario will become much more complex as already described due to the "opening" of the networks towards a multi-stakeholder scenario. In this case with different players, one important aspect between players cannot be assumed as something that is granted and so, somehow has to be properly generated in the future distributed architectures. This aspect is the trust among the participants.

### 2.4.3 Trust in multi-stakeholder scenarios

Trust is a human concept based on the perception that somebody or something will do what it has to do in an expected way. Implementing trust is not an easy task to do, for two main reasons: a) trust is a subjective concept which makes it difficult to implement it into an analytical field such as the management and control of network resources, and b) because the common networks infrastructure model deployed is based on a single network operator/infrastructure owner with a closed group of well-known providers to accomplish the contracts relationship (i.e., SLA) with the client.

Now, with the beyond 5G (B5G) path opening, new requirements and scenarios are being discussed and one of them is to disassociate as much as possible the dependency between providers (i.e., Radio Access Network (RAN) Controllers, NFV Orchestrators, Software and Hardware, etc.) as it happens for example with disaggregated networks. Evolving from a scenario where

each vendor is an isolated island with few providers to a multi-stakeholder one (i.e., multiple providers, operators, etc.) will allow each service client to not depend on a single operator and its few service providers and so, operators and service providers will have to compete among them to offer the best possible services in terms of performance and security. The multi-stakeholder scenario will imply a change on the way the domains are managed and operated. Currently the operators have 3-4 providers with a 100% implemented and operative solution in each domain (e.g., RAN, Transport, etc.) across their island. However, this will change towards a model where the vendor islands will be smaller with multiple providers participating in the E2E service as the maintenance and updates costs of all these infrastructures will be high.

The dependency on few providers and the multi-stakeholder model implementation will increase the relationships between the different players/stakeholders. Joining the NFV, SDN and Network Slicing technologies together with the multi-stakeholder model will imply that the related risks associated to the deployed E2E service do not depend on a single vendor but multiple vendors in terms of software and hardware. So, it becomes necessary to have a transparent and reliable way for the service clients to evaluate and identify whether a provider is trustworthy enough to participate in the service deployment or not. This is a complex decision to take because trust is based on the confidence towards a trustee and, at the same time, this confidence is based on the entity's own and/or others' experience (i.e., reputation) with the trustee doing an action. For this reason, it becomes necessary to find a way to define trust. Trust cannot be directly mapped into the analytical world due to its subjective nature, but it can be generated using an element that generates measurable trust which is "Reputation" [101][102]. Based on the reputation that a trustee have about doing a set of tasks, that trustee will generate a level of trust. Keeping this in mind, in Chapter 6 the designed reputation and trust mapping is described.

The multi-stakeholder scenario implies that the creation and management of E2E services will involve more than just a single provider environment. For this reason, the multiple stakeholders should participate and collaborate to create the final E2E service in a trustworthy, transparent and reliable way. Following the path from the previous subsection and as one of the most promising tools to assist on the design and the creation of more transparent and secure systems, Blockchain brings the possibility to generate trust among the different participants in the multi-stakeholder scenario thanks to

its transparency, immutability and distributed capabilities.

The idea of managing resources and generating trust among the multiple players is being widely studied for its importance in terms of security. C.Benzaid et al. [103] presented a set of trust models to propose how trust should be understood within the 5G and future networks. N.Lo et al. [104] focuse on a two module-based system to manage the resources allocation in a multi-cloud scenario in which the multiple cloud managers are aware of the actions done regarding their resources. M.Arasteh et al. [102] designed an access control model for the resources owners to participate together in a single virtual organization. While both works [102][104] focus on a multi-provider scenario, the solution they presented follows a centralized design (i.e., with a single element on top) which centralizes the whole process and so, it may lead to the possibility of becoming a bottleneck and cause a central point of failure situation that some entity might gain advantage over. An article whose topic is closer to the work later described was presented by V.Casola et al. [105] with a Trust Manager component called "TruMan" that aims to manage the best provider selection possible based on a set of requested requirements. While their work is quite similar to previous ones, we foresee that the problem of centralizing trust in a single element is that the information within this element may somehow be tampered. In terms of distributing the trust management, D.Wang et al. [106] presented a blockchain-based trust management framework specifically designed for vehicular scenarios, S.Malik et al. [101] presented an IoT supply chain environment and C.Balachandran et al. [107] designed an agnostic blockchain system focused on SDN domains. Compared to these three last works, the work later described shows both: a) the mathematical model designed to compute trust in the multi-stakeholder scenario, and b) the designed system and how the trust and reputation values evolve based on the service requests management.

### 2.4.4 Beyond SoTA

Based on the literature presented in the previous subsections, Chapters 5 and 6 describe the work and research done to complement the gaps commented.

Chapter 5 focuses on studying the possibility of changing the management model of an E2E multi-domain hierarchical NFV/SDN/Network Slicing architecture towards a mesh architecture using Blockchain where each domain in the multi-domain NFV/SDN network scenario has its own NFV/SDN architecture with a Network Slice Manager or Transport SDN Controller and

a new Blockchain-based component on the top. Through this new component, the different domains share their local resources and collaborate among their equals using the Blockchain technology to deploy E2E network slices (i.e., computing and transport resources) across domains. Then, when a vertical needs a new service, it requests the E2E network slice to its domain Network Slice Manager which will take care of the whole Slice deployment with the collaboration of the other Network Slice Managers and SDN Transport Controllers using Blockchain. Moreover, Chapter 5 presents the new Blockchain-based component through its internal architecture and functionalities, the process to deploy E2E network slices from following a distributed management point of view instead of a centralized one. Within the architecture and component description, a step by step process is followed. At first only computing resources were used to introduce the main concepts and later the networking resources were added to complete the E2E network slices and ensure that the different slice-subnets are properly stitched across transport domains by deploying the appropriate CS. Regarding the CS, a specific study is presented focused on comparing three abstraction models (i.e., transparent, VLink and VNode) for transport context information to evaluate the performance of the Blockchain when dealing with different amounts of information to deploy E2E CSs with an ensured spectrum continuity alongside all its Domain CSs.

Based on the experience obtained in the work of Chapter 5, Chapter 6 focuses on how Blockchain may bring trust on a multi-stakeholder scenario using again a Blockchain-based architecture. In this case, a new Blockchain-based Trust Manager component was developed and evaluated with the objective of assigning a continuously evaluated trust value to the different players involved in the services deployment. The internal architecture and capabilities are described and then experimentally evaluated.

# Chapter 3

# Management and control of multi-domain and hybrid network slices

## Contents

Networks are composed with multiple domains. Each one with specific characteristics based on their closeness to the final users and the amount of requests/information that had to reach that domain or go through it. Three are the basic domains: a) access - where the users are placed and their traffic is aggregated, b) transport - the paths used for the aggregated user traffic towards where the services are placed and, finally, c) core - where the services were stored and saved to be accessed by the users. While this basic structure was kept for so many years, an equal powerful step was to distribute some resources from the core domain towards the access domain. With that in mind, small scale DCs were placed with enough computing resources (i.e., processing, memory and storage) to solve either the most common service requests and also those with low resource demands in the access (or edge) domain while keeping the biggest centralized DCs in the core for those service requests requiring more powerful resources. Moreover, the fact of having DCs in the access domain, allowed the reduction of traffic congestion in the transport networks, making the service deployments that really need to access the core to have less obstacles and interference. With this in mind, Network Slicing deploys virtual networks following the same principle when possible. And so, using the access domain DC resources or if necessary, both of them in a single network slice by using the connectivity resources offered by the transport network domains.

Because each domain (and its resources) has its own characteristics, also the technologies used in the different network elements evolved and got adapted to improve the performance in each domain. Initially, from Physical Network Functions (PNFs) based on hardware applications to the VNFs [108] (i.e., software applications) using a KVM. Finally, the latest evolution is the use of containers-based solutions which allow to move from software application to micro-based applications. So, now, VNFs may be substituted by CNFs (i.e., container-based elements) [109] bringing an agile methodology, short updates cycles and more control over the applications.

This chapter describes the work done on the design and use of network slices. On the one hand, it focuses on different technologies to compose and deploy what we called "hybrid network slices" and, on the other hand, it describes the implementation and management of network slices with different NSs deployed in different domains and linked across Transport domains. This chapter is based on the work presented in [110] and [111].

## 3.1 Network slicing management on transport multi-domain infrastructures

Before working on security aspects around Network Slicing, it is important to understand the element in charge of managing the network slices (i.e., Network Slice Manager) and how it interacts with the rest of the NFV architecture as defined by standardization bodies such as the ETSI [112]. As introduced in Chapter 2, nowadays there are multiple open-source NFV-O options available in the market such as OSM [36], ONAP [35], SONATA-NFV [32], etc. Whichever is the chosen NFV-O solution, the overall NFV architecture will have the same elements as presented in Figure 3.1.

Four element types can be found in the architecture: a) the SDN Controllers (i.e., WIMs) which control the connectivity service between different points in order to interconnect the services deployed, b) the DC Controllers each with the VIMs able to manage one or another type of virtualization technology between VNFs and CNFs, c) the NFV-O on top to manage the previous two types of controllers and finally, d) the Network Slice Manager which, through the NFV-O, takes care of controlling and managing the correct deployment of network slices using the best combination of domain resources.

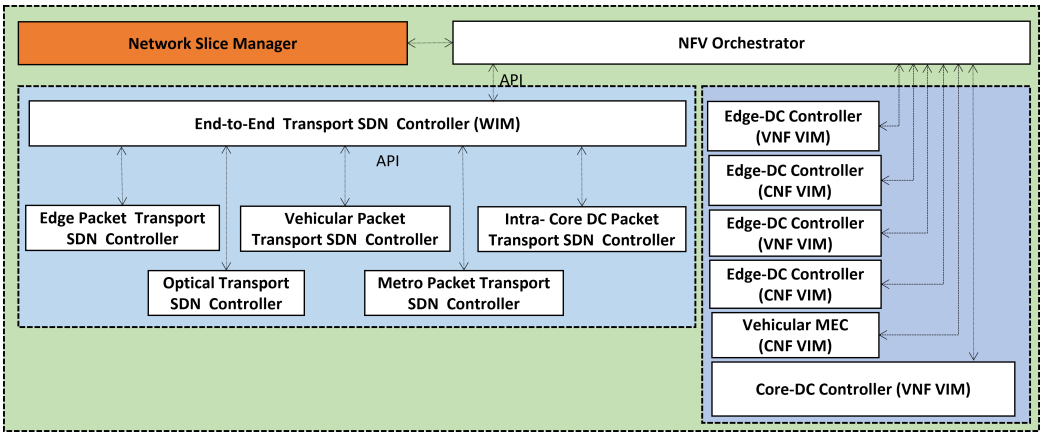


Figure 3.1: Network slicing and NFV management architecture.

Network slices were thought to be an element to improve the resources (in DCs) distributed in different domains while having multiple instances over the same physical infrastructure. During the work done to develop

the Network Slice Manager, the following functionalities were identified: a) the possibility to share a NS between different network slices if possible, b) the deployment management in different domains with DC resources (i.e., edge and core), c) related to the previous, the capability to interconnect the deployed elements in different DC domains and finally, d) the management of network slices using different technologies (this topic is later discussed in the following section).

As any manager module, a set of data objects is necessary to define and describe what it can be, it is or it was deployed. Due to the close relationship with the NFV data objects, a Network Slice Manager follows the same way than NFV-Os and it bases its way of working on the following two objects:

- Network Slice Template (NST): Firstly defined by the GSMA[113] and later on adopted by the ETSI. An NST is a static definition of the elements and characteristics to create the virtual network for the requested service with specific characteristics like the NSs composing the network slice, how they are interconnected and more information such as the QoS data.

- Network Slice Instance (NSI): Using the NSTs as reference and as described in [112], an NSI is the information of a deployment based on a NST. Each NSI keeps a high level information to manage the virtual resources deployed in each VIM and WIM composing a requested network slice.

When deploying a network slice defined in an NST, the final outcome is the creation of the associated NSI decomposed in NFV data objects (i.e., NSs and VNFs) which are finally mapped into two more types of data objects. On the one hand, the VIM requests to deploy the Virtual Machines (VMs) defined in the VNFs descriptors and, on the other hand, the connectivity requests managed by the WIM in order to interconnect the deployed VMs placed in different domains.

Based on this correlation of the different data objects, the deployment of a network slice in a multi-domain infrastructure was designed as Figure 3.2 with the following steps:

First, the vertical requests to the Network Slice Manager a network slice based on a NST (step 1). Once the Network Slice Manager receives the request, it creates the NSI and completes it with the NSs placement information (step 2). Once the NSI initial information is ready, the Network Slice

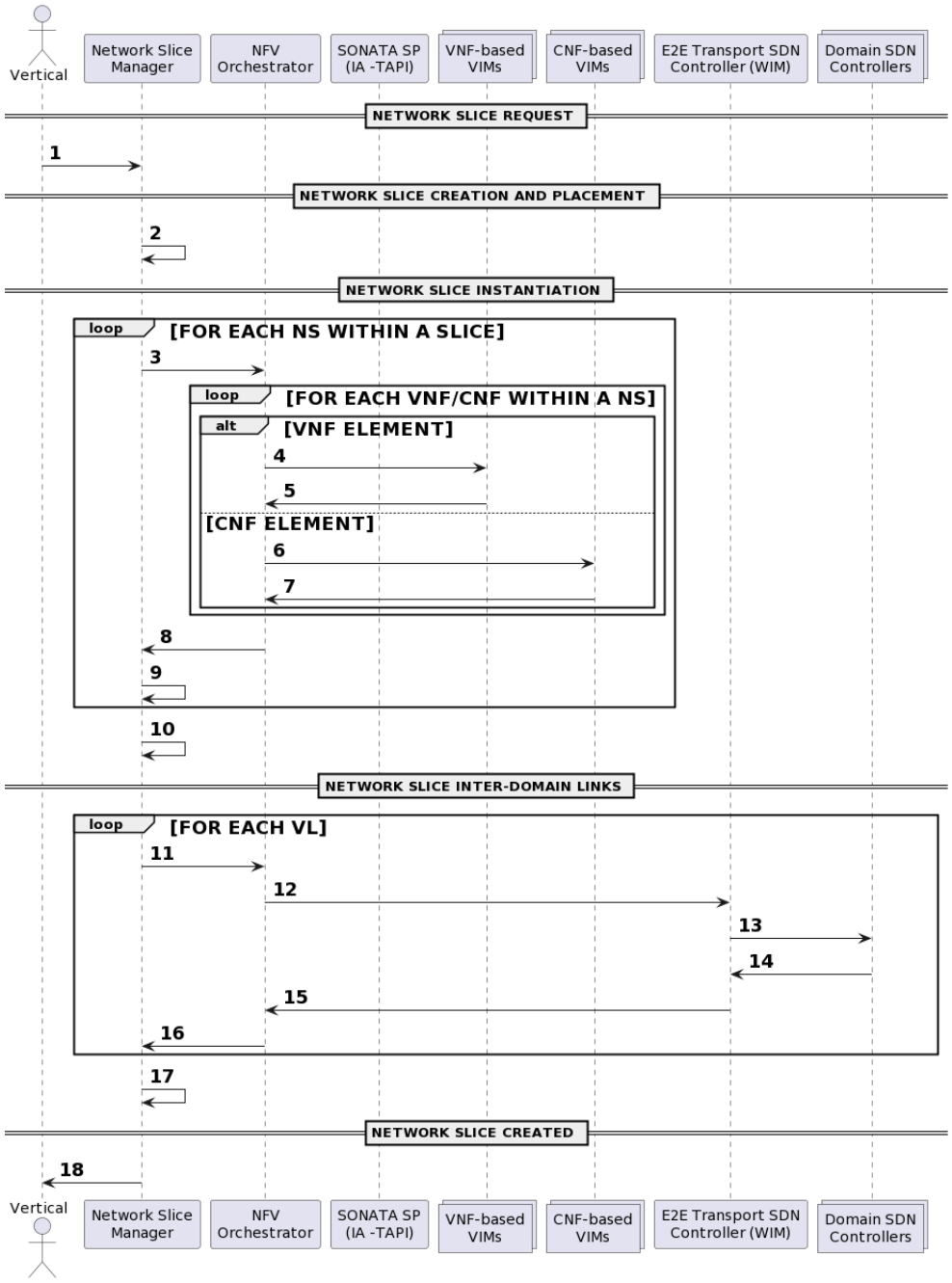


Figure 3.2: Network slice deployment.

Manager requests to the NFV Orchestrator the deployment of each NS (step 3) based on the NST and specifying which VIM must deploy them and so, the domain (steps 4 to 7). For each NS ready (step 8), the Network Slice Manager updates the NSI data object and follows with the missing NSs and it waits until all the NSs requests are deployed (step 9) to update the NSI data object (step 10). If the NSs are placed in different VIMs (i.e., domains), the Network Slice Manager requests through the NFV Orchestrator (step 11) to the E2E Transport SDN Controller to create the inter-domain virtual links (VLs) to connect the NSs (step 12). Then, the E2E Transport SDN Controller takes the necessary actions to configure the CSs to compose the VL within the network slices and forwards the requests to the selected Domain SDN Controller (step 13). Once the E2E Transport SDN Controller is informed, it forwards the information through the NFV-O (step 15) towards the Network Slice Manager. So this can update the NSI data object (step 17) and, if everything is ready, finally inform about the complete network slice deployment (step 18).

The CSs required in a multi-domain network slice deployment are configured with the following procedure. When the creation of a VL is requested, the E2E Transport SDN Controller receives the request with the QoS requirements and establishes the path either on the packet or the optical transport domains. Once the path is computed and the resources selected, two CS (i.e., two unidirectional flows) are configured allowing the possibility to perform load-balancing or priority routing.

It is important to remark that while the workflow includes the use of VIMs based on CNFs, the study case of network slices using different technologies is presented in the following section of this chapter, with its specific management details.

## 3.2 Hybrid network slices: combining technologies within a network slice

Up until some years ago, the most common technology of virtualization was KVM which was adopted by the NFV standard through the definition of VNFs. Each VNF is mapped to an amount of resources such as storage, memory, CPU and networking typically found in cloud environments. Despite this technology adoption allowed a very important improvement on the

efficiency of the resources management, operators still look for other parameters in order to have the best performance of their resources. Among other actions, an important one is the time that a service needs to be deployed and ready for the final users, the deployment time.

As previously described, one option was the distribution of computing resources in domains closer to the users, but it was not enough. At this point is where the use of CNFs appeared to be deployed together with VNFs, on what we called hybrid network slices. CNFs [114] make use of the container technology which allows a flexible, efficient and easy way to produce micro service-based applications and finally to deploy them faster than VNFs.

The usage of multiple technologies does not affect on the overall design of the Network Slicing and NFV management infrastructure (Figure 3.1) but it adds new requirements into the previously presented functionalities that a Network Slice Manager must have (i.e., NST/NSI life-cycle, NS sharing management, multi-vim deployment, etc.). Now the Network Slice Manager must be aware of the type of components composing each NST, so during the deployments it is necessary to select the correct VIM element in terms of the technology (i.e., KVM or container) aside of its location (i.e., edge or core). As illustrated in Figure 3.3 where two deployed hybrid network slices are represented, the Network Slicing and NFV management infrastructure keeps the same structure as presented in the previous section with the Network Slice Manager and the NFV-O on top controlling all the network slice and network services life-cycle [26], and below, all the VIMs and WIM elements managing the virtual deployed resources.

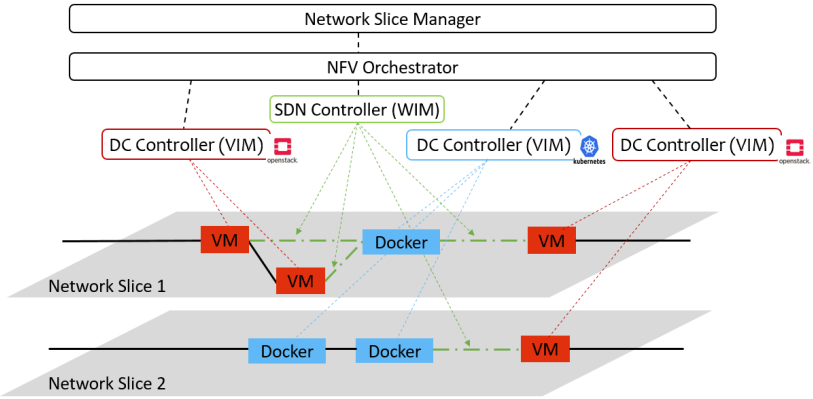


Figure 3.3: Hybrid network slices architecture.

While this work was being implemented, an important condition that affects the way hybrid network slices should be managed was found: the order in which the VNFs and CNFs are deployed for their interconnections (i.e., virtual links). The reason for this is due to the way each one of these two implemented technologies works. A VM is independent from other VMs (e.g., each VM has its own IP address) but containers are not independent in terms of networking as they are identified by port. Because of this reason, when deploying a Hybrid Slice it is important that all VNFs are deployed before the CNFs in order to pass the correct VMs information to the containers, so this second element may point to the correct IP address.

### 3.2.1 Data objects evolution to manage both technologies

By the time our work on hybrid network slices was done and presented, the two most known open-access NFV-O projects had not implemented this capability yet. While on the ONAP [35] side the management was done through the usage of Cloudify, on the OSM [36] side, the CNF orchestration was not even available. With our work in the EU 5GPP 5GTANGO [33] and its NFV solution called SONATA-NFV [32], among the different components within its architecture (Figure 3.4), the main contribution was on the design and development of the Network Slice Manager. The work well-done in that process, allowed us to contributed to the development of the Network Slicing Manager in the ETSI OSM and later, served as an initial reference for their CNFs management within the following Network Slice Manager versions.

In order to have the possibility to deploy both technologies and have hybrid network slices within the SONATA-NFV solution, the CNF descriptor (CNFd) data model and a Kubernetes wrapper were implemented to cover the deployment of the CNFs. To do so, the VNFd data model was used as a reference and converted into a Kubernetes object allowing the functions (previously deployed as VNFs) to be deployed as a CNFs over a Kubernetes cluster.

#### VNF/CNF descriptors and records

In order to manage both technologies, the data models used in our work had to take into account their singularities. Listings 3.1 and 3.2 shows two yaml file examples with a CNF and a VNF information, respectively. In both their

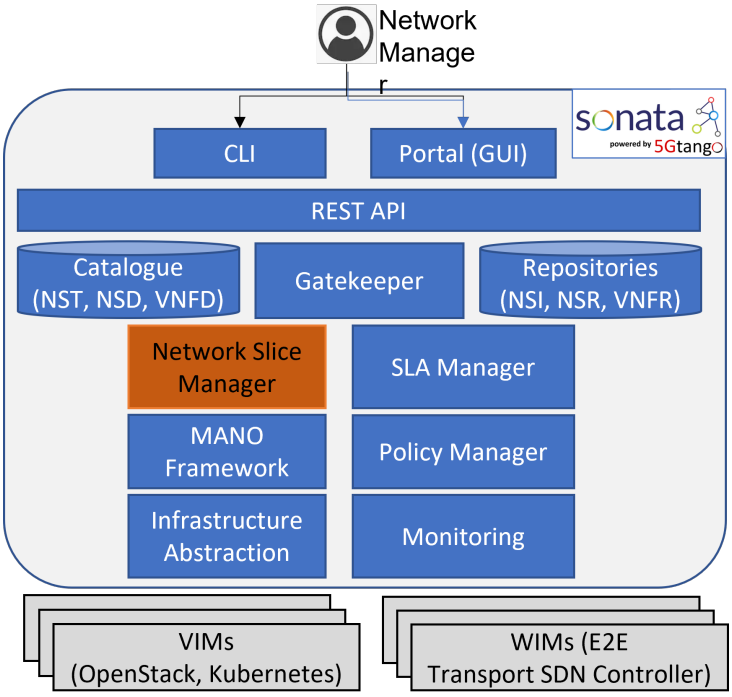


Figure 3.4: SONATA-NFV service platform architecture.

deployment requirements demanded different types of information. As it can be seen in the yaml items, the CNFs required to define the container ports to model the Kubernetes load balancer and allow the access from the outside world, meaning to configure its internal virtual links as an "E-TREE" [115] (i.e., point-to-multipoint) type to map external port to the internal ones. On the other hand, the VNFs defined the resources (i.e., CPU, Random Access Memory (RAM) and storage) necessary to work properly and the virtual links to connect the VMs in a VNFs could use the "E-LINE" type to have point-to-point connectivity. Having the similar data model for both technologies, allows the NFV-O to optimise the functionalities developed initially for the VNF and just adapt them to the CNF objects.

Listing 3.1: Example of CNF descriptor..

```
vendor: "eu.5gtango"
name: "cnf-mse-cache"
version: "0.9"
author: "eu.5gtango"
```

```
description: "Media-se cache VNF descriptor."
cloudnative_deployment_units:
  - id: "cdu01"
    image: 5gtango/media-cache:dev
    connection-points: ...
    parameters: ...
  - id: "cdu02"
    image: sonatanfv/tng-stats-collector:test
    parameters: ...
connection_points:
  - id: "hls"
    interface_cp: "ipv4"
    type: "serviceendpoint"
    port: 80
  - id: "api"
    interface_cp: "ipv4"
    type: "serviceendpoint"
    port: 5000
virtual_links:
  - id: "hls-link"
    connectivity_type: "E-TREE"
    connection_points_reference:
      - "hls"
      - "cdu01:int-hls"
  - id: "api-link"
    connectivity_type: "E-TREE"
    connection_points_reference:
      - "api"
      - "cdu01:int-api"
```

Listing 3.2: Example of VNF descriptor.

```
vendor: "eu.5gtango"
name: "vnf-cms"
version: "0.1"
author: "eu.5gtango"
description: "Media-cms VNF descriptor."
virtual_deployment_units:
  - id: "vdu01"
    vm_image: "http://google.es"
    vm_image_format: "qcow2"
    vm_image_md5: "dba700c13ddd8019ffb3c897a0de38da"
    connection_points: ...
    resource_requirements:
      cpu: ...
      memory: ...
```

```
      storage: ...
connection_points:
  - id: "cpexternal"
    interface_cp: "ipv4"
    type: "external"
  - id: "cpinternal"
    interface_cp: "ipv4"
    type: "internal"
virtual_links:
  - id: "vlexternal"
    connectivity_type: "E-Line"
    connection_points_reference:
      - "vdu01:external"
      - "cpexternal"
  - id: "vlinternal"
    connectivity_type: "E-Line"
    connection_points_reference:
      - "vdu01:internal"
      - "cpinternal"
```

Another important aspect we had to take into consideration for the correct deployment of the hybrid network slices is the following: within each element in the "cloudnative_deployment_units" key section in the CNF (Listing 3.1), a new yaml key was added called "parameters". This new key allowed to influence the deployment of the Cloud-native Deployment Units (CDUs) composing the CNF by sending the environmental variables for the Kubernetes config-map to have the reference towards the previously deployed VNFs.

Finally, the data objects to manage the deployed elements (i.e., VNF/CNF records [116]) were also extended with the key element "cloudnative_deployment_units". This was done so that the NFV-O could store runtime information of the CNFs such as the "load_balancer_ip", the "cdu_reference", the "cdu_instances" and other.

### NS descriptors and records

At the NS level, the use of CNFs did not involve any modification in their descriptor or record data objects.

### Slice descriptors and records

At the network slice level, as already introduced, the most interesting change was applied on the way the hybrid network slices were deployed compared

to those composed only by VNFs; first the VNF-based NSs and then, the CNF-based NSs. Regarding the network slice descriptor (i.e., NST) and record (i.e., NSI) data models, they did not suffer as many modifications as it happened at VNF/CNF level because from the point of view of a network slice, the elements below are treated as nodes and links without the need to know which service is being deployed. The main modification within both, the NST and the NSI, was to have the right VIM information based on the location and the technology for each of its NSs.

## 3.3 Experimental validation

In order to evaluate and validate all the aspects related to the architecture presented in Figure 3.1 and the related workflows and management of bot: multi-domain and hybrid network slices part of the work done was the design and development of a Network Slice Manager solution.

Based on the initial design presented in [117], the final solution made possible to manage all the previously described actions and data objects. Originally, the developed Network Slice Manager was an external element with its own DB that later was integrated within the SONATA-NFV [32]. This implied some modifications such as the removal of the DBs and merging with those already available in the SONATA-NFV. Moreover, the way this NFV-O worked required that any request had to pass through a centralised component called "Gateway" that kept control of all the asynchronous processes and added certain security level.

All the experimental results presented in the following sections were done using the ADRENALINE testbed infrastructure (belonging to CTTC) presented in Figure 3.5. On top of it, the Network Slice Manager developed together with the SONATA NFV-O were placed to manage the service deployments across the different VIMs and the transport domains. Below it has the different VIMs placed in the multiple Edge and Core DCS with both OpenStack (for the VNFs) or a Kubernetes (for the CNFs) solutions for the virtual computing resources. In parallel to the VIMs, the WIM with the Transport SDN Controller [118] on top with a set of SDN controllers below following the IETF Application-Based Network Operations (ABNO) architecture [119]. The SDN controllers use an ODL solution on the packet-based domains and an Open Line System (OLS) in the optical domain. Finally, as an important remark, the Transport SDN Controller communicates with the

NFV-O above and the Transport SDN Controllers below using the Transport-Application Programming Interface (T-API) [120] data model defined by the Open Networking Foundation (ONF).

Finally, the physical network infrastructure has different computing domains (edge and core) interconnected by multiple transport networks based on packet and optical technologies. Ten OpenFlow switches are distributed in the multiple packet-based networks and are controlled using Open vSwitch (OVS). Regarding the optical-based network, it is designed as a Photonic Mesh Network (PMN) managed by an OLS SDN controller. One last and important aspect is the fact that the Transport SDN Controller may use the packet-based or the optical transport networks based on the QoS parameters requested. The testbed has a "Vehicular packet Transport Network" domain dedicated to Vehicular communications scenarios and also the possibility to interact with another testbed focused on mobile communications.

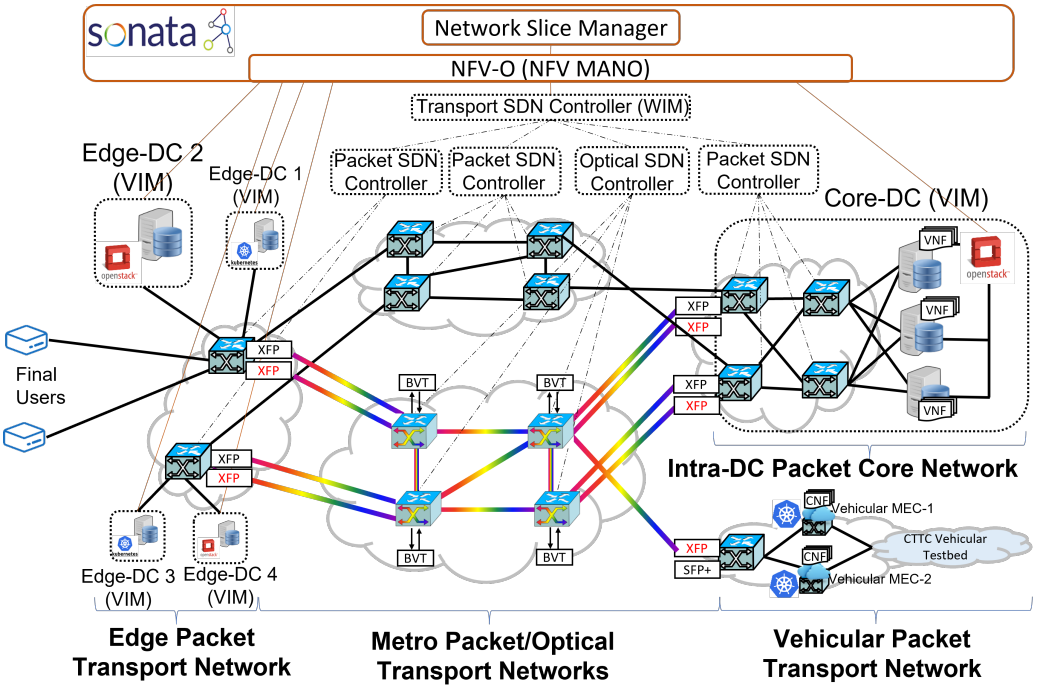


Figure 3.5: CTTC's ADRENALINE testbed and SONATA NFV MANO infrastructure.

### 3.3.1 Multi-domain Network Slicing service validation

The first of the results to be presented is the validation of the presented workflow in Figure 3.2. To do so, the Hypertext Transfer Protocol (HTTP) messages exchange between the Network Slice Manager and the rest of the SONATA-NFV modules during a network slice deployment procedure were collected and they are presented in Figure 3.6. In there, an NST with three NSs placed across the Edge-DC1 and the Cloud DC in Figure 3.5 is deployed.

Figure 3.6 shows how the vertical requests to the Network Slice Manager an E2E network slice based on a NST (1). The Network Slice Manager gets and processes the NST descriptor, creates and stores the NSI into the Repositories in less than 17s (2). With the NSI ready, the Network Slice Manager takes around 80s to request the next elements (3): a) the intra-domain VLs creation (/slices/networks) to connect the VMs within a single NS, b) the NSs deployment (/requests) to create the VMs in the correct VIM, and finally, c) the NSI record (/records/nsir/ns-instances/...) is updated and stored in the Repositories (i.e., SONATA-NFV DB). While deploying, the Network Slice Manager keeps checking that all the NSs are ready (4). Once they are ready (after 409s), the Network Slice Manager requests through the Gatekeeper to Infrastructure Abstraction (i.e., SONATA-NFV module in charge of requesting the CS to the WIM) all the necessary inter-domain links creation. Then, the Infrastructure Abstraction requests to the E2E SDN Controller to create the inter-domain links over the physical network, requiring around 11s to do it (5). Finally, when the network slice is ready, the Network Slice Manager updates the NSI information and informs the Gatekeeper that the whole process is done (6).

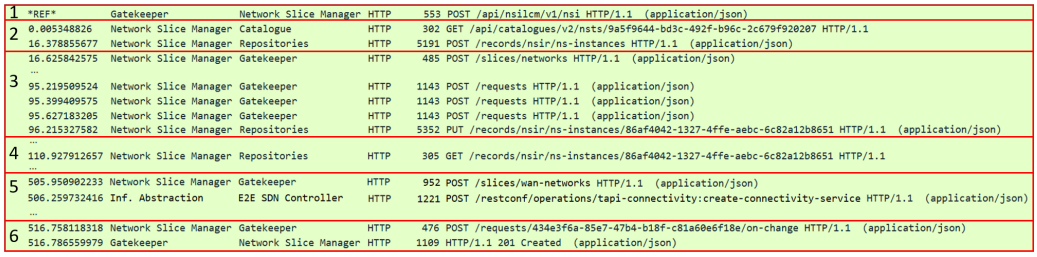


Figure 3.6: Network slice deployment.

This whole process took 516 seconds -i.e., 8 min and 36 seconds- to deploy 3 NSs composed by 2VNFs each -i.e., 6 VMs- placed in different VIMs (i.e.,

edge and core) and to interconnect them through the transport domain.

### 3.3.2 Definition, deployment and experimental validation of hybrid network slices

Taking the work done in [121] in which a smart manufacturing NS based on CNFs was implemented, the work done with hybrid network slices presented the use of CNF technology on another 5G scenario; an IMmersive-Media (IMM) use case. Compared to [121] where they use two individual NSs (Figure 3.7 up), the results presented in our work were obtained using an hybrid network slice that combined NSs based on both technologies: VNF and CNF (Figure 3.7 down). The IMM use case was designed to allow a user to receive multiple information flows in parallel by watching a video-streaming event while having the possibility to check social media accounts. The use of hybrid network slices within the IMM use case was part of a presented pilot in [122, 123] that combined both technologies to overcome the weaknesses of each technology with the strength of the other (i.e., the lightness of containers together with the safeness of virtual machines).

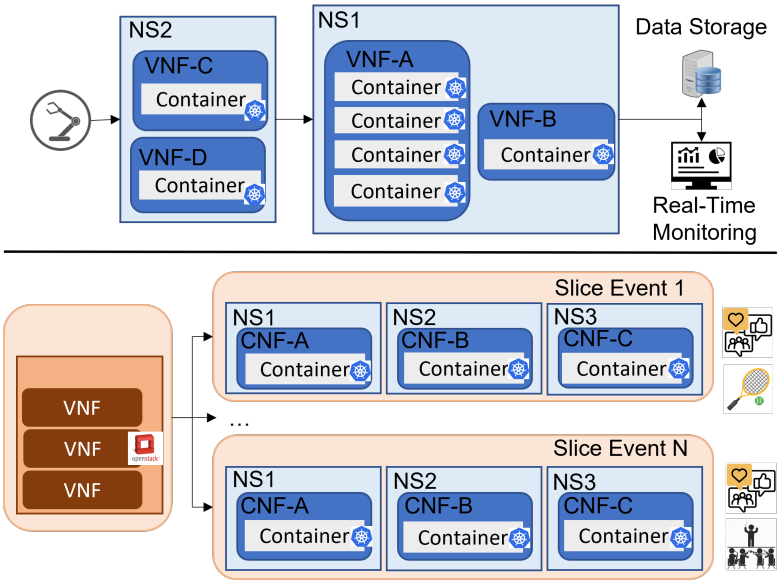


Figure 3.7: Smart manufacturing scenario (up) and IMM scenario (down).

As Figure 3.8 presents, the internal architecture of the IMM deployed ser-

vices requires the design and deployment of two NSTs. First a NST (green square) with a NS based on VNF technology called Content Management System (CMS-NS). The CMS-NS had the capability to be shared among multiple deployed network slices. The second NST (red square) was composed of the same shared CMS NS and three more, all of them using CNF-based NSs: Media Aggregator (MA-NS), Media Streaming Engine Transcode (MSE_transcode-NS) and Media Streaming engine Cache (MSE_cache-NS).

The way this use case worked had the following steps: The NST with the CMS-NS (green square) was deployed and left active to wait for users to connect and watch an event. Then, for each new user (i.e., N users means N deployments), a new deployment using the second NST (red square) is used to deploy only the CNF-based NSs connected towards the already deployed and shared NST (green square), composing the final hybrid network slices. As already introduced, this order had to be strictly followed because to interconnect the CNFs-based NSs with the VNF-based NS, the first required a set of instantiation parameters for the CNFs to point towards the shared CMS-NS to have a allow data and control traffic to flow among them.

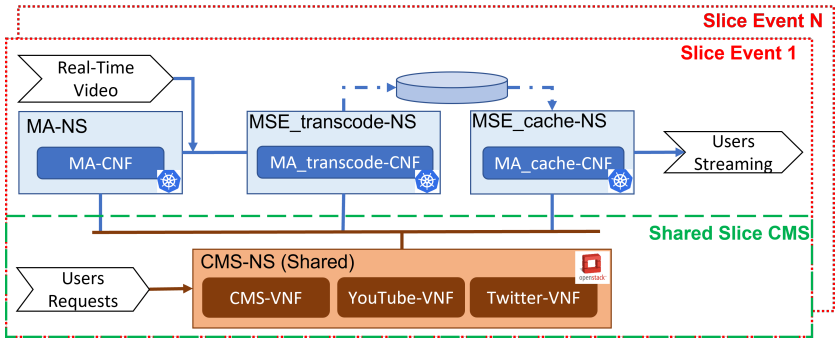


Figure 3.8: IMM hybrid network slices architecture.

The used NSs illustrated in Figure 3.8 are:

- CMS-NS: It is used in both NSTs and composed by three VNFs, one being the entry-point of the immersive streaming service (i.e., CMS-VNF) and the other two VNFs with the Social Network applications to be watched in parallel to the main video streaming.

- MA-NS: It is the proxy receiving the Real-Time Messaging Protocol (RTMP) video flow from the cameras event and it redirects those videos to the right Media Streaming Engine inside the same network slice.

- MSE_transode-NS: It takes care of processing the RTMP data sent by the MA-NS by adapting the video into different support bit rates, to segment the video in chunks (HTTP Live Streaming protocol) and finally, to save it.

- MSE_cache-NS: It has a web server serving the multiple stored video events.

Similarly to the previously presented results in Subsection 3.3.1, the deployment time is used in here too to evaluate the difference between the two possible cases: hybrid network slice using both VNF and CNF based NSs and a non-hybrid network slice composed only with VNF-based NSs. Finally, as the two defined network slices cases (i.e., Hybrid and non-Hybrid) use the same CMS-NS, its deployment time was not taken into account and only the deployment time for the second NST and its three NSs (i.e., MA-NS, MSE_transcode-NS and MSE_cache-NS) was used.

Figures 3.9 and 3.10 show the Cumulative Distribution Function (CDF) with the probability that an instantiation or termination process lasts less than X seconds. Checking both figures, the instantiation of an hybrid network slice has a 90% probability to be deployed in less than 100s while a non-hybrid network slice obtained a 80% probability to require more than 600s to be fully ready.

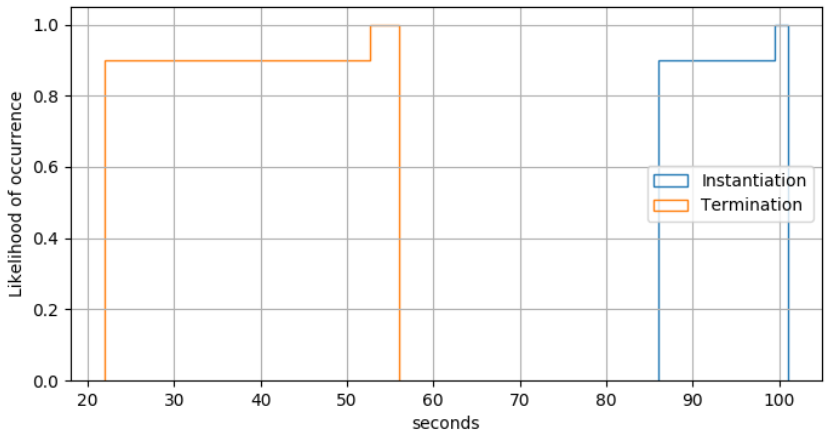


Figure 3.9: Hybrid network slice instantiation/termination CDF vs. likelihood of occurrence.

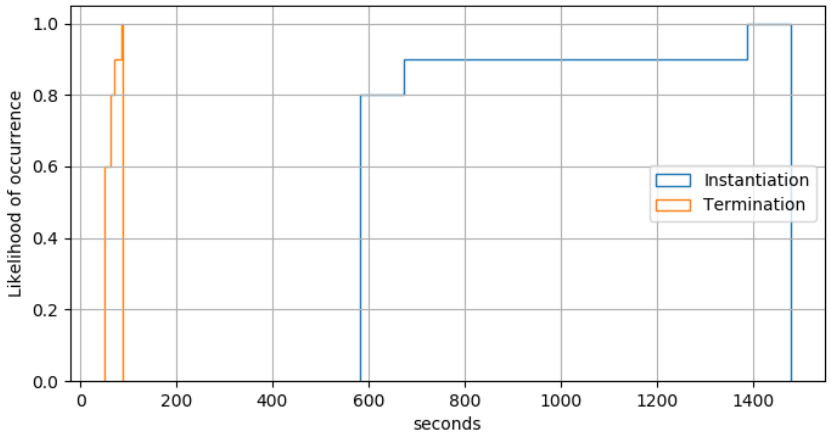


Figure 3.10: VNF-based network slice instantiation/termination CDF vs. likelihood of occurrence.

In terms of time deployment, the Hybrid network slice has a mean value of 87.5s (standard deviation of 4.5s) compared to the other case that requires 707.1s (standard deviation of 258.556s), around 8.1 times higher. On the termination procedure side, the hybrid network slice required 27.7s (standard deviation of 10.11s) against the 60.1s (standard deviation of 11.99s) required by the non-hybrid network slice, around 2.17 times.

Finally, comparing the results from the previous Subsection 3.3.1 (i.e., 516s of deployment time) and those just presented (i.e., 707.1s and 27.7s), the advantages obtained with the combination of virtualization technologies on the creation and management of network slices in terms of deployments and termination times but also to compensate the weaknesses of each virtualization technology with the strengths from the other are clear.

## 3.4 Conclusions

In order to reinforce and complement the introductory knowledge about SDN, NFV and Network Slicing technologies described in Section 2.2, this chapter illustrated how these three technologies were used in a multi-transport scenario to deploy network slices. Moreover, the use of different virtualization solutions to compose network slices is presented.

Despite its complexity, the obtained results show that it is worthy to implement the Network Slicing infrastructure because of the benefits obtained

from it. First, because the control and management of networking and computing resources improves in terms of dynamism when deploying multiple service in parallel over the same infrastructure. Second, dividing the resources in different domains allows a more controlled management of these resources. Finally, the use of the different virtualization solutions depending on the domains specific characteristics allows an improvement of the deployment time and so, the control and management becomes more efficient.

# Chapter 4

# Enforcing Quality of Service and security on network slices

## Contents

As previously described, Network Slicing [7] means the coexistence of the multiple verticals and their services over a single and common 5G infrastructure. This implies that the control and management system must take care of ensuring and enforcing the expected QoS based on the requested KPIs) -e.g., bandwidth, latency, etc.- defined in the metrics of the SLAs between the client and the service provider. For example, the requirements when deploying an e-health application will be quite different compared to those defined for an automotive service, in both terms: network performance and security.

The enforcement of the QoS for each deployed service is accomplished through the use of a monitoring system to detect when the KPIs are not respected, the SLAs are violated and, if possible, to trigger the right solution to correct the situation. The designed and implemented architecture focused on the service performance. However, there is another way to keep the desired QoS: the service security. Instead of requesting network resources to increase the bandwidth, it might be possible to maintain the QoS by defining a set of parameters related to information security instead of information traffic. A SSLA looks towards defining a set of requirements in order to ensure the safeness of a service -e.g., information integrity, encryption, etc.- in front of a possible problem affecting the information and accessing the service.

This chapter is divided in three sections and it describes the work done regarding monitoring systems and automation loops to ensure the correct behavior of deployed network slices in terms of service performance and security. The first section introduces the main concepts related to the monitoring actions and describes the architecture required to enforce the expected QoS. The second section extends the work presented in the first section by describing a framework that allows a network slice provider (i.e., the owner of the network slice descriptors with the verticals high-level requirements) to act as a broker relying on several service providers offering multiple network services to deliver network slices associated to SSLAs for the verticals/end-users. This task was done with a complementary architecture specifically dedicated to deploy and monitor the necessary security elements to enforce the expected QoSec. The third section presents the obtained experimental results related to all the work done in the first two sections. Finally, this chapter is based on the work presented in [124], [125] and [126].

## 4.1 Quality of Service on network slices

The control and management of network slices and the actions required to ensure the appropriate QoS involve a set of multiple and different elements such as the control and management infrastructure (i.e., NFV MANO, SDN Controllers, Computing and Networking resources, etc.) and the players that interact with the infrastructure. Due to the existence of multiple players, each with its specific role, it is important to identify who and what is involved in the multiple steps composing the process to deploy network slices and to enforce the QoS on them. To do so, the whole process is described; first by mapping vertical KPIs within the network slice descriptor using a QoS classification. Then, based on this QoS classification, by selecting the appropriate SLA for each NS composing the network slice and, finally, based on the selected SLAs and NSs,by deploying the corresponding virtual instances with the correct resource configuration to enforce the expected QoS.

To accomplish all the process description, a vertical KPI-enabled NFV MANO architecture was designed and it is illustrated in Figure 4.1. In this figure, three different entities may be identified: the set of components composing the architecture (rectangular-shape modules), the different actors (circle-shape modules) such as Verticals, Developers and service providers that interact with the corresponding components and, finally, the multiple actions between the components architecture and the actors.

The designed system allows to define customized SLA with different QoS parameters and to monitor them in parallel allowing the coexistence of different requirements defined by the 5G services (i.e., Enhanced Mobile Broadband (eMBB), Ultra-reliable and Low-latency Communications (uRLLC), and Massive Machine Type Communications (mMTC)). For this reason, the implemented system can manage different network slices and each with its own QoS requirements being monitored in parallel.

### 4.1.1 KPI-enabled NFV-MANO architecture

The designed architecture improves the one described in Chapter 3 (Figure 3.1). The new architecture complements those components already presented (i.e., Network Slice Manager, NFV MANO, VIM and WIM) with the new one to fulfill the QoS monitoring. As presented in Figure 4.1, the components are illustrated in a rectangular-shape:

- Network Slice Manager: Based on the ETSI NFV standard [26], it

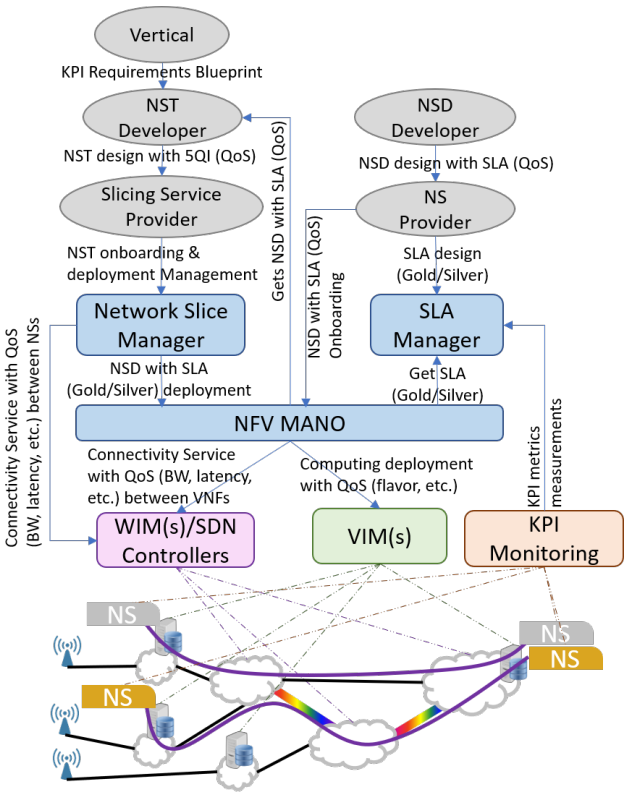


Figure 4.1: Generic architecture with Network Slicing and SLA life cycle management.

controls the network slices life cycle by managing NSTs and NSIs data objects and sends the requests to the NFV MANO to apply any action over the virtual elements composing the network slice.

- NFV MANO: It manages the life cycle of NSs and VNFs elements. The NFV MANO makes use of NS descriptor (NSd) and VNFd elements to identify how a NS is composed and to instantiate the corresponding VNFs and the VLs with the right VIM and WIM elements.

- SLA Manager: One of the new elements compared to the architecture in Figure 3.1. It manages SLA descriptor (SLAd) elements which contain the QoS requirements. Moreover, it applies them over the corresponding NS and, finally, it monitors and controls if any SLA metric has been violated while the network slice is deployed.

- VIMs & WIM(s)/SDN Controllers: The controllers to manage the computing and networking resources based on the required characteristics such as CPU, memory and storage for the computing elements and parameters like throughput, latency, spectrum or others for the CSs corresponding to the VLs composing the network slice.

- KPI Monitoring: Another new element together with the SLA Manager. It receives metrics from the different VNFs, gathers them and generates SLA violation alerts for the SLA Manager. The KPI Monitoring provides several mechanisms like monitoring probes deployed next to the VNFs, Application Programming Interfaces (APIs) in which VNFs can directly push metrics or well-known protocols like SNMP to make the collection of the performance metrics easier.

- 5G Physical Infrastructure: The complete set of resources organized in domains (i.e., edge, transport, core) and ready to be used when a network slice is requested. Each domain has it own set of computing and networking resources based on its specific characteristics.

### 4.1.2 Chaining QoS elements: from the KPI requirements to the network QoS enforcement

In order to understand how the process of setting the monitoring elements works when different players are involved, it is important to understand how the QoS is defined and implemented across the whole infrastructure through the evolution of the multiple data objects involved. Starting from the mapping of the vertical KPIs to the network QoS parameters and the relationships done between the layers in Figure 4.1.

The first mapping is done at the network slice layer by an NST Developer. When this player designs a NST, it takes the KPI requirements and maps them to the most appropriate set of metrics available at the NS/VNF layer. Among different options, the selected one was based on a standardized QoS classification, more specifically, the selected option was the 5G QoS Identified (5QI) parameters, which is defined by the 3GPP [127]. The 5QI classification identifies different sets of 5G parameters to define specific KPI values and identifies them in a single scalar value that is associated to each type of service. Despite this parameter was specially defined to be used in 5G scenarios with RAN and Core domains, it was selected based on two as-

pects: a) it gives the possibility of defining a complete QoS profile within the NST by using a single value, and, b) of opening to the possibility of using the Network Slicing work done in this thesis to other colleagues researching on topics that make use of scenarios with RAN domains. Finally, with the 5QI selected and based on the vertical KPI requirements, the NST Developer must select: a) the NSs involved in the desired service to be deployed in the network slice, b) the most suitable 5QI value based on the vertical KPIs and finally, c) the appropriate SLAs descriptors with the monitoring metrics and SLOs that will guarantee the expected vertical KPIs.

While the 5QI is used within the NSTs, in the lower layer (i.e., NSs and their VNFs), the QoS is designed to be enforced by using the SLAs to define which NS flavor is used. A flavor in a NS descriptor defines specific deployment parameters to configure the deployed virtual element (i.e, VM or containers) such as the computing resources (i.e., CPU, memory, RAM), bandwidth, latency and other possibilities that should allow to fulfill the SLOs in the SLA and so, to achieve the expected QoS. The use of flavors in a single NS descriptor generates a very important benefit in terms of the data objects management because in a single NS descriptor there are different versions (in terms of resources) of the same NS to be deployed. This aspect allows a more efficient data objects management from the point of view of a DB element. When a NS within a network slice with an associated SLA has to be deployed, the best flavor fulfilling the SLA requirements among the possible options is selected and so, the deployed NS (and its VNFs) should accomplish the QoS expected by the vertical KPI requirements.

Based on the previous descriptions, the vertical KPI requirements are mapped in all the infrastructure layers and so, the QoS chain (vertical KPI - 5QI - SLAS - flavors) is created. Based on this, once a network slice deployment is requested, the Network Slice Manager sends the NSs and the selected SLA information to the NFV MANO. Finally, and based on the SLA, the NFV MANO will check which NS flavor needs to be deployed and will manage the final step to have the complete network slice perfectly deployed and ready to fulfill the expected QoS defined by the vertical KPI requirements.

### 4.1.3 Descriptors design and management

Having described how the QoS is mapped across all the layers, it is interesting to understand which role each player has in the design and creation of all

the elements before the service provider may request them for its clients. As illustrated in Figure 4.2, to offer a service for a certain vertical there are two actions to have the system ready. Ont he one hand to prepare the NSs, its VNFs and the SLAs associated to each NS (steps 1-4 Figure 4.2). On the other hand, to prepare the network slices (steps 5-9 Figure 4.2).

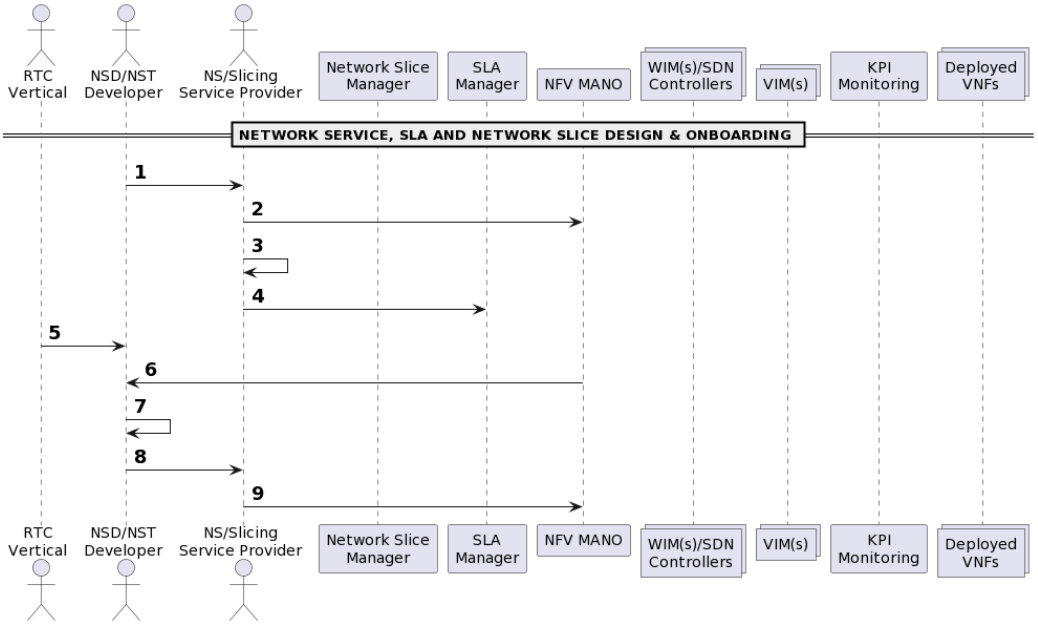


Figure 4.2: Design and on-boarding workflow steps.

The process for the NSs (and VNFs and SLAs) is triggered by the NS descriptor Developer as presented in Figure 4.1 (process on the top right side). The different steps of the process and further details of this process are described in Figure 4.2. The NSd Developer begins with the creation of the descriptor data object with the following information: a) the appropriate set of VNFs that will compose the NS, b) the interconnections among VNFs composing the NS and, finally, c) the set of flavors. Once the NSD is done, the NSD Developer gives it to the NS Service Provider (Figure 4.2, step 1). Once received, the NS Service Provider on-board the NSD in the NFV MANO (step 2). In addition to on-board the received NSD elements into the NFV MANO, the NS Service Provider has to define the SLA descriptors (step 3). To do so, the NS Service Provider has to define the metrics and SLOs to be monitored once the NS (within a network slice) is deployed. Moreover,

the NS Service Provider has to associate each of its SLAs to a specific NS flavor in order to obtain the previous QoS chain and so, the most suitable NS flavor will be deployed using the appropriate computing and networking resources to fulfill the SLA and achieve the vertical KPIs. Once the SLA descriptors are ready, the NS Service Provider will on-board them into the SLA Manager (step 4).

Finally, before having the system ready to offer vertical-specific services to the possible clients, there is the need to prepare the network slices descriptors in the system as presented in Figure 4.1 (process on the top left side). This process begins with a blueprint of each vertical KPIs requirements which are received by the NST Developer (Figure 4.2, step 5). Based on the received requirements, the NST Developer requests and verifies the available NSDs and their associated SLAs (step 6). Then, the NST Developer uses all the received information (i.e., vertical KPIs, NSDs and SLAs) to design and create the NST (step 7). To do so, it selects the following elements: a) the best 5QI value possible that maps to the vertical KPI requirements, b) the NSDs and the most suitable SLA that each NSD might have associated to fulfill the QoS defined by the 5QI. Once the NSs and SLAs are selected, the NST Developer defines how the selected NSs composing the network slice will be interconnected among them and forwards the NST to the Slicing Service Provider (step 8). Finally, the Slicing Service Provider on-boards the NST to the Network Slice Manager (step 9), leaving the NST completely ready to be deployed when required.

### 4.1.4 Network slice deployment and KPI monitoring

Once all the resources (i.e., computing, networking and services software) and the descriptors (i.e., VNF, NS, NST and SLA) are ready, the vertical clients may request the deployment of network slices with specific QoS characteristics. Illustrated in Figure 4.3, this process begins when the Slicing Service Provider requests to the Network Slice Manager (step 1) for the NST to be deployed. Based on the expected QoS by the service client, the NST with the most suitable 5QI is selected and its deployment triggered. When the request reaches the Network Slice Manager, the last one begins with the creation of the NSI (step 2). Based on the selected NST, it adds the context deployment information such as the ID, the name, the description and the instantiation parameters. Once the basic NSI information is ready, the NSs specific deployment is triggered when the Network Slice Manager forwards

the identifiers of the required NSDs and their associated SLAs to the NFV MANO (step 3).

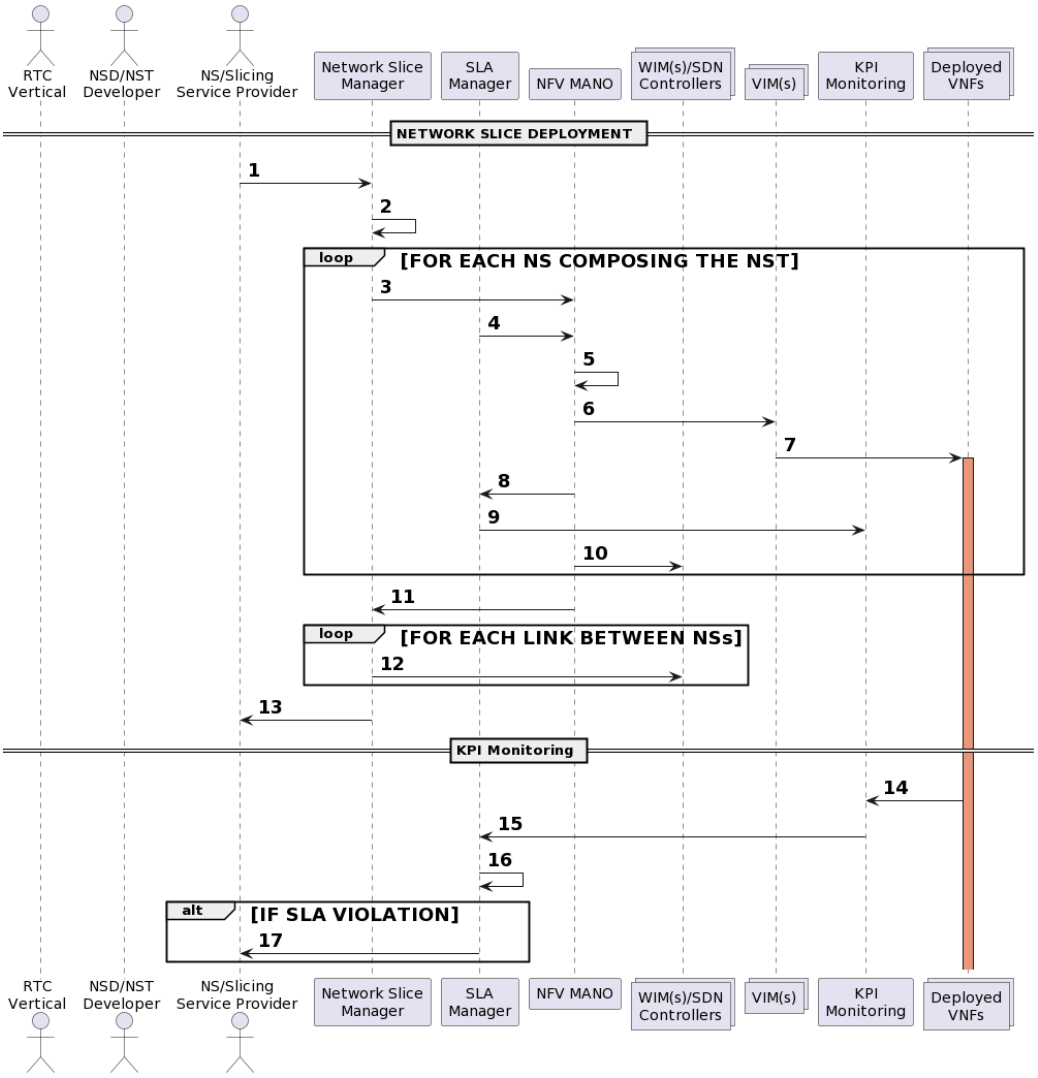


Figure 4.3: Network slice deployment and monitoring workflow steps.

Based on the SLAs received within the requests, the NFV MANO obtains the SLA information (step 4) to choose which flavor is needed. Once all the NS flavors are selected, the NFV MANO manages the placement procedure (step 5) to identify and decide which VIM, among the available ones, is the

best option where each NS may be deployed to accomplish the SLA (i.e., its metrics and SLOs) and therefore, to achieve the expected QoS. Once the NS flavor and its placement are selected, the NFV MANO follows by requesting to the selected VIM the NS deployment (step 6) and the set of VNFs composing the NS (step 7). When the NS (and its VNFs) are ready, the NFV MANO requests to the SLA Manager to configure the SLA monitoring rules (step 8) and this last one, sets these rules into the KPI Monitoring (step 9). The last action to have the NS completely deployed is the connection between VNFs composing the NS (step 10), so the NF MANO requests to the WIM/SDN Controller the required CSs to connect the VNFs among them. When done, the NS is completely ready. At this point, all NSs are allocated and ready across the multi-domain infrastructure (step 11), but the network slice still misses few more actions. First, the interconnection among the NSs defined in the NST are requested by the network slice to the WIM/SDN Controller (step 12). After all the required VLs are configured, then the network slice is ready and the Slicing Service Provider is informed (step 13).

The moment the network slice is ready, the Slicing Service Provider will inform the final clients about the procedure to access and use it properly. However, the Slicing Service Provider did not finish its responsibilities yet. When the service is deployed, the Slicing Service Provider has to monitor the performance of the service in order to ensure that the SLA (and therefore the QoS) is properly accomplished while the service is being used and alive. The monitoring procedure designed within our system bases its actions on the fact that the KPI Monitoring has the capability to configure the reception of metrics to be sent using SNMP messages directly from the VNFs. Taking this into account, and as illustrated in Figure 4.3, the monitoring procedure begins when each VNF exposes a set of performance metrics (through SNMP messages) to the KPI Monitoring (step 14). This evaluates the received data and compares it with the status associated to the SLA monitoring rule and if an SLA is violated, it informs with an alert message the SLA Manager (step 15). The SLA Manager applies the final check (step 16) and takes the final decision whether the QoS is accomplished or not. In case the SLA is violated, the SLA Manager alerts the Slicing Service Provider so this may take the appropriate actions to enforce a reaction (step 17).

An interesting detail about how the monitoring system was implemented is the fact of giving the responsibility to send the metrics to the VNFs implied an extra task for the NS Developer to do by the time the NS was designed.

The NS Developer has to add some extra information (e.g., metrics, SNMP OID, etc.) within the VNF descriptor in order to have the SNMP mechanism properly configured towards the KPI Monitoring module.

## 4.2 Network slices security enforcement

The enforcement of QoS on a deployed service is of absolute importance towards the fulfillment of the agreement between the provider and the client. This performance may be affected in many ways, being the most common the coexistence of multiple services using the same infrastructure. Despite having reserved the right amount of resources, the performance of the service may be altered from by an external entity which has to be avoid. This situation leads towards the need to design a solution with the capabilities to define and enforce the most appropriate QoSec around the network slices (i.e., services) deployed.

Taking the concepts learnt in the previous section, an evolution of the SLA monitoring system is presented with a new set of modules and data objects to ensure that the SSLA is respected and that, in case of a violation, it is resolved before the termination of the deployed service is required.

### 4.2.1 Evolving the NFV MANO architecture for security monitoring

Based on the work from the previous subsection and the architecture illustrated in Figure 4.1, an evolution has been studied with the objective to add the necessary elements to ensure that the deployed and monitored network slice may be secured and, in case of a threat or a possible attack, a solution is applied to keep the used resources safe and reliable.

The initial design of the architecture to make network slices secure and monitor their security KPIs is presented in Figure 4.4. This architecture was designed based on the idea to have, on the one hand the already presented network slice elements, and on the other hand, another deployed element called Security Function (SF) (i.e., a NS focused on security with for example probes). The objective of the SFs is to be deployed and become a security complement attached to the deployed network slice (i.e., service) requested by the service provider. Because of this aspect, in this architecture there is a clear difference between the modules focused on the service itself and the

modules in charge to secure the deployed service. Based on this idea, the first proposed architecture was composed of two main manager solutions: a) the Slice Manager/Service Orch. and b) the Security Manager. While the first management solution manages all the actions related to the network slice, the second one interacts with the first to deploy the requested network slice and the necessary elements to monitor the security.

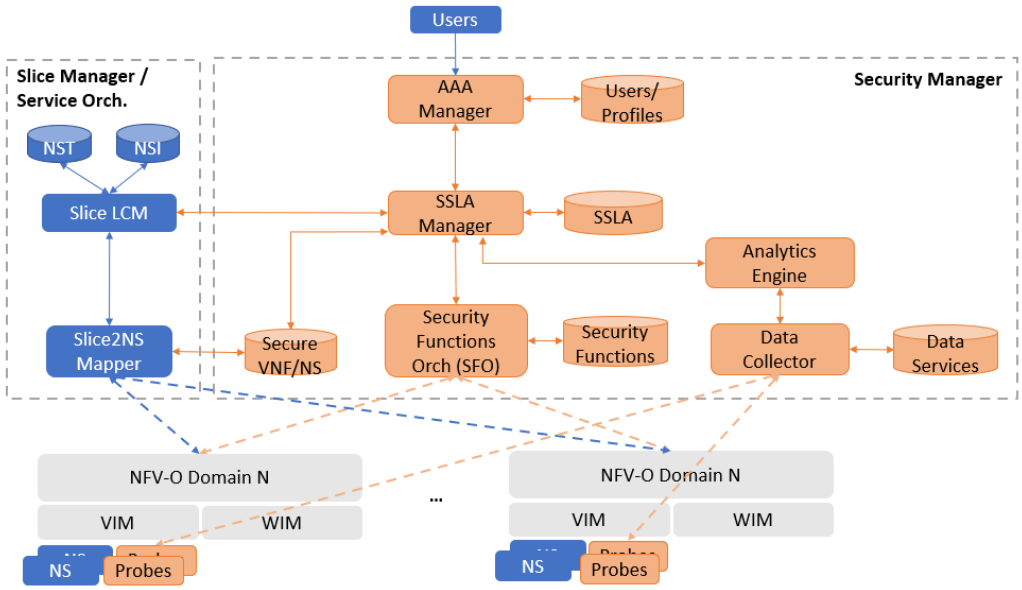


Figure 4.4: Secured Network Slicing architecture.

The Slice Manager/Service Orch. is in charge of managing and controlling all the actions related to the network slice elements and to interact with the NFV-Os below. The internal architecture of this element keeps the same design than the one presented in [117]. In there, the main modules are:

1. Slice LCM: Manages the NSTs and NSIs data objects describing any network slice object.

2. Slice2NS mapper: It decomposes the information within an NSI into the necessary requests to deploy/terminate all the individual NSs (and VNFs) composing the network slices and it stores the information related to the instantiated elements.

3. NST and NSI DBs: They contain the NSTs and the NSIs with the information of either which NSs need to be deployed or the deployed virtual elements respectively.

The Security Manager is in charge of the SSLAs management associated to the network slices by controlling the following actions: a) it gathers the verticals/end-users security requirements; b) it deploys the necessary SFs to enforce the agreed SSLA; c) it monitors whether the SSLAs are fulfilled; d) it detects SSLA violations based on an analytic engine and notifies about them; e) reacts in real-time to adapt the provided level of security or to apply proper countermeasures. Its main modules are:

1. Authentication, Authorization and Accounting (AAA) Manager: It takes care of allowing the access into the Security Manager and its internal features. Its main functionality is to control the users access and reject possible non-authorised access attacks.

2. Security Functions Orchestrator: It controls the SFs lifecycle for each network slice instance and monitors its security performance to warn the SSLA Manager about any SSLA violation.

3. SSLA Manager: It manages the lifecycle of the SSLA assigned to a network slice. Its functionalities are: a) to get the SSLA data object from the SSLA DB and pass it to the Slice LCM, b) to command the SFO to launch the SFs and so, to apply the Security Service Function Chaining (SSFC) in order to start monitoring and, finally, c) the management of the resolution policy action to apply when a SSLA violation occurs.

4. Analytics Engine (AE): Evaluates the data forwarded by the Data Collector and determines whether a SSLA has been violated or not based on parameters defined by the SSLA Manager.

5. Data Collector (DCol) : It gathers and organises all the information coming from all the SFs in order to forward it to the AE.

6. Users/Profiles DB: It keeps a list of who can access to the framework.

7. SSLA DB: It saves the generated SSLA data objects with the following three main elements: Security Controls, Metrics and SLOs.

8. SFs DB: Contains the set of SFs to be launched when a network slice is instantiated and gets the related data to monitor and ensure the SSLA is fulfilled. Together with probes, another possible SF could be the security controls defined by the NIST Control Framework [63] in order to address different security purposes (e.g., privacy).

9. Data Services DB: It keeps the incoming data/events collected by the DCol module, in order to keep track and log the performance.

10. Secure VNF/NS DB: This last DB has the list of Validated/Certified VNFS and NSs by a trust authority that a network slice can use. If a VNF/NS which is not in that DB is needed, it will not be deployed as it is not considered a secured element.

After a study and analysis process and thanks to the work done during the INSPIRE5G-Plus [128] project, the previous architecture had some aspects that could be improved. First of all, the internal architecture was quite complex in terms of the relationships between the modules. Having closed relationships (i.e., one to one interfaces) reduces the possibilities to keep a constant evolution of the solution. So, if a new module needs to be added, it might require to "break" the relationship leading to other unexpected issues. Based on this, the new architecture had to have a common element that would allow an easy and fast integration of the existing and any new modules. Second, based on the fact that networks are composed of multiple domains, to have a single control and management architecture having to manage multiple NFV-Os Domains while keeping their specific characteristics entails an increment of the complexity in terms of the operations and resources allocation. So, it was interesting to clearly differentiate the E2E domain and the multiple lower domains so that each control architecture would be specialised on its domain resources and, from the E2E point of view, an abstracted information of all the below domains would be necessary. Finally, and related to the act of having multiple domains, when an SSLA is violated its resolution will have two levels of action: either at domain level which avoids the E2E and other domains to participate and so reducing cost operations or at the E2E domain level which then requires multiple domains to act. In order to solve these three issues, the ETSI ZSM [69] was selected. The ZSM defined an architecture based on an E2E Management Domain on top and a set of Management Domains below and all of them interacting among them through an element called Cross-domain Integration Fabric. Despite

this hierarchical design, the ZSM architecture allows to keep certain independence of each domain while at the same time, they work together from an E2E point of view. Based on the ETSI ZSM architecture, the previously presented security architecture (Figure 4.4) evolved to the one illustrated in 4.5.

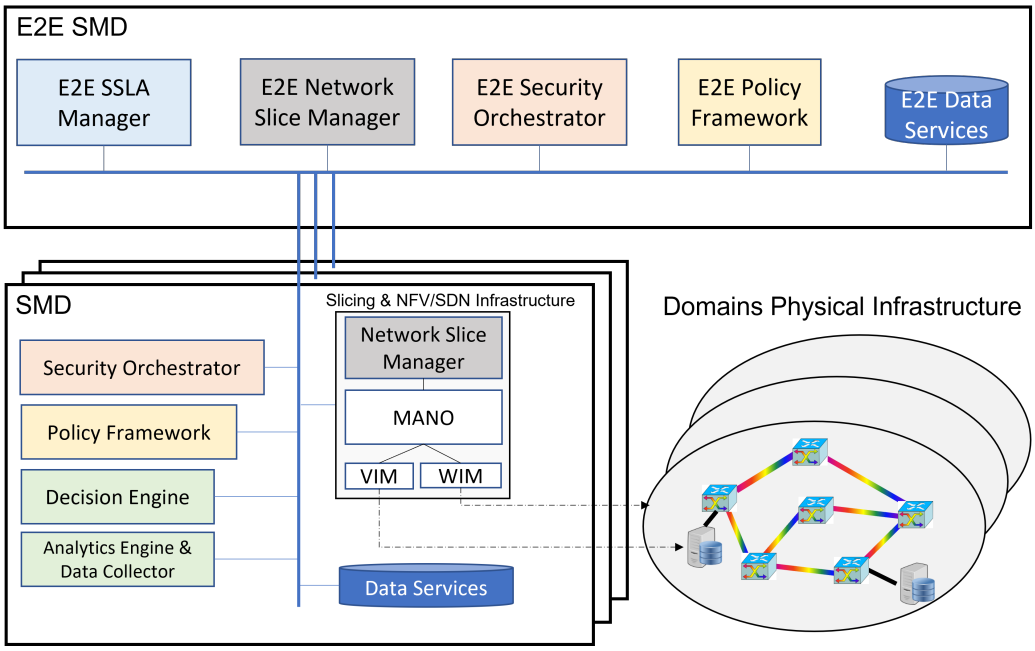


Figure 4.5: INSPIRE5G-Plus based architecture design for managing secured network slices.

The architecture is illustrated in Figure 4.5 and it is composed of three layers:

### E2E Security Management Domain (E2E SMD)

This layer is placed on top of the architecture and the elements in there must take care of those actions with an E2E point of view. Based on the information stored available in the layers below, this layer manages the E2E network slices actions. As first introduced by the 3GPP and later defined by the ETSI [129], the slice-subnets composing a network slice may be a set of independent NSs, a set of other network slices or a mix of both. With this idea, during the architecture design process, it was defined that each

slice-subnet within an E2E network slice would be a domain network slice. For this reason, the E2E SMD modules will interact among them but also with their corresponding "siblings" in each domain in order to control the deployed E2E network slice. The modules at this layer will act either when a new action needs to be implemented (i.e., architecture entry point) or when an action that happened in one domain below may affect another domain. The modules composing the E2E SMD are:

- E2E Network Slice Manager: The responsible of managing any action during the life-cycle of an E2E network slice. Similarly to the Network Slice Manager elements described up until now, the E2E Network Slice Manager uses E2E NSTs to define the slice-subnets composition (i.e., services and domains where they are), how are they going to be interconnected (i.e., the connectivity services to interconnect the domains). Now, as the focus on the evolved architecture is the security around the network slices, the slice-subnets within an E2E NST must also contain either domain NSTs with SFs or SFs themselves. Equally important, when an E2E NST is requested to be deployed, the deployment-related information is stored in E2E NSI data objects, which will contain the references to the NSIs deployed in the corresponding domains.

- E2E SSLA Manager: This component manages the definition of E2E KPI requirements (i.e., metrics, Security SLOs, etc.) to be accomplished by the whole E2E infrastructure when a service is deployed. Once a set of SSLAs is defined and available and an E2E NST is requested with a set of security requirements, this component has to select the most appropriate SSLA that will fulfil the expected QoS. To do so, it has to take into account the singularities of each domain (defined in the NST) that might participate in the secured E2E network slice by decomposing the E2E requirements into specific domain requirements.

- E2E Security Orchestrator (E2E SO): Its main focus is the correct implementation of the network slices and their security orchestration. For this reason, the E2E SO is in charge of controlling the multiple steps to deploy the E2E network slice by distributing the right policy actions across the involved domains through the domain SOs. The E2E SO manages all its actions (i.e., deployments and configurations) using policy data objects.

- E2E Policy Framework (PF): This component manages the creation and storage of pre-defined E2E policies (e.g., isolate element, reconfigure element, etc.) to be used in case they are required (i.e., to solve an SSLA violation). Moreover, and more importantly, this component detects any possible conflict between a policy to be applied and those already applied at the E2E level. In case the resolution of a violated SSLA implies the participation of different domains, the E2E PF will check if the selected policy can be applied on all of them or another one is needed.

- E2E Data Services: A centralised Data Base where all the data generated by all the previous components is stored and ready to be accessed. Moreover, when a new data object is stored, it makes sure that it follows the right data model. Finally, the data is stored in a central methodology for these reasons: a) in case of failure in a module, the data is protected and not lost, and b) as the data stored contains critical information related to different domains (e.g., computing and transport instances), there is one unique access to it and so, it is easier to verify "who or what" aims to access it.

**Single Management Domain**

While there is only one E2E SMD, below this, there might be as many SMDs as required and the components composing an SMD are:

- Security Orchestrator (SO): The entry point of each SMD, this element receive from its "parent" (i.e., E2E SO) the policy actions to apply in its SMD. Once a policy is received, the SO triggers the following processes: a) to analyze the policy requirements and to generate an orchestration plan based on them, b) to request the slice-subnet to the Network Slice Manager/MANO element and, c) to apply the security configuration according to the orchestration plan to enforce the expected QoSec. To ensure the last process is done, the SO relies on the Decision Engine and the Analytics Engine & Data Collector modules to configure the monitoring elements using the required metrics.

- Policy Framework: Like the E2E PF, this module helps the SO with the final infrastructure configurations associated to the security policies in the orchestration process. Moreover, it contains the policies with the

specific domain characteristics to be applied when an SSLA violation is detected.

- Decision Engine (DE): This component does not exist at the E2E SMD, the reason is because of its functionality, which consists on triggering and enforcing the SSLA violations resolution on the corresponding slice-subnet. This element has the capability to receive a set of monitored data and decide based on the received data whether an SSLA in its domain has been violated and, if so, it decides the best policy (store in the PF) to apply in order to resolve the SSLA violation. To decide which is the best policy to apply the DE requests the policies associated to the violated SSLA (or the specific violated metric) and translates the policy into the most convenient action.

- Analytics Engine & Data Collector (AE/DCol): The AE/DCol takes care of gathering the monitored data from different components within a deployed slice-subnet (i.e., monitoring agents, SFs, NSs, etc.), then it analyses this data and gathers the important metrics to be sent to the DE.

- Slicing & NFV Infrastructure: This component contains the Network Slice Manager and the NFV architecture [26] (i.e., MANO, VIM, WIM) already illustrated in Figure 3.1 which manages the domain NST/NSIs, the NSs and the VNFs. Moreover, the SFs resources are also managed by these elements as they are an evolution of the NS objects but focused on security. As an NST placed in an SMD is a slice-subnet within an E2E NST, the more SMDs are available with their specific characteristics, the highest is the number of possible E2E NSTs (i.e., secured services) to define and so, to offer to the final clients.

- Data Services: The data management applied in an SMD is the same that was presented in the E2E SMD, with the centralization of the data objects but only of those belonging to the specific SMD.

### Physical infrastructure

The lowest layer in the architecture presented is the physical infrastructure belonging to the SMDs (i.e., edge, transport and cloud), each with its specific resources (i.e., optical/packet networks, computing resources, etc.) ready to be used to instantiate and monitor the E2E network slices.

### 4.2.2 Deployment and enforcement of secured E2E network slices

Now, compared to the deployment workflows previously described, due to the division of E2E and domains infrastructures as defined by the use of the ETSI ZSM, the deployment and monitoring workflows have evolved into the following processes:

#### Secure E2E network slice deployment

Based on the architecture components previously described and as illustrated in Figure 4.6, the process to deploy an E2E network slice with the corresponding security requirements has four phases: a) the E2E network slice & QoS selection and definition, b) the Domain network slice deployment, c) the setting monitoring rules and d) the deployment confirmation.

The "E2E network slice & QoS selection and definition" phase starts with a request from a tenant (1) specifying both the E2E NST and the SSLA requirements to generate the secured E2E NSI. The E2E SSLA Manager receives the request, it retrieves the SSLA information and forwards it together with the selected NST identifier to the E2E Network Slice Manager (2). The E2E Network Slice Manager processes the request to generate the E2E NSI object, which allows have a record of the resources selected in each SMD and, finally, it generates a policy for the E2E SO (3 and 4). The E2E SO, proceeds to generate an E2E security orchestration plan to fulfill the correct E2E NST deployment (5).

Then, the "Domain network slice deployment" phase starts with the E2E SO distributing (6 and 7) the corresponding policies to the involved domain SOs to deploy the multiple domain network slices (i.e., slice-subnets) and from this moment all the domains involved apply the same actions (Note "A"). Once each domain SO receives the policy to deploy the slice-subnet, it generates a domain security orchestration plan (8) to manage the SMD slice-subnet and SSLA deployment. Once the plan is ready, it forwards the corresponding request to the domain Network Slice Manager/MANO element (9) to deploy the corresponding slice-subnet (i.e., a domain NSI based on a domain NST) (10) and it receives its confirmation (11).

When a slice-subnet is deployed, the "Setting Monitoring Rules" phase is triggered by the SO, which configures the monitoring rules based on the corresponding policy towards the AE/DCol through the DE (12). The DE

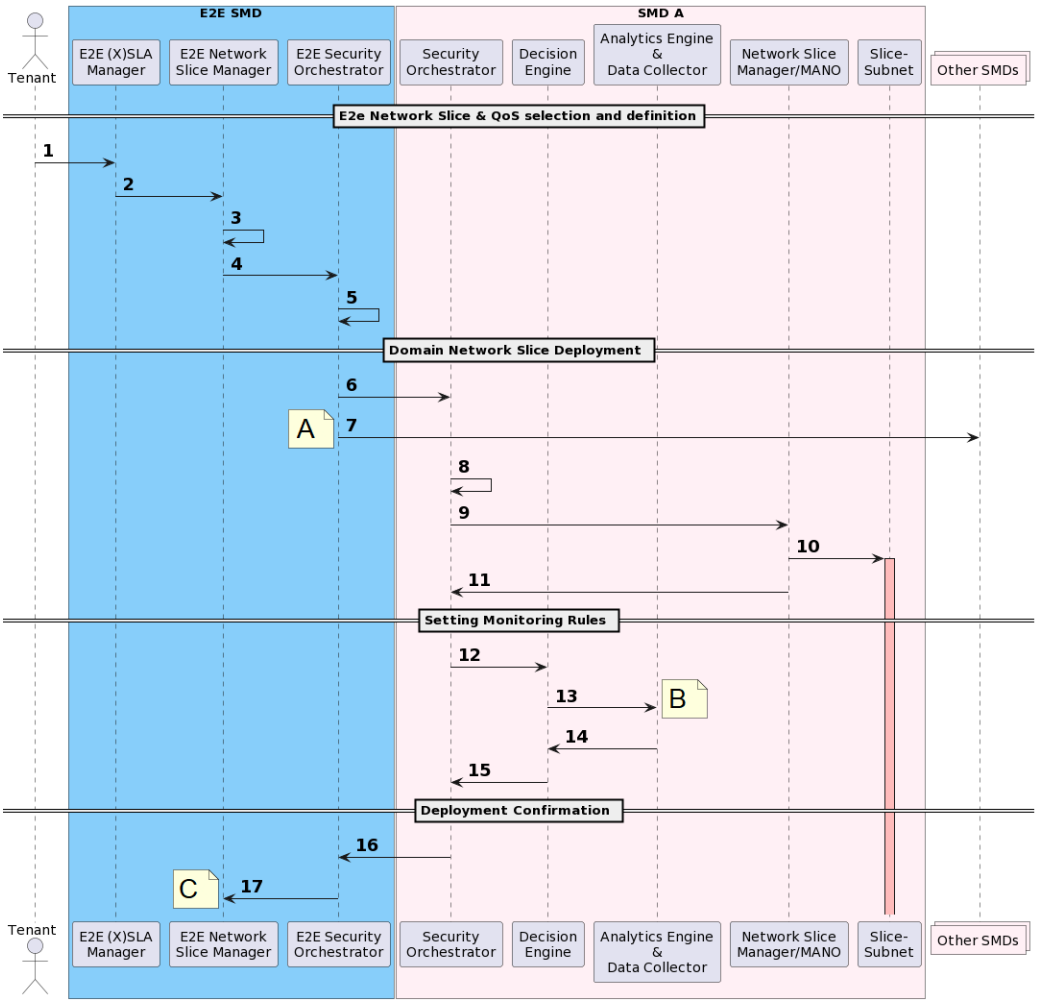


Figure 4.6: Secure E2E network slice deployment.

requests to the AE/DCol to select the best algorithm to collect and process the data associated to the SSLA metrics that need to be monitored from the deployed slice-subnet resources (13). The AE/DCol confirms the correct configuration (14) and, finally, the DE does the same to the SO (15). The Note "B" within the workflow, remarks the fact that, despite having the slice deployed, its metrics are not monitored until the step 11 is applied. Then, from that moment on, the SSLA is continuously validated with monitored data from the deployed resources.

Finally, for each slice-subnet deployed and SSLA configured, the "Deployment Confirmation" phase starts with the SMD SOs informing the E2E SO (16), which forwards the same information to the E2E Network Slice Manager (17). The E2E Network Slice Manager is the responsible of confirming that the deployment request has been fulfilled and so, the E2E network slice is secured and monitored properly. If so, it notifies the availability as marked in Note "C".

### SSLA monitoring and enforcement

Once the E2E network slice is completely deployed and being monitored, the users may start using it. While the best scenario for a monitoring system is to never do any actions other than evaluating and deciding that no SSLA has been violated, this possibility is quite remote because networks have many variables (e.g., other coexisting services, software or hardware bugs, external attacks, etc.). Through the described architecture, any deployed E2E network slice is monitored and, as illustrated in Figure 4.7, the monitoring process includes the following phases: a) the SSLA metrics Gathering & Evaluation and b) the SSLA Violation Management.

The "SSLA metrics Gathering & Evaluation" phase begins for any new set of monitored data that is received (1) by the AE/DCol, which processes it (i.e., grouped in sets of data based on SSLA metrics) and forwards it to the DE (2). Then, the DE evaluates with the monitored data whether the associated SSLA has been violated or not (3). If no violation has occurred, no other action is required and new monitored data is expected to start the process again (Note "A").

If an SSLA violation is detected, the "SSLA Violation Management" phase begins. Having decided that the SSLA has been violated, the DE retrieves the possible policies from the domain PF to fix the SSLA violation (4), it decides which one to apply and informs the SO about the policy (5). The SO requests the corresponding actions to the Network Slice Manager/MANO (6), which applies the actions over the deployed slice-subnet (7). While steps 6 and 7 happen, in parallel, the SO forwards the policy decided by the DE to the E2E SO (8). At this point, the E2E SO requests to the E2E PF if the policy received generates a conflict with other deployed policies and informs the E2E SO (9), which then forwards the information to the corresponding domain SOs (10) to apply the required actions using the corresponding SMD module.

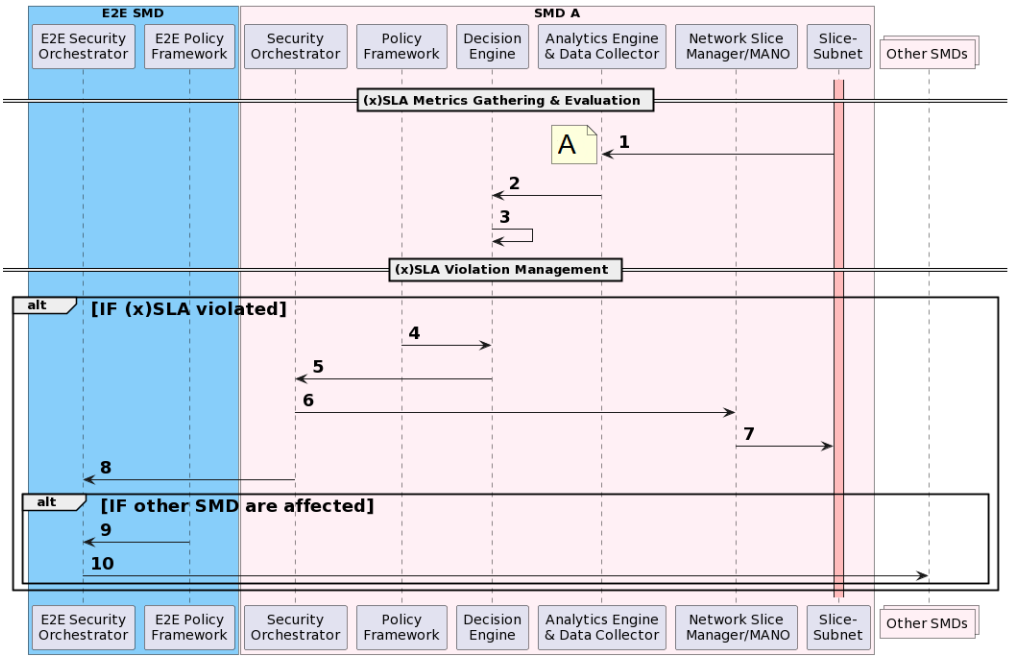


Figure 4.7: Secure E2E network slice monitoring.

As a final remark, the previous two workflow descriptions (Figures 4.6 and 4.7) omitted the actions of data retrieval in which the Data Service Modules should participate to avoid the repetition of those actions (i.e., getting or storing the data objects).

### 4.2.3 Data objects

The multiple components involved in the presented architecture make use of a set of different data objects in order to interact among them and to manage and control the actions implemented and the resources selected. All the domains (either E2E or not) have a Data Service module used to store the information.

#### E2E and domain NST & NSI

As Network Slicing functionalities were designed at both levels (i.e., E2E and domain), the idea was to generate a data object structure that could be used in both layers in order to keep the maximum homogeneity and correlation

possible between the E2E and the domain Network Slice Managers. As presented by the GSMA since its first round of technical reports about Network Slicing to the latest one [113], the idea was to generate a generic NST.

Due to the complexity of the architecture presented in our initial implementations, the NST (Listing 4.1) was kept as simple as possible with the most essential metadata (i.e., id, name, description, etc.), the list of slice-subnets (i.e., a reference to a domain NST id, location of that NST, etc.) and how the slice-subnets were linked. At the domain level, the same structure was used but the slice-subnets items had the reference identifiers of the NSs (and their VNFs) composing the NST in that domain.

Listing 4.1: NST structure example.

```
{
  "id": "nst_uuid",
  "name" : "nst_name",
  "description" : "nst_description",
  "slice-subnets":[
    {
      "id-ref": "subnet_id",
      "name": "subnet_name",
      "location": "domain_id"
    }
  ],
  "virtual-links":[
    {...}
  ]
}
```

Once a NST is deployed, an NSI (Listing 4.2) based on the NST is generated. In addition to the referenced NST identifier, the NSI contains the information related to the instantiated elements (i.e., NSs, VNFs, etc:) but also the selected SSLA reference.

Listing 4.2: NSI structure example.

```
{
  "id": "nsi_uuid",
  "name" : "nsi_name",
  "nst-ref" : "nst_uuid",
  "ssla-ref": "ssla_uuid",
  "slice-subnets":[
    {
      "id-ref": "subnet_id",
      "location": "domain_id"
    }
  ],
  "virtual-links":[
    {...}
  ]
}
```

### E2E SSLA

Regarding the SSLA data object, an extension to support Network Slicing was done on the machine readable format described in the SSLA model presented in [130]. A basic example of an SSLA is presented in (Listing 4.3). In there, first the basic data object information such as the identifier, name, and other possible metadata parameters can be found. But more important, the core of the SSLA is made of the following three blocks:

- slice-resources: It is the list of available E2E NSTs that can be deployed and monitored using that SSLA.
- security-capabilities: It is the list of security aspects that this SSLA defines to be monitored. A capability is a set of security controls and a security control is one or more elements and/or processes implemented in a system to manage the possible risks and to generate protection in terms of confidentiality, integrity, availability of the system and its information. Well-known examples of security controls are those defined by the NIST's Control Framework [63] such as Access Control or Incident Response.

- metrics: This third block of information includes the parameters and Security Service Level Objectives (SSLOs) to be monitored in the multiple SMDs. This information is referenced in the E2E NSI and the domain NSIs in order to configure correctly the monitoring elements and so, to verify the collected data and decide whether the SSLA is respected. Each metric element contains its definition and its measurement scale among other properties for the monitoring configuration process.

Listing 4.3: SSLA structure example.

```
{
  "id": "ssla_uuid",
  "name": "ssla_name",
  "description": "ssla_description",
  "slice-resources": [
    {
      "nst-ref": "nst_uuid"
    },
    {...}
  ],
  "security-capabilities": [
    {
      "id": "capability_id",
      "description": "cap_desc",
      "security-controls": [...]
    },
    {...}
  ],
  "metrics": [
    {
      "id": "metric_id",
      "service-level-objectives": {...}
    },
    {...}
  ]
}
```

### Policies

Because of the collaboration done with the partners in [126], some modules within the architecture in Figure 4.5 work and interact among them using a "Policy" data object. These modules are the multiple SOs (E2E and domain), the DEs or the PFs. An example of a policy and its structure is presented in the Listing 4.4. The policy data object was designed to define two types of actions: a) "DEPLOYMENT" - to manage the SMD network slices deployment when the E2E SO informs all the SMD SOs, and b) "CONFIGURATION" - to prepare the monitoring elements based on the selected SSLA requirements and to apply the corrections once an SSLA violation is detected.

In both actions, using the "nst-ref" and "ssla-requirements" fields allows, on the one hand, to inform the SMD Network Slice Manager about the right NST to deploy and to the SMD DE and AE/DCol about the metrics and SSLOs to configure. On the other hand, to ask to the SMD PF the policies that may be applied on the deployed NST in case of an SSLA violation.

Listing 4.4: Policy structure example.

```
{
  "id": "policy-instance-uuid",
  "subnet-name": "E2ENSI-name_SMD",
  "nst-ref": "nst_uuid",
  "location": "domain_id",
  "type": "DEPLOYMENT/CONFIGURATION",
  "status": "status",
  "ssla-requirements": [
    {
      "metric": "metric_id",
      "slo": {...}
    },
    {...}
  ]
}
```

## 4.3 Experimental validation

The previous QoS and QoSec related architectures described in the previous sections were implemented in different levels of detail to validate the workflows described and the multiple designed modules. Two different use cases are presented with their corresponding experimental results to demonstrate the aspects previously described. First, a Real-Time Communications (RTC) service for the content presented in Section 4.1 and then, the resolution of a DoS attack for theory described in Section 4.2.

### 4.3.1 Use case I: RTC network slices and their QoS

Figure 4.8 shows the elements managed within the CTTC ADRENALINE testbed [131] with the implemented architecture presented in Figure 4.1. For this use case, the Edge, the Transport (packet and optical) and the Core domains were used to achieve the objective of deploying a RTC service in the Core domain and configure the appropriate Transport CSs across one or the other Transport domains based on the SLA requirements from the vertical KPIs.

Despite other possibilities[132], the already introduced SONATA Service Platform (SP) was used on top of the infrastructure because the developed Network Slice Manager is integrated with the other necessary elements (i.e., NFV MANO, SLA Manager and KPI Monitoring) integrated in a single piece of software. Below, the same infrastructure presented in Section 3.3 was used.

#### RTC vertical KPIs and SLA description

The use case done over this infrastructure had the objective of deploying two network slices with the same RTC service but each with a different associated SLA each. The two SSLA defined were called "GOLD" and "SILVER" with high and lower requirements respectively. In both cases, the selected KPIs were the Packet loss (Pl), the Symmetric User Datagram Protocol (UDP) throughput and the Round-Trip time (RTT). Based on the three KPIs, the following SLAs were defined:

- GOLD SLA with a Pl $< 1$ %, a Symmetric UDP throughput $> 100$ Mbps and an RTT $< 40$ ms.

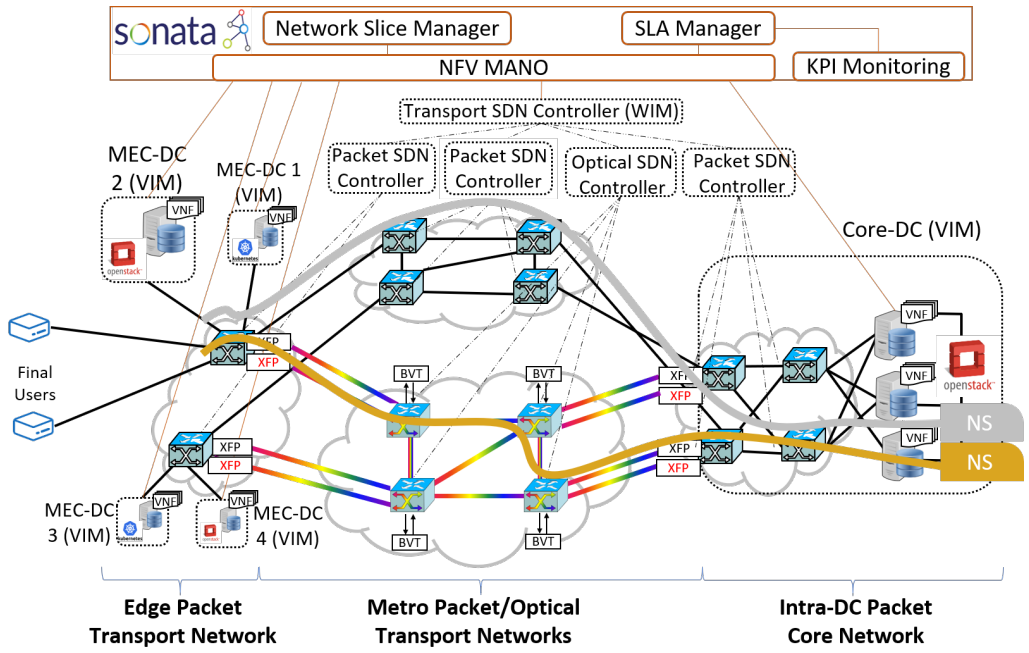


Figure 4.8: Experimental infrastructure and illustration of the two network slices and data flows (GOLD/SILVER) across the CTTC ADRENALINE testbed.

- SILVER SLA with a Pl $<$ 2 %, a Symmetric UDP throughput $>$ 80 Mbps and an RTT $<$ 60 ms.

Despite it was possible to define other SLA for better granularity with the QoS, by the time of this experimental actions, only the two presented SSLAs were proposed for simplicity. To apply this better granularity, a service profiling solution could be used as the one presented in [133]. Finally, more information about the implemented use case and other details that were not required for this document may be found in [134].

### RTC network service

As Figure 4.9, the selected RTC NS[1] was composed by the following set of VNFs:

[1]Service implemented by Quobis, partner in the 5GTANGO project.

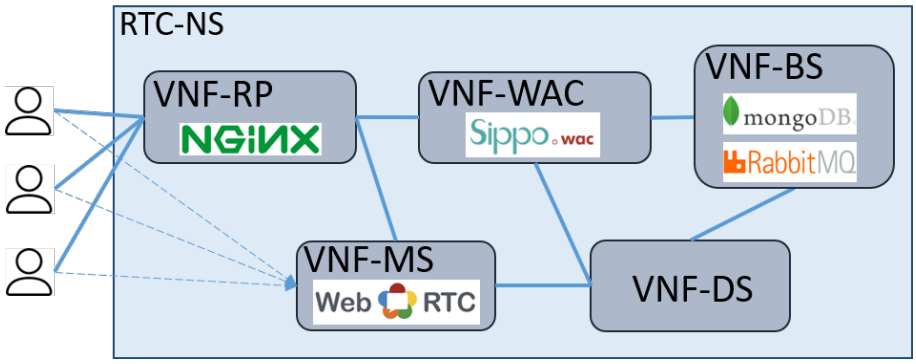


Figure 4.9: RTC NS (and VNFs) internal architecture.

- Media-Server (VNF-MS): This VNF manages the exchange of Real-Time Protocol (RTP) traffic among the users.
- Reverse proxy VNF (VNF-RP): This VNF takes the incoming HTTP/WebSocket traffic and distributes this traffic to the other VNFs.
- WebRTC Application Controller (VNF-WAC): This VNF controls all the AAA tasks related to the users together with the signaling logic to setup the videoconferences.
- Dispatcher (VNF-DS): This VNF manages the multiple Selective Forwarding Units (SFUs) generated and sends their information to the VNF-WAC which is in charge of creating new multimedia sessions for each SFU.
- Back-end Services (VNF-BS): This VNF implemented the DB and the queue system to store the status information and the information needed for the interactions between the different VNFs.

As already presented, two SLA were defined, and so, two flavors were designed within the NSD. The first one called "gold" (associated to the GOLD SSLA) has the minimum bandwidth requirement of 1000 Mpbs. The second flavor was called "silver" (associated to the SILVER SSLA) and it required a minimum bandwidth of 500 Mpbs.

### 5QI selection for the network slice

Having selected and designed the vertical KPIs, the SSLAs and the NS to deploy, the design of the network slice and its 5QI parameter is missing. Because of the RTC NS design, the composition of NSs at network slice level was not complex as only the RTC NS was in it. The task done at the Network Slicing level was the creation of two different NSTs. On the one hand, the first NST was the GOLD NST which was associated to the GOLD SLA and it had an associated 5QI value of 3 which defines a packet delay limit lower than 50 ms (GOLD SLA RTT ¡ 40 ms). On the other hand, the SILVER NST was associated to the SILVER SLA with a 5QI value of 2 defining a packet delay limit lower than 150 ms (GOLD SLA RTT ¡ 60 ms).

### Results

With all the elements ready, a set of tests were triggered to validate the NS deployment but specially to demonstrate that the QoS requirements defined at the network slice level were achieved at the NS/VNF level with the deployed RTC service. A set of multiple tests were done by deploying the RTC network slice in the Cloud DC within the Core domain, with the constraint to use the transport domains (depending on the SLA) and a time duration of 180 s. The tests were defined to have different QoS requirements, using the GOLD and SILVER SLAs and also without any SLA, looking for a Best Effort (BE) case. In this last case, two options were studied: a) without any constraint, and b) the worst best effort case before the videoconference becomes not possible. The difference of QoS can be clearly seen in Figure 4.10 with the GOLD SLA at the top left, the SILVER at the top right, the non-constrained BE at the bottom left and the constrained BE at the bottom right.

As illustrated within each image in Figure 4.10, the bandwidth at the application layer was monitored, obtaining a 300 Kbps for the GOLD SLA case, 128 Kbps for the SILVER SLA case and 80 Kbps for the non-constrained BE case. Regarding the worst BE case, we found that the lowest bandwidth to have a minimum stable Quality of Experience (QoE) was with a 50 Kbps bandwidth, below that the image would block.

In addition to the graphical and the bandwidth results, a more statistical set of results were obtained by looking the most demanding case: the network slice with the GOLD SLA. The first of these statistical results is the relation

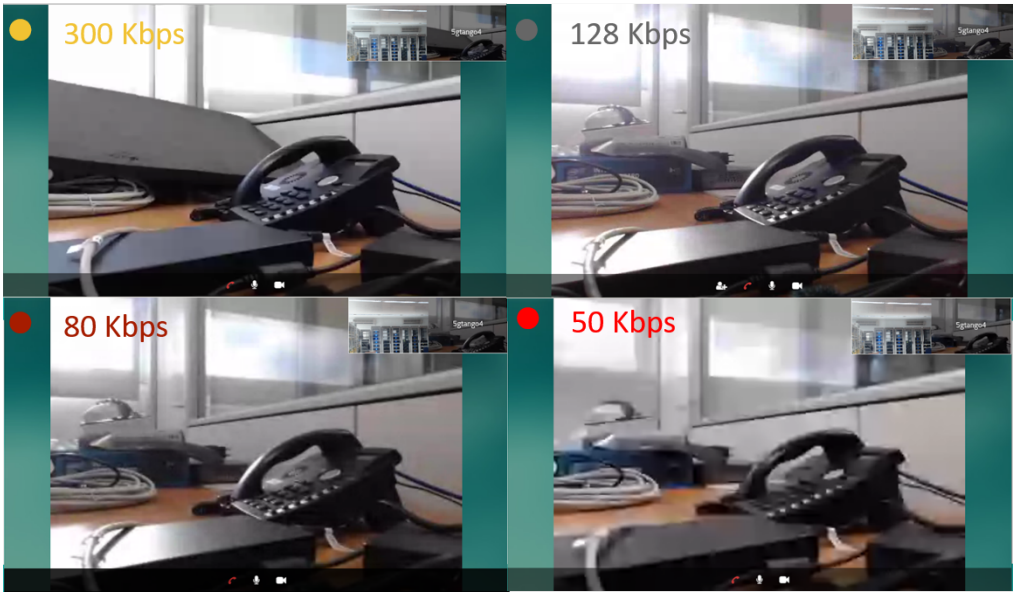


Figure 4.10: Examples of QoS degradation.

of transmitted packets and the number of packets lost presented in Figure 4.11. As previously said, each call had a duration of 180 s and a mean packet transmission rate of 50 packets/s (red line), which resulted on a total number 9000 transmitted packets. Taking this value and comparing it with the lost transmitted packets cumulative value (blue line in Figure 4.11), the packet-loss experienced was of a 0.022 %. This value (0.022 %) is much lower than the required by the GOLD SLA (Pl $< 1\%$), which confirms the correct implementation of the RTC NS and so that the architecture worked properly and deployed what it was expected offering to the final user the expected QoS and QoE.

In addition to this good result, a second parameter included in the SLAs was evaluated, the GOLD SLA RTT. In this case the audio and video data flows RTT values were studied and are presented in Figure 4.12. The RTT in the GOLD SLA was defined to be lower than 60 ms. As Figure 4.12 illustrates, the RTT mean value of both data flows are lower than the SLA RTT. So, the audio flow RTT (Figure 4.12-left) had a mean value of 0.5 ms while the video flow RTT had a mean value of 0.41 ms, much lower than the defined threshold.

Comparing the obtained results with the defined KPI requirements in the

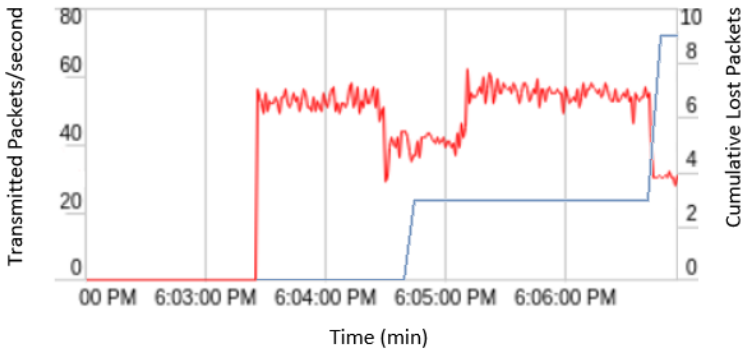


Figure 4.11: Transmitted packets per second (red line) versus cumulative value of lost sent packets (blue line).

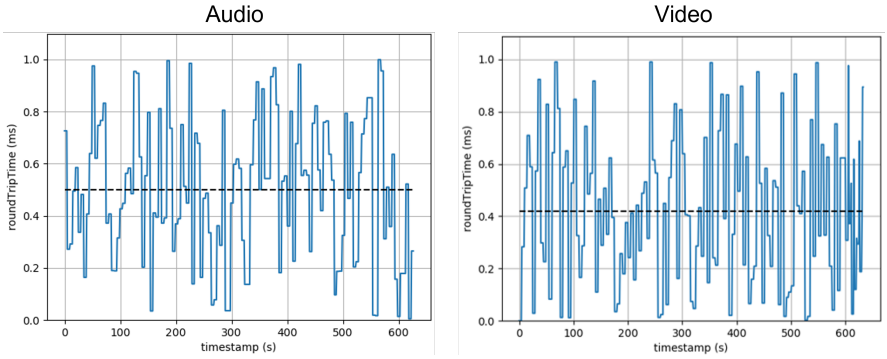


Figure 4.12: RTT values for the audio and video data flows.

GOLD SLA, the difference is quite important. The origin of these differences is because of the CSs deployed based on the NS flavors characteristics. As it was described when presenting the NS and its flavors, the gold and silver flavors demanded a minimum bandwidth of 500 Mbps and 1000 Mbps respectively. Because of these requirements, when each NS flavor was deployed and the NFV MANO passed to the Transport SDN Controller the corresponding requirement, the Transport SDN Controller selected the most suitable transport domain to deploy the CSs. The optical transport domain for the case of the gold flavor and the packet-based transport domain for the silver flavor. Following with the flavors bandwidth, it is important to remark the difference of bandwidth resources between the requested by the flavors and the obtained at the application layer results, meaning that the deployment

done had and over-assignment of bandwidth resources for the conditions in which the tests were done. To solve this over-assignment, a possible solution could be the implementation of policies to adjust the resources used based on what it is really need.

### 4.3.2 Use case II: solving a DoS attack on an E2E network slice

Having presented the deployment of network slices and how their associated SLAs are well respected as the expected QoS is achieved, with the following use case the objective is to validate the actions to apply by the evolved architecture presented in Section 4.2 when an SLA, more specifically a SSLA, is violated. For this use case, the ADRENALINE testbed was not available because of professional activities and so, the environment was done using a simulated multi-domain scenario.

Based on the use case described in [124] and [135], which introduced the idea to monitor and re-configure (when requested) a Firewall (FW) in a Vehicular Communications scenario, a similar use case was implemented to validate the evolved architecture in Figure 4.5 and the deployment and monitoring processes related. The final use case implemented is illustrated in Figure 4.13.

The used environment infrastructure was composed of an E2E SMD on top and two SMDs (i.e., Edge and Core) below connected among them through transport domain. Using this architecture, a deployed E2E network slice composed with a FW and a web server were deployed, more specifically, the FW at the Edge and the web server in the Core. Each one of the two services was defined in a different NST from the other and they were deployed as the slice-subnets of the E2E NST.

Despite nowadays there are very complex DoS attacks (i.e., data flows of Tbps) [136], in this experimental tests the DoS was not the focus but how the proposed architecture reacted in front of a simple DoS attack of 60 requests/min. So, based on this information, an E2E SSLA was created and used with the E2E NST to avoid a DoS attack that may block the web server functionality. The E2E SSLA had a metric defining the maximum threshold for the number of requests coming from a single user in a period of time with an SLO of 30 requests/min.

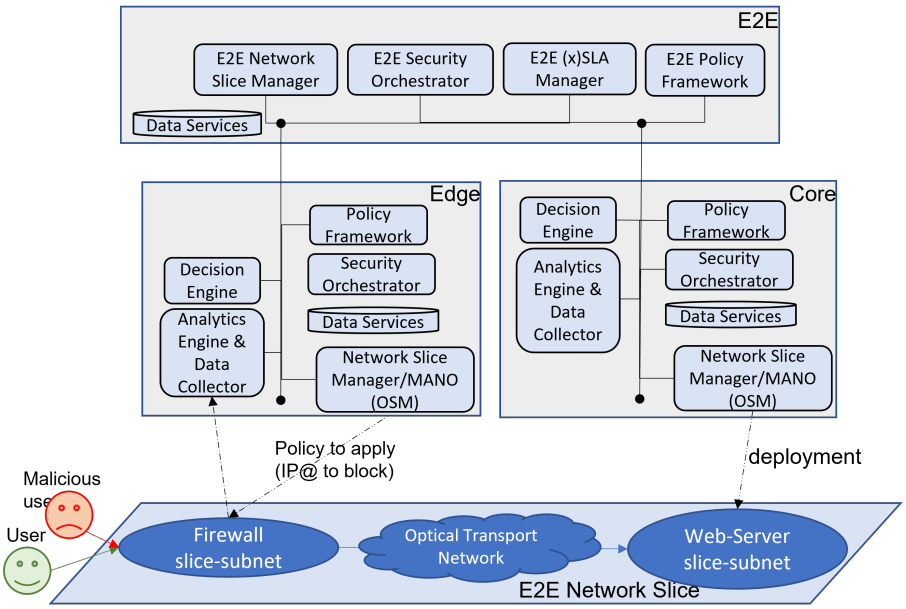


Figure 4.13: DoS use case design.

### Architecture interactions validation

Because of the architecture evolution from a single domain to the multiple SMDs and the E2E SMD, the deployment and monitoring actions were reviewed and validated through the acquisition of Wireshark samples.

On the one hand, the deployment procedure illustrated in Figure 4.6 is now validated in Figure 4.14 with the different HTTP requests done among the modules composing the architecture. The four main phases defined in the original workflow (Figure 4.6) can be identified. The first three requests correspond to the "E2E network slice & QoS selection and definition" phase, then the "Domain network slice deployment" phase is mapped with the requests 4,5,6,7,8 and 11. The third phase, "Setting monitoring rules", started first in the edge domain with the requests 9 and 10 and then in the core domain with the requests 12 and 14. Finally, the fourth and last phase called "Deployment confirmation" is done with the requests 13, 15, 16 and 17. All the requests are done accordingly to what it was expected and so, the architecture was implemented as expected for the E2E network slice deployment action.

On the other hand, while the service is alive and running and the users

| | Source | Destination | Lengt | Info |
|---|---|---|---|---|
| 1 | Tenant | E2E SSLA Mngr | 202 | POST /e2e_sla/deploy_slice_sla HTTP/1.1 |
| 2 | E2E SSLA Mngr | E2E Network Slice Mngr | 474 | POST /e2e_slicer/instance/deploy HTTP/1.1 , JavaScript Object Notation (application/json) |
| 3 | E2E Network Slice Mngr | E2E Sec. Orchestrator | 682 | POST /e2e_so/policies HTTP/1.1 , JavaScript Object Notation (application/json) |
| 4 | E2E Sec. Orchestrator | EDGE Sec. Orchestrator | 396 | POST /so/policies HTTP/1.1 , JavaScript Object Notation (application/json) |
| 5 | EDGE Sec. Orchestrator | EDGE Network Slice Mngr./MANO | 406 | POST /slicer/instance/deploy HTTP/1.1 , JavaScript Object Notation (application/json) |
| 6 | E2E Sec. Orchestrator | CORE Sec. Orchestrator | 396 | POST /so/policies HTTP/1.1 , JavaScript Object Notation (application/json) |
| 7 | CORE Sec. Orchestrator | CORE Network Slice Mngr./MANO | 406 | POST /slicer/instance/deploy HTTP/1.1 , JavaScript Object Notation (application/json) |
| 8 | EDGE Network Slice Mngr./MANO | EDGE Sec. Orchestrator | 90 | POST /so/policies/e14d7886-bf40-487b-841a-211c1000f227 HTTP/1.1 , JavaScript Object Notation (application/json) |
| 9 | EDGE Sec. Orchestrator | EDGE Decision Engine | 453 | POST /de/monitoring_rules HTTP/1.1 , JavaScript Object Notation (application/json) |
| 10 | EDGE Decision Engine | EDGE Analytics Engine & Data Collector | 285 | POST /ae/metrics HTTP/1.1 , JavaScript Object Notation (application/json) |
| 11 | CORE Network Slice Mngr./MANO | CORE Sec. Orchestrator | 90 | POST /so/policies/8ada5fbb-f636-4312-bd7c-a71322b8431e HTTP/1.1 , JavaScript Object Notation (application/json) |
| 12 | CORE Sec. Orchestrator | CORE Decision Engine | 453 | POST /de/monitoring_rules HTTP/1.1 , JavaScript Object Notation (application/json) |
| 13 | EDGE Sec. Orchestrator | E2E Sec. Orchestrator | 90 | POST /e2e_so/policies/e14d7886-bf40-487b-841a-211c1000f227 HTTP/1.1 , JavaScript Object Notation (application/json) |
| 14 | CORE Decision Engine | CORE Analytics Engine & Data Collector | 285 | POST /ae/metrics HTTP/1.1 , JavaScript Object Notation (application/json) |
| 15 | E2E Sec. Orchestrator | E2E Network Slice Mngr | 94 | POST /e2e_slicer/instance/policy-update/e14d7886-bf40-487b-841a-211c1000f227 HTTP/1.1 , JavaScript Object Notation (application/json) |
| 16 | CORE Sec. Orchestrator | E2E Sec. Orchestrator | 90 | POST /e2e_so/policies/8ada5fbb-f636-4312-bd7c-a71322b8431e HTTP/1.1 , JavaScript Object Notation (application/json) |
| 17 | E2E Sec. Orchestrator | E2E Network Slice Mngr | 94 | POST /e2e_slicer/instance/policy-update/8ada5fbb-f636-4312-bd7c-a71322b8431e HTTP/1.1 , JavaScript Object Notation (application/json) |

Figure 4.14: Wireshark capture with the secure E2E network slice deployment actions.

make use of it (requests 1 and 3), the monitoring process (Figure 4.7) may be reviewed with the Wireshark samples illustrated in Figure 4.15 which shows the two main phases. First, the "SSLA Metrics Gathering & Evaluation" phase with the requests 2 and 4 where the monitored data is gathered and sent to the DE. The DE applies the corresponding procedure to establish whether the received monitored data indicates if the metric defined in the E2E SSLA is not respected and so, the SSLA is violated. If the decision taken is that the violation has happened, the second phase "SSLA Violation Management" is triggered and applied. This procedure can be mapped with the HTTP requests 5,6,7,8,9 and 10 in Figure 4.15.

A final remark, and an important aspect to take into account, is that the whole process and its phases were done in a transparent way compared to the non-malicious user. While the E2E SSLA was being recovered, the benign user kept using the service and doing requests without any negative impact on its performance.

| | Source | Destination | Protocol | Lengt | Info |
|---|---|---|---|---|---|
| 1 | User | FireWall-subnet (EDGE) | HTTP/JSON | 87 | POST /fw/pings HTTP/1.1 , JavaScript Object Notation (application/json) |
| 2 | FireWall-subnet (EDGE) | EDGE Analytics Engine & Data Collector | HTTP/JSON | 87 | POST /ae/metrics/data HTTP/1.1 , JavaScript Object Notation (application/json) |
| 3 | FireWall-subnet (EDGE) | WebServer-subnet (CORE) | HTTP/JSON | 87 | POST /web/pings HTTP/1.1 , JavaScript Object Notation (application/json) |
| 4 | EDGE Analytics Engine & Data Collector | EDGE Decision Engine | HTTP/JSON | 101 | POST /de/collected_data HTTP/1.1 , JavaScript Object Notation (application/json) |
| 5 | EDGE Decision Engine | EDGE Policy Framework | HTTP | 263 | GET /pf/get_policies/dos HTTP/1.1 |
| 6 | EDGE Decision Engine | EDGE Network Slice Mngr./MANO | HTTP/JSON | 81 | POST /slicer/instance/update_config HTTP/1.1 , JavaScript Object Notation (application/json) |
| 7 | EDGE Network Slice Mngr./MANO | FireWall-subnet (EDGE) | HTTP/JSON | 81 | POST /fw/config HTTP/1.1 , JavaScript Object Notation (application/json) |
| 8 | FireWall-subnet (EDGE) | EDGE Network Slice Mngr./MANO | HTTP/JSON | 90 | HTTP/1.0 200 OK , JavaScript Object Notation (application/json) |
| 9 | EDGE Decision Engine | EDGE Sec. Orchestrator | HTTP | 297 | POST /so/policies/security_notification HTTP/1.1 |
| 10 | EDGE Sec. Orchestrator | E2E Sec. Orchestrator | HTTP/JSON | 86 | POST /e2e_so/policies/security_notification HTTP/1.1 , JavaScript Object Notation (application/json) |
| 11 | User | FireWall-subnet (EDGE) | HTTP/JSON | 87 | POST /fw/pings HTTP/1.1 , JavaScript Object Notation (application/json) |
| 12 | FireWall-subnet (EDGE) | EDGE Analytics Engine & Data Collector | HTTP/JSON | 87 | POST /ae/metrics/data HTTP/1.1 , JavaScript Object Notation (application/json) |
| 13 | FireWall-subnet (EDGE) | WebServer-subnet (CORE) | HTTP/JSON | 87 | POST /web/pings HTTP/1.1 , JavaScript Object Notation (application/json) |

Figure 4.15: Wireshark capture with the secure E2E network slice monitoring actions.

### Results

Having validated how the elements within the architecture correctly interact among them, another set of results is to check whether the actions done when the E2E SSLA violation is detected, allowed its resolution. Figure 4.16 shows the time evolution of the two data flows and the resolution of the DoS attack scenario.

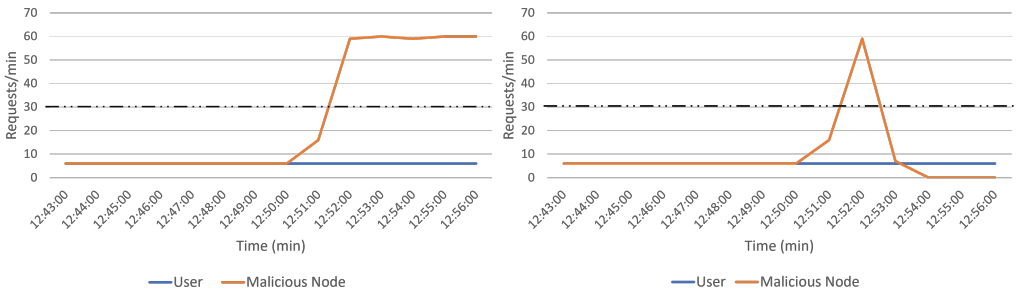


Figure 4.16: Received traffic in the firewall (left) and the web server (right).

As both graphs show, the two users send the same amount of requests (i.e., 6 requests/minute) to the web server (through the Firewall) but when the malicious user (orange line) decided to attack (minute 12:50), there is an increment of requests on the left graph passing from 6 requests/minute to 60 requests/minute (an increment of 10) with the objective to block smooth access to the web service and so, to decrease the performance of the service. As previously defined, the SLO was a maximum amount of 30 requests/min. So, when this threshold was not respected, the monitoring system reacted and applied the workflow procedure defined and demonstrated in Figures 4.7 and 4.15. By applying these actions, the result was the resolution of the DoS attack as it can be seen in the right graph of Figure 4.16. In there, the traffic from the malicious user reached a peak of 60 requests/minute and suddenly decreased to 0 requests/minute and an absolute blockage of its traffic in the FW slice-subnet.

One last set of results used to evaluate the implemented architecture is through the CDF. As illustrated in Figures 4.17 and 4.18, the CDFs in there show the probability of the DoS attack being detected and resolved in less than X seconds. Checking both figures, the detection of the DoS done by the security monitoring system has an 80% of probability to be detected in less than 82s; whereas the resolution action is done with a very low time, giving a

100% probability to require less than 0.4s to solve the threat. Furthermore, after conducting all the tests, the resulted detection mean time from the tests samples was of 49.5728s with a standard deviation of 20.07s. In parallel, the resulted resolution mean time had a value of 0.0463s with a standard deviation of 0.1005s.

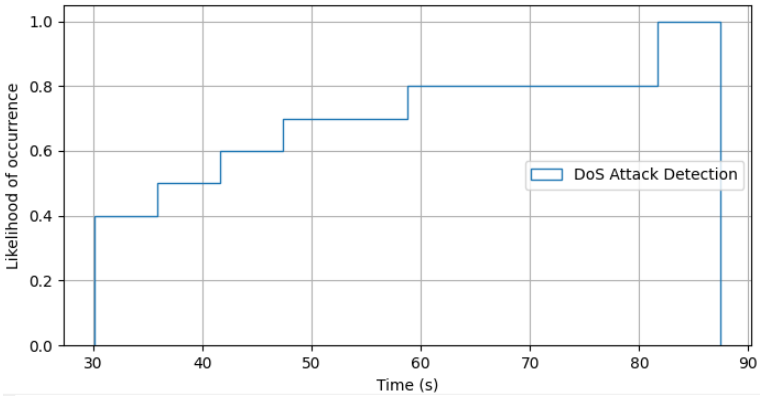


Figure 4.17: DoS detection time CDF.

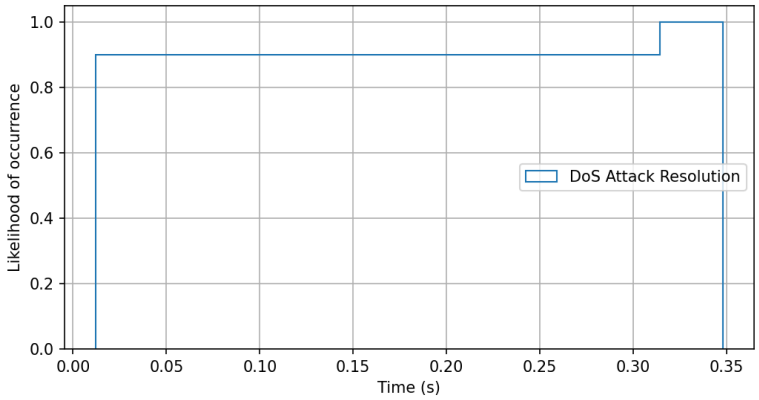


Figure 4.18: DoS (SSLA violation) resolution time CDF.

With the different results described, it is possible to conclude that the evolved architecture implementation to deploy E2E network slices with the enforcement of security elements based on SSLA requirements was achieved. The results allowed an evaluation from different points of view: a) the processes to deploy and monitor a secured E2E network slice, b) the correct

functionality of the deployed service and finally, c) the needs in terms of time for the monitoring system to reach and solve a threat/attack situation.

## 4.4 Conclusions

The results illustrate the need to monitor the services and resources composing the network slices from different points of view. While the most common way is to constantly evaluate service performance metrics, security metrics are equally important. Having different dedicated systems for each point of view is necessary as, when an issue appears, the source of it might seem to come from a performance cause when it actually comes from a security cause. The results previously described showed this need through the implementation of two different cases that monitor and enforce the QoS and the QoSec, respectively.

Finally, two different monitoring systems were developed and used in the experimental actions but they have some common functionalities that could be placed together. Unifying the right elements would simplify the overall monitoring actions but the limit in this unification is to keep the independence of each point of view (i.e., service performance and security).

# Chapter 5

# Cooperative management of network slices and transport connectivity services

## Contents

All the work described in the previous chapters has been done using a hierarchical control and management infrastructure. As hierarchical infrastructure models are based on having one layer over another, it is easy to add a new layer although this leads towards a centralization of the management actions and so, in terms of security, to what is called a central point of failure. For this reason, the work presented in this chapter studies a new possible architecture using a DLT such as Blockchain. This new architecture bases its way of working on a collaborative and cooperative model among control and management elements participating in the multi-domain NFV/SDN infrastructures.

This chapter is divided in three sections. Section 5.1 focuses on the management and control of computing resources in order to compose network slices across multiple domains. In addition it shows how the Network Slice Managers interact among them to share their own NST resources with other domains which do not have those resources but need them to compose an E2E network slice. Section 5.2 follows the path started in the first sections and focuses on the management and control of CSs to interconnect multiple transport domains in order to compose an E2E connectivity service across multiple transport domains. Finally, Section 5.3 describes the experimental validation and results obtained from the theory described in the previous two sections.

This chapter is based on the work presented in [137], [138], [139], [140] and [141].

## 5.1 Blockchain-based multi-domain network slices

As it happened in the last years, the MANO architecture associated to the NFV/SDN domains incremented their functionalities with the addition of new control and management elements, one over another creating a set of multiple layers. For each new layer, the management actions have become more centralised, leading to a possible central point of failure threat scenario. To avoid this possibility, the element on top (e.g., E2E orchestrator/controller/manager) of hierarchical architectures with the complete view across the multiple domains is removed. With the objective of using a Blockchain network that allows computing and transport domains to become a peer of

the Blockchain network and, finally, to share their domain information (i.e., per-domain slice-subnets) among them in a trustworthy way. Blockchain was designed to be a transparent and public way to share information among peers, but due to the fact that the proposed architecture is thought to be used by operators, a certain level of access restriction is necessary. To do so, a Permissioned Distributed Ledger (PDL) (i.e., private Blockchain) is used with the Network Slice Managers as its peers, which allows them to share their own resources with the other domains in a reliable and transparent way.

### 5.1.1 A Blockchain-based Network Slicing architecture

Figure 5.1 presents the Blockchain-based Network Slicing architecture with three NFV orchestration domains (i.e., access, metropolitan and core) used to create a P2P network in which each peer shares its available Network Slicing resources to instantiate part of an E2E network slice requested by another peer. In our study case, it is considered that all domains follow the same architecture presented in [26] in which the ETSI merged the 3GPP Network Slicing proposal with its standardised NFV architecture (blue components in Figure 5.1).

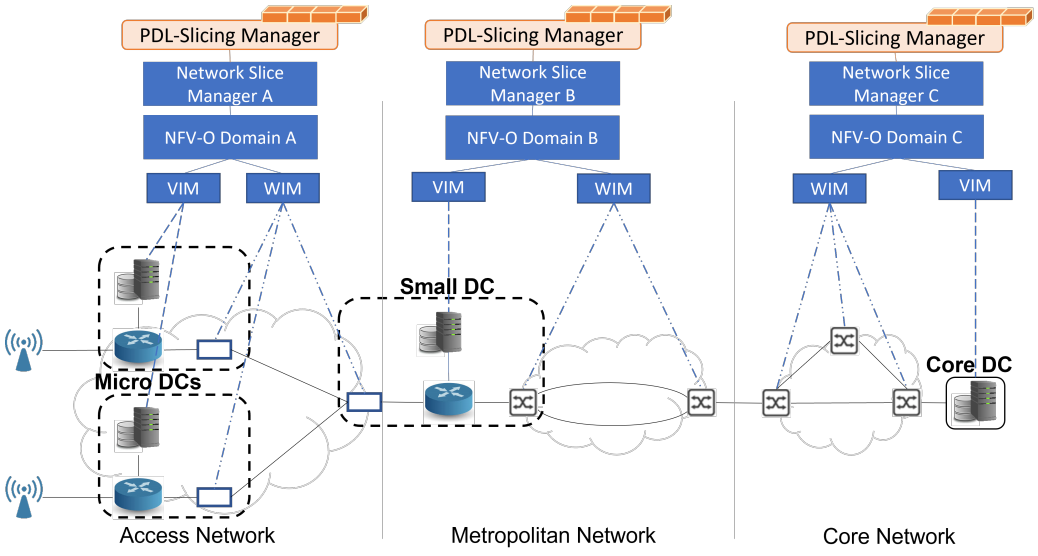


Figure 5.1: Blockchain-based Network Slicing architecture.

On the top of each NFV orchestration domain there is a Network Slice

Manager in charge of all the local domain Network Slicing actions and at the same time, thanks to a new module called PDL-Slicing Manager it becomes a peer of the Blockchain, allowing each Network Slicing Manager to request a slice-subnet from another domain to be part of the E2E network slice. Each Network Slice Manager is the owner of the resources in its domain, but the management of these resources changes depending on whether the instantiation is requested by the Network Slice Manager in the same domain or by a Network Slice Manager in a different domain. If a Vertical requests an E2E network slice instantiation to its domain Network Slice Manager and it only uses resources placed within the same Network Slice Manager domain, that Network Slice Manager is the unique owner and, through the NFV-O, it can apply any necessary E2E network slice related action to the resources in the network. On the other hand, if a vertical requests an E2E network slice instantiation using resources of different domains, the Network Slice Manager which has requested the E2E Network Slice is the unique owner, but any action to apply to the resources placed in other domains must go through the Blockchain and then, to the other Network Slice Managers. These will then request to their NFV-Os to apply the corresponding actions to their domain network resources. Keeping this in mind, if some domain network resources of a Network Slice Manager are being used for an E2E network slice requested by another domain Network Slice Manager, these computing resources cannot be modified unless the E2E network slice owner asks for it.

As previously presented, the Network Slice Manager component has a secondary but essential functionality for the collaborative architecture to work: being a peer in the Blockchain network. Despite Blockchain is known to be a secure and trustworthy technology and one of its main key stones is to be a public database, it is also possible to have a Blockchain with only a certain set of peers allowed (i.e., permissioned/private) to be part of it. This last idea is how the proposed architecture makes use of Blockchain: only the Network Slice Managers belonging to known domains can be a peer in the collaborative network. Despite the Blockchain being private, the rest of its characteristics are kept. So, there is no central point of authority (i.e., no Network Slice Manager has more power than the others), the information in the Blockchain is public (i.e., if a peer replies a NST from another peer, all the peers will realise), and all information is still secure as it is encrypted and can be read only by the accepted peers.

Under each Blockchain-based component and the Network Slice Manager, in the middle and lowest levels of the architecture presented in Figure 5.1

there are the NFV-Os in charge of the NSs and VNFs life-cycle management and orchestration. Then, below each NFV-O, there are the VIMs and WIM to deploy and configure the computing (i.e., VIMs) and networking (i.e., WIMs) resources placed in the latest level, the physical infrastructure.

**Designed smart contract**

Among the available Blockchain technologies, the selected one had to be composed only by peers fulfilling a set of requirements to write and validate data. Moreover, the Blockchain system also needed the capacity to have some autonomy to trigger some processes when data could be written. The solution for this last requirement is the use of a SC. As previously described, a SC is a program with a few set of functions known by all the Blockchain peers executed when requested. In the context of this work, the functionalities of the designed SC are:

- To distribute and store the Network Slicing elements such as the NSTs that one domain offers to the others, the requests and the NSIs generated from them.
- To trigger the deployment of slice-subnets (i.e., NSTs) belonging to other domains and composing the E2E network slice requested by one domain. When a domain owner has selected the necessary NSTs for the desired E2E network slice, a transaction is distributed and generates the events with the specific NST identifier and the associated Blockchain address of the peer owning that resources. Only the owner (i.e., the peer with the specific Blockchain address) will take the event and deploy the right NST in its domain.
- To constrain what each peer is able to do with the deployed resources. A deployed slice-subnet should only be terminated (under normal circumstances) when the original E2E network slice requester requests its termination.

### 5.1.2 The PDL-Slicing Manager

The PDL-Slicing Manager has the following functionalities:

- NST resources distribution: when a new Network Slice Manager joins the Blockchain network, the information of its local NSTs is added

into the Blockchain and distributed to all the other peers so they have a copy with the new available NSTs.

- E2E NSIs management: Based on the designed E2E network slice, it manages the generated E2E NSI data object and controls that all its slice-subnets (i.e., NSTs) are deployed or terminated by interacting with its local Network Slice Manager or those placed in other peer domains.

As illustrated in Figure 5.2, the internal architecture of the PDL-Slicing Manager is composed by the following components:

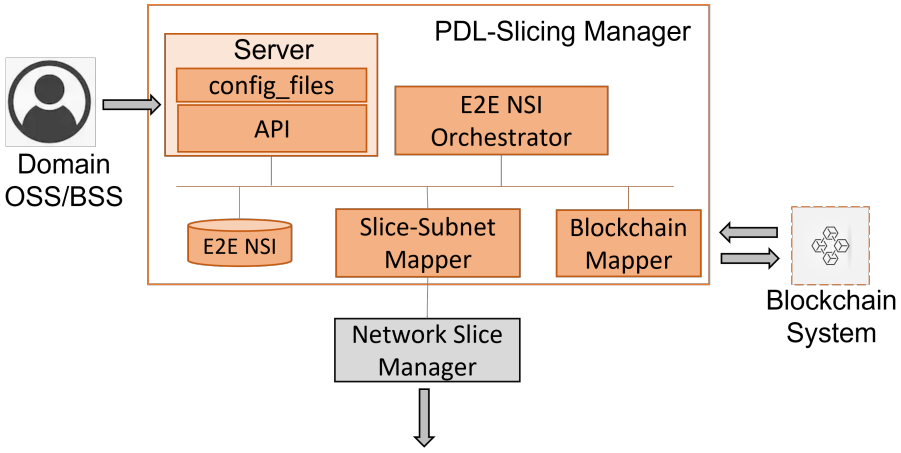


Figure 5.2: PDL-Slicing Manager internal architecture.

- Server: It contains the API with all the possible actions to be requested and the configuration parameters.
- E2E NSI Orchestrator: It takes care of managing the requested actions coming from the API to deploy or terminate the E2E NSI requested.
- E2E NSI: This component is the database where the E2E NSI data objects are stored.
- Network Slice & Blockchain Mappers: These two components take care of translating all the internal information into the required requests format; the set of slice-subnets composing the E2E network slice to the Blockchain System or the specific slice-subnet (i.e., NST) to the local Network Slice Manager.

### 5.1.3 Instantiation procedure description

While in a hierarchical scenario with a multi-domain E2E Network Slice Manager all the network slice instantiation process is done in that single manager, now the process must involve all the domain managers/controllers through the use of the Blockchain they are part of.

The complete process to deploy an E2E network slice in a multi-domain architecture involving different managers and controllers is divided in two main phases: the first phase focuses on the design of the desired E2E network slice to be deployed (Figure 5.3), and the second phase its deployment procedure (Figure 5.4). Before describing them, one last important remark is the fact that in both figures there is one more element called PDL-Transport Manager. This element is similar to the PDL-Slicing Manager but focused on the management of multi-domain transport CSs and it is described in Section 5.2 as in the current section the focus is on the Network Slicing resources.

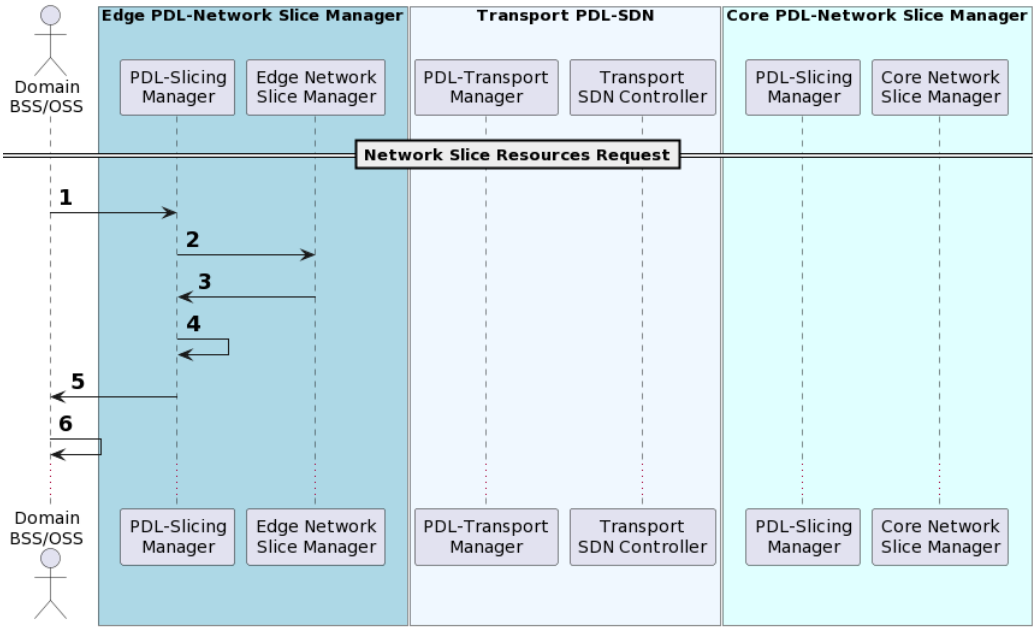


Figure 5.3: NST catalogue requests and E2E network slice design.

The E2E network slice design procedure (Figure 5.3) begins when a Domain OSS/BSS needs an E2E network slice with a set of services requirements to be fulfilled. It requests to the PDL-Slicing Manager the catalogue with

the available NSTs (step 1). Then, the PDL-Slicing Manager requests the local NSTs (steps 2/3) to its Edge Network Slice Manager and joins them with its copy of the Blockchain-stored NSTs information (step 4). Once all the available NSTs are joined, the PDL-Slicing Manager sends back the information to its Domain BSS/OSS (step 5). So the last step, the number 6, corresponds to the Domain BSS/OSS designs the E2E network slice by selecting the necessary NSTs.

Once the E2E network slice is designed (the set of NSTs involved is selected), the collaborative deployment procedure may be triggered. In order to have a control on the requests, the E2E network slice deployment can only be triggered through a PDL-Slicing Manager, which avoids the possibility that other domains may request them. As Figure 5.4 illustrates, this procedure is divided in four phases: the E2E network slice request, the E2E network slice components instantiation, the E2E network slice components stitching and finally the E2E network slice confirmation.

The "E2E network slice request" consists of a unique step (1 - Figure 5.4), when a PDL-Slicing Manager receives a request from their Domain OSS/BSS.

The second phase, the "E2E network slice components instantiation" begins once a request is received by the PDL-Slicing Manager and this module starts to request each of the slice-subnets deployment; on the one hand, to its local Network Slice Manager (step 2) that manages their deployment with the NFV-O below (step 3) and, on the other hand, if the slice-subnet belongs to another domain, it distributes the requests to all the peers in the Blockchain (steps 4/5). Only the owner (step 4) of the requested slice-subnet will forward the request to its associated Network Slice Manager (step 6) and acknowledge the slice-subnet ownership (step 7). While the second action is done, the associated Network Slice Manager manages the slice-subnet deployment with the NFV-O below (step 8). At this point the PDL-Slicing Manager managing the E2E deployment must wait until all the slice-subnets are deployed. On the one hand, it checks the local slice-subnets with its Network Slice Manager and, once it is deployed, updates the E2E network slice information (step 9). On the other hand, for the slice-subnets from other domains, it must wait until a transaction arrives (steps 11/12). The transaction is originated when the Network Slice Manager in another domain finishes its slice-subnet deployment and informs its PDL-Slicing Manager about it (step 10). The PDL-Slicing Manager distributes the transaction to all the peers (steps 11/12) and only the original requester keeps it (step 12) and acknowl-

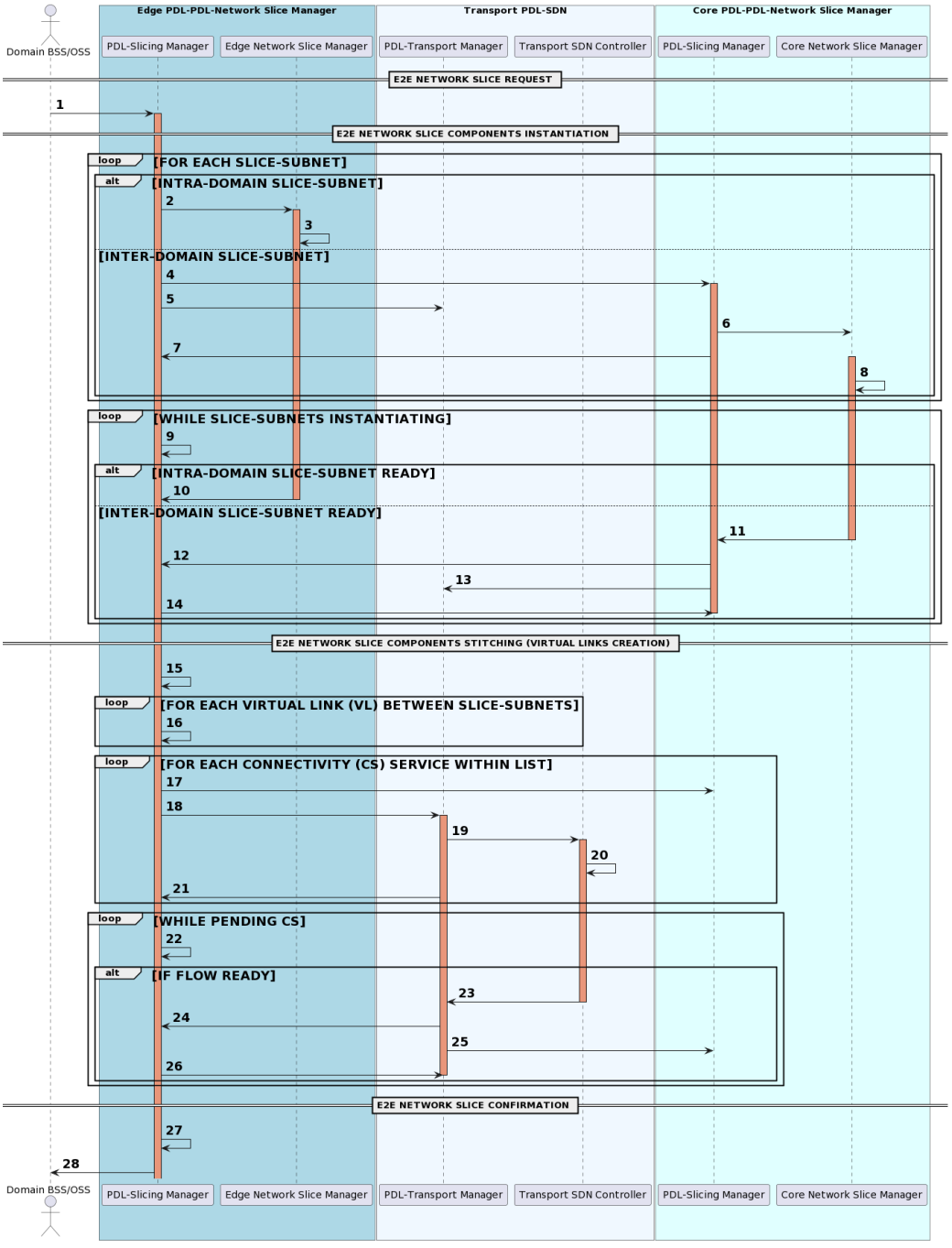


Figure 5.4: Blockchain-based network slice deployment.

edges (step 13) its reception.

Once all the slice-subnets are ready, the "E2E network slice components stitching" phase begins with the PDL-Slicing Manager computing the path for each pair of interconnected slice-subnets and generates a list of CS (step 14). Based on the list, it distributes a transaction per each CS (steps 15/16) to all the peers although only the PDL-Transport Manager with the ownership will take it (step 16), forward it to its associated Transport SDN Controller (step 17) and acknowledge its reception (step 19). Meanwhile, the Transport SDN Controller configures the CS (step 18). Once a CS is ready, the PDL-Transport Manager is informed (step 20) and then it generates another transaction for all the other peers (steps 21/22). Again, only the CS requester will take it (step 21) and confirm its reception (step 23).

Finally, the fourth and last phase, the "E2E network slice confirmation" starts once all the CS are ready and the slice-subnets stitched. On that moment, the PDL-Slicing Manager updates the E2E NSI and informs about its completeness (step 24).

## 5.2 Collaborative composition of E2E CSs

The work presented in the previous Section (5.1) focused on presenting a cooperative system to manage computing resources to compose the E2E network slices and assumed that the interconnection of these resources was done across only one Transport domain (Figures 5.3 and 5.4). Based on that, the following steps were focused on having multiple Transport domains working together similarly to what it was presented in the previous section. So, now the objective was to make the SDN Controllers on top of each transport domain an equal participant of another P2P network. To do so, the same approach done with the network slices was applied on a hierarchical transport multi-domain scenario. Now, the E2E SDN Controller on top of a transport multi-domain scenario is substituted by a new (per domain) module designed with the functionalities to complement each domain SDN Transport Controller to join in the Blockchain system and interact with other domain SDN Transport Controllers as illustrated in Figure 5.5. As presented in this figure, now on top of the controlling system there is a Blockchain Network composed by a set of "PDL-Transport Manager" modules (i.e., peers). These elements allow each (Optical) Transport SDN Controller to become a part of a Blockchain Network without any difference in terms of power influence

on the overall E2E vision compared to the other equal elements.

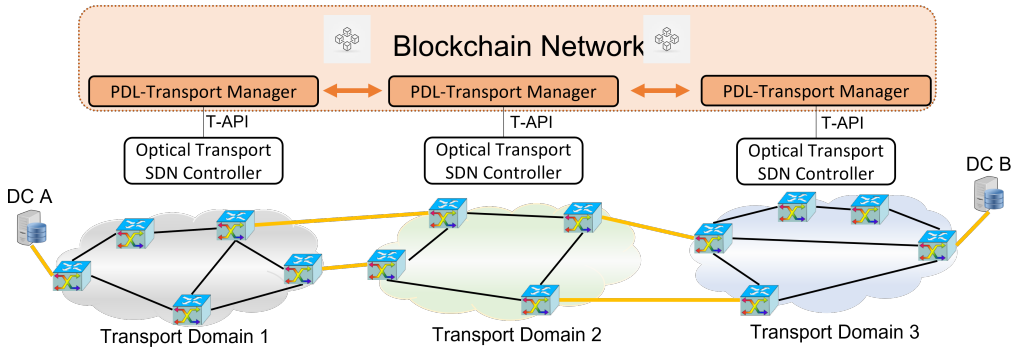


Figure 5.5: Generic Blockchain-based SDN Controller architecture.

The way this evolved architecture works is based on the cooperation and collaboration among the multiple domains. So, when a CS (i.e., an interconnection between two endpoints/node ports) requires the participation of multiple domains, the involved SDN peers interact among them to create the necessary domain CSs to compose the final E2E CS. Based on this, this new module takes assigned events generated from the Blockchain requests done by other peers, and maps the events into a domain data model requests for the underlying Transport SDN Controller to create the corresponding domain CS. While there are multiple data models (i.e., YANG, NETCONF, etc.) available to define the context and topology of a transport network domain (optical or packet-based), the selected one was the T-API [120] data model. An example of its implementation can be found in the T-API-enabled Transport SDN controller described in [119].

### 5.2.1 Modelling transport domains

The use of T-API was selected as it allows the deployment of per-domain CS to configure the E2E transport connection (i.e., E2E CS). T-API allows the definition of SDN control plane functions to a set of service interfaces. T-API bases its data model on a set of elements that allows to abstract the existing physical resources information into a single data object and to apply actions over the physical resources based on the abstracted information. One of the benefits of using T-API is the use of REST commands and the fact that it allows to constantly abstract the topology from one layer to another

one above it as many times as required through a relationship between a T-API provider (i.e., SDN Controller) and a T-API Client (i.e., Application, Orchestrator or a parent SDN Controller). By using the T-API v2.1.3 [9] photonic media model it is possible to manage the connectivity, topology and path computation services. So, the WDM is modelled as a (single layer) T-API forwarding domain (FD) and the Media Channel (MC) protocol qualifier is covered within the "PHOTONIC_MEDIA" layer (within the context data object).

Regarding the essential T-API elements, the most basic set of information is the definition of a context with a list of Service Interface Points (SIPs). As described in [9], a context is an abstraction allowing logical isolation or grouping of other abstracted network resources, and a SIP is the logical representation of the external view of any port placed on the edge of a node. So, with only a list of SIPs in the context, it is possible to request and configure CSs between two edge points in the domain without the need to know the details of the network resources infrastructure associated to the context domain. Having the list of SIPs enables a T-API client to request MC CSs between endpoints, optionally specifying optical spectrum to be provisioned. To support this possibility, each SIP element within the context is augmented with specific MC resource availability, referred to as MC pool. The MC pool encompasses information about the supportable, available and occupied frequency slots. In addition to the SIPs list, a context may contain a more detailed definition of the real network infrastructure. To do so, the use of Node Edge Points (NEPs), Nodes, Links and Topology becomes essential. A NEP is the representation of each physical port available in a real network element. A NEP may be considered internal (i.e., no SIPs associated) or external. If a NEP is internal, it contains the information regarding the available, supportable and occupied spectrum (i.e., MC pool). On the other hand, if a NEP is external, one or more SIPs may be associated to allow multiple CSs over the same physical port and the spectrum information is specific for each SIP. A Node is the abstract representation of a set of network resources (i.e., a single physical network element or a set) and it essentially aggregates a set of NEPs. Moreover, a Link is the representation of the association between two or more NEPs. Finally, a Topology defines the abstracted topological characteristics from a set of network resources. To sum up, a Topology is composed by a set of Nodes and Links, a Node is a set of NEPs and a Link is the association of at least two NEPs.

From the T-API point of view, a CS is the representation of the intercon-

nection request between two SIPs. As a result, a Node Connection element is configured between each pair of NEPs belonging to the same node for each of the nodes involved in the route between the source and the destination SIPs. So, as previously presented, the creation of a CS only needs two different SIPs. However, more complex requests are possible. For example, using optical network resources required to specify the spectrum slot and optionally the internal links for each requested CS in order to ensure the spectrum continuity alongside all the optical domains involved.

One last key concept related to the use of the T-API within the Blockchain-based architecture is the differentiation between E2E CS and Domain CS: an E2E CS is a composition of multiple Domain CSs. Once the Domain CSs are provisioned, each domain topology with any of the Domain CSs is updated with the corresponding Connection Endpoints (CEPs), which are the elements that define T-API Connection elements composing each Domain CS.

### 5.2.2 Blockchain advantages & drawbacks in the networks management

Together with the advantages brought by the use of the T-API data model, and based on the characteristics described in Section 2.4, the use of Blockchain within the described SDN scenario brings the following advantages:

- Any request for networking resources (i.e., CS) is public, transparent and immutable once done, which makes it highly difficult to tamper it in case traceability is needed.
- The avoidance of an element on top of a hierarchical architecture removes the central point of failure threat that may block E2E CS actions.
- Due to the permissioned feature, when a new peer joins the Blockchain network, its information is dynamically added and the other peers update their vision of the whole E2E transport infrastructure.
- As there is no hierarchy and all the peers are equally important, if a peer becomes unavailable, the others can still work together. This gives independence and autonomy to all the SDN domains.

- The way the architecture is designed, only one Transport SDN domain is able to configure/control each domain CS as its creation request is linked to a unique Blockchain address identifier.

On the other hand, the current architecture has some drawbacks too:

- A domain not being available may be included in a path computation because the domain computing the path does not have updated infrastructure information.
- Blockchain is designed to avoid unfinished transactions using an associated cost per transaction, which means that any transaction must be generated with precision and the security/confidence to be accomplished.
- Due to the use of costs, the information in a transaction must be as precise as possible and avoid possible redundancies.

Despite having a strong dependency on the Blockchain technology, the influence of the fore-mentioned drawbacks may be solved by checking each domain resources availability when the path is computed or by improving the amount of information to be distributed and stored in the Blockchain using, for example, abstraction models.

### 5.2.3 Abstraction models for the Blockchain architecture

When managing a Blockchain-based SDN architecture an important aspect to consider is the amount of information required to manage the network resources exposed by each domain controller to the upper domain controller. The more details a controller has about its below domain, the more precise its network control and management becomes. However, in parallel, it also takes longer due to the need of processing the higher amount of information. This trade-off (i.e., detailed management vs. processing time), together with the fact that a network is composed by multiple transport domains, raised the necessity to use abstraction models to control the available resources. The process to abstract the topology of one optical network domain is a complex procedure due to the massive amount of information required to define all the nodes, links and the spectrum information for each of the previous elements.

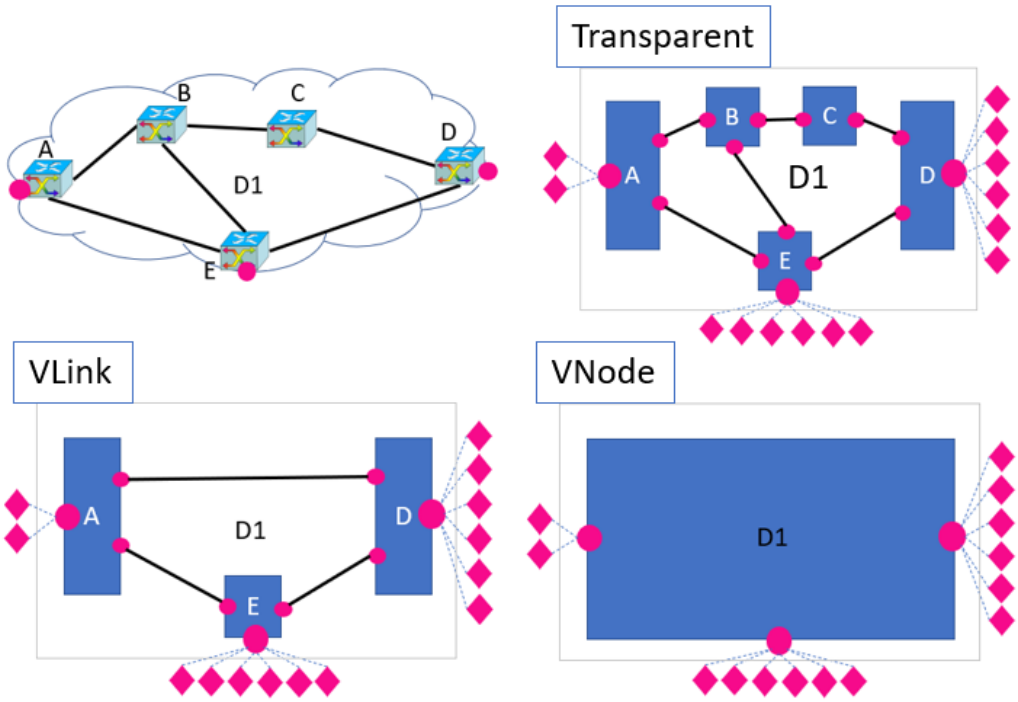


Figure 5.6: Original domain and its abstracted topologies.

When dealing with abstraction processes, the concept of "node" has to be understood as an entity that may be a single physical network element (i.e., router, switch, bridge, etc.) belonging to a domain but that may also be a complete domain infrastructure, for example when dealing with other domains. As illustrated in Figure 5.6, the network resources in the optical transport domain (top left) may be abstracted with all the physical nodes in detail (top right) or the whole optical transport domain may become a single node (bottom right) in the resulted abstraction process. The elements defined in the T-API data model are used in Figure 5.6, so the SIPs are the pink diamonds, the NEPs are the pink circles, the nodes are the blue boxes and the links are the black lines between nodes. Moreover, Figure 5.6 shows the three abstraction models that were implemented and compared in the experimental actions later described in Section 5.3.

The first abstraction model is the so-called "transparent" model. As its own name shows, there is a transparent abstraction procedure which means that the complete T-API context coming from the local SDN Controller is

not processed and it is simply distributed among all the peer domains. As presented in Figure 5.6, all the nodes, NEPs, SIPs and links in the physical infrastructure (top left) are selected and the abstraction resulting (top-right) is an exact copy of the real resources.

Secondly, the VNode abstraction model processes the original network topology (Figure 5.6 top left) to generate a new T-API context in which the whole domain becomes a single node (Figure 5.6 bottom-right). In this abstraction model, all the real nodes and links information is omitted. Instead, the new abstracted T-API context contains the list of SIPs (mandatory in all three abstraction models) and the topology has a single node with the NEPs and their associated SIPs. So, the internal domain infrastructure is not distributed to all the other peer domains in the Blockchain.

The last and third model is the VLink model. The basic idea behind this model is to select a specific group of the real nodes and define a new set of "virtual" links that interconnect the selected nodes among them. The condition used to select the nodes was to keep those that had NEPs with associated SIPs. As illustrated in Figure 5.6, the abstracted T-API context (bottom left) had only three nodes and three virtual links when originally (top right) they were five and six respectively. This abstraction model has a peculiarity compared to the other abstraction models in order to equalize the three of them in terms of routing path computation costs. While in the Transparent and VNode all the links have a cost equal to 1 (i.e., hooping), in the virtual links created when using the VLink model, each one had a weight equally proportional to the number of hoops in the real physical infrastructure. So, using the domain abstraction example illustrated in Figure 5.6, the link between the nodes A and D has a weight of 3, which is equal to the number of hoops in the transparent model domain.

Finally, there is one more element related to the abstracted domains information which is how they are related to each other with the so-called Inter-Domain Links (IDLs). Together with each domain context, the different peers need to share their knowledge about the IDLs around them in order for the other peers to complete the E2E vision of the whole physical infrastructure. Using the transport domains example in Figure 5.5, the IDLs are represented with the yellow lines and they are identified with two NEP identifiers, one of each transport domain involved in that IDL.

### 5.2.4 The PDL-Transport Manager

The PDL-Transport Manager allows, as introduced, that each SDN Controller becomes part of the collaborative system where they share a set of information related to the internal resources available in the different (optical) transport network domains without needing an E2E SDN Controller on top managing the complete E2E transport network actions. The communication between the PDL-Transport Manager and the underlying Transport SDN Controller in the south-bound interface is done through the use of the T-API data model [120] and, on the north-bound interface, REST requests are defined to get, distribute and manage the network information and the E2E CSs deployment and terminate actions.

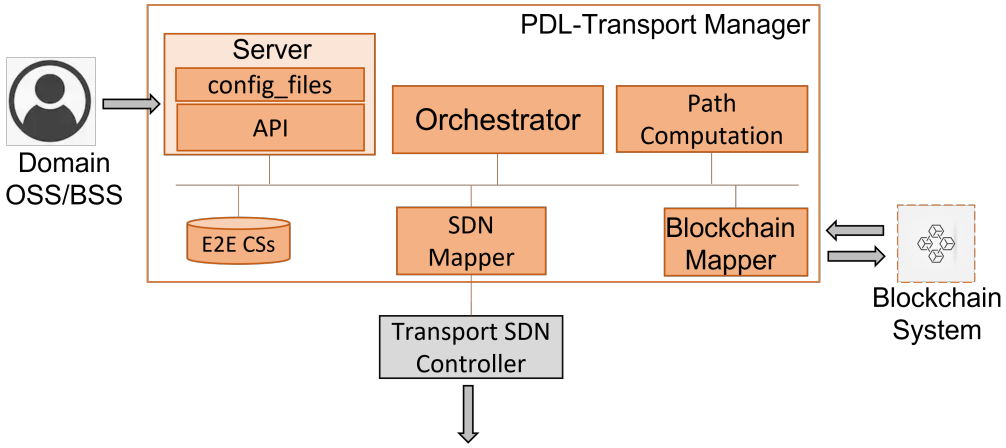


Figure 5.7: PDL-Transport Manager internal architecture.

The PDL-Transport Manager internal architecture is presented in Figure 5.7 and its main components are:

- Server: It contains the API with all the possible actions to be requested and the configuration parameters (e.g., abstraction model used).
- Orchestrator: It takes care of managing the requested actions coming from the API to control the E2E CS deployments.
- Path Computation: This component manages the local graph with the E2E multi-domain topology to generate the possible routes for each E2E CS.

- E2E CSs: This component is the database where the information related to the requested and deployed E2E CS are stored.

- SDN & Blockchain Mappers: These two components are in charge of generating the T-API Domain CS requests depending if they have to be requested to the local (SDN Mapper) or to another domain (Blockchain Mapper) SDN Controller.

This new module is in charge of the following functionalities:

- Context abstraction: Once the PDL-Transport Manager is launched and depending on the configured abstraction model, the PDL-Transport Manager obtains the local optical domain context and processes it in order to generate the new (transparent, VLink or VNode) abstracted context.

- Context resources distribution: The abstracted context is split in SIPs, NEPs, Nodes, Links and Context metadata (i.e., UUID, name, list of SIPs UUIDs, topology) and distributed in the Blockchain. This division was done to avoid the distribution of the complete context once a resource (i.e., NEP, SIP) is selected in an E2E CS, so only the specific resource updated information is distributed in a new transaction.

- IDLs distribution: In addition to the context, each PDL-Transport Manager is in charge of distributing the information of the physical links that interconnect the domains among them. Before distributing this information, a PDL-Transport Manager checks if an IDL has already been distributed and, in that case, the IDL is omitted. This means that the last domain to join does not distribute anything as all IDLs were already shared. The IDLs are not abstracted as they are not part of a single optical SDN domain but they are a shared element between two optical SDN domains.

- E2E CS management: Deploying and terminating the requests to connect or disconnect two SIPs from two different optical transport domains. During the deployment process, there are some important actions such as the path computation, the spectrum continuity enforcement and checking and updating of the resources availability information in the Blockchain.

#### Routing and Spectrum Assignment

As previously presented, all the PDL-Transport Manager peers have the complete E2E vision of the infrastructure. So, each PDL-Transport Manager peer is in charge of the Routing and Spectrum Assignment (RSA) process for any E2E CS request received.

Regarding the routing phase, a K-Shortest Path procedure is applied. All the route possibilities between the source and destination nodes are generated but only a few set (i.e., 20) of the shortest paths selected. Then, the PDL-Transport Manager validates if the first route in the list is feasible. Instead, if the PDL-Transport Manager discovers that there are no resources available on the selected route, it searches for the next viable route within the routes list until it finds one that fulfills the requirements.

The requirement used to decide if a route is feasible or not is the spectrum continuity among all the SIPs and NEPs involved in the selected route. To ensure it, the PDL-Transport Manager managing the requested E2E CS gets the selected NEPs (only for the transparent and VLink models) and the SIPs involved in the route from the Blockchain and takes their available spectrum slots. Among all the available slots, a list with the common available slots is created. Then, to select the slot among all the options, the PDL-Transport Manager applies a two-step policy selection: first an Exact-Fit policy and, if that does not exist, it applies a Best-Fit policy with the closest spectrum slot to the requested capacity. Finally, with the common spectrum continuity selected, the E2E CS can be created with the composition of all the required Domain CSs.

### 5.2.5 Blockchain-based management of transport domains

Having described the technologies used and the new module to let the transport SDN domains to be part of the Blockchain network, the last elements to introduce are how the multiple peers (i.e., domains) interact among them to deploy the E2E CSs. To do so, the SC and the different workflows to share the context information and deploy the E2E CSs are presented.

### Designed smart contract

As it happened with the management of network slice resources described in the previous section, a new SC was designed and implemented in order to determine who may participate in the Blockchain (those with the implemented SC ID) and to define the actions to trigger when something could be requested to the Blockchain related to the Transport Domains. The functionalities of the designed SC are:

- To distribute and store SDN information from each abstracted domain (e.g., T-API SIPs and T-API topology), together with the IDLs data between the SDN domains.
- To automate the multiple and specific requests generation to deploy Domain CSs. When a PDL-Transport Manager has selected the best route based on the spectrum resources available, it distributes a transaction that generates an event with the specific Domain CS information and the associated Blockchain address of the peer owning those resources. Then, all the peers receive the event but only the peer with the specific Blockchain address will take the event information and process it to generate the Domain CS through a T-API CS request sent to the SDN Controller below.
- To limit the rights to apply actions over other domains physical resources. Only the peer that has requested a Domain CS (as part of an E2E CS) is able to request its termination aside the Domain CS owner itself, which can terminate it locally through its local SDN Controller.

### A collaborative E2E topology

Before any E2E CS may be requested, it is necessary that all domains distribute their local abstracted context to the other peer domains, so each of them may create the local graph with the E2E infrastructure view as presented in Figure 5.8. Using Figure 5.5 as a reference, there are two sets of information to be distributed by each optical SDN domain (called PDL-SDN peer in Figure 5.8): a) the abstracted context and, b) the set of known IDLs. Once these two information sets are distributed, each PDL-Transport Manager is able to update the vision of the whole E2E physical infrastructure.

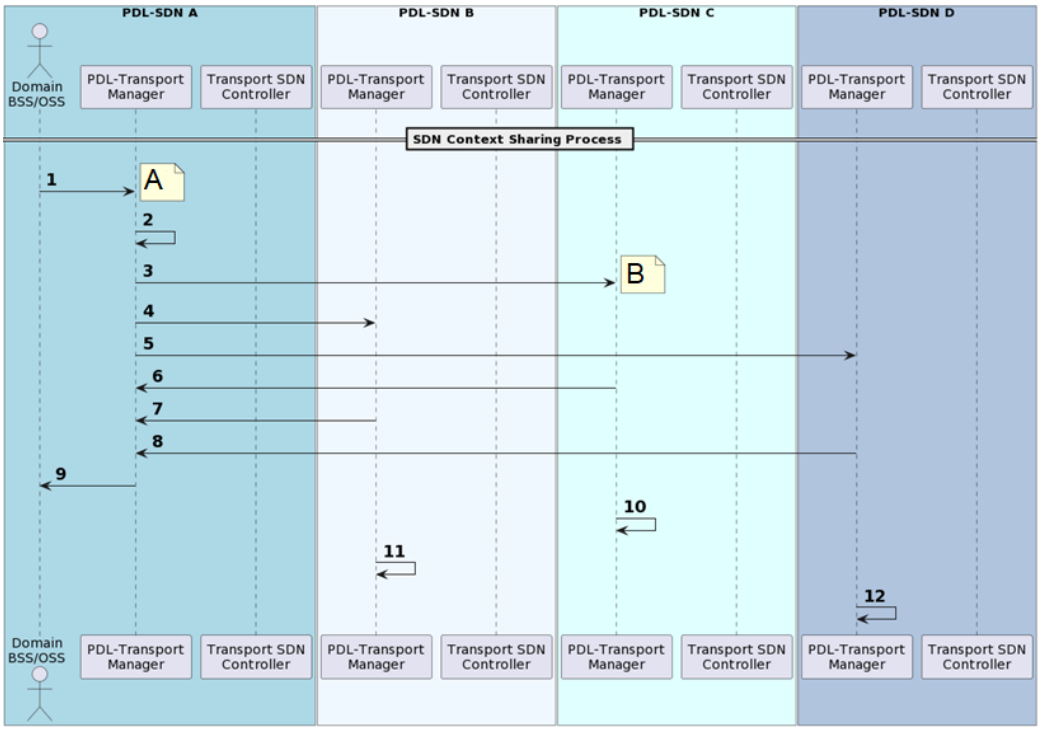


Figure 5.8: SDN context and IDLs distribution.

The workflow (Figure 5.8) follows these steps. The Domain OSS/BSS passes the IDLs information (previously defined between optical domain operators) and requests to distribute all this information (step 1) together with the known IDLs (Note A). Then, based on the IDL information, the local E2E network topology view (e.g., a graph element) is updated with the IDL data by adding to the graph only the new nodes and links (step 2). Once done, a transaction to distribute both sets of data (i.e., the IDLs and the local abstracted SDN Context (based on the T-API data model)) is generated. Once the transaction is done, all the other PDL-SDN peers (i.e., domains) receive an event (steps 3/4/5) informing that a "NEW DOMAIN" is distributing its data (Note B). After that, the event reception is confirmed by all the PDL-SDN peers (steps 6/7/8). Finally, the Domain OSS/BSS is informed about the correct distribution (step 9) and all the PDL-SDN peers update their local graph with the new SDN and IDL information (steps 10/11/12).

Once this procedure is done, each single PDL-SDN peer knows about the existence of all the other PDL-SDN peers that together compose an E2E

optical infrastructure with the distributed abstracted T-API contexts information and the associated IDLs. Once all this information is distributed, one final aspect to take into consideration is the possibility of data leakage to a peer that should not be part of the Blockchain system. To avoid this possible situation, the use of a permissioned (i.e., private) Blockchain becomes necessary as only known peers are involved in the p2p system because the access to it is not public. Moreover, the use of the predefined data models (i.e., T-API) defines the same information to be shared by all the peers, making it equal among them.

### Blockchain-based E2E CS requests management

Once the multiple PDL-SDN peers domain contexts and the IDLs information are distributed, the whole E2E transport topology is ready to be used to deploy and terminate E2E CSs. The process to deploy an E2E CS is divided in four main phases as presented in Figure 5.9.

The first phase, "E2E CS request & data object creation" begins with the Domain OSS/BSS (i.e., Cloud Operator) requests an E2E CS defining the source and destination (domain, node and SIP identifiers) and the desired spectrum (e.g., GHz) (step 1). The PDL-Transport Manager does a first check on the SIP availability by checking if they are already used. If so, the Domain OSS/BSS is informed about it and the E2E CS deployment finished. Otherwise, the E2E CS data object is created (step 2).

The second phase called "Path Computation (Routing & Spectrum Assignments)" consists of the next steps. First, the PDL-Transport Manager checks for a set of possible routes between the source and destination nodes using the local E2E graph (step 3). Then, the shortest one is selected and the nodes information are mapped to the SIPs and internal NEPs information (step 4). With the chosen SIPs and internal NEPs, the PDL-Transport Manager checks if they all have a common spectrum available with the requested capacity (step 5). After that, if a common spectrum is found, a set of Domain CSs is defined to be deployed to compose the E2E CS. Otherwise, steps 4 and 5 are started again to search for the following route possibility, always from shortest to the longest one (steps 6/7).

The third phase is called "Connectivity Services Creation" and it is triggered when the PDL-Transport Manager reads the list of Domain CSs and if they belong to the local PDL-SDN peer, the Domain CS request is sent to the Transport SDN Controller (Transport SDN Ctrl. in Figure 5.9) below

(step 8). Then, the local Transport SDN Controller deploys the Domain CS and confirms it back to the PDL-Transport Manager (steps 9/10).

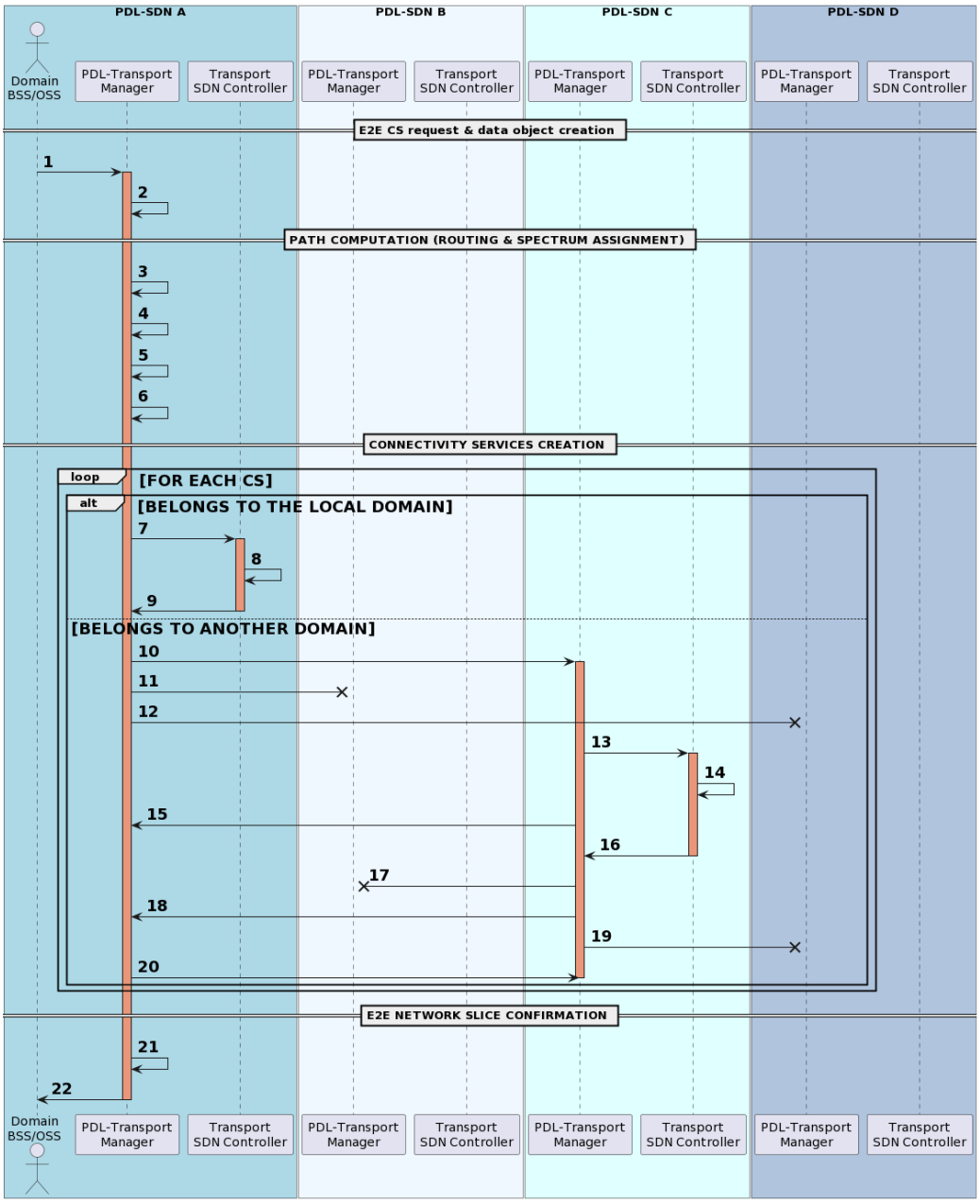


Figure 5.9: E2E network slice deployment.

If the Domain CS belongs to another PDL-SDN peer, a Blockchain transaction with the Domain CS information (i.e., domain SIPs, capacity and, when using the transparent abstraction model, the internal links) is generated and distributed to all the other peers (steps 11/12/13). As described, only the owner of those resources takes the transaction, maps it and forwards it to its local Transport SDN Ctrl. below to deploy the Domain CS (steps 14/15). Alongside of steps 14 and 15, the PDL-Transport Manager confirms that the transaction has been taken on (step 16) and, once the Domain CS is ready, the Transport SDN Ctrl. informs its upper PDL-Transport Manager which generates a new Blockchain transaction (step 17). Once generated, the new Blockchain transaction with the updated Domain CS information is distributed (steps 18/19/20) and, finally, only the E2E CS owner processes the transaction with the updated Domain CS information and confirms its reception (step 21).

The last and fourth phase is the "E2E CS Confirmation" and is composed of two steps. First, the E2E CS data object information is updated with the latest Domain CS status (i.e., ready) (step 22) and second, the Domain OSS/BSS is informed about the requested E2E CS being deployed and available to be used (step 23).

Finally, the process to terminate an E2E CS is not illustrated because it follows a simpler procedure. Once a terminate request is received, the PDL-Transport Manager checks the multiple Domain CSs information composing the E2E CS and generates a set of Blockchain transactions (one per Domain CS) with the necessary information to terminate them (i.e., Domain CS identifier and PDL-SDN peer address owner). Once the Domain CSs are terminated and the E2E CS owner is informed about it, the resources availability is updated and distributed to the Blockchain for future E2E CSs.

## 5.3 Experimental validation

Based on the theoretical and design work presented, some experimental results were obtained in order to learn and start to open the path towards the possibility to use Blockchain as another possible element to assist on the networks management. This first subsection presents the Blockchain technology selected to carry out the multiple experiments. Then, the second subsection describes the results obtained starting with a small use case with only Network Slicing resources, then the deployment of CSs using a single transport

domain is added and, finally, the third subsection describes a set of results focusing on the use of abstracted models to determine how the amount of multiple transport domains affects the deployment of E2E CSs.

### 5.3.1 Blockchain selection

Among all the Blockchain possibilities the two most known solutions are Bitcoin [77], Ethereum [78] and Hyperledger [79]. The first two are widely known, specially in terms of cryptocurrency transactions, while the last is used in industrial environments. Among these three possibilities, one had to be chosen. In terms of wide capabilities aside of the cryptocurrency transactions, Ethereum allows the possibility to design and create decentralised applications (dapps) using SCs. Moreover, the time required to create the data blocks between Bitcoin and Ethereum is quite different. While Bitcoin uses SHA-256 and takes minutes [142] to encrypt a block, Ethereum uses ethash which requires few seconds [143], a delay the would surely affect the results presented. For this reason, Bitcoin was removed among the initial possibilities. Instead, Hyperledger also allows this possibility and is quite focused on industrial environments. Despite this, an Ethereum-based solution was finally selected. The reasons for this decision are: a) creating, configuring and maintaining a Blockchain is a complex process which requires a set of stable computing resources, b) the ADRENALINE testbed was not always available as it was used in professional research projects and, because of this, most of the experimental results were obtained by using an emulated environment.

The selected Ethereum-based solution is the Ganache [144] emulator illustrated in Figure 5.10. Ganache allows the definition of multiple aspects such as the number of peers, the amount of Ether (i.e., Ethereum's cryptocurrency) per peer and others. The reasons to select Ganache as part of the testbeds environment were: a) it allowed to build a Blockchain system able to accept and manage SCs without the need of applying long configuration actions in only few seconds, b) it allowed to have the complete integrated environment with the rest of the use cases elements and finally, c) Ganache has the capability to keep logs and a list of the Blockchain transactions, allowing a smooth PDL-Slicing/Transport Manager module development, integration and testing with the rest of the environment. As Ethereum was the selected Blockchain, the SCs were developed using its own programming language called Solidity. As the use of Ganache means that the Blockchain selected

is a standard implementation of an Ethereum network, it implies that the consensus mechanism [145] used is the "de facto" Proof of Work (PoW) and the incentives for the Blockchain peers to participate in the network are the standard values defined in [146].

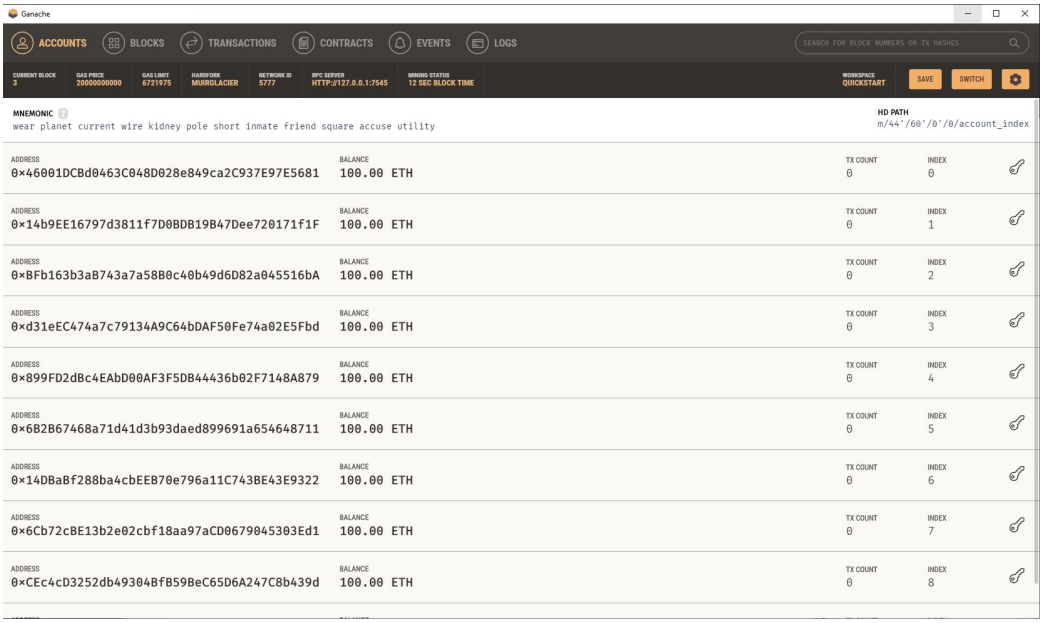


Figure 5.10: Ganache emulator.

### 5.3.2 Network slices deployment validation & delays

The main focus of the experimental tests done was on the delay increment that the use of Blockchain would add into the deployment of multi-domain network slices. To do so, the following process was done: first a set of results were done with the focus on those actions involving Blockchain as presented in [137]. Secondly, based on the previous results and as illustrated in [138], a similar experimental process was done using computing resources available in the CTTC ADRENALINE testbed. In a third phase and based on the work described in [139], the focus was placed on the creation of the connectivity services to interconnect two different endpoints and finally, both elements (i.e., computing and networking resources) were deployed to have the complete E2E network slice deployment using the blockchain-based infrastructure as demonstrated in [140].

### Initial Blockchain deployment time values

The first set of results were obtained using a use case illustrated in Figure 5.11 with the two computing domains (A & B), each with its own Network Slice Manager and NFV orchestration infrastructure (NFV-O A & NFV-O B). In each Network Slice Manager there are different NSTs available, Network Slice Manager A has two NSTs (NST_1 & NST_2) and Network Slice Manager B has one (NST_3). Finally, on top of each domain, there are two PDL_Slicing Managers. Each of them is a node (i.e., peer) in the Blockchain network and they shared in there a resumed version of the NSTs available in each Network Slice Manager The test done was the simulation of deploying an E2E network slice composed by the NST1 and the NST3 (in domain A and B respectively). Finally, this first experimental phase was done in an emulated scenario in which the Network Slice Managers were not sending the requests down to the NFV-Os. It was designed this way for two reasons: first, to avoid waiting periods and, second and more important, to clearly show the influence of the time values in which the Blockchain layer was involved. For this reason, it was expected to have T1, T3 and T5 nearly to 0s. With this

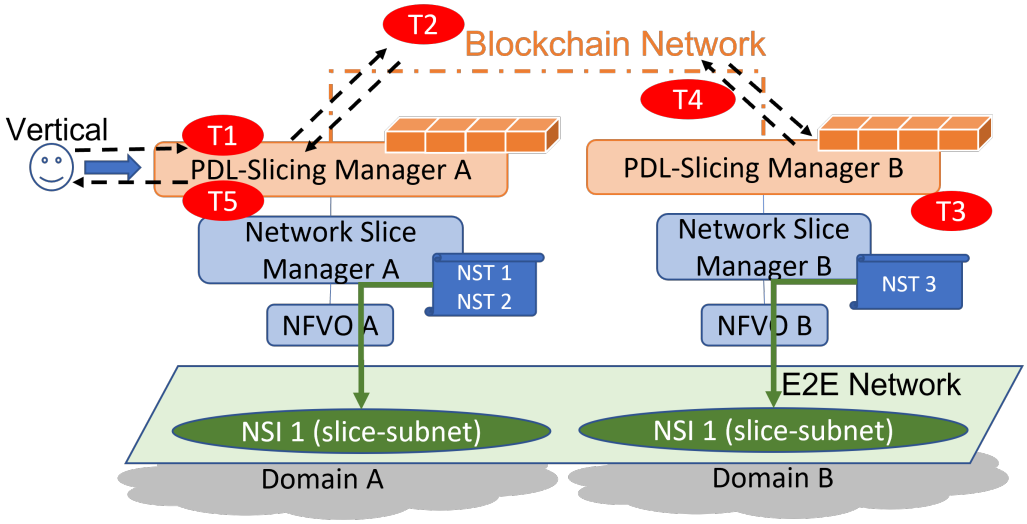


Figure 5.11: Instantiation use case.

initial infrastructure, the objective was to obtain an initial set of time values associated to the different deployment process steps in which the Blockchain elements were involved. Based on those times, we could decide whether the

use of Blockchain could be interesting and so, proceed with further research and more complex elements. These initial time values are referenced with red circle numbers in Figure 5.11 which are:

1. Slice request (T1): The time between the vertical request (i.e., E2E network slice composed by NST_1 and NST_3) reaches the PDL-Slicing Manager A and all the slice-subnets instantiation requests are ready to be sent either to the local NFV-O A (i.e., NST_1) and to other NFV-Os (i.e., NST_3) through the Blockchain.

2. Blockchain request distribution (T2): The time for the PDL-Slicing Manager A to distribute the event with the information about each slice-subnet (i.e., NST_3) and to get the confirmation from the corresponding owner.

3. Slice deployment (T3): The time to deploy the corresponding local slice-subnet and, once done, to send the updated information about the slice-subnet deployment to the PDL-Slicing Manager B.

4. Blockchain update distribution (T4): The time required to add the updated information into the Blockchain, to distribute the corresponding event to warn about the deployments status the original E2E network slice requester and to receive its answer.

5. Slice end-deployment (T5): The time to process the last actions and leave the instance ready to be used by the Vertical.

With the use case ready and once the experimental actions were done, the deployment workflow validation and the time results were obtained. Figures 5.12 and 5.13 present the HTTP traffic and the Ethereum transactions respectively, which demonstrate the design process illustrated in Figure 5.3 and the part of the computing resources deployment process in Figure 5.4 (i.e., from step 1 to 14 and 27 and 28).

First, all NSTs were added into the local DB (Figure 5.12 step 1) of each Network Slice Manager (Slicer A/B in Figure 5.12) and uploaded in the Blockchain (Figure 5.13 transactions A and B) and so, all the NSTs are available (Figure 5.12 step 2) for any vertical. When the Vertical A requests the deployment of an E2E network slice (Figure 5.12 step 3) to the Network Slice Manager A, this creates the NSI object with its slice-subnets (i.e., selected NSTs) and requests their deployment to corresponding Network

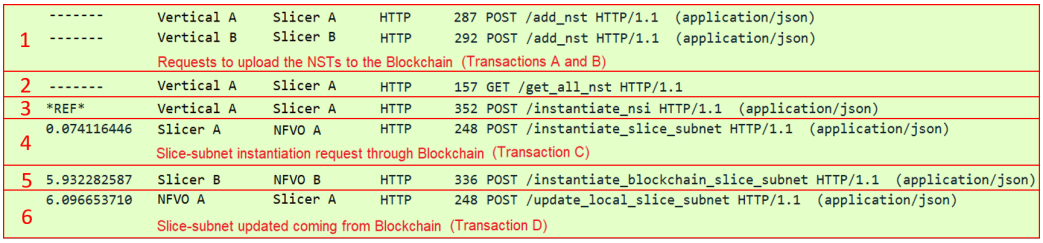


Figure 5.12: HTTP traffic.

Slice Manager domain: its own NSTs are requested (Figure 5.12 step 4) to its domain NFV-O A, while the external NSTs requests are sent to the Network Slice Manager B through the Blockchain (Figure 5.13 transaction C).

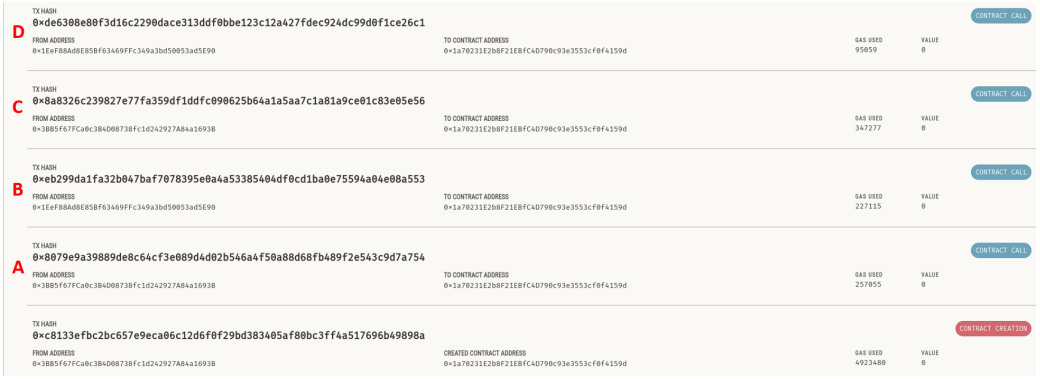


Figure 5.13: Ethereum transactions.

Then, the Network Slice Manager B creates an NSI to keep the local track of the computing resources used and requests (Figure 5.12 step 5) to its local NFV-O B the deployment of the NST. Once all the subnet-slices (i.e., NSTs) are deployed, the E2E network slice owner (i.e., Network Slice Manager A) is informed directly by its local NFV-O A (Figure 5.12 step 6), or through the Blockchain (Figure 5.13 transaction D) about those deployments done in other domains.

Having validated the corresponding workflow, the obtained time results are illustrated in Figure 5.14 which shows the mean values of each one of the five time samples previously defined. Furthermore, Tab.5.1 presents the corresponding standard deviation values of each one of the time samples.

Keeping in mind that no computing resources were deployed in this em-

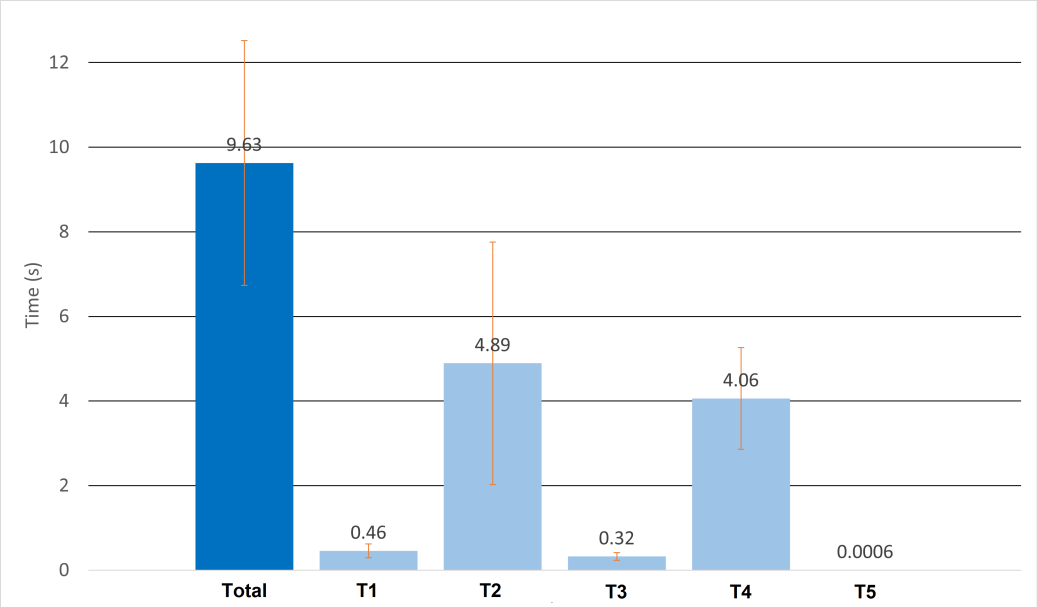


Figure 5.14: Set up phases delay time.

ulated process, the mean value of the total instantiation process time is of 9.627458s with T1, T3 and T5 having small values (close to 0) as these three columns are the actions done locally in each NFV domain. Regarding the two important time samples, on the one hand, T2 shows a value of 4.89s which is the time needed to request a non-local slice-subnet and create the Blockchain transaction and distribute it. On the other hand, T4 has a value of 4.05s which is the time to update the latest information of the slice-subnet and distribute it towards the original requester.

Table 5.1: Time steps standard deviation.

| $\sigma$ (s) | | | | | |
|---|---|---|---|---|---|
| Total | T1 | T2 | T3 | T4 | T5 |
| 2.89 | 0.16 | 2.86 | 0.092 | 1.20 | 0.0003 |

Using the deployment time values from the work done in [111] as a reference, where the authors compared the influence of KVMs and Containers creation during an instantiation of an hybrid network slice, the mean value KVM-based slice-subnets instantiation was around 11 minutes (i.e., 660s)

and 87.5s for the container-based slice-subnets. Taking the worst case possible from the previous graph and table (i.e., total time plus the standard deviation), the time value is of 12.523s. Removing the three time sample which didn't involve Blockchain actions, the time remains with a value of 11.5s. This means that on the KVM-based slice-subnet instantiation, the time added by the Blockchain actions is around 1.71% (11.5s over 660s). On the container-based slice-subnet instantiation, this percentage become bigger with a value of 11.61%.

### Computing resources vs. Blockchain

Based on the previous set of results, it was decided to go for the next step and use a real infrastructure testbed and validate with real computing resources deployment how Blockchain may affect. To do so, a set of similar tests were done using the CTTC ADRENALINE Testbed and the control infrastructure illustrated in Figure 5.15.

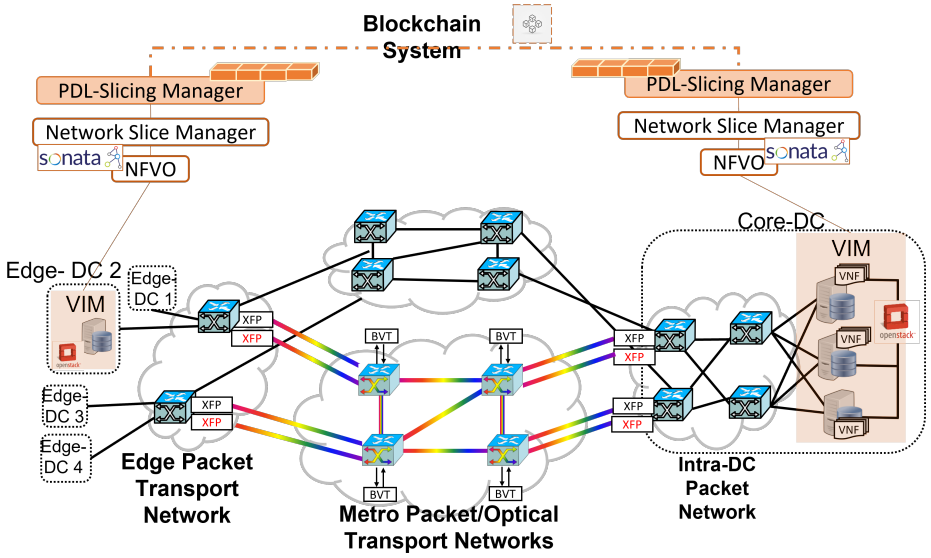


Figure 5.15: NFV Blockchain-based architecture on the ADRENALINE testbed.

As already described, this testbed is composed of different transport networks (both packet and optical-based) and different domains such as four edge DCs and one core DC. To develop the tests we used two computing

domains, the Edge-DC2 and the Core-DC. In both, there is a SONATA SP solution that has both Network Slicing and NFV-O functionalities and above them a PDL-Slicing manager acting as a peer of the created Ethereum Blockchain. Below each SONATA SP there is an associated VIM to control the domain computing resources.

To follow with the previous time results, the same use case is done with an E2E network slice composed by 2 NSTs distributed in the Edge-DC2 and Core-DC nodes presented in Figure 5.15. To make the deployment more realistic and based on the amount of resources available in each DC, the NSTs were composed of a single NS with 2 VNFs (i.e., 3 VMs in total) in the Edge-DC2 and 3 NSs using the same NS (i.e., 9 VMs in total) in the Core-DC.

Using the same time values defined previously (i.e., T1, T2, ..., T5) the same experimental test was done and the mean time values are illustrated in 5.16 with the standard deviation values in Tab. 5.2. Checking the different time values, it is clear that Blockchain has no big influence on the deployment time. Times T2 and T4 are the two periods of time in which Blockchain is involved (like in the previous test). T2 with a 6,867s and T4 with 5,003s. These two values compared to the others (i.e., T1, T3 and T5) are the lowest values and so, they are the time values that affect the least to the overall instantiation process. Adding the times of these two steps (i.e., T2+T4 = 11,87s) and comparing it to the total instantiation time (553,87s), the percentage of influence is equal to a 2,14% of the total time.

Table 5.2: Time steps standard deviation.

| $\sigma$ (s) | | | | | |
|---|---|---|---|---|---|
| Total | S1 | S2 | S3 | S4 | S5 |
| 24,7 | 0,3 | 2,8 | 35,8 | 2,1 | 20,7 |

It is important to remark that the last experimental test was done using only a KVM-based network slice, which allowed to obtain a similar behaviour than the one it was described in the initial test results previously presented in the first part where the deployment porcess was emulated without real computing resources being deployed.

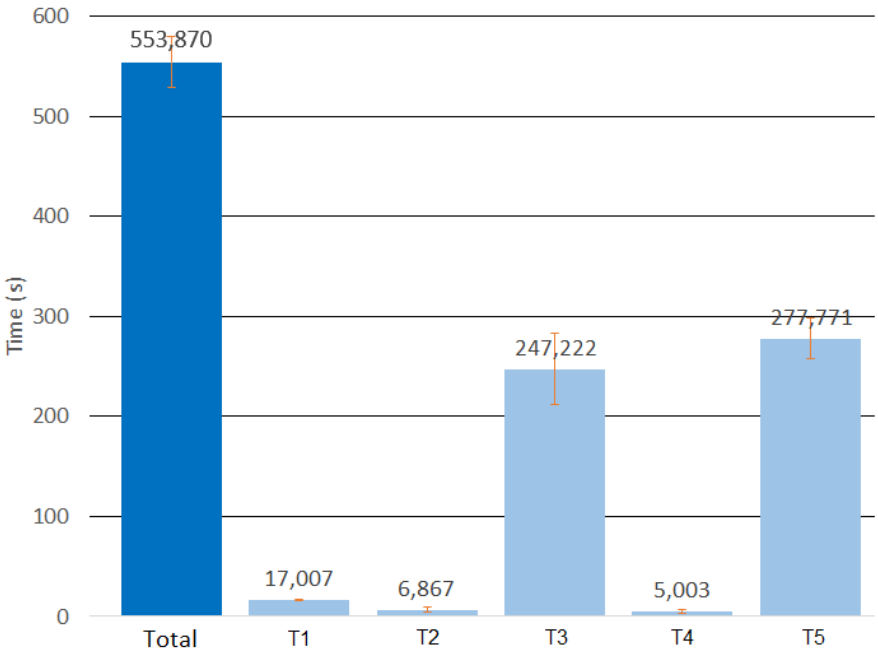


Figure 5.16: Measured setup delay.

### Networking resources vs. Blockchain

Having described the behaviour of the cooperative deployment of computing resources, one last element is missing in order to have the complete E2E network slice deployment: the cooperative deployment of networking resources. To do so, as illustrated in Figure 5.17, the PDL-Transport Manager module is used. Now, using the CTTC ADRENALINE testbed [131], a set of four transport domains (i.e., edge, transport and a core packet-based domains and a a single optical-based transport domain) are used. Each domain has an SDN Controller domain managing the incoming domain CSs requests and above it, the PDL-Transport Manager module to become a peer in the Blockchain and manage the cooperative actions. To validate the CSs deployment with the proposed solution, an experimental test was done by sharing the context of three different domains (edge, optical and core) and requesting a bidirectional CS (i.e., two unidirectional CSs) between the edge and core domains through the optical transport domain.

As similarly done with the computing resources, the set up and deployment procedures are validated as Figures 5.18 and 5.19 show with the HTTP

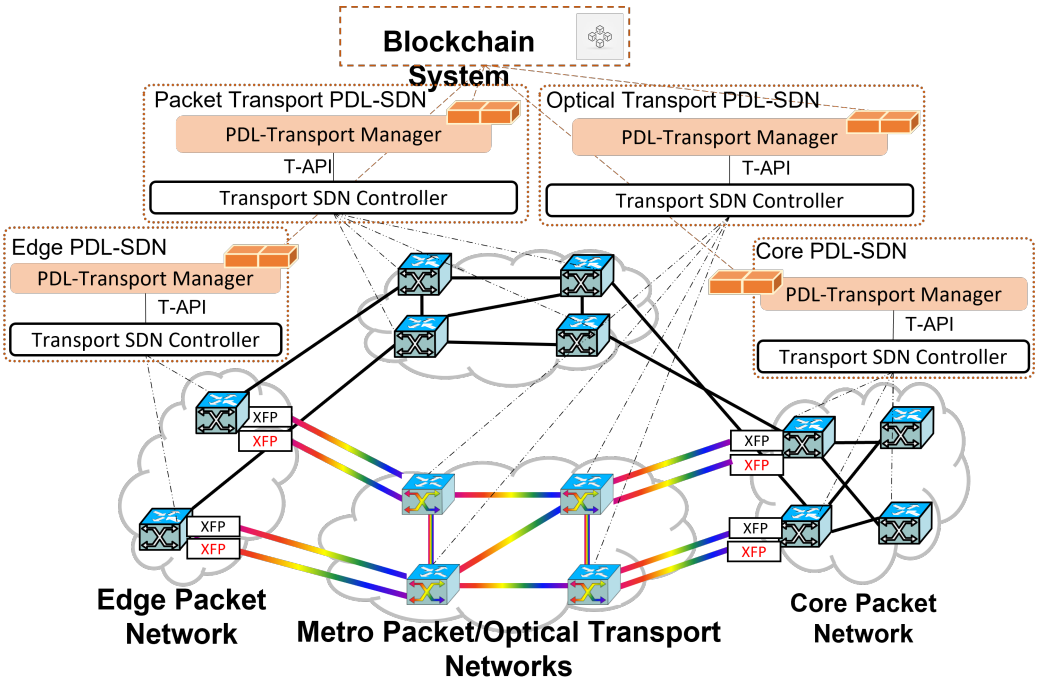


Figure 5.17: SDN transport Blockchain-based architecture on the ADRENALINE testbed.

requests and the Blockchain transactions generated among the peers. First, regarding the workflow to distribute the context of each SDN domain, Figure 5.18-A shows the three HTTP requests to share the context of each SDN domain. Once each HTTP request reaches the corresponding PDL-Transport Manager, a Blockchain transaction is generated and distributed as demonstrated in Figure 5.19-A. Secondly, regarding the deployment of CSs (i.e., steps 17 until 26 in Figure 5.4), Figure 5.18-B shows the HTTP request that triggers the whole process and the HTTP requests to create the different CSs between the PDL-Transport Manager and its associated Transport SDN Controller. The distribution of the CSs requests across the Blockchain network is presented with the two pairs of transactions log in Figure 5.19-B/C. Each transaction log belongs to the CSs creation distribution and the CSs update distribution.

From the multiple tests done, a set of measures were obtained. Table 5.3 presents the mean and standard deviation deployment time values for each

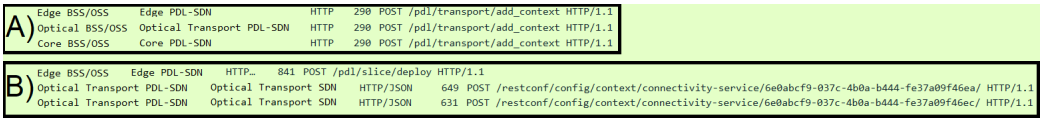


Figure 5.18: HTTP requests to order the distribution of the context and the CSs deployment.


Figure 5.19: Blockchain transactions log.

E2E CS (columns 1, 2), their total deployment time (column 3) and the total time associated to the different Blockchain transactions (column 4). Based on the results, a complete E2E CS configuration requires around 2s, giving a total mean time value of 4.08s to create the two unidirectional domain CSs (CS1 and CS2 in Table 5.3) and the Blockchain transaction mean time value is of 2.34s.

Comparing the 2.34s with the 4.08s, around 50% of the time delay is caused by the Blockchain transactions which is an important percentage in an isolated CS deployment context. Instead, if these values are placed into a complex E2E Network Slicing deployment context (i.e., joining computing and networking resources), the increment of time of 2s becomes much less

Table 5.3: CS deployment time vs. related Blockchain transactions time.

| | **Time (s)** | | | |
|---|---|---|---|---|
| | **Networking Deployment** | | | **Blockchain** |
| | **CS 1** | **CS 2** | **Total** | **Transactions** |
| Mean Value | 2.01 | 2.07 | 4.08 | 2.34 |
| Std. Dev. | 0.11 | 0.22 | 0.19 | 0.49 |

important because computing deployments need much more time. Moreover, compared to possible SDN situations such as a reconfiguration of optical amplifiers that may take minutes, it becomes less significant. Based on this, the trade-off to implement this new architecture might be an increment of seconds per each CS creation to keep the Blockchain advantages.

**Blockchain-based E2E network slice validation**

Finally, we joined all the previous elements in order to evaluate the two Blockchain-based modules (i.e., PDL-Slicing/Transport Managers) and, using the CTTC ADRENALINE testbed as previously presented, a complete E2E network slice was deployed. In this final use case phase, as the testbed was shared with other experiments using the Core domain, the E2E network slice resources were reduced. The Edge-DC2 kept the same NS (i.e., 2 VNFs with 3 VMs) and the Core-DC had two NSs (i.e., 4 VNFs with 6 VMs). As done in the previous experimental tests, before any deployment, the required NSTs information and the SDN controller (optical and packet) transport contexts were distributed to all the Blockchain peers. Then, it was possible to request E2E network slices from any computing domain.

Based on a new set of tests, the obtained results are illustrated in Figure 5.20. Checking the different columns, it is quite fast to validate that the Blockchain actions influence on the total E2E deployment time is very low. Comparing the mean time values between the total Blockchain actions (i.e., 4.51s) and the total E2E deployment actions (i.e., 202.28s), the Blockchain time means a 2.23%. The reason is that, as it was already demonstrated, the computing resource take a long time to deploy. This can be seen by checking the first two columns which show deployment times of 151.13s and 191.6s for the edge and cloud resources respectively. Instead, the time to stitch the

computing resources through the two unidirectional CSs was of 4.08s, similar to the Blockchain actions time (4.51s).

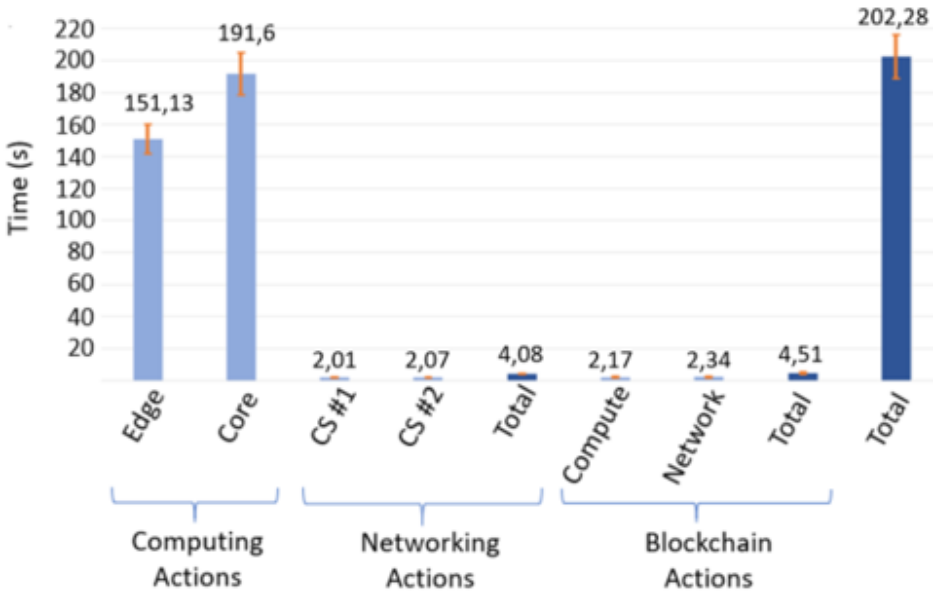


Figure 5.20: Deployment time values for each deployment action.

Moreover, another result illustrated in Figure 5.21 is the CDF for the total E2E deployment time. Based on these results, it is possible to conclude that long time E2E deployments (219.2s) have higher probabilities (89.3%) to occur. Finally, comparing the lowest time obtained (179s) with the Blockchain actions mean time (4.51s), the increment of time is a 2.55%, and so, the percentage will reduce for longer E2E deployments.

With this last set of results, the most essential elements composing an E2E network slice are deployed and studied using the new Blockchain-based NFV/SDN architecture. The main conclusion is that in terms of deployment time, the use of Blockchain does not add an important time increment while the benefits (i.e., transparency, security, etc.) brought by the Blockchain technology may be used.

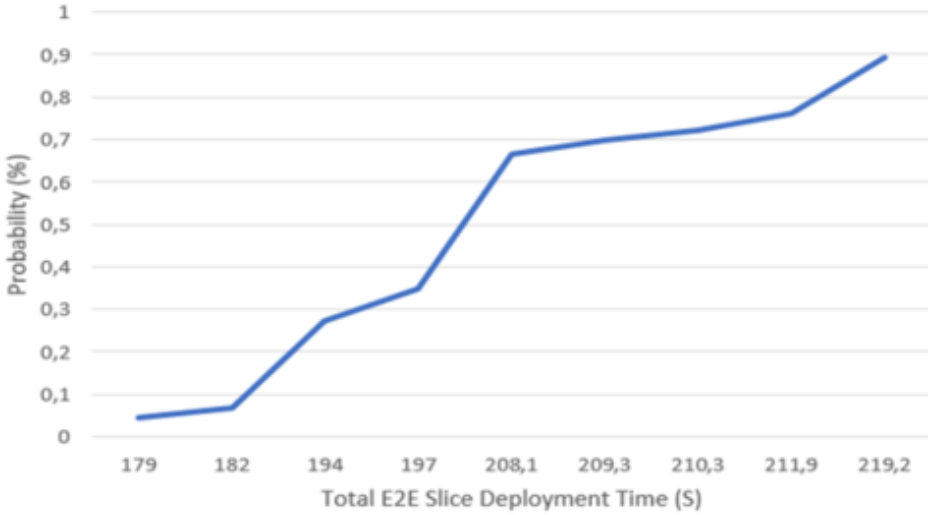


Figure 5.21: Total deployment time CDF.

### 5.3.3 Blockchain on multiple abstracted transport domains

In all the experimental data described until this point, the interconnection between computing resources was done using a CS crossing a single transport domain with known connection points. This is not quite close to the real world as the common scenario in terms of transport connectivity services is to cross more than a single transport domain. Using the elements described in Section 5.2, an experimental use case with four transport domains is studied to validate the Blockchain-based transport SDN architecture (i.e., the SDN Controller with the PDL-Transport Manager module) to manage the collaborative creation of an E2E CS composed of multiple domain CSs. Due to the fact that Blockchain has some limitations in terms of the data processing speed and because a single domain may have a massive amount of data, a study of multiple optical domains with the previously described abstraction models (Subsection 5.2.1) were compared to study the composition of E2E CS across multiple transport domains. This results were presented in [141].

### Use case description

An E2E network use case with four optical domains was designed and each one of them is individually managed by its own SDN Controller. Each optical domain had a different number of nodes, internal context and topologies (Subsection 5.2.1). Figure 5.22 shows the complete E2E network topology after each of the three studied abstraction models (Subsection5.2.3) was applied. On the top right, there is the E2E transparent abstracted topology which is exactly as the originally defined E2E topology (top left). On the bottom left, the VLink E2E abstracted topology and, finally, on the bottom right, the VNode E2E abstracted topology.

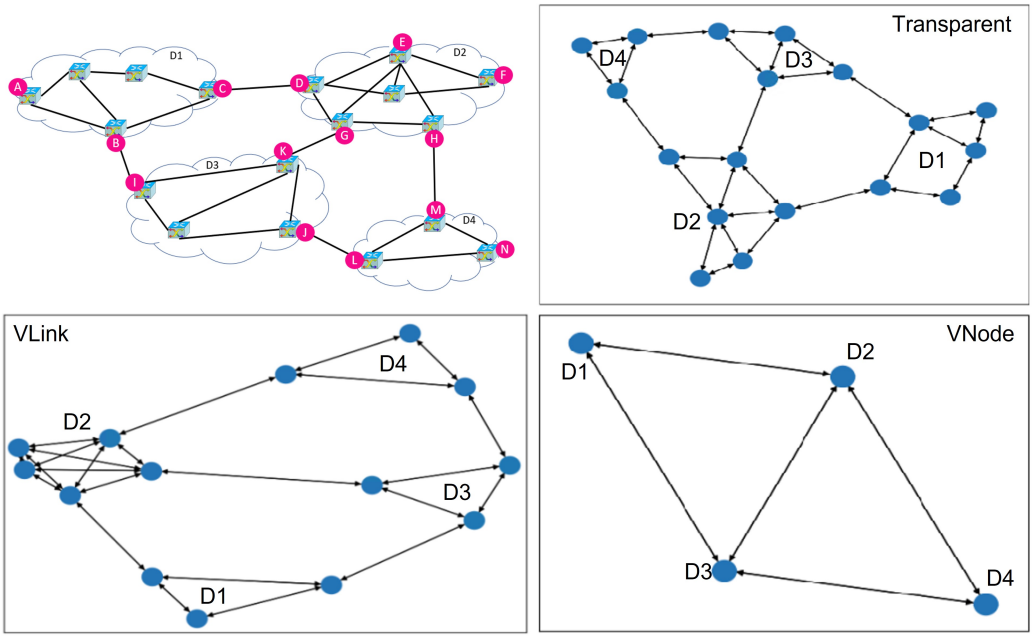


Figure 5.22: Original and abstracted network topologies graphs.

The interesting point of our use case is that, as previously explained, each PDL-Transport Manager has a vision of the whole E2E topology but it cannot directly request Domain CSs except to its local SDN Controller below. Regarding the Domain CS in other domains, it must distribute them to all the Blockchain peers and the right domain owner will take the newly arrived event and create the desired Domain CS. The only common network resources in the three abstraction cases are the IDLs (i.e., Domain 1 to Domain 2,

Domain 1 to Domain 3, etc.) among the optical domains.

Finally, using the four transport domain network designed, the different abstraction topologies are compared by validating the delay time values and the costs associated to the deployment and termination of E2E CSs.

### Testbed architecture

The use case presented was implemented using a set of four emulated optical transport domains as illustrated in Figure 5.23, each with its PDL-Transport manager on top, as introduced in Subsection 5.3.3.

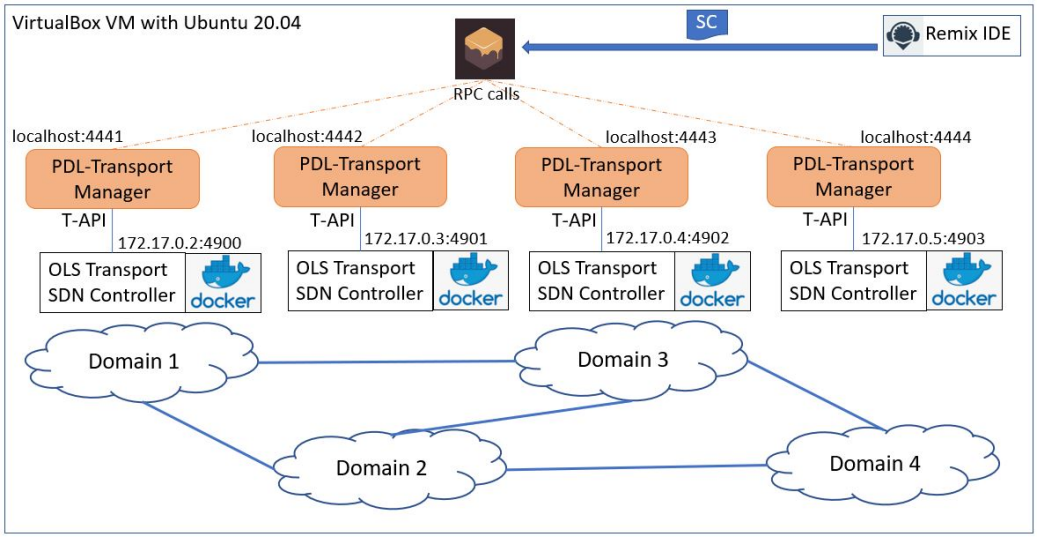


Figure 5.23: Testbed architecture.

At the top of everything there is a Blockchain network created using the Ganache [144] emulator. Once the Blockchain network is emulated, four instances (i.e., one per transport domain) of the PDL-Transport Manager were deployed and associated to one of the Blockchain peers created by the Ganache. Below each PDL-Transport Manager, the corresponding optical SDN Transport controller was also deployed with an associated transport domain context.

With the complete Blockchain and SDN optical domains environment ready, the next step was the creation and deployment of the SC. The SC was developed and deployed into the Blockchain using the Remix Integrated

Development Environment (IDE) which is one of the sub-projects composing the Remix Project [147]. Remix IDE is an open source web and desktop application that makes the design and development process of SCs faster and lighter than other similar options. REMIX IDE was used to write the previously presented SC using the Solidity language (e.g., a high-level and object-oriented language). One of the most powerful aspects from Remix IDE is the simplicity of deploying the written SCs to the Blockchain system. In order to communicate the PDL-Transport Manager with the Blockchain, Remote Procedure Calls (RPC) were done using the python web3 library [148] which is specially dedicated to interact with Ethereum-based Blockchains.

Finally, each optical domain was simulated using an SDN-controlled disaggregated optical network simulator [149] that allowed to define and create the context and topology for each optical transport domain with an SDN-Controller on top ready to receive requests to either manage (Domain) CSs and to retrieve the SDN domain context using the T-API data model. Moreover, it was deployed using docker which ensures the isolation of the optical domains as if they would be physically deployed in different geographical locations.

### Experimental results

Based on the multiple tests done using the different abstraction models applied on the designed network illustrated in Figure 5.22, a set of results are described to compare the abstraction models performance among them in terms of the time delay and the costs of managing their associated information volume in the Blockchain system. In the following column graphs, each column represents one of the three abstraction models; black for the transparent, gray for the VLink and white for the VNode.

Before discussing the results and in order to understand them better, it is important to be aware of the units used to study the costs results. Due to the fact of using an Ethereum-based Blockchain, each transaction generated requires an element called "Gas" to be processed. Gas [150] is referred to as the unit to measure the computational cost of a transaction in the Blockchain system. This cost depends on the amount of data within the transaction and the processes done with the SC action applied to the transacted data. So, the higher the amount of information and the number of code actions to be applied, the higher the cost will be.

- Time Delay Comparison

Regarding the results associated to the E2E CS deployment time in Figure 5.24. The first aspect to take into consideration is the high values of the total time needed in all three abstraction models. Compared with the values presented in [139], the main reason for the increment in the deployment time is due to the times values of the second, third and fourth columns in Figure 5.24 which belong to: a) the time to get the IDLs information from the Blockchain and compose the data object, b) the time to compose the SDN Context data object and check if the selected resources are available and, finally, c) the time to update the selected resources in the Blockchain. Checking the worst case (i.e., transparent model - blue columns), all these four time values give a total value of 384.405s which represents a 94.23% of the overall E2E CS deployment time, leaving 22.521s for the E2E CS data object creation and update and the Domain CS deployment composing the E2E CS.

A similar behaviour is illustrated in Figure 5.25 when terminating an E2E CS. The high values are due to the IDL composition and the update of the used (now free) resources in the Blockchain. In the E2E CS terminate case, there is no "SDN Composition" column as it was only necessary to update the specific SIPs and NEPs elements in the Blockchain, not like in the E2E CS deployment case where it was necessary to have all the SDN resources composed to find those that could ensure the spectrum continuity. When comparing the transparent abstraction model to deploy an E2E CS with the other two models, a time difference of 75.75% (Figure 5.24) is obtained compared with the VNode and a 38.42% compared with the VLink model. In the case of the termination process, the difference becomes even bigger with a 129.61% (Figure 5.25) difference between the transparent and the VNode models and a 17.11% between the transparent and the VLink models.

- Costs Comparison

From the costs point of view, the total mean value costs to complete an E2E CS deployment and terminate actions are presented in Figures 5.26 and 5.27, respectively. Both figures show the five procedure steps that generate transactions in the Blockchain. These steps are : a) requesting a Domain CS deployment or termination, b) updating the Domain CS data object once the action is done, c) updating the spectrum in the internal NEPs used, d) updating the spectrum in the SIPs used within the corresponding optical domain context and, finally, e) updating the IDLs information regarding the

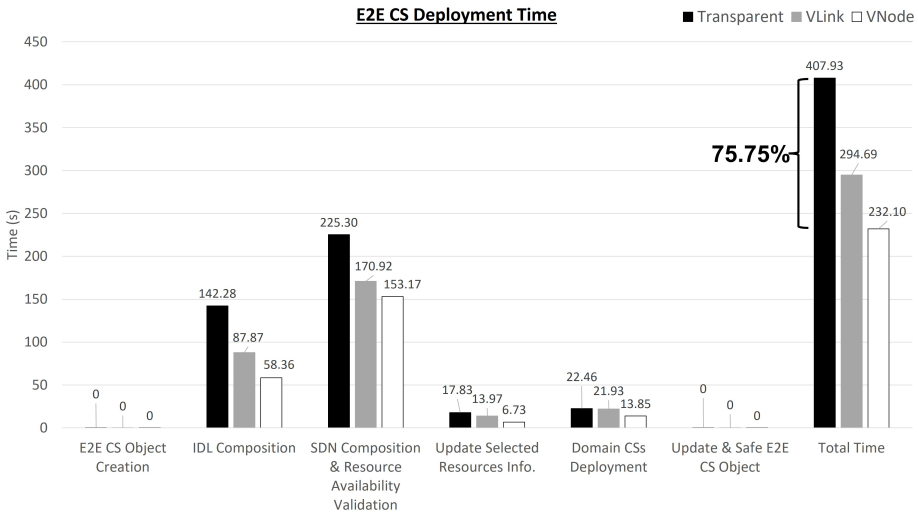


Figure 5.24: E2E CS deployment time delay.

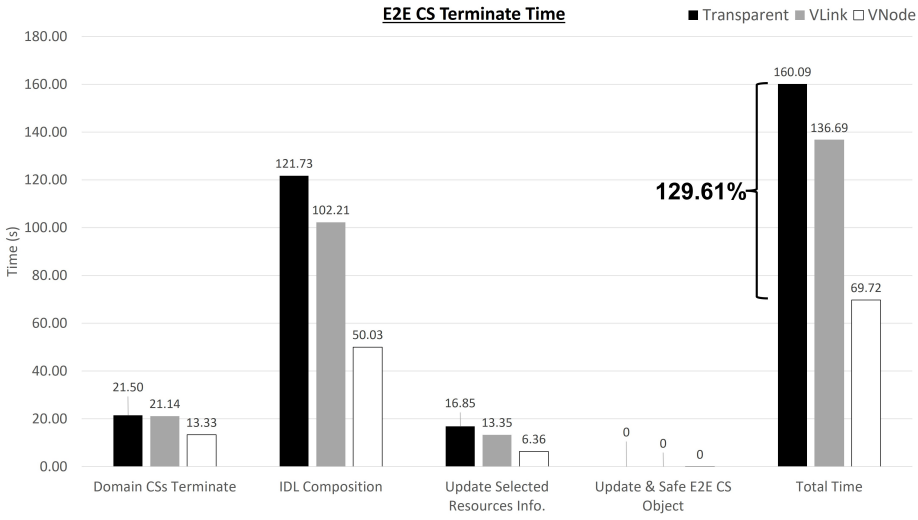


Figure 5.25: E2E CS terminate time delay.

SIPs used. As illustrated in both figures, the step with the highest cost is the one associated to the SIPs update. This is due to the need of finding the specific element among all the SIP elements. But the main difference between the two procedures is the distribution cost of the Domain CS deployment and termination requests. While the Domain CS deployment request (first

column in Figure 5.26) generates a transaction with multiple information parameters (i.e., uuid, SIPs, spectrum, etc.), the transaction generated for the Domain CS termination request (first column in Figure 5.27) only needs the Domain CS uuid.

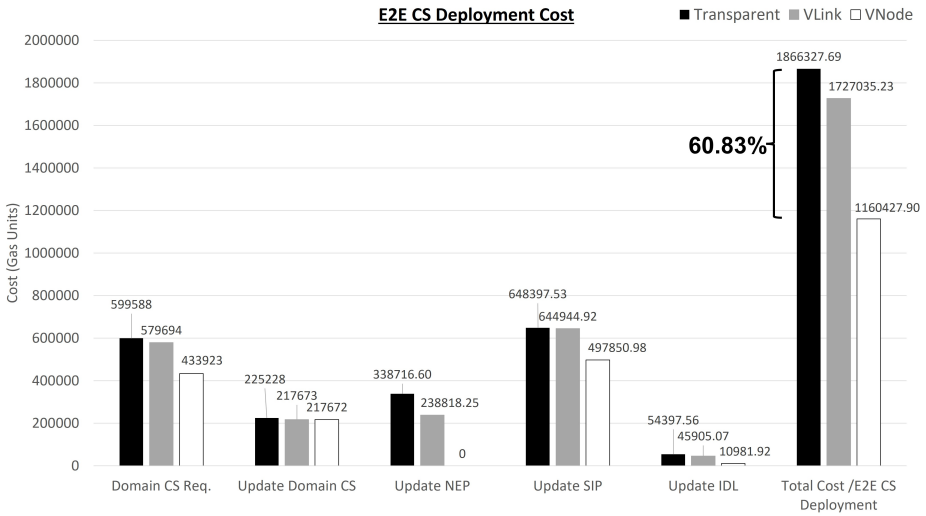


Figure 5.26: E2E CS deployment cost.

In both procedures, the results present the behaviour expected with the transparent model being more expensive than the other two models. For example, comparing the transparent abstraction model to deploy an E2E CS with the other two models, a cost difference of a 60.83% (Figure 5.26) is obtained compared to the VNode and a 8.06% compared to the VLink model. In the case of the termination action, the difference is bigger with a 93.01% (Figure 5.27) difference between the transparent and the VNode models and an 11.04% between the transparent and the VLink models. In addition to the amount of information stored in the initial Domain CS requests, another aspect that influenced the total values presented is the non-usage of NEP resources when deploying/terminating E2E CS in the VNode model. As it can be seen in the third column of both figures, the cost is equal to 0.

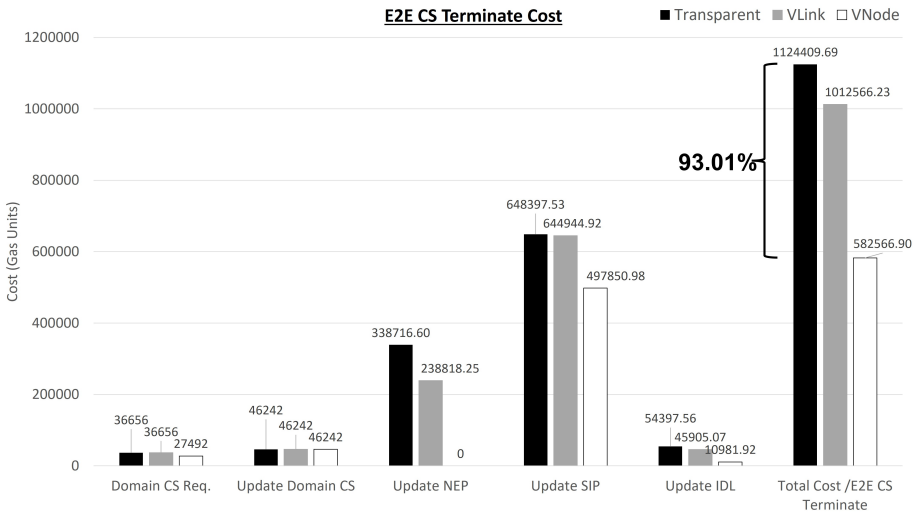


Figure 5.27: E2E CS terminate cost.

## 5.4 Conclusions

With the appearance of new technologies such as DLT, it is a must to see how they can be used to improve the design and performance of the current control and management architectures and infrastructures. As previously described, DLTs will not only be a new way to store and distribute the information but it will also become a key player as part of new mesh infrastructures with a cooperative way of working.

The results show that DLTs do not add an important time delay when the cooperative deployment procedure is applied to instantiate multi-domain network slices. Further works will be necessary to keep improving the results, specially to validate the (certain) evolutions that DLTs will get in the next years, specially in terms of dealing with higher amounts of information with faster processing delays. This will be very important to improve the work presented related to the abstraction network contexts, as with more details shared among peers, the control and management will become much more efficient.

# Chapter 6

# Control and orchestration of trusted resources for network Slices

## Contents

One of the most important evolved aspects around networks is the increment of roles and disassociation of functionalities and so, the definition of new players. This leads towards to what is called a multi-stakeholder environment with multiple providers, multiple-operators, multiple-domain owners, etc. Based on the work done in the previous chapter using Blockchain to distribute the Network Slicing Resources in different domains, one more step was thought to use Blockchain in order to manage trust among different service providers and customers. To do so, a Blockchain-based Trust Manager was designed to have for each provider type, a set of shared and tampered-proof values describing how trustworthy is each of the providers.

This chapter describes the main elements used on the Trust Manager design. First, we describe the process implemented to map a subjective concept like Trust towards an objective value associated to the risk of trusting each vendor called Trusted Risk (TR). Then, new Blockchain-based Trust Manger component is described followed with the actions to update and manage the trust among the multiple stakeholders. Finally, an experimental evaluation is presented.

The work done in this chapter is currently submitted and under review in [151].

## 6.1 Subjectivity towards objectivity

As previously defined, trust is a subjective feeling generated between two or more players. From the network services and resources management point of view, trust must be defined in an objective way, but the problem of Trust is that it cannot be measured by itself. As conceptually speaking, trust is based on the reputation that somebody or something is able to achieve a specific action and this can be measured (i.e., how many times a task is done?), reputation was selected to be the key parameter to generate the TR.

Once the TR is computed, it will define the risk of trusting one or another provider, so if a client desires to be served by a provider with a minimum TR of 80%, any provider with a higher value will be in the possible providers selection process. To do so, in the current approach of our Trust Manager, a set of high-level parameters was selected to compute the reputation values based on different actions. By doing so, the objective is to obtain a single mathematical value (i.e., a percentage) that allows any client to easily understand that the risk of trusting a provider goes from 0 (i.e., not trustworthy)

to 100 % (i.e., trustworthy) compared to the other providers.

Following a similar approach as in [101] or [102] where they compute Trust using Reputation, in this work the obtained TR value is computed using multiple "Reputation" parameters ($R_x$), each from a specific network management action. In the current work, the following three actions were selected to define the corresponding reputation parameters:

- Provisioning Rate: This value allows to check the percentage of requests being accepted and deployed. This parameter allows to identify how often a provider accepts or renounces to a request. While this action might not be as relevant as the following two, it is still important as it indicates whether the provider is ready to manage multiple services or not and so, if it has a correct and constant resources management plan.

- Non-forced Termination rate: This value checks how many accepted and deployed services were finished correctly and not forced because of a SLA violation that could not be fixed. By using this parameter, a client can verify the reputation of a provider to keep the deployed services and terminate them when expected and not before.

- SLA correction rate: This value validates how good each provider is in terms of reacting towards SLA violations. Despite planning problems prevention mechanism, it can still be expected that problems may rise while services are deployed. What is important is how good the provider is in terms of reaction, correction and solution in case of any unexpected problem or SLA violation.

### 6.1.1 Computing the reputation values to generate trust

Based on the previously selected concepts to generate trust, we designed the mathematical model to be used within the Trust Manager solution to compute the reputation values and, with them, the TR for each provider.

The computation of the TR related to the reputation parameters previously described is presented in Equation 6.1. The final result is the sum of the three reputation ($R_x$) values:

- The Provisioning Rate Reputation ($R_{prov\_rate}$) is obtained by dividing the total number of deployed services over the total number of requested services.

- The Non-forced Termination Rate Reputation ($R_{term\_rate}$) is obtained by dividing the total number of terminated services over the total number of deployed services.

- The SLA Correction Rate Reputation ($R_{correction\_rate}$) is obtained by dividing the total number of corrected SLA violation over the total number of SLA violations, allowing to check the reaction of each provider when having to correct any SLA violation.

Each $R_x$value in the TR has an associated weight parameter (i.e., $\alpha$, $\beta$ and $\gamma$) which is used to give to the model the possibility of giving more importance to one of the three reputations values (i.e., not all the systems are equal) and, also, to obtain a result with a percentage format for an easier and faster understanding. For these reasons, both $R_x$ and TR values, together with the three weight parameters, cannot be lower than 0 and higher than 1 ($0 \leq TR \leq 1$, $0 \leq R_x \leq 1$, $0 \leq \alpha/\beta/\gamma \leq 1$). Finally, to enforce that TR follows the previous condition, the three variables sum must be equal to 1 ($\alpha + \beta + \gamma = 1$).

$$\begin{aligned} TrustedRisk(TR) &= \alpha R_{prov\_rate} + \beta R_{term\_rate} + \gamma R_{correction\_rate} \\ &= \alpha \frac{Serv_{depl}}{Serv_{req}} + \beta \frac{Serv_{term}}{Serv_{depl}} + \gamma \frac{SLA_{corr}}{SLA_{viol}} \end{aligned} \tag{6.1}$$

Based on the three previous reputation values, the TR is computed by giving a single percentage value that a client will easily understand. To assist the providers selection when a request is generated, a set of five trust levels were defined within our design and illustrated in Table 6.1:

Table 6.1: Trust classification levels.

| **Trust Level** | $T_{min}$ | $T_{max}$ |
|---|---|---|
| Completely Untrustworthy | 0 % | 20 % |
| Untrustworthy | 20.01 % | 40 % |
| Weak Trustworthy | 40.01 % | 60 % |
| Trustworthy | 60.01 % | 80 % |
| Completely Trustworthy | 80.01 % | 100 % |

### 6.1.2 Trust SLA: a first approach

When requesting a service, the client needs to define which parameters are to be fulfilled in terms of trust that later will be used for the selection of the best provider possible. The concept of Trust SLA (TSLA) was implemented by evolving an SLA in terms of Trust requirements such as the reputation values related to the different aspects or the overall risk to accept.

Listing 6.1 shows the JSON used in our current Trust Manager solution to request a service based on a set of SLA requirements in terms of service performance (e.g., throughput or accepted requests) but also, and equally important, the TSLA requirements to select the best set of providers that fulfils the minimum values of reputation and trust.

Listing 6.1: Service request structure with TSLA requirements.

```
{
  "service-id": <uuid4>,
  "sla": {
    "accepted-requests": 500,
    "throughput": "100 Mbps"
  },
  "tsla": {
    "min-trust-score": 80,
    "min-rep-depl": 75,
    "min-rep-term": 85,
    "min-rep-sla": 80
  }
}
```

When the TSLA requirements within the service request are processed, any provider below those requirements will be excluded. Then, among the selected trustworthy providers, the Trust Manager will choose those with higher trust and reputation values which, at the same time, are also able to fulfil the performance requirements within the SLA.

### 6.1.3 Blockchain as the trust keeper

The computed reputation and trust values need to be distributed and known by all the stakeholders to have a collaborative management of service resources based on a transparent reputation and trust. To solve this aspect,

a private Blockchain system (i.e., PDL) was the selected solution. The Blockchain network will be the element to keep the record evolution of all the reputation values and the TRs associated to each provider because of its transparency strength and also, due to the capabilities brought when using SCs.

The SC designed for this Trust Manager solution have the following functionalities: a) to save and to distribute the service requests to the correct provider based on the TSLA requirements and the current reputation and TR values, b) to trigger the complete TR computation after a service is terminated (either forced or not) and, finally, c) to save and to distribute the updated reputation and TR values corresponding to the provider that gave the terminated service.

## 6.2 A Trust Manager

Within the following section, using the previous Figure 6.1 as reference, the Trust Manager architecture and the functionalities of its internal modules are introduced. Moreover, the two main processes managed by the Trust Manager are described.

### 6.2.1 Internal architecture

As Figure 6.1 illustrates, the Trust Manager is composed by three main modules: Trust SLA & Policies, Trust Controller, and PDL-Service Manager.

#### Trust SLA & policies

This module is in charge of two different but strongly linked elements: the management of SLAs and Policies. To do so, this module is composed of two components:

- TSLA Manager: It manages the TSLAs life-cycle defining the requirements requested by any customer to the provider. The TSLAs are defined by specific requirements focused on liability and trust associated to the services available for the users.
- Policy Manager: It manages the life-cycle of the available policies related to the services (i.e., deployment) and related to the TSLAs (i.e.,

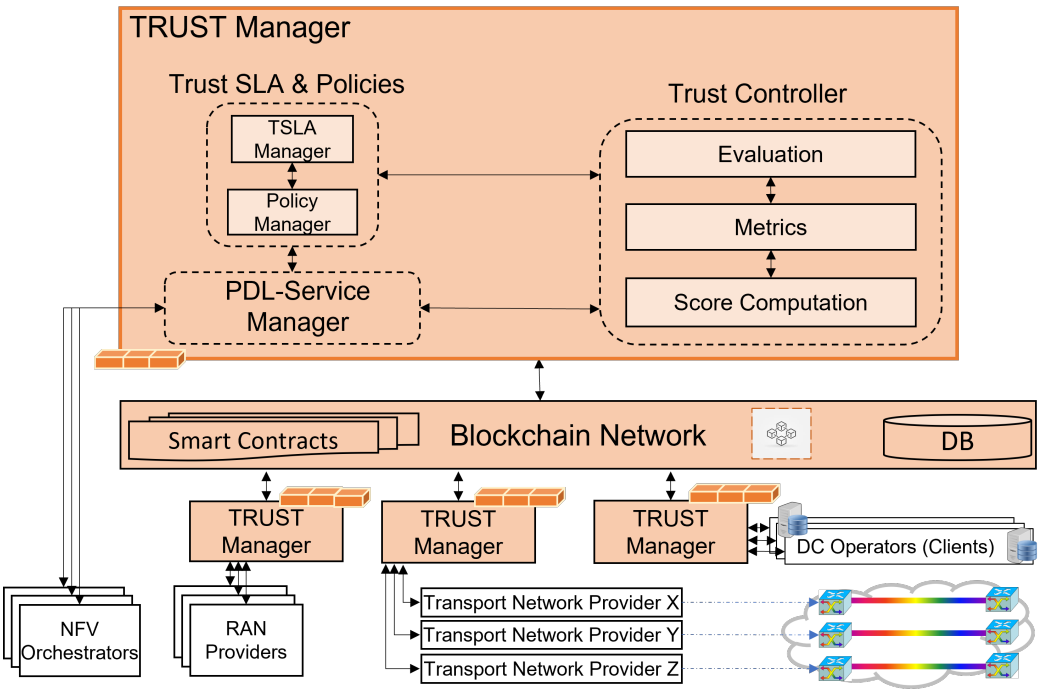


Figure 6.1: Blockchain-based system for trust and internal Trust Manger architecture.

violation resolution) to ensure that the commitment between client and provider is fulfilled.

### Trust Controller

The Trust Controller manages the process to compute the multiple reputation values together with the TR associated to each provider. After each deployed service is terminated, it obtains the set of metrics/parameters previously defined in Section 6.1. The data to compute these values may come from multiple sources such as the information gathered during the monitoring phase while the service is active and the historical data already available in the Blockchain. This module is composed of three components:

- Evaluation: It gathers and evaluates which data is going to be used based on the user experience, the data generated by the providers and other possible sources.

- Metrics: It makes use of data models and historical logs from the different providers to generate the reference data to pass to the Score Computation component.

- Score Computation: It processes the data coming from the previous two components to obtain the updated and latest reputation values and TR for those providers involved in the terminated services. By doing this action, the next TSLA will be correctly applied by choosing the most suitable policy based on the updated values. Finally, this component is in charge of generating the correct data object to be distributed in the Blockchain through the PDL-Service Manager.

#### PDL-Service Manager

PDL-Service Manager handles the collaborative deployment through the Blockchain of any service requested to the Trust SLA & Policies module. Moreover, it is the gateway for the previous two modules to access to the collaborative Blockchain-based system, specifically for the Trust Controller when this needs to distribute the updated reputation and trust values. Finally, this module also has the role of gateway that allows the Trust Manager to interact with the set of available providers and requests the required service based on the selection done using the client TSLA requirements.

### 6.2.2 Service selection and deployment based on trust requirements

The first of the two main actions that the Trust Managers do is the selection of the most trusted provider to deploy the required service based on the TSLA requirements.

This process is described in Figure 6.2 and it begins (step 1) when a client requests a service with a set of Trust requirements (i.e., KPIs) defined in a TSLA. The Trust SLA & Policies module takes care of selecting the best TSLA that fulfils the requirements and forwards a policy to deploy a service with the TSLA information to the PDL-Service Manager module (step 2). Then, based on the trust requirements and the reputation values of each provider, the PDL-Service Manager decides whether the service can be deployed by a local provider or by another Trust Manager domain (step 3). Once decided, if the provider is local, the request is forwarded to the right

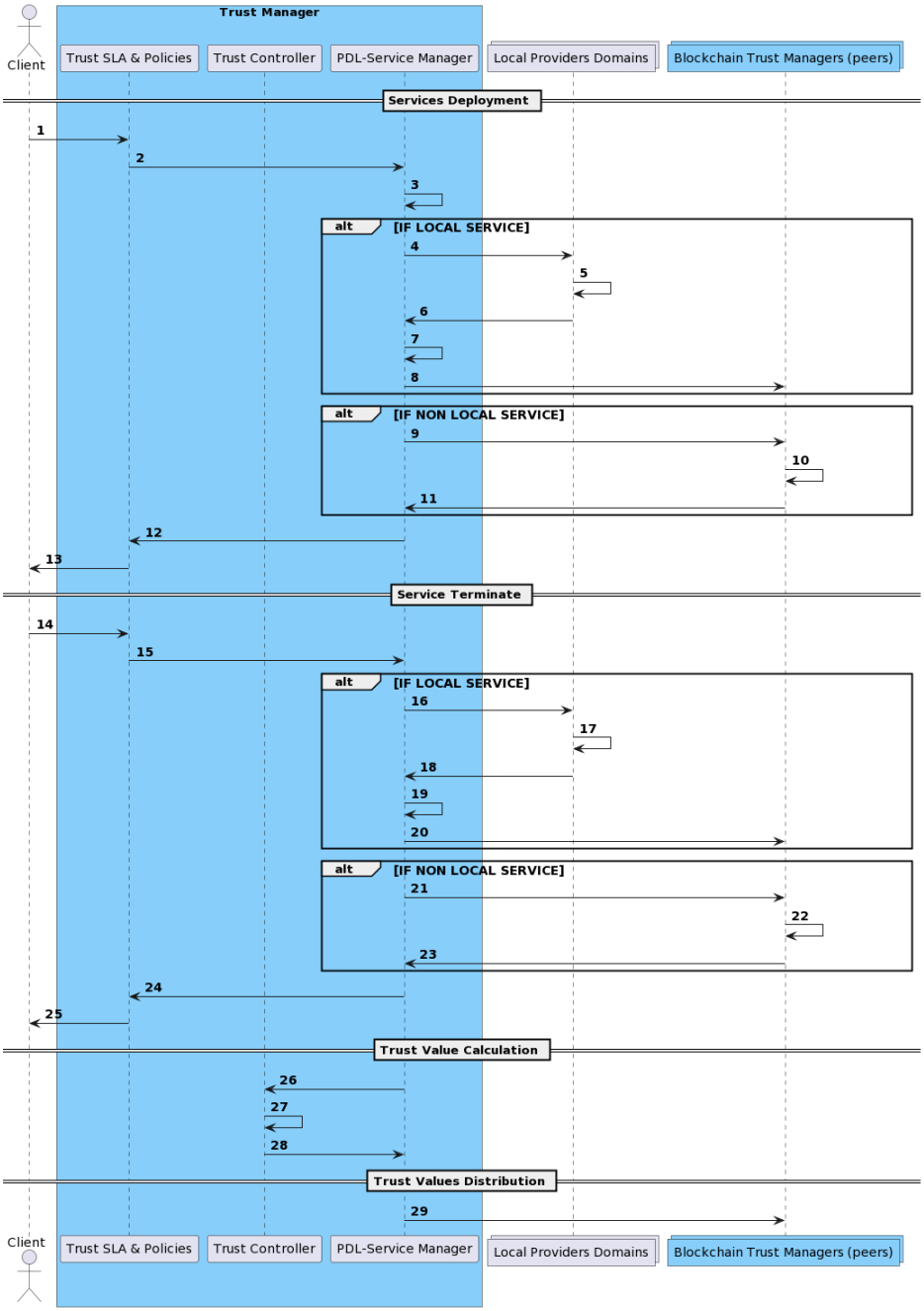


Figure 6.2: Service life cycle and trust value update workflows.

Local Providers Domains (step 4) which deploys it (step 5) and informs back (step 6). Once the PDL-Service Manager receives the confirmation about the deployed service, it updates its local information (step 7) and distributes it in the Blockchain (step 8). If the provider selected in step 3 is not local, the PDL-Service Manager distributes the request among the Blockchain peers (step 9), the one with the provider takes it and follows the same process from steps 4 to 8 (summarised in step 10). When the service is deployed, the Trust Manager peer that took the distributed service request distributes the updated information in the Blockchain (step 11) and only the original Trust Manager takes it back. Finally, this reports back to the client (step 13) through its PDL-Service Manager and Trust SLA & Policies modules (step 12).

### 6.2.3 Trust update

The Trust Managers second main action is the updating process of the reputation values and the TR to improve the providers selection of future similar service requests once a deployed service is terminated.

This process is described in Figure 6.2 and it begins when a service is requested to be terminated (step 14). The Trust SLA & Policies Module sends the termination policy order to the PDL-Service Manager (step 15), so this second module verifies whether the service is deployed locally or in another Trust Manager domain. If the deployed service is local, a request to terminate the service is sent to the corresponding Local Provider Domain (step 16) which terminates it (step 17). Once terminated, the Local Provider Domain reports about it together with the parameters required to calculate the reputation values and compute the TR (step 18). With the incoming information, the PDL-Service Manager updates the local service deployment data object (step 19) and distributes (and stores) this information to the Blockchain (step 20). Once the information is distributed, the PDL-Service Manager informs the client about it through the Trust SLA & Policies module (steps 24/25). If, on the other hand, the service deployed belongs to another Trust Manager domain, the terminate request is distributed in the Blockchain (step 21) and it is only taken by the right Trust Manager which follows the process described in steps 16 - 20 (resumed in step 22). Once terminated, the trust statistics related to the provider that managed the terminated service are distributed in the Blockchain and an event is taken only by the right Trust Manager (step 23) which then triggers two parallel actions: informing

the client (steps 24/25) and updating the trust and reputation values (steps 26 - 29).

The "Trust Value Calculation and Distribution" process (steps 26 - 29) begins once the service is terminated and its information is in the Blockchain. Through the use of the SC, the process to update the TR and reputation values of the provider that managed the terminated service is triggered when an event for the Blockchain peers arrives (step 23). With the information received in the event, the PDL-Service Manager forwards the required information to the Trust Controller module (step 26) and this processes the data and computes the updated values (step 27). Then it sends them back to the PDL-Service Manager (step 28). Finally, the PDL-Service Manager distributes the updated information among the Blockchain peers to have the latest TR and reputation values for any new service deployment request (step 29).

Finally, after the network has started working and the first service request is generated, there is one specific situation to take into account from a fairness point of view. To decide with which TR and $R_x$values should each new provider start joining the network in comparison to the other providers. At time 0, all providers begin with an equal $R_x$and TR values of a 100%, but once the first service request is triggered, any new provider could corrupt the system as it would enter with $R_x$and TR equal o 100% while the other have may have it below. To solve this possible situation, any new provider joining once the system started working should start with a $R_x$and TR values equal to the mean value obtained from all the providers of the same type (e.g., Transport Network Providers, RAN Providers, NFV-Os in Fig. 6.1).

## 6.3 Experimental validation

After the presentation of the mathematical model and the designed solutions, this section describes the use case designed and implemented to validate the previously described solution. Moreover, an initial set of experimental results are presented.

### Use case description

As presented in the introduction, the described Trust Manager architecture and its elements aim to become a solution towards the next generation of

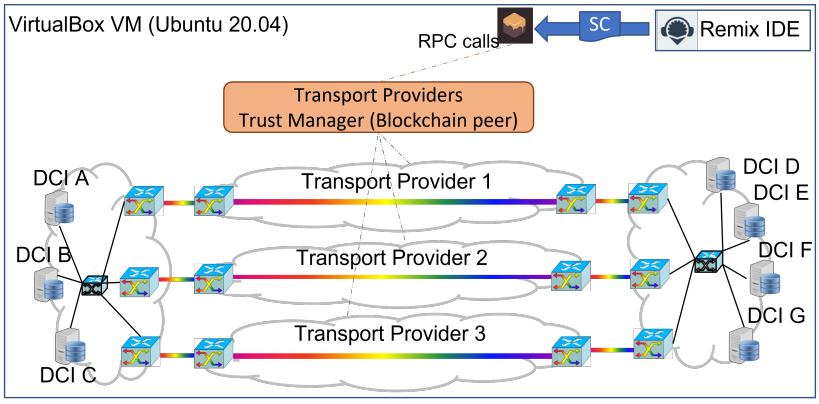


Figure 6.3: use case: DCI transport providers selection based on trust.

networks in which multiple and different infrastructure and service providers may offer their resources. Based on this, a DCI with multiple transport providers use case is illustrated in Figure 6.3. Two computing domains with a group of DCs each are interconnected through a set of three different transport connectivity services providers. Each DC is able to request an optical CS to the available transport providers specifying the trust-related requirements in the TSLA associated to the service request.

### Testbed architecture

Regarding the experimental environment, as illustrated in Figure 6.3, a VM with an Ubuntu 20.04 LTS was used. In there, a Trust Manager with the three providers information on a Blockchain. Following the testbed environment described in Subsection 5.3.3 within the previous chapter, the Blockchain was created using Ethereum-based emulator Ganache [144] due to its advantages. Moreover, the Remix IDE [147] was used to develop and deploy the SC into the Blockchain and once the SC was in the Blockchain, the Trust Manager was linked to it by using the python web3 library [148].

The executed experimental tests were planned to analyse the trust and reputation parameters evolution of each transport provider for a certain amount of time. The experimental test requested a set of 100 service requests based on a 1 Erlang Poisson distribution. A very important element to take into account to consider the outcomes as realistically as possible was the fact of having a minimum amount of failed services. Due to the fact that providers and operators do not present the failures in their generated

data, it was considered to perform the experimental task with a reliability of a 99% of received service requests fulfillment. By doing so, there is a small probability for each request to be considered as a failed request, affecting the final trust and reputation values of each provider. The developed test used the three $R_X$ parameters previously described, this mean the accepted requests, the SLA resolution and the non-forced terminated services. Each of these three elements had a weight value associated when computing the final TR. The weight values of each reputation parameter were of 0.33 for the accepted requests, 0.33 for the SLA resolution and 0.34 for the non-forced terminated services.

### Experimental results

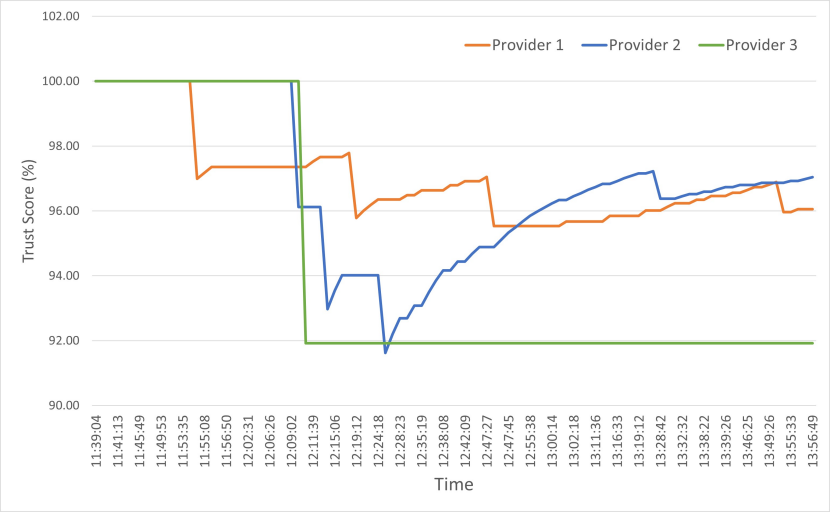


Figure 6.4: Trusted Risk ranking evolution in time.

Based on the tests done using the use case, the first round of results are presented in Figure 6.4. In there, it is possible to see how the TR of each provider evolves during the test time. The TR is updated based on the received requests and their final status. As shown in Figure 6.4, all the providers begin with a TR of 100 and, once a service is done, the TR is updated based on its statistics. As clearly seen in Figure 6.4 and based on the model described in Section 6.1.1, when a service is not properly accomplished, which means that an SLA violation was not solved and the service

was inevitably terminated, the provider TR decreases. This is the expected behaviour because service providers should be fully prepared to resolve an SLA violation before it is forced to terminate a service prior to its request. Despite failing means receiving a penalty on the TR, it does not imply that the provider is out of the game. In fact, when checking the evolution of providers 1 and 2 in Figure 6.4, it can be seen how their TR increases while services were requested. A special case was provider 3, which received only 5 requests during the whole test and forced a terminated service in the fifth request, leaving its TR at 91.92% while the others tend to increase it.

As it can be seen on the left side of Figure 6.4, the initial TR values of all providers were at 100%. After the test was done, as Figure 6.5 presents, the TR and its associated reputation values changed. Figure 6.5 shows the final values of the three providers (provider 1 in black, provider 2 in gray and provider 3 in white) and how each reputation value affects the TR value. The first column set shows the TR with provider 2 having the maximum value (97.04%) over the other two (96.06% provider 3 and 91.92% provider 2). The second column set shows the reputation value associated with the reception and deployment of a service request, with all providers having a 100%. These perfect values mean that the providers always had available resources. However, the other two reputation values decreased. The third column shows the Terminate Reputation and the fourth one shows the reputation on solving SLA violations.

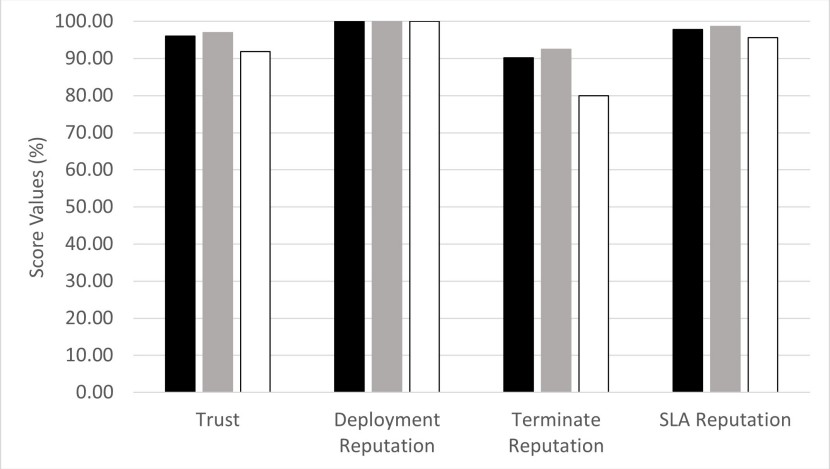


Figure 6.5: Final Trusted Risk and reputation parameters after test.

Finally, Figure 6.6 shows the TR evolution dependency with the three reputation values during the test time obtained for the provider 2. This figure shows how the fact of terminating a deployed service in a forced way has a very important effect on the TR, as the valleys in time 12:12:41, 12:26:38 and 12:28:42 show. Despite these strict penalties when failing, the model gives the option to improve and recover the TR of a provider in a long term.

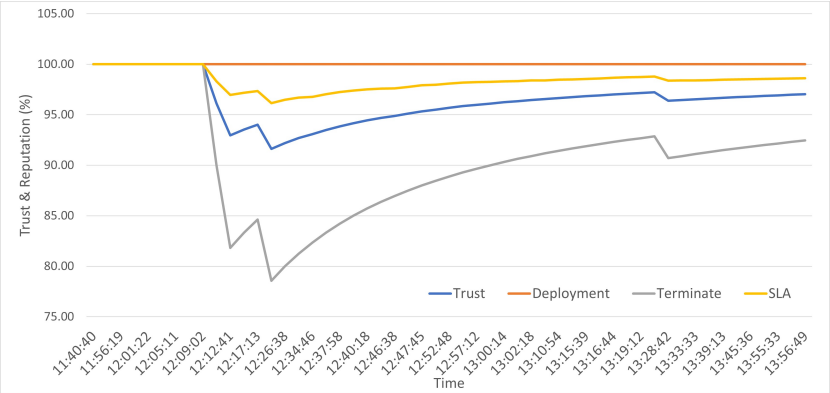


Figure 6.6: Final Trusted Risk and reputation parameters evolution in time.

The next generation of networks are expected to be an open scenario in which multiple domains, providers and even operators (i.e., multi-stakeholder scenario) should interact among them avoiding the strict dependency that there is in the current infrastructures. Having so many options means an increment of relationships between clients and providers created in run-time. For this reason, it is required to have a system to inform about how trustworthy a provider is and that this systems is widely accepted by all providers.

## 6.4 Conclusions

Trust is a crucial aspect that needs to be taken into account in any system where different and multiple players with their objectives coexist. If a provider and its resources can be trusted on what they are expected to do, customers will feel their needs are properly saved even before the required service is deployed and may be used. For this reason, computing and managing trust is important and needs to be properly studied.

Trusted third parties still have a long life, but the work done in this chapter showed a distributed model in which these players (i.e., the third

trusted parties) should become less and less necessary as DLTs may be the solution to turn a mistrusted relationship between two players into a trusted one.

# Chapter 7

# Dissemination and exploitation results

## Contents

This chapter presents the research and scientific articles published, accepted or submitted together with the research project in which the student has participated related to the work done in this PhD thesis. The chapter is divided in two sections, Section 7.1 presents the related publications classified in journals, leaded international conference papers and finally, collaborated articles. Secondly, Section 7.2 presents the international and national projects in which the student has participated with his studies.

## 7.1 Related publications

### 7.1.1 Journals

1. C. Manso, ***P. Alemany***, R. Vilalta, R. Muñoz, R. Casellas, R. Martínez, End-to-end SDN/NFV orchestration of multi-domain transport networks and distributed computing infrastructure for beyond-5G services , **IEICE Transactions on Communications**, Vol. E104-B, No. 3, March 2021.

2. ***P. Alemany***, A. Román, R. Vilalta, A. Pol, J. Bonnet, E. Kapassa, M. Touloupou, D. Kyriazis, P. Karkazis, P. Trakadas, J. Martrat, R. Casellas, R. Martínez, R. Muñoz, A KPI-Enabled NFV MANO Architecture for Network Slicing with QoS , **IEEE Communications Magazine**, Vol. 59, No. 7, pp. 44-50, July 2021. DOI: 10.1109/M-COM.001.2001077

3. ***P. Alemany***, R. Vilalta, R. Muñoz, R. Casellas, R. Martínez, Evaluation of the abstraction of optical topology models in blockchain-based data center interconnection, **Journal of Optical Communications and Networking**, vol. 14, pp. 211-221, April 2022.

4. ***P. Alemany***, A. Molina, C. Dangerville, R. Asensio, D. Ayed, R. Muñoz, R. Casellas, R. Martínez, A. Skarmeta, R. Vilalta, Management and Enforcement of Secured E2E Network Slices across Transport Domains, **Optical Fiber Technology**, Vol. 73 (103010), October 2022.

5. ***P. Alemany***, R. Muñoz, J. Martrat, A. Pastor, R. Díaz, D. Lopez, R. Martínez, R. Casellas and R. Vilalta, Trust Management through

Blockchain for Optical Data-Center Interconnection Providers, submitted in the Journal of Optical Communications and Networking (JOCN), January 2023.

### 7.1.2 International conferences

1. ***P. Alemany***, J. de la Cruz, R. Vilalta, R. Casellas, R. Martínez, R. Muñoz, A. Pol, A. Roman, Comparison of Real-Time Communications Service KPI in Edge and Cloud Domains Through Multi-Domain Transport Networks , in Proceedings of the 2020 International Conference on Optical Network Design and Modeling (ONDM), 18-21 May 2020, Virtual Event.

2. ***P. Alemany***, R. Vilalta, F. Vicens, I. Dominguez Gómez, R. Casellas, R. Martínez, S. Castro, J. Martrat, R. Muñoz, Hybrid Network Slicing: Composing Network Slices based on VNFs, CNFs Network Services , in Proceedings of the 2020 IEEE Conference on Network Softwarization (IEEE NetSoft), 29 June-3 July 2020, virtual event.

3. ***P. Alemany***, D. Ayed, R. Vilalta, R. Muñoz, P. Bisson, R. Casellas, R. Martínez, Transport Network Slices with Security Service Level Agreements , in Proceedings of the 22nd International Conference on Transparent Optical Networks (ICTON 2020), 19-23 July 2020, Virtual Event.

4. ***P. Alemany***, R. Vilalta, R. Muñoz, R. Casellas, R. Martínez, Peer-to-Peer Blockchain-based NFV Service Platform for End-to-End Network Slice Orchestration Across Multiple NFVI Domains , in Proceedings of the 2020 IEEE 3rd 5G World Forum (5GWF'20), 10-12 September 2020, virtual event.

5. R. Vilalta, ***P. Alemany***, R. Sedar, C. Kalalas, F. Vázquez-Gallego, R. Casellas, R. Martínez, J. Ortiz, A. Skarmeta, J. Alonso-Zarate, R. Muñoz, Applying Security Service Level Agreements in V2X Network Slices , in Proceedings of IEEE Conference on Network Function Virtualization and Software Defined Networks, 10-12 November 2020, virtual event.

6. ***P. Alemany***, R. Vilalta, R. Muñoz, R. Martínez, R. Casellas, Managing Network Slicing Resources Using Blockchain in a Multi-Domain

Software Defined Optical Network Scenario , in Proceedings of European Conference on Optical Communications (ECOC 2020), 6-10 December 2020, virtual event.

7. ***P. Alemany***, R. Vilalta, R. Muñoz, R. Casellas, R. Martínez, End-to-End Network Slice Stitching using Blockchain-based Peer-to-Peer Network Slice Managers and Transport SDN Controllers , in Proceedings of The Optical Networking and Communication Conference & Exhibition (OFC), 6-11 June 2021, virtual event.

8. ***P. Alemany***, R. Vilalta, R. Muñoz, R. Casellas, R. Martínez, Blockchain-Based Connectivity Provisioning in Multiple Transport SDN Domains , in Proceedings of the 25th International Conference on Optical Network Design and Modelling (ONDM), 28 June-1 July 2021, virtual event.

9. L. Lossi, ***P. Alemany***, J. F. Hidalgo, F. Moscatelli, R. Vilalta, R.Munoz, R. Sedar and M. Catalan-Cid, 5GCroCo Barcelona Trial Site Results: Orchestration KPIs Measurements and Evaluation, in Proceedings of 2022 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit), 7-10 June 2022, Grenoble, France.

10. R. Vilalta, ***P. Alemany***, LL. Gifre, R. Casellas, R. Martínez and R. Muñoz, End-to-End Inter-domain Transport Network Slice Management Using DLT-enabled Cloud-based SDN Controllers, accepted in The Optical Networking and Communication Conference & Exhibition (OFC), 7-9 March 2023, San Diego (California, USA).

### 7.1.3 Collaborations

1. R. Muñoz, F. Vázquez-Gallego, R. Casellas, R. Vilalta, R. Sedar, ***P. Alemany***, R. Martínez, J. Alonso-Zarate, A. Papageorgiou, Miguel Catalan-Cid, F. Moscatelli, Giada Landi, X. Vilajosana, Andrea Bartoli, Denis Guilhot, S. Kanti Datta, Jerome Harri, R. Silva, Laurent Dizambourg, Antonio Fernandez, M. Muehleisen, 5GCroCo Barcelona Trial Site for Cross-border Anticipated Cooperative Collision Avoidance , in Proceedings of European Conference on Networks and Communications (EUCNC), 16-17 June 2020, virtual event.

2. J. Ortiz, R. Sanchez-Iborra, J. Bernal, A. Skarmeta, C. Benzaid, T. Taleb, ***P. Alemany***, R. Muñoz, R. Vilalta, C. Gaber, J. Wary, D.

Ayed, P. Bisson, M. Christopoulou, G. Xilouris, E. Montes de oca, G. Gür, G. Santinelli, V. Lefebvre, A. Pastor, D. Lopez, INSPIRE-5Gplus: Intelligent Security and Pervasive Trust for 5G and Beyond Networks , in Proceedings of the 15th International Conference on Availability, Reliability and Security (ARES), 25-28 August 2020, virtual event.

3. A. Alcalá, S. Barguil, V. López, L. M. Contreras, C. Manso, ***P. Alemany***, R. Casellas, R. Martínez, D. Gonzalez-Perez, X. Liu, J.M. Pulido, J.P. Fernandez-Palacios, R. Muñoz, R. Vilalta, Multi-layer Transport Network Slicing with Hard and Soft Isolation , in Proceedings of The Optical Networking and Communication Conference & Exhibition (OFC), 6-11 June 2021, virtual event.

4. S. Barguil, V. López, L.M. Contreras-Murillo, O. González-de-Dios, A. Alcalá, C. Manso, ***P. Alemany***, R. Casellas, R. Martínez, D. González-Pérez, X. Liu, J.M. Pulido, J.P. Fernández-Palacios, R. Muñoz, R. Vilalta, Packet Optical Transport Network Slicing with Hard and Soft Isolation , Applied Sciences, Vol. 11, No. 13, pp.6219, July 2021.

5. LL. Guifré, R. Vilalta, S. Andreina, M. Xie, J.F. Pajo, H. Lonsethagen, S. lange, T. Zinner, G. Marson, M. Gajic, ***P. Alemany***, R. Casellas, R.Martínez, R.Muñoz, DLT-based End-to-end Inter-domain Transport Network Slice with SLA Management Using Cloud-based SDN Controllers, accepted in the 2022 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 14-16 November 2022, Chandler, Arizona (USA).

6. R. Asensio-Garriga, ***P. Alemany***, A. Molina Zarca, R. Sedar, C. Kalalas, J. Ortiz, R. Vilalta, R. Munoz and A. Skarmeta, A ZSM Security Framework for Autonomous management of E2E V2X Security Slices in Beyond-5G Networks, submitted in the IEEE Network Magazine, January 2023.

## 7.2 International and national R&D projects

### 7.2.1 5GTANGO - 5G Development and Validation Platform for global Industry–specific Network Services and Apps

**EC 5GPPP H2020 5GTANGO** [33] focused on the flexible programmability of 5G networks with the design, development and experimental validation of: i) a NFV-enabled Service Development Kit (SDK) ii) a Store platform with advanced validation and verification mechanisms for VNFs/Network Services qualification (including 3rd party contributions) and iii) a modular SP with an innovative orchestrator in order to bridge the gap between business needs and network operational management systems.

The project proposed an integrated vendor-independent platform where the outcome of the development kit that was a packaged NFV forwarding graph of composed services, was automatically tested and validated in the Store for their posterior deployment with a customized orchestrator compatible with common existing VIM and SDN controllers in the market. The combination of the proposed development toolkit, the validation and verification store and the SP were realised following an extended multi-modal NFV DevOps model between service developers, telecom operators and vertical industries, which allowed and increment of operational efficiency, facilitated the implementation and validation of new services and accelerated the adoption of NFV technologies.

#### PhD thesis relationship

The 5GTANGO project was the element used as the initial step from which this doctoral thesis has evolved. During this project the basic concepts related to SDN, NFV and Network Slicing such as the use of different virtualization technologies to deploy hybrid network slices or the SLA and Network Slicing relationship presented in Chapter 3 were acquired. Moreover, the 5GTANGO project allowed the student to learn the process about how to design and develop experimental use cases to obtain the results related to the theoretical concepts. To do so, the student contributed to the implementation of an SDN/NFV infrastructure used on the 5GTANGO use cases. Finally, based on the experience gained in this project it was possible to plan, think and identify which objectives and aspects were going to be studied in

this doctoral thesis.

### 7.2.2 5GCroCo - Fifth Generation Cross-Border Control

**EC 5GPPP H2020 5GCroCo** [152] focused on the cooperative, connected and autonomous mobility (CCAM) along Europe by looking for harmonized solutions to support cross-border traffic. The possibility of providing CCAM services along different countries when vehicles traverse various national borders has a huge innovative business potential. However, the seamless provision of connectivity and the uninterrupted delivery of services along borders also poses interesting technical challenges. The situation is particularly innovative given the multi-country, multi-operator, multi-telco-vendor, and multi-car-manufacturer scenario of any cross-border layout. Taking all these requirements, 5GCroCo defined a successful path towards CCAM services in cross-border scenarios and to reduce the uncertainties of a real 5G cross-border deployment. 5GCroCo experimentally validated 5G technologies in the Metz-Merzig-Luxembourg cross-border corridor, traversing the borders between France, Germany and Luxembourg.

#### PhD thesis relationship

The contribution in the 5GCroco project consisted on applying the experience gained during the 5GTANGO project regarding network slices and the implemented SDN/NFV infrastructure on a specific scenario such as the automotive one. While in the 5GTANGO project the student developed network slices for an optical and a media use cases, the objective of the contribution done in this project was to develop network slice in a cross-border scenario for vehicle communications.

### 7.2.3 INSPIRE5G-Plus - INtelligent Security and PervasIve tRust for 5G and Beyond

**EC 5GPPP H2020 INSPIRE-5GPlus** [128] aimed to make a revolutionary shift in the 5G and beyond security vision by progressing 5G Security and by devising a smart, trustworthy and liability aware 5G security platform for future connected systems, while contributing to its realization.

INSPIRE-5Gplus allowed, for the first time, the advancement of security vision for 5G and Beyond through the adoption of a set of emerging trends and technologies, such as zero-touch management, SD-SEC models, Artificial Intelligence/Machine Learning (AI/ML) techniques and TEE. New breed of SD-SEC assets and models were developed to address some of the challenges that remain (e.g., adaptive slice security) or are completely new (e.g., proactive security).

INSPIRE-5Gplus ensured that the provided security level is in conformance to legislation', verticals' and standard's security requirements. Trust and liability were fostered through integration of novel mechanisms supporting confidence between parties and compliance with regulation. INSPIRE-5Gplus planned to deliver innovative and actionable results (methodologies, enablers, services) of interest for related 5G-PPP projects (i.e., ICT-17, ICT-18 and ICT-19). Technical and/or societal innovation based on results achieved were promoted, while fostering the extendibility and convergence of 5G security within and beyond 5G-PPP.

#### PhD thesis relationship

While the previous two projects served to learn and improve the knowledge on Network Slicing, this project has been one of the two projects where the work developed in this doctoral thesis has been used. In detail, the student contributed to this project with the work done regarding SSLAs associated to network slices (Chapter 4, the management of trusted Network Slicing resources (Chapter 5) and trust in multi-stakeholder scenarios (Chapter 6).

### 7.2.4 TeraFlow - Secured autonomic traffic management for a Tera of SDN flows

**EC 5GPPP H2020 TeraFlow** [39] created a new type of secure cloud-native SDN controller to radically advance the state-of-the-art in beyond 5G networks. This new SDN controller is able to integrate with the current NFV and MEC frameworks as well as providing revolutionary features for both flow management (service layer) and optical/microwave network equipment integration (infrastructure layer), while incorporating security using ML and forensic evidence for multi-tenancy based on Distributed Ledgers. The target pool of stakeholders expands beyond the traditional telecom operators towards edge and hyperscale cloud providers. These actors can be benefited

from TeraFlow by a) exploiting a new type of secure SDN controller based on cloud-native solutions while, b) achieving substantial business agility with novel and highly dynamic network services with zero-touch automation features.

The novel TeraFlow architecture was validated through the implementation of the TeraFlow OS with the following innovations: a) Cloud-Native Architecture; b) Transport Network Integration; c) Unification of Network and Cloud Resource Management; d) ML-based security; and e) Distributed Ledger Technologies. The final demonstrations supported three use cases: Autonomous Networks Beyond 5G, Automotive, and Cybersecurity.

**PhD thesis relationship**

The TeraFlow project has been the second project where the work developed in this doctoral thesis has been used, together with the previous one. Following the focus of this project about the development of a cloud-native SDN controller, the student contributed to it with his research about the design and validation of a Blockchain-based SDN control infrastructure for a distributed creation and management of connectivity services to interconnect multi-domain computing resources. All this work is presented in Section 5.2 within Chapter 5. The work done by the student was widely accepted by the project and the knowledge generated used in one of the implemented use cases.

### 7.2.5 6G-OPENSEC - Seguridad y confianza en redes 6G desagregadas y abiertas

**6G-OPENSEC**, under the Spanish UNICO-5G [153] programm, aims to study new possibilities towards the next generation of networks (i.e., 6G). To do so, the project focuses on the research of different security aspects used in the current literature and how they can be adapted for the open and disagregated 6G networks. The 6G-OPENSEC is divided in three subprojects: a) the 6G-OPENSEC-SECURITY aims to investigate the design and development of a new Network Slice Manager able to deploy secured 6G network slices, b) the 6G-OPENSEC-KEYS focuses on the use of quantum keys to increase the security of the 6G data transmission, and c) the 6G-OPENSEC-TRUST investigates how trust in the multi-stakeholder scenario may be managed.

### PhD thesis relationship

Among the three previous 6G-OPENSE sub-projects, the student has been involved in two of them: the 6G-OPENSEC-SECURITY and the 6G-OPENSEC-TRUST. The topics and objectives defined in these projects are closely related to the topics discussed in this thesis. More specifically, part of the work presented in Chapter 4 is associated to the research done in the 6G-OPENSEC-SECURITY, while the content described in Chapter 6 is associated to the tasks done in the 6G-OPENSEC-TRUST.

# Chapter 8

# Conclusions and future work

## Contents

The last chapter of this thesis presents the main outcomes. Moreover, an assessment of objectives is presented together with potential future research directions that will allow to further develop the topics investigated by this thesis.

## 8.1 Conclusions

Despite Network Slicing has been a trend in these last years within the research of the control and management of network resources and even though there have been some Proof of Concept presentation using close-to-real scenarios [154], it is still in its youth in terms of being implemented by the real-world operators, infrastructure owners and service providers as it as it essentially depends on the use of SDN and NFV technologies. From RAN to transport, these three technologies will be implemented with the capabilities already defined in its first related technical reports [24] but also with those new ideas that might appear in the meantime. With this in mind, this thesis showed new possibilities regarding the security around Network Slicing.

The research and study of the security around any element is not an easy task to do due to two main reasons: a) because to study security aspects you need the element that needs to be secured, and also, b) because security itself is not something with closed and limited bounds as for any new threat, there's a new protection and for each new protection the threat is improved and evolved. In the case of Network Slicing and regarding the first reason, despite this technology is still quite young and there are few Network Slicing tools (i.e., Network Slice Manager), they are not fully closed and stable, therefore the security-specific work and tasks are slower and sometimes a bit more complex than expected.

Chapter 3 introduced the implementation of network slices with different virtualization technologies such as KVM and containers to use the strengths of each technology by deploying them where they fit the best and so, to improve the overall performance of the network slice. Moreover, the deployment of a multi-domain network slice across transport domains and the necessary control and management architecture was presented. While this second aspect served as the introductory work for the research described in the following chapters, the use of KVM and containers brought an improvement in terms of resources management.

Chapter 4 used the knowledge learned in the previous chapter and fo-

cused on the enforcement of first the correct QoS, and second the QoSec. To do so, SLAs and SSLAs are used to define the service performance and security requirements to be monitored during the deployment and operation of the network slice. Despite the procedure in both cases is quite similar, the complexity was in the fact of separating the monitoring of the service performance and the security as the sources of the violations may be blurred and for this reason, two different architectures were designed and validated.

Chapter 5 illustrated the possibility of taking advantage of Blockchain and to propose a change in the current control and management architectures based on hierarchical models. Using Blockchain, it becomes possible to have mesh architecture in which all domains interact among them to reach an agreement based on the available resources to avoid the need to keep increasing the amount of available resources in each domain.

Chapter 6 follows the path started in the previous chapter and illustrates the deployment of services based on a set of trust and reputation requirements in a multi-stakeholder scenario. Due to the complexity of having different players, Blockchain is used as the key element to fairly distribute among all the providers and clients, the information of the providers and how trustworthy are they to fulfill the expectations.

Finally, this thesis has focused mainly on the use of two possible elements to improve the security-related aspects around Network Slicing; the design and enforcement of SSLAs and the use of Blockchain as a tool for a distributed Network Slicing resources management system. The join of both elements with network slices allowed to consider the work done as quite innovative based on the following reasons: a) the work presented in Chapter 3 about the use of multiple technologies to deploy Hybrid network slices was used as a reference for some colleagues in their work [155], b) regarding the work in Chapter 4 using SSLAs, it illustrated a complete deployment of a secured E2E Network Slicing on a multi-domain scenario which, up to the best of our knowledge, it was not found in the literature, and c) the use of Blockchain presented in Chapters 5 and 6 focused on aspects that the ETSI ISG PDL has described as being interesting for the near-future networks in [156], specially on two of the three scenarios.

One remarkable achievement is that the work done in Chapter 5 was part of an Innovation Item identified by the European Commission's Innovation Radar team [157]. The identified item, called Blockchain-based Management of Certified Network Slices [158] joins the idea of the work presented in Chapter 5 (i.e., Blockchain-based management of network slices) with the use of a

solution that certifies the descriptors related to the VNFs, NSs (i.e., Network Services) and network slice resources. So, when there is a request, the network slice requester may have a guarantee about what the selected resources do and what they are expected to do. Moreover to the innovative identification, the innovative item was also pre-selected and registered to participate in the Innovation Radar Prize 2022.

Finally, in terms of dissemination, the objective has been properly achieved due to the production and publication of multiple articles and the participation of different EC funded projects. Regarding the articles production, four articles were accepted and published in journals or magazines with high impact such as the Journal of Optical Communications and Networking and the Communications Magazine, nine papers were presented in international conferences (ONDM'20, NetSoft'20, ICTON'20, 5GWF'20, NFV-SDN'20, ECOC'20, OFC'21, ONDM'21, EUCNC'22) and few more secondary contributions with researchers from the industry or research organizations. Moreover, the dissemination of the work done was also presented and contributed to a set of international projects and was positively accepted by the project consortium and so, increasing the professional relationship between CTTC and other organizations to collaborate in future projects.

## 8.2 Assessment of the technical objectives

With all the tasks presented in terms of theory and experimental results and the dissemination information in the previous chapters, it is possible to evaluate whether the technical objectives presented in Subsection 1.2.2 have been achieved or not.

The first technical objective consisted on reinforcing the current experience and acquiring new knowledge about NFV/SDN and Network Slicing systems. The work done in all chapters and specially in Chapter 3 (i.e., description and implementation of the multi-domain architecture and the use of different virtualization technologies) allowed the student to gain knowledge and think of the ideas described and the following chapters.

The second technical objective (i.e., to contribute to the telecommunications field) is properly achieved with all the experimental results and implemented systems presented in all the dissemination items and across all the chapters in this PhD thesis.

The third technical objective described the use of SSLAs to make network

slices a secured element and, as demonstrated in Chapter 4, the objective is achieved with the enforcement of SSLAs using the new designed and implemented architectures.

The fourth technical objective (i.e., to investigate the use of DLT on Network Slicing management and trust procedures) is completed with the work done in Chapter 5 with the new modules and processes for the cooperative network slices management and in Chapter 6 with the trust and reputation computation model for a multi-stakeholder scenario.

The fifth and last technical objective (i.e., to acquire new skills to improve the research career) was properly accomplished through all the works published (and rejected). The peer-reviewed process has challenged the student to find the right way to write due to certain restrictions and corrections in each publication or the way to communicate the ideas and technical concepts in the conferences oral presentations towards the audience.

Based on all the previous statements, it is possible to declare that all the objectives were properly fulfilled.

## 8.3 Future research work

From the work presented, there are two clear paths to follow for further research work, the enforcement of SSLAs and the use of Blockchain.

A very clear point regarding SSLAs is that they will be used in the future of Network Slicing. An aspect to be studied is the strengths and weaknesses to have two different monitoring architectures, one for the SLAs and service performance and one for the SSLAs and the service security. While the current path is to split security and service performance monitoring, this fact increases the implementation complexity of the overall architecture. So, a possible future work could be to study the common elements and try to join both monitoring systems with those elements that would allow the reduction of the aforementioned complexity while keeping the independence to find the right violation issue.

On what regards the use of Blockchain to manage Network Slicing resources and their deployment, the main issue is to identify the exact use cases in which the current Blockchain technology brings advantages. In some cases this technology might yet not be fast enough for certain management actions, but despite this, the use of Blockchain became a very interesting part of this PhD thesis and for this reason it will further be researched in international

research projects such as ACROSS, Hexa-X II or current national project like the 6G-OPENSEC.

Finally, about the reputation and trust, one topic to keep analysing is the mathematical model presented and its potential improvement by considering new possible reputation parameters and by evaluating the weight of each of these parameters within the overall trust value. Moreover, more experimental tests in a more complex testbed would greatly improve the understanding of this technology in real-world scenarios.

# Chapter 9

# Bibliography


[1] E. Bassot, "Ten issues to watch in 2020," standard, European Parliamentary Research Service (EPRS), Brussels, Belgium, 2020. 1.1

[2] EUROSTAT, "Digital economy and society statistics-households and individuals," *EUROSTAT website*, no. 1, 2020. 1.1

[3] 5GPPP, "https://5g-ppp.eu/verticals/." Online. Accessed in February 13, 2022. 1.1, 2.2

[4] E. Haleplidis, K. Pentikousis, S. Denazis, J. Salim, D. Meyer, and O. Koufopavlou, "Rfc 7426: Software-defined networking (sdn): Layers and architecture terminology," *IRTF*, 01 2015. 1.1, 2.1

[5] M. Chiosi et al., "Network functions virtualisation: An introduction, benefits, enablers, challenges & call for action," *ETSI Introductory Whitepaper*, no. 1, 2012. 1.1, 2.2.2

[6] ETSI, "Network functions virtualisation (nfv); infrastructure overview," standard, European Telecommunications Standards Institute (ETSI), Sophia Antípolis, Fr, Dec. 2015. 1.1

[7] NGNM, "Description of network slicing concept," tech. rep., NGMN Alliance, Jan. 2016. 1.1, 2.2.3, 4

[8] ETSI, "Network functions virtualisation (nfv); terminology for main concepts in nfv," standard, European Telecommunications Standards Institute (ETSI), Sophia Antípolis, Fr, Jan. 2020. 2.1

[9] Open Networking Foundation (ONF), "Tapi v2.1.3 reference implementation agreement," tech. rep., Open Networking Foundation, 1000 El Camino Real, Suite 100, Menlo Park, CA 94025, 7 2020. 2.1, 5.2.1

[10] ETSI, "Zero-touch network and service management (zsm); end-to-end management and orchestration of network slicing," standard, European Telecommunications Standards Institute (ETSI), Sophia Antípolis, Fr, June 2021. 2.1

[11] ISO, "https://www.iso.org/obp/ui/iso:std:iso:22739:ed-1:v1:en." Online. Accessed in August 8, 2022. 2.1

[12] ETSI, "Permissioned distributed ledgers (pdl); smart contracts; system architecture and functional specification," standard, European Telecommunications Standards Institute (ETSI), Sophia Antípolis, Fr, June 2021. 2.1

[13] N. Anerousis, P. Chemouil, A. A. Lazar, N. Mihai, and S. B. Weinstein, "The origin and evolution of open programmable networks and sdn," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1956–1971, 2021. 2.2.1

[14] L. Csikor, M. Szalay, G. Rétvári, G. Pongrácz, D. P. Pezaros, and L. Toka, "Transition to sdn is harmless: Hybrid architecture for migrating legacy ethernet switches to sdn," *IEEE/ACM Transactions on Networking*, vol. 28, no. 1, pp. 275–288, 2020. 2.2.1

[15] M. B. Jiménez, D. Fernández, J. E. Rivadeneira, L. Bellido, and A. Cárdenas, "A survey of the main security issues and solutions for the sdn architecture," *IEEE Access*, vol. 9, pp. 122016–122038, 2021. 2.2.1

[16] Q. Yan, F. R. Yu, Q. Gong, and J. Li, "Software-defined networking (sdn) and distributed denial of service (ddos) attacks in cloud computing environments: A survey, some research issues, and challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 602–622, 2016. 2.2.1

[17] J. Gil Herrera and J. F. Botero, "Resource allocation in nfv: A comprehensive survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, 2016. 2.2.2

[18] I. Farris, T. Taleb, Y. Khettab, and J. Song, "A survey on emerging sdn and nfv security mechanisms for iot systems," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 812–837, 2019. 2.2.2

[19] D. Behnke, M. Müller, P.-B. Bök, S. Schneider, M. Peuster, H. Karl, A. Rocha, M. Mesquita, and J. Bonnet, "Nfv-driven intrusion detection for smart manufacturing," in *2019 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pp. 1–6, 2019. 2.2.2

[20] Red Hat, "https://www.redhat.com/en/topics/containers/containers-vs-vms." Online. Accessed in September 22, 2022. 2.2.2

[21] OpenStack, "https://wiki.openstack.org/wiki/main_page." Online. Accessed in September 22, 2022. 2.2.2

[22] Docker, Inc., "https://www.docker.com/resources/what-container." Online. Accessed in February 17, 2022. 2.2.2

[23] The Linux Foundation, "https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/." Online. Accessed in February 17, 2022. 2.2.2

[24] 3GPP, "Study on management and orchestration of network slicing for next generation network," technical report, 3rd Generation Partnership Project (3GPP), Jan. 2018. 2.2.3, 8.1

[25] ETSI, "Network functions virtualisation (nfv); architectural framework," standard, European Telecommunications Standards Institute (ETSI), Sophia Antípolis, Fr, Dec. 2014. 2.2.3

[26] ETSI, "Network functions virtualisation (nfv) release 3; evolution and ecosystem; report on network slicing support with etsi nfv architecture framework," standard, European Telecommunications Standards Institute (ETSI), Sophia Antípolis, Fr, Dec. 2017. 2.2.3, 3.2, 4.1.1, 4.2.1, 5.1.1

[27] ETSI, "Network functions virtualisation (nfv); management and orchestration," standard, European Telecommunications Standards Institute (ETSI), Sophia Antípolis, Fr, Dec. 2014. 2.2.3, 2.2.4

[28] 3GPP, "Management and orchestration; architecture framework," technical specification, 3rd Generation Partnership Project (3GPP), Sophia Antípolis, Fr, Mar. 2022. 2.2.3

[29] IETF, "https://tools.ietf.org/id/draft-defoy-coms-subnet-interconnection-04.html." online. Accessed in February 3, 2022. 2.2.3

[30] R. Vilalta, A. Mayoral, R. Muñoz, R. Casellas, and R. Martínez, "Hierarchical sdn orchestration for multi-technology multi-domain networks with hierarchical abno," in *2015 European Conference on Optical Communication (ECOC)*, pp. 1–3, 2015. 2.2.3

[31] T. Soenen, R. Banerjee, W. Tavernier, D. Colle, and M. Pickavet, "Demystifying network slicing: From theory to practice," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pp. 1115–1120, 2017. 2.2.3

[32] SONATA-NFV Platform, "https://www.sonata-nfv.eu." Online. Accessed in February 02, 2022. 2.2.3, 3.1, 3.2.1, 3.3

[33] EUC 5GTANGO Project, "https://www.5gtango.eu/." Online. Accessed in February 02, 2022. 2.2.3, 3.2.1, 7.2.1

[34] P. Alemany, J. L. de la Cruz, A. Pol, A. Roman, P. Trakadas, P. Karkazis, M. Touloupou, E. Kapassa, D. Kyriazis, T. Soenen, C. Parada, J. Bonnet, R. Casellas, R. Martínez, R. Vilalta, and R. Muñoz, "Network slicing over a packet/optical network for vertical applications applied to multimedia real-time communications," in *2019 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pp. 1–2, IEEE, 2019. 2.2.3

[35] ONAP Platform, "https://www.onap.org/." Online. Accessed in February 02, 2022. 2.2.3, 3.1, 3.2.1

[36] ETSI, "https://osm.etsi.org/." Online. Accessed in February 02, 2022. 2.2.3, 3.1, 3.2.1

[37] OpenDayLight, "https://www.opendaylight.org/." Online. Accessed in November 14, 2022. 2.2.3

[38] Open Networking Foundation, "https://opennetworking.org/onos/." Online. Accessed in November 14, 2022. 2.2.3

[39] Teraflow project, "https://www.teraflow-h2020.eu/." Online. Accessed in November 14, 2022. 2.2.3, 7.2.4

[40] R. Vilalta, R. Muñoz, R. Casellas, R. Martínez, V. López, O. G. de Dios, A. Pastor, G. P. Katsikas, F. Klaedtke, P. Monti, A. Mozo, T. Zinner, H. Øverby, S. Gonzalez-Diaz, H. Lønsethagen, J.-M. Pulido, and D. King, "Teraflow: Secured autonomic traffic management for a tera of sdn flows," in *2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*, pp. 377–382, 2021. 2.2.3

[41] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca, and J. Folgueira, "Network slicing for 5g with sdn/nfv: Concepts, architectures, and challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 80–87, 2017. 2.2.3

[42] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5g: Survey and challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 94–100, 2017. 2.2.3

[43] J. Ordóñez et al., "Cross-domain slice orchestration for advanced vertical trials in a multi-vendor 5g facility," in *2020 European Conference on Networks and Communications (EuCNC)*, 06 2020. 2.2.3

[44] M. Vincenzi et al., "Multi-tenant slicing for spectrum management on the road to 5g," *IEEE Wireless Communications*, vol. 24, no. 5, pp. 118–125, 2017. 2.2.3

[45] M. Richart, J. Baliosian, J. Serrat, and J. Gorricho, "Resource slicing in virtual wireless networks: A survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 462–476, 2016. 2.2.3

[46] Y. Cui et al., "An intelligent coordinator design for network slicing in service-oriented vehicular networks," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, pp. 1–6, 2020. 2.2.3

[47] F. Ansah et al., "Worst-case delay slicing for time-sensitive applications in softwarized industrial networks," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, pp. 1652–1659, 2020. 2.2.3

[48] L. Gifre, C. Natalino, S. Gonzalez-Diaz, F. Soldatos, S. Barguil, C. Aslanoglou, F. J. Moreno-Muro, A. N. Q. Cornelio, L. Cepeda, R. Martinez, C. Manso, V. Apostolopoulos, S. P. Valiviita, O. G. de Dios, J. Rodriguez, R. Casellas, P. Monti, G. P. Katsikas, R. Muñoz, and R. Vilalta, "Demonstration of zero-touch device and l3-vpn service management using the teraflow cloud-native sdn controller," in *2022 Optical Fiber Communications Conference and Exhibition (OFC)*, pp. 1–3, 2022. 2.2.4

[49] R. Vilalta, L. Gifre, M. Xie, J. F. Pajo, H. Lønsethagen, S. Lange, H. Øverby, T. Zinner, R. Muñoz, R. Casellas, and R. Martínez, "End-to-end interdomain transport network slice management using cloud-based sdn controllers," in *2022 27th OptoElectronics and Communications Conference (OECC) and 2022 International Conference on Photonics in Switching and Computing (PSC)*, pp. 1–3, 2022. 2.2.4

[50] D. Sattar and A. Matrawy, "Towards secure slicing: Using slice isolation to mitigate ddos attacks on 5g core network slices," *CoRR*, vol. abs/1901.01443, 2019. 2.3

[51] GSMA, "E2e network slicing architecture," technical report, Global System for Mobile Communications Association (GSMA), London, UK, 2021. 2.3

[52] D. A. Chekired, M. A. Togou, L. Khoukhi, and A. Ksentini, "5g-slicing-enabled scalable sdn core network: Toward an ultra-low latency of autonomous driving service," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 8, pp. 1769–1782, 2019. 2.3

[53] A. Marotta, D. Cassioli, M. Tornatore, Y. Hirota, Y. Awaji, and B. Mukherjee, "Reliable slicing with isolation in optical metro-aggregation networks," in *Optical Fiber Communication Conference (OFC) 2020*, p. M4D.3, Optical Society of America, 2020. 2.3

[54] A. Marotta, D. Cassioli, M. Tornatore, Y. Hirota, Y. Awaji, and B. Mukherjee, "Multilayer protection-at-lightpath for reliable slicing with isolation in optical metro-aggregation networks," *Journal of Optical Communications and Networking*, vol. 14, no. 4, pp. 289–302, 2022. 2.3

[55] A. Papageorgiou et al., "Sla management procedures in 5g slicing-based systems," in *2020 European Conference on Networks and Communications (EuCNC)*, pp. 7–11, 2020. 2.3.1

[56] K. D. R. Assis, R. C. Almeida, H. Waldman, A. F. Santos, M. S. Alencar, M. J. Reed, A. Hammad, and D. Simeonidou, "Sla formulation for squeezed protection in elastic optical networks considering the modulation format," *Journal of Optical Communications and Networking*, vol. 11, no. 5, pp. 202–212, 2019. 2.3.1

[57] X. Xue, B. Pan, F. Agraz, A. Pagès, X. Guo, F. Yan, S. Spadaro, and N. Calabretta, "Sdn-enabled reconfigurable optical data center network with automatic network slicing to provision dynamic qos," in *2020 European Conference on Optical Communications (ECOC)*, pp. 1–4, 2020. 2.3.1

[58] C.-Y. Chen, P.-R. Lin, Y.-C. Chen, P.-H. Chang, and S.-S. Jeng, "Optimization of spectrum resource allocation based on network slicing," in *2021 30th Wireless and Optical Communications Conference (WOCC)*, pp. 61–65, 2021. 2.3.1

[59] R. R. Henning, "Security service level agreements: Quantifiable security for the enterprise?," in *1999 Workshop on New Security Paradigms ser. NSPW '99*, pp. 54–60, 1999. 2.3.2

[60] V. Casola, A. De Benedictis, M. Eraşcu, J. Modic, and M. Rak, "Automatically enforcing security slas in the cloud," *IEEE Transactions on Services Computing*, vol. 10, no. 5, pp. 741–755, 2017. 2.3.2

[61] ENISA, "Survey and analysis of security parameters in cloud slas across the european public sector," tech. rep., European Union Agency for Cybersecurty, 2011. 2.3.2

[62] G. Hogben, M. Dekker, and ENISA, "Procure secure: A guide to monitoring of security service levels in cloud contracts," tech. rep., European Union Agency for Cybersecurty, 2011. 2.3.2

[63] NIST, "Security and privacy controls for information systems and organizations," tech. rep., National Insitute of Standards and Technology, 2020. 2.3.2, 8, 4.2.3

[64] SPECS Project, "https://cordis.europa.eu/project/id/610795." Online. Accessed in August 30, 2022. 2.3.2

[65] MUSA Project, "https://www.musa-project.eu/." Online. Accessed in August 30, 2022. 2.3.2

[66] N. Kaaniche, M. Mohamed, M. Laurent, and H. Ludwig, "Security sla based monitoring in clouds," in *2017 IEEE International Conference on Edge Computing (EDGE)*, pp. 90–97, 2017. 2.3.2

[67] K. W. Ullah and A. S. Ahmed, "Demo paper: Automatic provisioning, deploy and monitoring of virtual machines based on security service level agreement in the cloud," in *2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pp. 536–537, 2014. 2.3.2

[68] A. De Benedictis, V. Casola, M. Rak, and U. Villano, "Cloud security: From per-provider to per-service security slas," in *2016 International Conference on Intelligent Networking and Collaborative Systems (INCoS)*, pp. 469–474, 2016. 2.3.2

[69] ETSI, "Zero-touch network and service management (zsm); reference architecture," group specification, European Telecommunications Standards Institute (ETSI), Sophia Antípolis, Fr, 2019. 2.3.3, 4.2.1

[70] R. Muñoz, R. Vilalta, R. Casellas, R. Martínez, F. Vicens, J. Martrat, V. López, and D. López, "Hierarchical and recursive NFV service platform for end-to-end network service orchestration across multiple NFVI domains," in *in Proc. of International Conference on Transparent Optical Networks (ICTON), 1-5 July 2018, Bucharest (Romania)*, July 2018. 2.4

[71] D. Gkounis, N. Uniyal, A. S. Muqaddas, R. Nejabati, and D. Simeonidou, "Demonstration of the 5GUK exchange: A lightweight platform for dynamic end-to-end orchestration of softwarized 5g networks," in *2018 European Conference on Optical Communication (ECOC)*, pp. 1–3, 2018. 2.4

[72] BitTorrent Platform, "https://www.bittorrent.com/es/." online. Accessed in February 3, 2022. 2.4

[73] Emule Platform, "https://www.emule.es/." online. Accessed in February 3, 2022. 2.4

[74] M. Fouda, T. Taleb, M. Guizani, Y. Nemoto, and N. Kato, "On supporting p2p-based vod services over mesh overlay networks," in *GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference*, pp. 1–6, 2009. 2.4

[75] F. Di Cerbo, P. Bisson, A. Hartman, S. Keller, P. H. Meland, M. Moffie, N. G. Mohammadi, S. Paulus, and S. Short, "Towards trustworthiness assurance in the cloud," in *Cyber Security and Privacy Forum*, pp. 3–15, Springer, 2013. 2.4.1

[76] S. Shetty, C. Kamhoua, and L. Njilla, *Blockchain for Distributed Systems Security*. Wiley-IEEE Press, 1 ed., 5 2019. 2.4.1

[77] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Cryptography Mailing list at https://metzdowd.com*, 03 2009. 2.4.1, 5.3.1

[78] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014. 2.4.1, 5.3.1

[79] The Linux Foundationm, "https://www.hyperledger.org/." online. Accessed in February 3, 2022. 2.4.1, 5.3.1

[80] W. Wang, D. T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wang, Y. Wen, and D. I. Kim, "A survey on consensus mechanisms and mining strategy management in blockchain networks," *IEEE Access*, vol. 7, pp. 22328–22370, 2019. 2.4.1

[81] S. Kou, H. Yang, H. Zheng, W. Bai, J. Zhang, and Y. Wu, "Blockchain mechanism based on enhancing consensus for trusted optical networks," in *2017 Asia Communications and Photonics Conference (ACP)*, pp. 1–3, 2017. 2.4.1

[82] H. Yang, H. Zheng, J. Zhang, Y. Wu, Y. Lee, and Y. Ji, "Blockchain-based trusted authentication in cloud radio over fiber network for 5g," in *2017 16th International Conference on Optical Communications and Networks (ICOCN)*, pp. 1–3, 2017. 2.4.1

[83] S. Boukria, M. Guerroumi, and I. Romdhani, "Bcfr: Blockchain-based controller against false flow rule injection in sdn," in *2019 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1034–1039, 2019. 2.4.1

[84] S. Misra, K. Sarkar, and N. Ahmed, "Blockchain-based controller recovery in sdn," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 1063–1068, 2020. 2.4.1

[85] R. V. Rosa and C. E. Rothenberg, "Blockchain-based decentralized applications for multiple administrative domain networking," *IEEE Communications Standards Magazine*, vol. 2, no. 3, pp. 29–37, 2018. 2.4.1

[86] S. Fichera, A. Sgambelluri, A. Giorgetti, F. Cugini, and F. Paolucci, "Blockchain-anchored failure responsibility management in disaggregated optical networks," in *2020 Optical Fiber Communications Conference and Exhibition (OFC)*, pp. 1–3, 2020. 2.4.1

[87] H. Yang, Y. Li, S. Guo, J. Ding, Y. Lee, and J. Zhang, "Distributed blockchain-based trusted control with multi-controller collaboration for software defined data center optical networks in 5g and beyond," in *2019 Optical Fiber Communications Conference and Exhibition (OFC)*, pp. 1–3, 2019. 2.4.1

[88] Y. Liang, H. Yang, Q. Yao, S. Guo, A. Yu, and J. Zhang, "Blockchain-based efficient recovery for secure distributed control in software defined optical networks," in *2019 Optical Fiber Communications Conference and Exhibition (OFC)*, pp. 1–3, 2019. 2.4.1

[89] J. Backman, S. Yrjölä, K. Valtanen, and O. Mämmelä, "Blockchain network slice broker in 5g: Slice leasing in factory of the future use case," in *2017 Internet of Things Business Models, Users, and Networks*, pp. 1–8, 2017. 2.4.1

[90] B. Nour, A. Ksentini, N. Herbaut, P. A. Frangoudis, and H. Moungla, "A blockchain-based network slice broker for 5g services," *IEEE Networking Letters*, vol. 1, no. 3, pp. 99–102, 2019. 2.4.1

[91] M. Fiorani, A. Rostami, L. Wosinska, and P. Monti, "Abstraction models for optical 5g transport networks," *J. Opt. Commun. Netw.*, vol. 8, pp. 656–665, Sep 2016. 2.4.2

[92] N. Đerić, A. Varasteh, A. Basta, A. Blenk, and W. Kellerer, "Sdn hypervisors: How much does topology abstraction matter?," in *2018 14th International Conference on Network and Service Management (CNSM)*, pp. 328–332, 2018. 2.4.2

[93] M. Licciardello, M. Fiorani, M. Furdek, P. Monti, C. Raffaelli, and L. Wosinska, "Performance evaluation of abstraction models for orchestration of distributed data center networks," in *2017 19th International Conference on Transparent Optical Networks (ICTON)*, pp. 1–4, 2017. 2.4.2

[94] R. Casellas, R. Martínez, R. Vilalta, and R. Muñoz, "Abstraction and control of multi-domain disaggregated optical networks with open-roadm device models," *Journal of Lightwave Technology*, vol. 38, no. 9, pp. 2606–2615, 2020. 2.4.2

[95] S. Ding, G. Shen, K. X. Pan, S. K. Bose, Q. Zhang, and B. Mukherjee, "Blockchain-assisted spectrum trading between elastic virtual optical networks," *IEEE Network*, vol. 34, no. 6, pp. 205–211, 2020. 2.4.2

[96] H. Yang, Y. Liang, J. Yuan, Q. Yao, A. Yu, and J. Zhang, "Distributed blockchain-based trusted multidomain collaboration for mobile edge computing in 5g and beyond," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 11, pp. 7094–7104, 2020. 2.4.2

[97] A. Derhab, M. Guerroumi, M. Belaoued, and O. Cheikhrouhou, "Bmc-sdn: Blockchain-based multicontroller architecture for secure software-

defined networks," *Wireless Communications and Mobile Computing*, 2021. 2.4.2

[98] P. Gorla, V. Chamola, V. Hassija, and N. Ansari, "Blockchain based framework for modeling and evaluating 5g spectrum sharing," *IEEE Network*, vol. 35, no. 2, pp. 229–235, 2021. 2.4.2

[99] P. Fernando and J. Wei, "Blockchain-powered software defined network-enabled networking infrastructure for cloud management," in *2020 IEEE 17th Annual Consumer Communications Networking Conference (CCNC)*, pp. 1–6, 2020. 2.4.2

[100] S. Fichera, A. Sgambelluri, F. Paolucci, A. Giorgetti, N. Sambo, P. Castoldi, and F. Cugini, "Blockchain-anchored disaggregated optical networks," *Journal of Lightwave Technology*, vol. 39, no. 20, pp. 6357–6365, 2021. 2.4.2

[101] S. Malik, V. Dedeoglu, S. S. Kanhere, and R. Jurdak, "Trustchain: Trust management in blockchain and iot supported supply chains," in *2019 IEEE International Conference on Blockchain (Blockchain)*, pp. 184–193, 2019. 2.4.3, 6.1

[102] M. Arasteh, M. Amini, and R. Jalili, "A trust and reputation-based access control model for virtual organizations," in *2012 9th International ISC Conference on Information Security and Cryptology*, pp. 121–127, 2012. 2.4.3, 6.1

[103] C. Benzaïd, T. Taleb, and M. Z. Farooqi, "Trust in 5g and beyond networks," *IEEE Network*, vol. 35, no. 3, pp. 212–222, 2021. 2.4.3

[104] N.-W. Lo, K.-H. Yeh, and P.-Y. Liu, "An efficient resource allocation scheme for cross-cloud federation," in *Anti-counterfeiting, Security, and Identification*, pp. 1–4, 2012. 2.4.3

[105] V. Casola, L. Coppolino, A. Mazzeo, N. Mazzocca, and M. Rak, "Design and implementation of truman, a trust manager component for distributed systems," in *Second International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing (SecPerU'06)*, pp. 7 pp.–40, 2006. 2.4.3

[106] D. Wang, X. Chen, H. Wu, R. Yu, and Y. Zhao, "A blockchain-based vehicle-trust management framework under a crowdsourcing environment," in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pp. 1950–1955, 2020. 2.4.3

[107] C. Balachandran, P. A. C, G. Ramachandran, and B. Krishnamachari, "Edison: A blockchain-based secure and auditable orchestration framework for multi-domain software efined networks," in *2020 IEEE International Conference on Blockchain (Blockchain)*, pp. 144–153, 2020. 2.4.3

[108] ETSI, "Network functions virtualisation (nfv); virtual network functions architecture," standard, European Telecommunications Standards Institute (ETSI), Sophia Antípolis, Fr, Dec. 2014. 3

[109] IBM, "https://www.ibm.com/cloud/blog/containers-vs-vms." Online. Accessed in February 14, 2022. 3

[110] C. Manso, P. Alemany, R. Vilalta, R. Muñoz, R. Casellas, and R. Martínez, "End-to-end sdn/nfv orchestration of multi-domain transport networks and distributed computing infrastructure for beyond-5g services," *IEICE Transactions*, vol. E104-B, pp. 188–198, Mar 2021. 3

[111] P. Alemany, R. Vilalta, F.Vicens, I. Dominguez Gómez, R. Casellas, R. Martínez, S. Castro, J. Martrat, and R. Muñoz, "Hybrid network slicing: Composing network slices based on vnfs, cnfs network services," in *2020 IEEE Conference on Network Softwarization (NetSoft)*, 2020. 3, 5.3.2

[112] ETSI, "Next generation protocols (ngp); e2e network slicing reference framework and information model," standard, European Telecommunications Standards Institute (ETSI), Sophia Antípolis, Fr, Sept. 2018. 3.1, 3.1

[113] GSMA, "Generic network slice template," technical report, Global System for Mobile Communications Association (GSMA), London, UK, 2021. 3.1, 4.2.3

[114] ETSI, "Network functions virtualisation (nfv) release 3; architecture; report on the enhancements of the nfv architecture towards "cloud-native" and "paas"," standard, European Telecommunications Standards Institute (ETSI), Sophia Antípolis, Fr, 2019. 3.2

[115] NetworkLessons.com, "https://networklessons.com/cisco/ccna-routing-switching-icnd2-200-105/introduction-to-metro-ethernet." Online. Accessed in February 16, 2022. 3.2.1

[116] Sonata NFV GitHub Repository, "https://github.com/sonata-nfv/tng-schema/blob/master/function-record/vnfr-schema.yml." Online. Accessed in February 02, 2022. 3.2.1

[117] R. Vilalta, P. Alemany, R. Casellas, R. Martínez, C. Parada, J. Bonnet, F. Vicens, and R. Muñoz, "Zero-touch network slicing through multi-domain transport networks," in *2018 20th International Conference on Transparent Optical Networks (ICTON)*, pp. 1–4, 2018. 3.3, 4.2.1

[118] A. Mayoral et al., "Sdn orchestration architectures and their integration with cloud computing applications," *Optical Switching and Networking*, vol. 26, pp. 2 – 13, 2017. Advances on Path Computation Element. 3.3

[119] D. King et al., "Rfc7491: A pce-based architecture for application-based network operations," *Internet Engineering Task Force (IETF)*, 03 2015. 3.3, 5.2

[120] V. Lopez, R. Vilalta, V. Uceda, A. Mayoral, R. Casellas, R. Martinez, R. Muñoz, and J. P. Fernandez Palacios, "Transport api: A solution for sdn in carriers networks," in *ECOC 2016; 42nd European Conference on Optical Communication*, pp. 1–3, 2016. 3.3, 5.2, 5.2.4

[121] S. Schneider, M. Peuster, K. Hannemann, D. Behnke, M. Muller, P.-B. Bök, and H. Karl, "Producing cloud-native": Smart manufacturing use cases on kubernetes," in *2019 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pp. 1–2, 2019. 3.3.2

[122] EUC 5GTANGO Project, "D7.1 evaluation strategy and testbed setup report," project deliverable, EUC 5GTANGO Project, 2017. 3.3.2

[123] EUC 5GTANGO Project, "D7.2 implementation of pilots and first evaluation," project deliverable, EUC 5GTANGO Project, 2019. 3.3.2

[124] P. Alemany, D. Ayed, R. Vilalta, R. Muñoz, P. Bisson, R. Casellas, and R. Martínez, "Transport network slices with security service level agreements," in *2020 22nd International Conference on Transparent Optical Networks (ICTON)*, pp. 1–4, 2020. 4, 4.3.2

[125] P. Alemany, A. Román, R. Vilalta, A. Pol, J. Bonnet, E. Kapassa, M. Touloupou, D. Kyriazis, P. Karkazis, P. Trakadas, J. Martrat, R. Casellas, R. Martinez, and R. Muñoz, "A kpi-enabled nfv mano architecture for network slicing with qos," *IEEE Communications Magazine*, vol. 59, no. 7, pp. 44–50, 2021. 4

[126] P. Alemany, A. Molina, C. Dangerville, D. Ayed, R. Muñoz, R. Casellas, R. Martinez, A. Skarmeta, and R. Vilalta, "Management and enforcement of secured multi-domain network slices," *submitted in the Elsevier Optical Fiber Technology*, 2022. 4, 4.2.3

[127] 3GPP, "Technical specification group services and system aspects; system architecture for the 5g system (5gs); stage 2," technical specification, 3rd Generation Partnership Project (3GPP), 2018. 4.1.2

[128] Inspire5G-plus Project, "https://www.inspire-5gplus.eu/." Online. Accessed in January 3, 2023. 4.2.1, 7.2.3

[129] ETSI, "5g; management and orchestration; concepts, use cases and requirements (3gpp ts 28.530 version 15.0.0 release 15)," technical specification, European Telecommunications Standards Institute (ETSI), Sophia Antípolis, Fr, 2018. 4.2.1

[130] V. Casola, A. D. Benedictis, M. Rak, and U. Villano, "Sla-based secure cloud application development: The specs framework," in *2015 17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pp. 337–344, 2015. 4.2.3

[131] R. Muñoz et al., "The adrenaline testbed: An sdn/nfv packet/optical transport network and edge/core cloud platform for end-to-end 5g and iot services," in *2017 European Conference on Networks and Communications (EuCNC)*, pp. 1–5, 2017. 4.3.1, 5.3.2

[132] P. Trakadas et al., "Comparison of management and orchestration solutions for the 5g era," *Journal of Sensor and Actuator Networks*, vol. 9, p. 4, 1 2020. 4.3.1

[133] M. Peuster et al., "Introducing automated verification and validation for virtualized network functions and services," *IEEE Communications Magazine*, vol. 57, no. 5, pp. 96–102, 2019. 4.3.1

[134] EUC 5G TANGO Project, "D7.3 final demonstrators and evaluation report," project deliverable, EUC 5GTANGO Project, 2020. 4.3.1

[135] R. Vilalta, P. Alemany, R. Sedar, C. Kalalas, R. Casellas, R. Martínez, F. Vázquez-Gallego, J. Ortiz, A. Skarmeta, J. Alonso-Zarate, and R. Muñoz, "Applying security service level agreements in v2x network slices," in *2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pp. 114–115, 2020. 4.3.2

[136] WIRED Magazine, "https://www.wired.com/story/github-ddos-memcached/." Online, 2018. Accessed in March 19, 2022. 4.3.2

[137] P. Alemany, R. Vilalta, R. Muñoz, R. Casellas, and R. Maríinez, "Peer-to-peer blockchain-based nfv service platform for end-to-end network slice orchestration across multiple nfvi domains," in *2020 IEEE 3rd 5G World Forum (5GWF)*, pp. 151–156, 2020. 5, 5.3.2

[138] P. Alemany, R. Vilalta, R. Muñoz, R. Martínez, and R. Casellas, "Managing network slicing resources using blockchain in a multi-domain software defined optical network scenario," in *2020 European Conference on Optical Communications (ECOC)*, pp. 1–4, 2020. 5, 5.3.2

[139] P. Alemany, R. Vilalta, R. Muñoz, R. Casellas, and R. Martínez, "Blockchain-based connectivity provisioning in multiple transport sdn domains," in *2021 International Conference on Optical Network Design and Modeling (ONDM)*, pp. 1–3, 2021. 5, 5.3.2, 5.3.3

[140] P. Alemany, R. Vilalta, R. Muñoz, R. Casellas, and R. Martínez, "End-to-end network slice stitching using blockchain-based peer-to-peer network slice managers and transport sdn controllers," in *2021 Optical Fiber Communications Conference and Exhibition (OFC)*, pp. 1–3, 2021. 5, 5.3.2

[141] P. Alemany, R. Vilalta, R. M. noz, R. Casellas, and R. Martínez, "Evaluation of the abstraction of optical topology models in blockchain-based data center interconnection," *J. Opt. Commun. Netw.*, vol. 14, pp. 211–221, Apr 2022. 5, 5.3.3

[142] NASDAQ, "https://www.nasdaq.com/articles/bitcoin-has-hit-a-historically-slow-average-block-time-2021-06-29." online. Accessed in June 13, 2022. 5.3.1

[143] The Ethereum Foundation, "https://ethereum.org/en/developers/docs/blocks/block-time." online. Accessed in June 13, 2022. 5.3.1

[144] ConsenSys Software Inc. 2021, "https://www.trufflesuite.com/ganache." online. Accessed in November 3, 2021. 5.3.1, 5.3.3, 6.3

[145] The Ethereum Foundation, "https://ethereum.org/en/developers/docs/consensus-mechanisms/." online. Accessed in Janury 9, 2022. 5.3.1

[146] The Ethereum Foundation, "https://ethereum.org/en/developers/docs/consensus-mechanisms/pow/." online. Accessed in Janury 9, 2022. 5.3.1

[147] The Ethereum Foundation, "https://remix-project.org/." online. Accessed in November 3, 2021. 5.3.3, 6.3

[148] P. Merriam and J. Carver, "https://web3py.readthedocs.io/en/stable/." online. Accessed in November 3, 2021. 5.3.3, 6.3

[149] R. Casellas, F. J. Vílchez, L. Rodríguez, R. Vilalta, J. M. Fàbrega, R. Martínez, L. Nadal, M. S. Moreolo, and R. Muñoz, "An ols controller for hybrid fixed / flexi grid disaggregated networks with open interfaces," in *Optical Fiber Communication Conference (OFC) 2020*, p. M3K.2, Optical Society of America, 2020. 5.3.3

[150] The Ethereum Foundation, "https://ethereum.org/en/developers/docs/gas/." online. Accessed in November 3, 2021. 5.3.3

[151] P. Alemany, R. Muñoz, J. Martrat, A. Pastor, R. Díaz, D. Lopez, R. Martínez, R. Casellas, and R. Vilalta, "Trust management through blockchain for optical providers and services," *submitted in the Journal of Optical Communications and Networking (JOCN)*, January 2023. 6

[152] 5GCroco Project, "https://5gcroco.eu/." Online. Accessed in January 12, 2022. 7.2.2

[153] CTTC, "https://www.cttc.cat/unico-5g-success-project/." Online. Accessed in November 17, 2022. 7.2.5

[154] Telefonica, "https://www.telefonica.com/en/communication-room/telenor-and-telefonica-demonstrate-network-slicing-with-open-source-mano-osm-at-mwc/." Online. Accessed in Sptember 2, 2022. 8.1

[155] J. Baranda, J. Mangues-Bafalluy, L. Vettori, R. Martínez, and E. Zeydan, "Deploying hybrid network services: Mixing vnfs and cnfs in multi-site infrastructures," in *2021 18th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pp. 1–2, 2021. 8.1

[156] S. Shetty, C. Wang, M. Dohler, D. López, R. Forbes, S. Steiff, T. Faisal, S. Backia Mary B., Q. Liu, and I. Arribas, *ETSI White Paper No.48 - An Introduction of Permissioned Distributed Ledger (PDL)*. Wiley-IEEE Press, 1 ed., 1 2022. 8.1

[157] European Commission's Innovation Radar Team, "https://www.innoradar.eu/." Online. Accessed in October 2, 2022. 8.1

[158] European Commission's Innovation Radar Team, "https://www.innoradar.eu/innovation/41970." Online. Accessed in October 2, 2022. 8.1