ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN

COMPUTER SCIENCE AND ENGINEERING

Ciclo 37

Settore Concorsuale: 01/B1 - INFORMATICA

Settore Scientifico Disciplinare: INF/01 - INFORMATICA

CROSS-DOMAIN NETWORK SERVICE MANAGEMENT AND ORCHESTRATION

Presentata da: Chiara Grasselli

Coordinatore Dottorato

Ilaria Bartolini

Supervisore

Franco Callegati

Esame finale anno 2025

Contents

Abstract

The introduction of virtualization and cloud computing technologies in the telco industry has significantly changed how network services are delivered. The Network Function Virtualization paradigm leverages these technologies to replace physical network appliances with software network functions decoupled from the hardware. Moreover, the emergence of software-defined approaches such as Software Defined Networking and programmable data plane has increased network programmability. Although the combination of these paradigms provides unprecedented flexibility, a seamless orchestration of all network components is required to meet the functional and performance requirements of different types of services. This thesis addresses the management and orchestration of network services with a cross-domain study, presenting the orchestration solutions implemented during the three years of PhD. The work exploits the capabilities offered by the mentioned technologies and studies the benefits and challenges of their application in different scenarios. First, this thesis focuses on 5G network slicing, reporting the design and implementation of a network slice for mission-critical communications. Then, it discusses the orchestration of an industrial network digital twin for cybersecurity testing in a realistic virtualized environment. Finally, it presents a novel orchestration system to deploy secure machine learning pipelines for near-real-time control of network services. Two solutions are considered to secure the communications between the agents composing the pipelines. The former exploits IPsec secure channels using a Distributed Ledger Technology network for key exchange. The latter proposes in-network encryption performed with P4 programmable switches.

Chapter 1

Introduction

Over the last decade, the emergence of paradigms that foster the decoupling between hardware and provided functionality has reshaped the way the network infrastructure is managed and controlled, with software and virtualization taking a prominent role. Two complementary trends underlie this evolution.

On the one hand, the advances in virtualization technologies and the adoption of cloud computing models in the telco industry have opened up the possibility of providing network services in a more agile and cost-effective manner. The Network Function Virtualization (NFV) paradigm leverages these technologies to replace traditional physical network appliances with software "building blocks" called Virtualized Network Functions (VNFs) that can run on Virtual Machines (VMs) or containers on Commercial Off-The-Shelf (COTS) hardware. The European Telecommunications Standards Institute (ETSI) has defined a reference architecture to implement NFV principles in [B1], where the NFV-MANO (Management and Orchestration) components provide new capabilities to operate network services that complement traditional management procedures. The decoupling of network functions from dedicated hardware allows network operators to take full advantage of cloud-based infrastructures to deploy and manage network services with increased flexibility. That ensures a higher degree of automation in the service lifecycle management and the possibility to deploy, scale, and migrate VNFs dynamically. The

relevance of this trend is reflected by initiatives such as the Cloud iNfrastructure Telco Task Force (CNTT) [B2], which designed guidelines for a common "telco-cloud" (i.e., a cloud infrastructure for NFV-based telco applications) to drive this innovation, and the proposal of several platforms implementing NFV capabilities [B3].

On the other hand, the emergence of software-defined approaches that foster a similar decoupling principle for network devices has increased network programmability and reduced the management burden. This second trend revolves around the idea of programming network resources dynamically through software. The Software Defined Networking (SDN) paradigm enables logically centralized and directly programmable network control, allowing for dynamic traffic steering across the infrastructure. SDN achieves this by decoupling the control plane (i.e., the part that controls how packets are forwarded) from the data plane (i.e., the part that physically receives, stores, and forwards the packets) and by providing a unified, open, and programmable interface to control network devices from different vendors. Furthermore, data plane programmability has introduced a novel "top-down" approach that allows direct programming of how data plane devices process packets. This approach provides unmatched flexibility, enabling the creation of custom processing pipelines to perform additional functionalities beyond regular forwarding [B4].

This thesis delves into the trends outlined in this overview and investigates the management and orchestration of network services from different perspectives. Three different application domains are considered in the following chapters. Chapter 2 deals with 5G network slicing, presenting a practical application of NFV and SDN principles to provide a network slice tailored for mission-critical communications in a multi-domain scenario. Based on the previous work, Chapter 3 addresses the application of the NFV-MANO approach to enable the automated provisioning of digital twins to enhance the security of connected systems in modern industrial networks. The aim is to provide a virtualized network infrastructure to perform cybersecurity analysis and validation without interfering with the real production

plant. Chapter 4 discusses the implementation of AI-driven control mechanisms in next-generation mobile networks. Then, it presents a novel orchestration system to deploy secure ML pipelines for near-real-time control of network services.

Chapter 2

5G Network Slicing

Network slicing is a key paradigm of 5G networks. Current technological trends in network virtualization and programmability, along with flexible service management and orchestration procedures, play a pivotal role in implementing its principles. This chapter explores the topic, presenting the design and implementation of a network slice tailored for Mission Critical (MC) communications. As will be covered more in detail in the following section, the MC ecosystem is one of the vertical sectors leaning toward a new generation of services based on 5G technologies. The work reported here integrates the technologies introduced in the previous chapter and aligns with 3rd Generation Partnership Project (3GPP) standards on 5G and MC services. The NFV-MANO approach is applied to orchestrate the slice components on a cloud-based infrastructure spanning different data centers and automate the slice lifecycle management. In addition, a software-defined transport network is proposed to manage the QoS in the interconnection between the network slice sections. The network slice architecture includes all the 5G core network components and 3GPP-compliant MC service elements. The network slice implementation was tested in a private data center using open-source tools and a platform for MC services provided by Leonardo S.p.A, an Italian company active in the defense, aerospace, and security sectors that collaborated in this research activity.

In recent years, many research works in the literature have delved

into the network slicing paradigm, but only a few explore MC services as a field of application. Moreover, although standardization bodies have already defined an architectural framework compatible with 5G networks for this specific domain, related real-life experiments and applications are still in their infancy. How to achieve effective slice management and orchestration is a relevant topic of discussion [B5],[B6]. Some solutions explored in the literature propose a slice-based customization of mobile networks at different granularity levels [B7], and combining the 5G architecture with an NFV-based network store to provide on-the-fly resource reservation, deployment, and slice management that matches end-users demand [B8]. However, further investigation of fine-grained service composition strategies and approaches to implement end-to-end slice orchestration that can guarantee specific performance and functionalities is needed [B9]. In relation to some aspects addressed in this chapter, the virtualization of mobile core networks and the effects of user/control plane separation have been studied before, but in a different application scenario [B10]. Among the works exploring the mission critical scope, complementary approaches have been proposed that focus on slicing the Radio Access Network (RAN) segment [B11],[B12].

The work presented here aimed to contribute to this research field and to bridge the gap between standards and real-life experiments with a comprehensive study covering the design and subsequent demonstration of the proposed network slice architecture. The results presented here have been published in [P2].

2.1 Network slicing for Mission Critical communications

5G networks are expected to serve efficiently heterogeneous vertical applications [B13]. The increasing diversification of QoS requirements makes the traditional "one-size-fits-all" approach no longer feasible, demanding a more flexible infrastructure and service management. Network slicing has emerged as a key paradigm to address this is-

sue, by enabling the coexistence of multiple logical networks (i.e., the network slices) on the same physical infrastructure. Several standardization bodies and industry associations, such as the ITU-T [B14], the NGMN Alliance [B15], the GSMA [B16], and the 3GPP [B17], have discussed the main principles of this paradigm. Apart from the different focus of the organizations that drafted these documents, the underlying concepts are the same. Each network slice can have distinct characteristics and is tailored to meet the requirements of a specific use case. Complete isolation between different slice instances must be ensured so that they do not interfere with each other.

End-to-end network slices span the access, transport, and core network segments. As discussed before, a seamless creation of separated logical networks across these main segments builds on top of network virtualization and programmability technologies. NFV and SDN paradigms are considered key enablers of the network slicing concept [B5],[B6]. For instance, the NFV-MANO approach allows to dynamically orchestrate the slice components as VNFs, taking advantage of virtualization technologies and a cloud-based infrastructure. At the same time, software-defined technologies enable to create overlay network sections in the transport segment that ensure traffic flow isolation and QoS policy enforcement.

The Mission Critical ecosystem is one of the vertical sectors that can benefit from current technological trends. MC communications play a key role in serving Public Protection and Disaster Relief (PPDR) forces (e.g., police officers, firefighters, and first-aid teams) during their operations in critical scenarios, representing a strategic asset for any nation. MC networks must guarantee reliable, low-latency, and priority-access communications even in case of disruptive events, such as large-scale emergencies or natural disasters, during which commercial communication systems may fail. For this reason, MC services traditionally rely on dedicated networks and radio access spectrum, using legacy narrowband technologies such as TErrestrial Trunked RAdio (TETRA), Tetrapol, Digital Mobile Radio (DMR), and Project 25 [B18]. These technologies ensure effective voice communications

but limited data throughput for advanced services. Therefore, PPDR organizations have started to complement the MC offering with enhanced data services delivered over dedicated or mobile operators' broadband networks, with the two options entailing different control and costs [B19]. A critical aspect of using commercial mobile networks is implementing differentiated priority and pre-emption mechanisms to ensure service continuity to PDDR forces by overriding lower-priority users when needed [B20]. Running MC services over a dedicated network is therefore the safest option in terms of security and control, but results in very high initial capital expenditure costs as well as continuous support and maintenance costs.

The 3GPP standards define the architectural framework to deliver MC services over LTE and 5G [B21]. This set of services, including Push-To-Talk (PTT), Data, and Video services, is usually denoted by Mission Critical Everything (MCX). In particular, given the architectural evolution telco infrastructures have been undergoing in recent years, economic and technological factors have opened to a new generation of MC systems based on 5G state-of-the-art technologies. The integration of 5G network slicing in the MC ecosystem promises to offer a solution to the performance and network isolation problem, at a lower cost compared to deploying and maintaining a dedicated infrastructure. In fact, operators could instantiate a dedicated network slice customized to fulfill the specific functional and performance requirements of this use case. The slice instance could be activated across different data centers or, in general, wherever COTS hardware with enough computing resources is available. Moreover, a cloud-based network infrastructure promises to be more flexible with the possibility to scale, migrate, or redeploy the virtual components as needed. All these aspects make the delivery of services more agile in case of disruptive events, an important characteristic for mission critical scenarios.

2.2 Network slice design

The network slice considered in this work is split into four logical sections:

1. access network, either mobile or fixed;
2. edge network components located in an Edge Data Center (DC), virtualizing the user plane part of the mobile core network and the MC proxies for the exchange of the media data flows (voice, video, etc.);
3. core network components located in a Core Data Center, virtualizing the control plane part of both the mobile core network and of the MC system;
4. interconnection network between the data centers, which could be either a public network or a private geographical interconnection.

Figure 2.1 shows a high-level view of the network slice. This architectural design supports the separation between the control and data plane and the distribution of the key network slice components where they best fit the purpose. The core network control and user plane functions are fully separated, according to the 5G Control User Plane Separation (CUPS) principle. Therefore, the network signaling is logically separated from the user traffic transported in the network for service provisioning. Moreover, the user and control plane separation is also applied at the MC service level to keep the media servers as close to the user as possible to provide better communication performance.

The implementation poses some interesting challenges, most notably:

- multi-data center and possibly multi-domain orchestration;
- traffic management for QoS guarantee in the transport network;
- cross data center applications and traffic management.

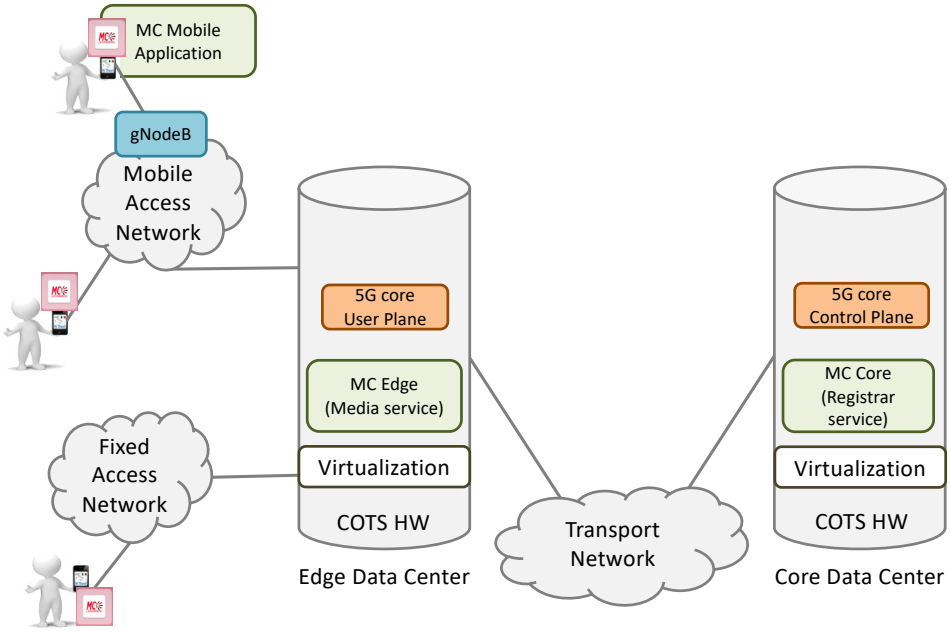

Figure 2.1: The high-level architecture of the network slice for MC services with the related main components.

These challenges require a network slice design properly tailored to support them, as we will discuss later. Before proceeding, however, it is important to recall some concepts that will be used extensively in the rest of this chapter.

2.2.1 Actors and roles

The network slicing approach involves three main types of actors.

- Infrastructure Providers or infrastructure owners: they own the infrastructure and provide all the infrastructural management actions. A single network slice may span multiple infrastructure domains. Therefore, its deployment and lifecycle management require interactions with each Infrastructure Provider involved.
- Network Slice Provider: the provider of the communication service implemented with the network slice.
- Network Slice Customers: the users of the communication ser-

vice.

According to their respective roles, these actors must have different rights, with the Infrastructure Providers and the Network Slice Provider having specific management roles to keep the infrastructure and the service up and running.

In our case, an Infrastructure Provider can be identified as a mobile network operator, a mobile virtual network operator, or a public body operating the infrastructure for the PPDR forces. At the same time, the Network Slice Provider is the entity that directly manages the mission critical communication services. Depending on the organizational model chosen, this may be a public body serving all the various PPDR forces of the country or a specific body inside a PPDR force (e.g., firefighters, police officers, etc.). Consequently, the Network Slice Customers are the PPDR forces that will use the service for communication.

From the brief discussion above, it follows that the organizations acting as Infrastructure Providers and Network Slice Providers might be different from case to case, either being closely bound to each other or just linked by a conventional commercial agreement. Therefore, the slice architecture must be very flexible to adapt to these diverse organizational models, ensuring a seamless co-existence of these actors while providing all of them with the required functionalities. For example, management is an important issue for both infrastructure and service providers, since no service can be properly set up or guaranteed in real production environments without management capabilities. This is considered in the NFV-MANO framework, where the management components are clearly outlined. Specifically, it is assumed here that:

1. the Infrastructure Provider must have management access to the whole infrastructure, including all the VNFs, to be able to interact with the various components active in the cluster whenever some high-level general configurations or recovery actions are needed;

2. the Network Slice Provider must have management access to its own infrastructure and VNFs to implement all the management

Table 2.1: Example of characterizing NEST parameters for the MC communications network slice

ATTRIBUTE	**VALUE**
Coverage	Local (Outdoor)
Guaranteed Downlink Throughput per Network Slice	391600 (391.6Mbps, band 3, channel 20MHz(100RB), 256QAM, 4x4MIMO)
Mission Critical Support	1: mission critical
+ Mission Critical Capability Support	1: Inter-user prioritization, 2: Pre-emption, 3: Local control
+ Mission Critical Service Support	1: MCPTT, 2: MCData, 3: MCVideo

actions related to the production phase of the service, including modification of the VNF configurations, performance monitoring, etc.

2.2.2 Network slice description

A correct interaction between all the actors mentioned above has to be guaranteed. The GSMA standard specifies how to describe the characteristics of each network slice in a standardized way, starting with the Generic Slice Template (GST) [B22].

The GST can be used to describe a network slice type. It is a dictionary containing common slice attributes, such as supported throughput/functionality and provided Application Programming Interfaces (APIs). Once the GST is filled with values based on specific vertical use cases, it gives birth to the NEtwork Slice Type (NEST), which can be used by vendors, vertical industry customers, and network operators to reach their objectives. Table 2.1 shows an example of NEST for the network slice type considered here. Once the NEST is available, it has to be translated into a description that allows the real-life implementation of the network slice. Neither the GST nor the NEST specifies the steps required to achieve this. The collection of all the technical details that are necessary to implement a particular net-

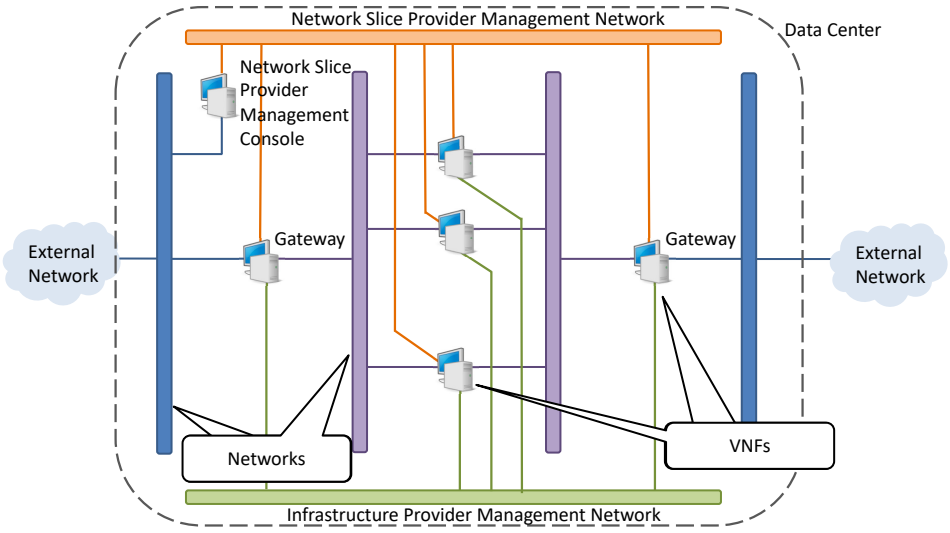

Figure 2.2: The network slice blueprint for a single data center.

work slice is usually referred to as the slice *blueprint*. This description depends on the technological approach taken by each Infrastructure Provider and it is not standardized.

The general slice blueprint designed to match the high-level implementation goals and the NEST specifications is described in the following subsection.

2.2.3 Network slice blueprint

At first, we designed and tested a network slice blueprint that was quite general, to be sure it was suitable to meet all the requirements mentioned above in terms of architecture, role splitting, and performance characteristics. We started by considering a single data center and designed the blueprint plotted in Figure 2.2. In the figure, the horizontal or vertical bars represent virtual networks defined in the data center, whereas the computer icons represent VNFs. This blueprint aims at satisfying the following characteristics:

- separate management networks for the Infrastructure Provider and the Network Slice Provider;

- isolation and protection of the VNFs providing the required func-

tionalities, avoiding the direct exposure of their network interfaces to external networks;

– maximum flexibility of interconnection between the VNFs composing the service.

We introduced two separate management networks since the Infrastructure Provider must be able to talk to all its customers (tenants) at once, whereas the Network Slice Provider, acting as a tenant of the Infrastructure Provider, must be able to talk to its dedicated infrastructure only, isolated from those of other tenants. Therefore, two different management networks were implemented:

– the *infrastructure management network*, set up at system start-up, devoted to the Infrastructure Provider, and shared among all tenants;

– the *tenant management network*, set up as part of the network slice, seen only by the Network Slice Provider running the slice.

This general architecture can be composed to create network slices spanning across multiple data centers, according to the schematic presented in Figure 2.3. These data centers might belong to the same provider or different ones. Regardless of that, this should be transparent from the Network Slice Provider's point of view, given the existence of the interfaces required for these interactions.

The basic idea of this design is the following: production VNFs run inside the data center, connected to two different management networks, the former devoted to the Infrastructure Provider and the latter to the Network Slice Provider. Moreover, a slice-specific management console connected to the management network is provided to the Network Slice Provider, thanks to which it can manage the slice components directly from the data center where they are deployed.

The VNFs of a slice section, like the one depicted in Figure 2.2, are not directly connected to the data center networks providing access to the outside world, but there are gateways in between. This choice is motivated by two main reasons:

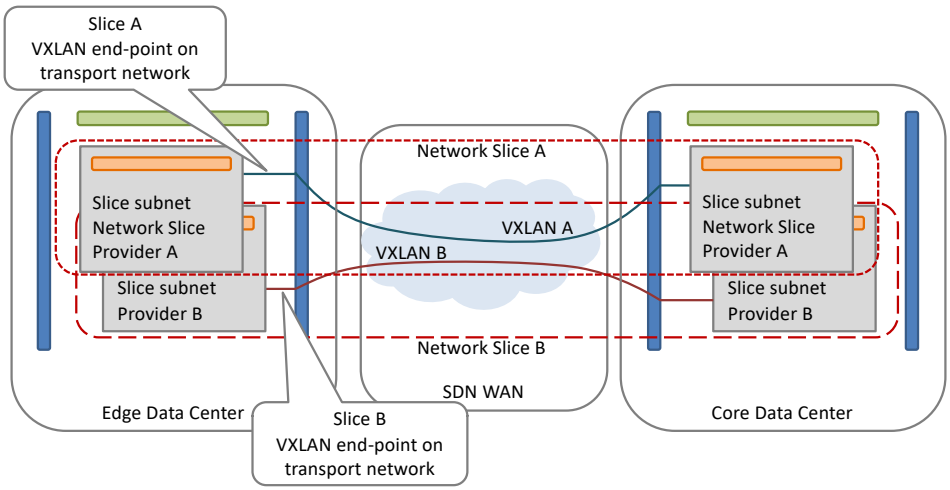

Figure 2.3: Full network slice spanning two data centers. In this figure an example with two slices deployed in parallel is shown to provide a better understanding of the different management infrastructures for Infrastructure Provider and Network Slice Provider.

- security: the network gateway provides the required traffic isolation and acts like a firewall protecting the production section of the slice;
- functionality: the network gateways can work as end-points of tunnels (in this example VXLAN tunnels) providing an overlay network between the involved data centers, thus allowing seamless slice management over different sections, even belonging to distinct providers.

2.2.4 Network slice delivery and lifecycle management

The management and orchestration of the network slice during its entire lifecycle is a crucial issue to allow effective usage of this technical approach. 3GPP and ETSI describe the lifecycle of a network slice in their documents [B23] and [B24], identifying all the steps required to provide the performance requirements for the network slice design, as well as to instantiate, run, and terminate it. The steps implementing

the complete network slice lifecycle management are depicted in Figure 4.3.1.1 in [B23]. The overall process consists of two main phases: a *preparation phase* containing the description of the network slice blueprint and the preparation of its run-time environment, and a *lifecycle management phase* where the network slice instance is created, run, and eventually terminated. The former phase is a matter of the Infrastructure Providers, which will prepare all the necessary components based on the NEST provided by the Network Slice Provider and the chosen blueprint. In our case, it refers to the networks in the cloud platform that must be shared between slices and must exist before the single network slice instance is started. In particular, these are:

- the management network of the Infrastructure Provider, which will be connected to the parts of the network slice that the provider has to control to handle some emergency event (either collaborating with or overriding the management actions from the Network Slice Provider);
- the inter-DC interconnection network;
- the physical interconnection to the access networks, either mobile or fixed.

Furthermore, the NEST and the slice blueprint are translated to a set of Network Slice Templates and/or Network Service Descriptors, which are then onboarded in the orchestration platform. These descriptive files represent the list of VNFs and their interconnections for each slice segment (e.g., for each Infrastructure Provider domain), adopting a language understandable by the NFV-MANO system. On the other hand, the latter phase involves the Network Slice Provider that can start, run, modify, monitor, and stop the network slice at will, using the interfaces provided by the Infrastructure Providers or through the native interfaces of the applications deployed in it.

The testbed implemented in this work follows this paradigm. But before proceeding with the description in the next section, it is relevant to introduce an approximation adopted for the testbed realization. To

simplify the deployment process, we assumed having a single Infrastructure Provider offering two data centers, one at the edge and the other at the core. Nevertheless, the same considerations made for the blueprint description and slice lifecycle management hold.

2.3 Network slice implementation

Following the general description discussed in the previous part of the chapter, this section goes on to describe the approach employed to build the slice and the system supporting it. To this end, we will also introduce the software tools chosen to implement the proposed system.

To support the performance requirements of the service, the network slice is split into access and core parts, the former hosted in the Edge DC and the latter in the Core DC. The actual implementation of the access and core parts of the network slice are plotted respectively in Figures 2.4 and 2.5. For the sake of readability, the connections of the various VNFs with the Infrastructure Provider and Network Slice Provider management networks (green bar at the bottom and orange bar at the top, respectively) are omitted, but they follow the general blueprint template in Figure 2.2. The motivation behind this design choice is to place everything user-related as close as possible to the user itself, with the aim of reducing the load in the network core and improving the performance (e.g., by reducing the latency) for the end-users. As mentioned before, it is also in line with the CUPS principle.

The VNFs of the access section in Figure 2.4 are the User Plane Function (i.e., the packet forwarder for the 5G data plane) and the MCX edge component acting as a media server, forwarding media streams from and to users. The core section in Figure 2.5 is simpler since there is no “transit” traffic and a single internal interconnection network is enough. The specific VNFs are all the components of the 5G control plane and the MCX control element, which acts as a registrar server for the MC applications, managing user registrations and their

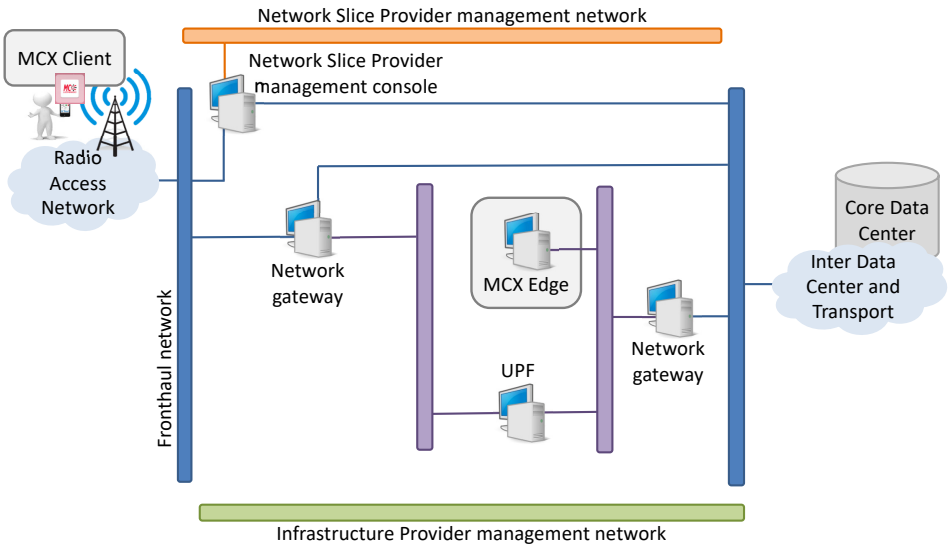

Figure 2.4: Architecture of the access section of the network slice in the Edge DC.

communication profiles. It is worth noting that the slice architectures presented here are just a graphical sketch. In practice, following the NFV-MANO architecture [B1], each of the VNFs is actually deployed as a pair of virtual machines: the former for production and the latter for management, with an additional network in between to connect them.

Regarding the tools selected for the implementation, we first introduce the platforms for managing the infrastructure supporting the proposed architecture (i.e., the tools of the Infrastructure Provider), and then we present the software components running inside the slice VNFs. Following the directives proposed by the CNTT group, we chose OpenStack [B25] as the cloud management platform for the two data centers. Specifically, we deployed the Stein release with Kolla Ansible in both data centers. Then, to orchestrate the virtual functions of the slice over these cloud-based infrastructures, we opted for Open Source MANO (OSM) [B26], the open-source project backed by ETSI that implements a standard-compliant NFV-MANO platform. In particular, we used OSM Release 10 with descriptors following the ETSI SOL006 specifications [B27]. Finally, to emulate the behavior

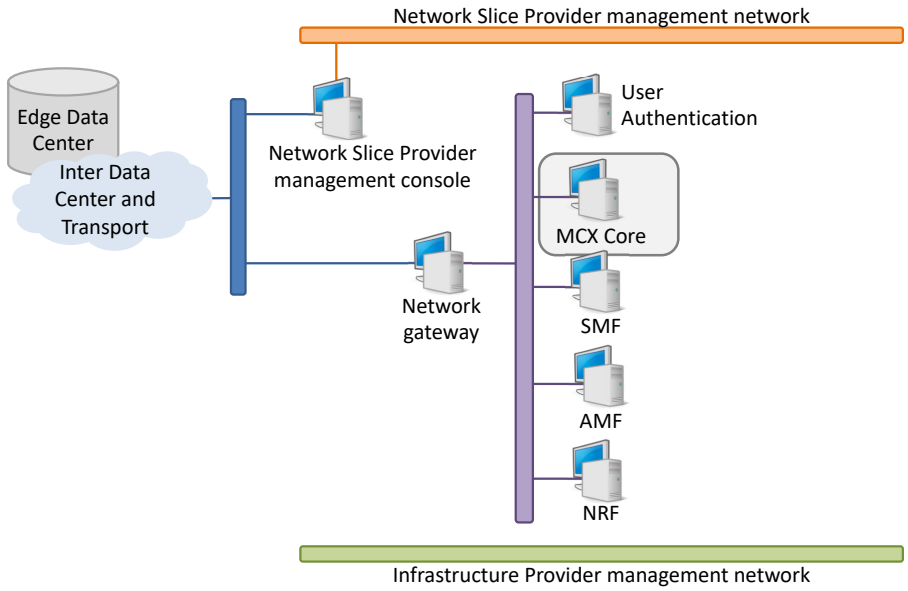

Figure 2.5: Architecture of the control plane section of the network slice in the Core DC.

of the transport network between the two data centers, we introduced emulated delays in the outgoing interfaces connected to this network with the Linux Traffic Control (*tc*) utility, which allows manipulating packet transmission settings in the kernel. In addition, we added a single SDN-enabled Open vSwitch [B28] switch controlled by the ONOS [B29] controller acting as the WAN Infrastructure Manager (WIM).

Following the slice lifecycle presented in Subsection 2.2.4, after the network environment preparation phase, we designed and onboarded the descriptors and configuration files required by the NFV-MANO platform. The implementation of the network slice considered in this work is rather complex; for this reason, the deployment was split into three steps to make the configuration and debugging process more controllable. Three NFV-MANO descriptors have been designed for the two sections of the slice. Descriptors can be reused and some are common to the various slice sections. Therefore, the network slice blueprint is represented by the complete set of these descriptors and related configuration files (called packages).

Specifically, the descriptors provide to the NFV-MANO platform all the information about:

1. the VNF packages to be run in the slice;
2. the interconnections between them (Virtual Links in NFV-MANO terminology), described in the Network Service Descriptors (NSDs) and Virtual Link Descriptors (VLDs);
3. the Network Slice Template (NST) as a combination of Network Service Descriptors;
4. the details of the Virtualized Infrastructure Managers (VIMs) where the network slice has to be instantiated;
5. the VNF Forwarding Graph Descriptor (VNFFGD), specifying the traffic path from one VNF to another, which has to be implemented in the network slice.

Recalling the blueprint description (Subsection 2.2.3), the Infrastructure Provider management network is already part of the cloud environment, even before the deployment of the slice. Therefore, the first step of the slice deployment phase is the creation of the Network Slice Provider management elements. This initialization deploys the Network Slice Provider management network and the management VNF, which is connected to the data center external networks and the Network Slice Provider management network. Moreover, the Network Slice Provider management VNF can automatically create an overlay network (e.g., VXLAN) on top of the inter-DC network, allowing a seamless interconnection between components deployed in different data centers. In the second step, the NFV-MANO triggers the deployment of the 5G core network elements based on the Open5GS [B30] software package. It is an open-source implementation of a hybrid 4G/5G core network compliant with 3GPP Release 16. Finally, the orchestrator instantiates the control and data plane elements of the MCX application provided by Leonardo S.p.A.. More details on this will be given in the remainder of this section.

2.3.1 Mission critical components

As introduced at the beginning of the chapter, the MC services were deployed with a product provided by Leonardo S.p.A. The Leonardo Mission Critical Services is part of the Leonardo Communications Service Platform product family [B31]. It is a complete Mission Critical solution compliant with 3GPP MCX standards that extends the portfolio of standard solutions for PPDR communications, ranging from DMR to TETRA technologies, with next-generation broadband capabilities. It offers Push-to-Talk communication, enhanced with voice, video, multimedia chat, and a set of APIs for third-party application development. It can be deployed over both commercial and private mobile networks and can provide users with advanced functionalities such as:

- instantaneous group and private high-quality voice communications;
- mobile broadband multimedia applications (real-time video streaming, multimedia messaging, file/video/photo transfer, database access);
- location-based services;
- emergency, man-down/immobility, and Land Mobile Radio standard interaction via InterWorking Function for augmentation of traditional systems [B32].

The complete solution to provide MCX services is composed of the following components:

- An Android client designed for on-field operations with a complete set of functionalities providing all the MC service implementations as per the 3GPP standard, namely MCData, MCVoice, and MCVideo. It can be installed on off-the-shelf smartphones, as well as on ad-hoc terminals, with a fully customizable Human-Machine Interface. It can be customized to provide a differentiated User Experience, ranging from a traditional push-to-talk

radio to a multimedia client similar to a conventional smartphone.

- A web-based dispatcher providing control, monitoring, and management of the operations of the teams.
- A dedicated interface for the management and monitoring of the platform KPIs.
- A Session Initiation Protocol (SIP) Core server for user registration, location, and authorization, as well as for call signaling management as per the 3GPP standard.

The SIP core is a cloud-native platform designed to be deployed either as a virtual machine or as a containerized application. It also supports a full separation of user and control planes, according to the aforementioned 5G CUPS principle. In particular, the registration server used for signaling can be decoupled from the media servers, which will manage and deliver the media streams. Moreover, the internal SIP Core component can be easily plugged "in" and "out" at run-time by using the MCX dashboard. External IP Multimedia Subsystem (IMS) core servers are supported for large-scale deployments.

In this work, we took advantage of the control and user plane separation offered by the mission critical components by deploying them in the core and edge data centers, respectively. In detail, the "MCX Core" in Fig. 2.5 refers to the web-based dispatcher and the SIP components in charge of registration and signaling, while the "MCX Edge" in Fig. 2.4 represents the external media server used to exchange users' voice or video messages. This choice allows us to keep the media servers as close as possible to the final users, thus guaranteeing optimal performance in line with the 5G edge computing concepts.

2.4 Experimental validation

All the experiments were run in a private data center with two separate OpenStack clusters, one for the Edge DC and the other for the

Core DC. Each cluster consists of two physical servers, equipped as follows: 64 GB of RAM, 40 CPUs, 1.2 TB of disk, 1 Gbit/sec interfaces, and Ubuntu 18 LTS as OS. We emulated the 5G Radio Access Network (RAN) elements with UERANSIM [B33], both the gNodeB base station and the 5G User Equipments (UEs). Given the limited laboratory setup, the results presented here were not meant to assess the absolute performance achievable with the proposed approach. It is expected that a more powerful configuration will likely lead to better performance overall. Instead, the experimental phase was aimed at demonstrating the feasibility of the proposed approach and investigating any performance issues and critical bottlenecks that might occur at the various stages of the slice deployment.

2.4.1 Network slice instantiation

The first aspect that was evaluated is the time needed to create a new network slice instance based on the proposed architecture. In particular, we measured the time required to instantiate all the network slice components. As explained in the previous section, the process is split into three phases:

1. initialization of the Network Slice Provider management infrastructure in the data centers, with the dedicated management network and consoles;

2. deployment of the 5G mobile core network and gateways, with all the required components split among the edge and the core data centers;

3. provisioning of the service to the Network Slice Customers, running the MCX edge and core components respectively in the edge and core data centers.

For each phase, ETSI MANO SOL006 [B27] compliant descriptors were implemented and onboarded in the OSM platform for the deployment in each OpenStack cluster.

Table 2.2: Number of virtual components of the network slice instantiated in each phase.

	VNFs	Virtual Machines	Virtual Networks
Init. Core Data Center	1	1	1
Init. Edge Data Center	1	1	1
5G core net. Core Data Center	5	10	6
5G core net. Edge Data Center	3	6	5
MCX Core Data Center	1	1	0
MCX Edge Data Center	1	1	0

As reported in Table 2.2, in terms of virtual components, in the first phase, we instantiate 1 VNF on a single virtual machine and 1 virtual network per data center for the Network Slice Provider management infrastructure. During the second phase, we activate in the Edge DC 3 VNFs running on 6 virtual machines (one for the management and one for the service functionalities) and 5 virtual networks for the 5G mobile core and the gateways components at the edge. Instead, in the Core DC, we instantiate 5 VNFs on 10 virtual machines and 6 virtual networks for the other slice components at the core. In the third phase, we run 1 VNF on a single virtual machine per data center for the MCX services.

We performed ten instantiation experiments, measuring the time required to complete the various deployment phases in the edge and core data centers. Table 2.3 reports the average time needed to instantiate the network slice components at every step and the related two-tailed 90% confidence interval computed assuming a Student's t distribution. The table also shows the values of the coefficient of variation, calculated as the ratio of the standard deviation to the average estimated from the experimental results. The second phase is more complex than the others since it involves a larger number of virtual components. So, as expected, it takes more time for deploy-

Table 2.3: Average time required by the orchestration system at each stage to instantiate the various components of the network slice. The values are averaged over the results of 10 different experiments. The lower and upper bounds of the 90% confidence interval (min 90% and max 90%, respectively) and the Coefficient of Variation (CV) are reported along with the estimated average.

	Average	Min (90%)	Max (90%)	CV
Init. Core DC	49.9 s	45.72 s	54.08 s	0.14
Init. Edge DC	63.6 s	59.95 s	67.25 s	0.10
5G core net. Core DC	404.8 s	394.36 s	415.24 s	0.04
5G core net. Edge DC	273.3 s	267.41 s	279.20 s	0.04
MCX Core DC	75.5 s	67.23 s	83.77 s	0.19
MCX Edge DC	64.9 s	58.95 s	70.85 s	0.16

ment. The other two phases are of similar complexity and involve a smaller number of components than the second phase. Therefore, their instantiation requires a shorter time.

The variability of the measured values is due to the fact that we run the experiments on physical servers in a realistic cloud environment. Even though there are no other active network slices, the servers still run the basic management tasks required by the cloud management platform. These tasks share the CPU with all the others and introduce some random delay in the execution of the slice instantiation. Intuitively, this sort of "white noise" in the measurements should affect the short tasks more, while it should be less evident in longer ones. That is what happens in practice. It is possible to see in Table 2.3 that the confidence interval, with respect to the average time, is approximately between 15% and 20% for the phases that require less time, while it is around 5% for the second phase, which takes more time. Similarly, the coefficient of variation values show greater variability in instantiation times in the shorter phases than in the second phase, with values ranging from 10% to 19% for the former and around 4% for the latter.

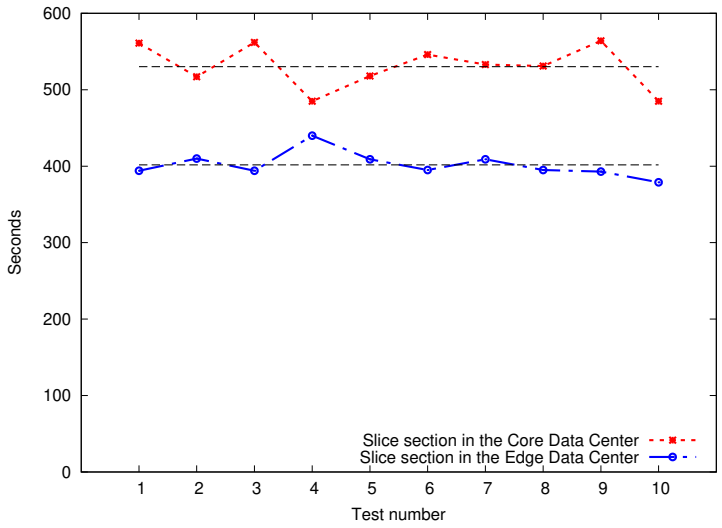

Figure 2.6: Total time required to instantiate the full network slice. The plot shows 10 measurements taken from 10 different experiments on the same infrastructure. The horizontal dashed lines represent the average values.

In Figure 2.6 we also graphically report the total time needed to instantiate the edge and core network slice sections in each of the ten experiments and their averages. The figure shows that the complete network slice instance can be deployed in a few minutes. There is some variability in the measured times, as discussed above, but the reported averages provide a rather clear indication of the values we are facing. Given the scenarios in which MC services are used, the PPDR communication infrastructure supported by the considered network slice should run for a reasonably long time, from a few days in the case of an ad-hoc deployment because of a specific emergency to months or even years for a stable deployment at the national level. Therefore, an initialization time of the whole slice within a few minutes is reasonable. This is also true in case of failure or infrastructure disruption. If the images of the various virtual network functions are available, the entire network slice can be restarted in a reasonably small amount of time at a different location.

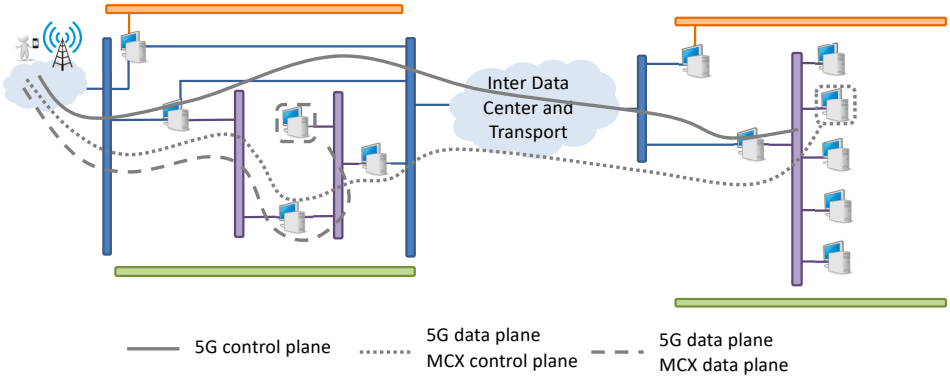

Figure 2.7: Overall scenario and an example of the data flows showing the separation between the control and data plane of the 5G network as well as of the MCX infrastructure.

2.4.2 MCX service delivery

The network slice implementation was also evaluated from the MC service perspective, testing the correct functional split of the MC components and the split effect on the service delivery. The Mission Critical service delivery scenario and the paths of the various traffic flows are shown in Figure 2.7. From the SIP Uniform Resource Identifier (URI) point of view, the domain is called `test` and two UEs are registered as `user1@test` and `user2@test`. As discussed, the testbed guarantees separation between the control and data plane. That is true at the 5G level, as the standard implies, but also at the MC service level since the core MCX server is dedicated to handling signaling traffic, such as SIP registration and call set-up messages, while the edge MCX server acts only as a media server. Therefore, the signaling for service and call set-up follows a different path than the data flows carrying the communication media streams.

Coming to the experiments, we tested at first the correct functional splitting of roles of the two MCX servers according to the planned split of workloads. Figure 2.8 shows the flow of an MCVIDEO call from the point of view of the caller (`user1@test 10.250.123.101`) and of the callee (`user2@test 10.250.123.102`). The two packet sequences shown in the figure were obtained by capturing the packet traffic with

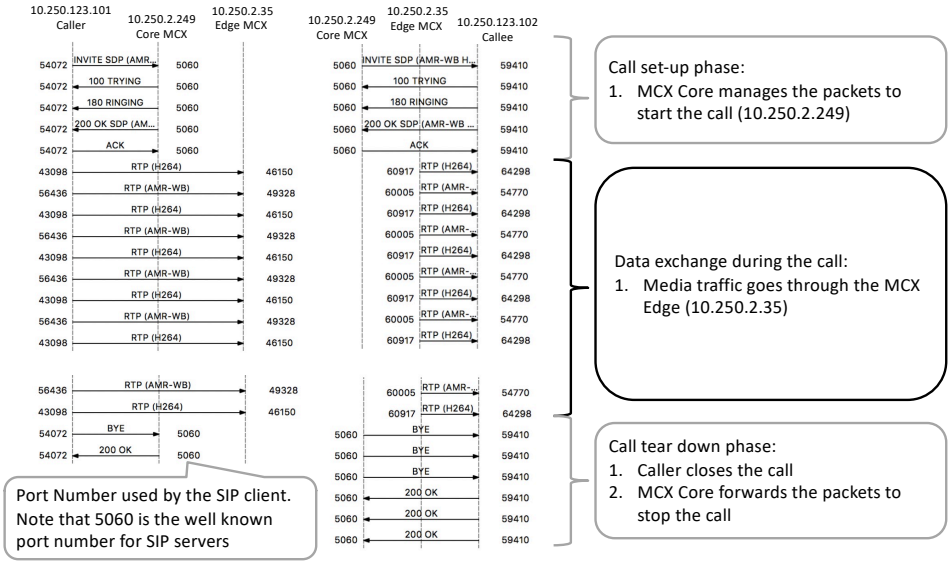

Figure 2.8: Packet flows of an MCVIDEO call, capturing the traffic on the caller (`10.250.123.101`) and on the callee (`10.250.123.102`).

Wireshark. The core MCX is located at `10.250.2.249`, while the edge MCX is located at `10.250.2.35`. The figure shows that the split of roles is implemented correctly. As planned, the SIP traffic required to set up and close the multimedia call between the two users goes through the core MCX. All SIP signaling messages such as INVITE, TRYING, and RINGING flow to and from the core MCX server (`10.250.2.249`). Instead, the Real-time Transport Protocol (RTP) media traffic flows to and from the edge MCX server (`10.250.2.35`). Then, to prove the effectiveness of the Control User Plane Separation approach, we exploited a built-in feature of the MC mobile app. This feature provides a series of evaluation tools for measuring network latency and capacity, as shown in Figure 2.9 and Figure 2.10.

To emulate a higher latency when connecting to the core infrastructure, we forced a delay of $T = 200$ ms on the inter-DC connection by setting up the Linux traffic control on the interface towards the transport network of the Core DC network gateway. We asked the app to register on both the MCX core and the MCX edge. The MCX core is the only one that allows the registration of a SIP user since it is the only one running the control functions. When we ask the MC app

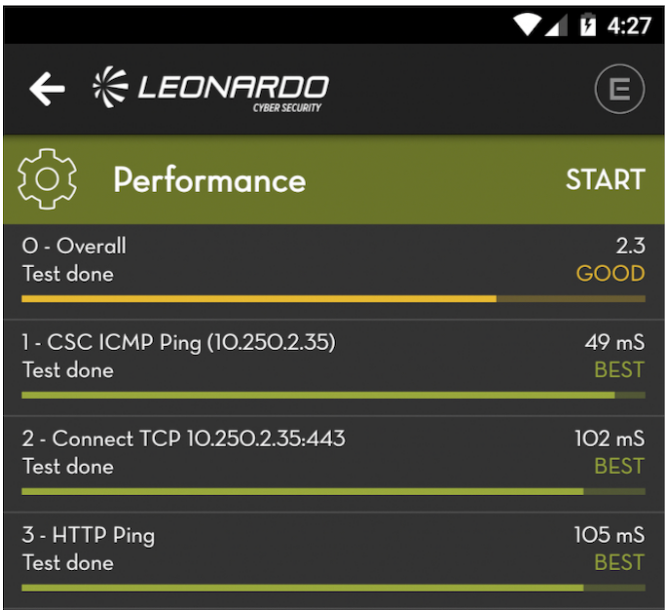

Figure 2.9: Screenshot of the MC smartphone application executing performance measurements while communicating with the MCX in the edge.

to register on the MCX edge, acting only as a media server, the registration is not successful, but the app still allows the execution of the performance test, albeit in a limited way. As a consequence, the two screenshots are different. For the scope of this research, the relevant fields to compare are: `2. CONNECT TCP` and `3. HTTP PING`. The former reports the time required to complete the three-way handshake of TCP between the Android application user and the MCX server. The latter reports the time taken to complete an HTTP request from the user to the server. The values obtained depend on the Round-Trip Time (RTT) of the data connection. We can see that both fields are approximately 200 ms larger in the connection towards the MCX core than to the MCX edge. That is perfectly in line with the additional latency introduced in the path towards the Core DC, which is 200 ms in this experiment. Therefore, we can conclude that in the case of a real call the RTT of the media flows (voice and video) would be significantly lower than the RTT of the signaling towards the MCX in the core. This is one of the expected advantages of the CUPS approach.

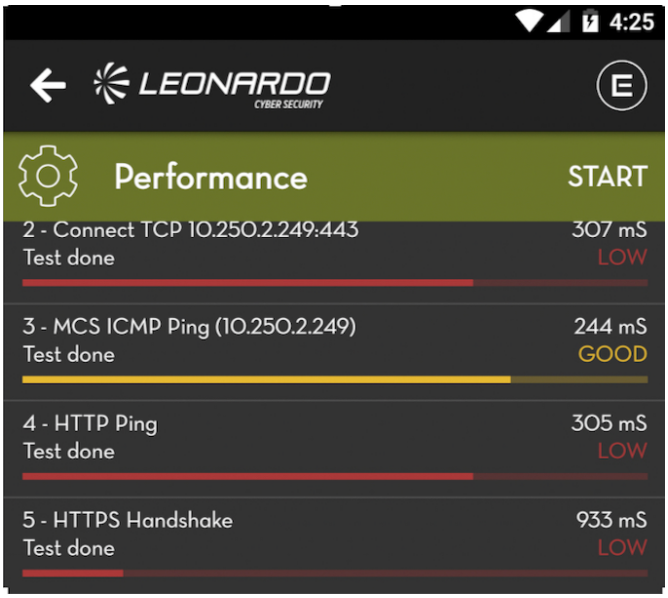

Figure 2.10: Screenshot of the MC smartphone application executing performance measurements while communicating with the MCX in the core.

2.4.3 Inter-DC QoS management

Finally, we evaluated the QoS management in the inter-DC connection. The QoS management in the interconnection network between the data centers will depend on the features made available by the network owner and/or provider. In this work, we assumed Software Defined Networking capabilities to test the possibility of managing the QoS in an integrated way with the network slice management.

The scenario considered is again the one sketched in Figure 2.3:

- network slices A and B are deployed, serving customers $C1$ and $C2$ respectively;
- the network slices are split in two sections and share an interconnection link at 10 Gbit/s;
- the transport link determines the end-to-end bandwidth availability (given that inter-VNF bandwidth inside the same data center is typically larger);

- $C1$ and $C2$ negotiated the following service level agreements:
  - Slice A minimum guaranteed interconnection capacity $C_A =$ 1 Gbit/s
  - Slice B minimum guaranteed interconnection capacity $C_B =$ 3 Gbit/s

The interconnection network is emulated as a virtual switch (implemented with Open vSwitch) controlled by the ONOS SDN controller. We set up a token bucket mechanism to enforce the negotiated QoS for traffic flows belonging to the network slices A and B. In the switch, token bucket queues at the minimum guaranteed capacity of the network slices were implemented with a higher priority over a standard FIFO queue used by other traffic flows. We used the Linux Traffic Control utility to configure the rules to shape the bandwidth as requested, setting as one of the parameters provided to *tc* the guaranteed bandwidth equal to the target capacity negotiated for the slices. In addition, traffic forwarding rules were set by exploiting the ONOS intent framework [B34], forcing the switch to push the packets of the two slices into their specific queue. With reference to the slice lifecycle presented in Subsection 2.2.4, these forwarding rules can be prepared during the network slice design and can be instantiated when the network slice is created. The QoS control is reactive and safeguards the minimum required bandwidth of the slices when a network overload happens. We forced these overload events by generating a very high bandwidth background traffic into the interconnection link.

In Figures 2.11 and 2.12, we show results proving the effectiveness of the QoS management strategy. In both cases, 2 minutes (120 seconds) of communication are shown. Slices A and B generate traffic trying to saturate the available bandwidth. The greedy background traffic causing the link overload is at 10Gbit/s and causes link congestion with high traffic losses in two different ways, as described below. In Figure 2.11, the background traffic starts when the one generated by slices A and B is already active with an almost even share between them. When the background traffic starts and congestion arises, the

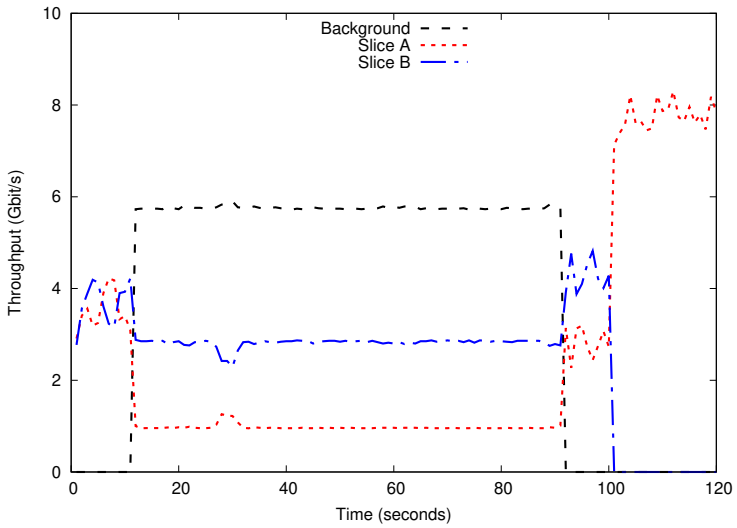

Figure 2.11: Guaranteed bandwidth per slice interconnection when a background traffic able to saturate the link is applied. The background traffic starts when the inter-slice traffic flows are already established.

traffic control strategy safeguards the minimum guaranteed bandwidth requested by the two network slices. After 90 seconds, the background traffic stops and the two network slices can take over. Finally, when slice B stops, slice A can consume the whole link capacity. In Figure 2.12 the background traffic is already active and saturates the link for the entire duration of the experiment. Instead, slice B and slice A start generating traffic at about 10 s and 20 s, respectively. As before, the traffic control strategy throttles the background traffic to enforce the minimum capacity required by the two slices.

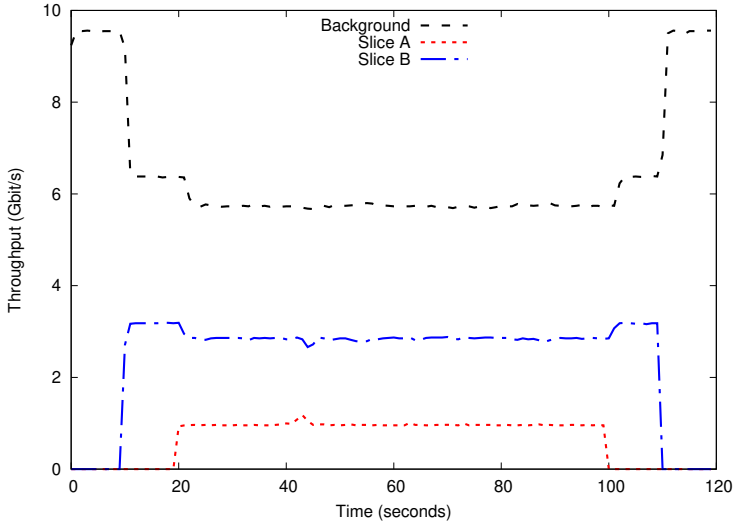

Figure 2.12: Guaranteed bandwidth per slice interconnection when a background traffic able to saturate the link is applied. The inter-slice traffic flows are started sequentially when the background traffic is already established.

Chapter 3

Digital twin orchestration

The previous chapter addressed the application of NFV orchestration principles to network slicing and demonstrated the automation of the entire lifecycle management of a network slice instance. Building on this research work and related insights, this chapter presents the application of the NFV-MANO approach to enable the automated provisioning of digital twins to enhance the security of connected systems in modern industrial networks. The digital twin replicates the industrial network infrastructure in a realistic virtualized environment that can be used as a cyber range to perform cybersecurity assessment and countermeasure validation without interfering with the real industrial plant. In the proposed approach, the lifecycle management of the digital twin follows the one defined for network slices. Moreover, the same orchestration techniques from the previous chapter are applied to deploy and configure the digital twin components in a flexible and automated manner on a cloud platform. Following the proposed methodology, the work reports a proof-of-concept implementation of a digital twin of a typical industrial network infrastructure that includes network and security-related components. The work shows the flexibility offered by combining NFV concepts with cloud computing technologies, which enables the orchestration of virtualized network environments according to users' needs in different application domains, even outside typical telco scenarios.

The chapter presents the results published in [P1]. In principle,

the proposed implementation methodology can also be extended to other application scenarios where it is useful to perform testing and validation without impacting the real-world counterpart, as presented in [P4].

3.1 Digital twin for enhanced security in industrial networks

The manufacturing industry is increasingly relying on networking to empower machinery and processes. The rise of interconnected systems in Operational Technologies (OT) is the basis of the Industry 4.0 paradigm, one of the major industrial turning points of the XXI century [B35]. The opening of the OT network to the enterprise Information Technology (IT) network and also to the Internet exposes the manufacturing environment to novel cybersecurity threats. Several attacks have been reported in recent years, making it clear that this is a key issue to be addressed [B36, B37]. Standardization efforts are also paving the way for a more systematic and uniform definition of cybersecurity features for manufacturing connected systems [B38]. As a result, assessing the cybersecurity characteristics and possible weaknesses of industrial networks, identifying mitigation techniques, and implementing possible countermeasures in case of attack are all strategic practices. However, industrial networks pose a whole new challenge to the application of these principles. This is because production systems should not be perturbed to any extent. Every anomaly in their behavior may result in a productivity loss or even permanent damage. Therefore, performing cybersecurity tests directly in a real industrial environment is challenging. One potential solution lies in leveraging the concept of digital twin, a virtual replica that accurately mimics the behavior and characteristics of the physical counterpart.

The work presented in this chapter proposes the implementation of a digital twin that replicates a typical industrial network architecture in a realistic virtualized environment. The aim is to provide a cyber range to perform cybersecurity testing and "what-if" scenario

validation without interfering with the real plant. In particular, the work focuses on the following aspects: first, to define a methodology that enables a flexible deployment of different digital twins from the same tools and procedures; then, to support this methodology with a platform to automate the orchestration of all components and resources needed for the deployment of the selected digital twin; finally, to implement and demonstrate the possibility of controlling the entire digital twin lifecycle.

3.2 Implementation methodology

Implementing digital twins of industrial networks involves the creation and provision of virtual replicas of physical network infrastructures that may differ in terms of topology, components, and offered service functionalities. Moreover, effective management and orchestration mechanisms are essential to ensure an automated and dynamic deployment of the digital twin components according to manufacturers' needs. In fact, the digital twin should be a flexible infrastructure that:

- can be created and deleted at will;
- can easily integrate new software components;
- is easy to modify in terms of network architecture.

Therefore, the digital twin should be a virtualized infrastructure manageable during its whole lifecycle in a flexible way. To meet these requirements, the proposed implementation methodology integrates NFV and cloud computing technologies. The NFV-MANO approach is adopted to provide automated management of the entire digital twin lifecycle, starting from high-level MANO descriptors and including all the operations to enforce configurations during the instantiation phase and at run time (usually called Day 0/1/2 operations).

Figure 3.1 shows a schematic representation of the reference scenario and the idea behind the digital twin implementation. The

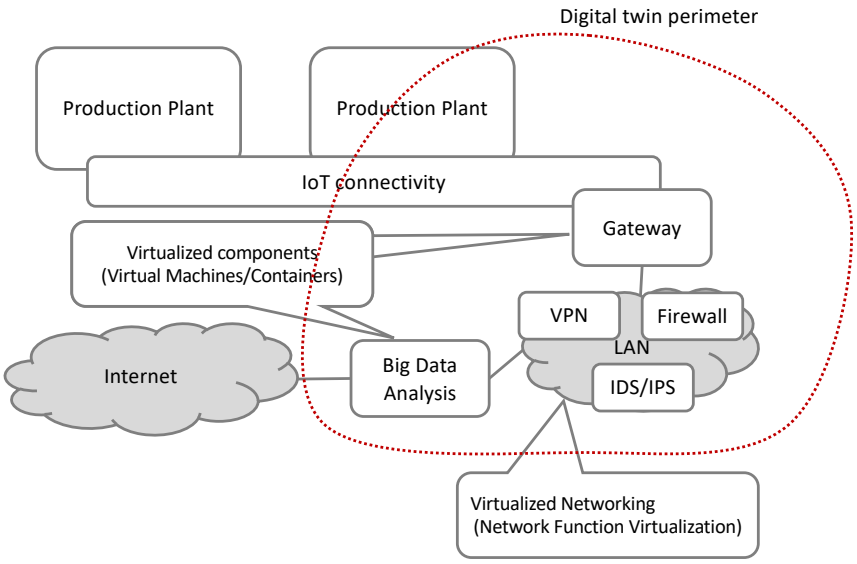

Figure 3.1: The perimeter of the digital twin, embracing the industrial network and part of the field network. Real-life field networks can be connected to the digital twin as well.

industrial environment comprises several elements such as network nodes, middleboxes, and security-related components (e.g., Intrusion Detection System (IDS) and firewall) that are provisioned as Virtual Network Functions (VNFs) interconnected with virtual networks in a cloud-based infrastructure. The digital twin thus provides a realistic virtual replica of the industrial network.

However, two main challenges usually arise while building such a twin. First, the field segment of the network is populated by sensors, actuators, Programmable Logic Controllers (PLCs), and other components that are closely bound to the hardware, making its virtualization difficult. We addressed this aspect following two possible alternatives: the field segment may be simulated in the digital twin with software components that implement the protocols of interest and simulate data exchanges on it, or real sections of the field network may be connected directly to the digital twin to feed samples of data. In this latter case, the connection can be active in parallel to the real one, thus not interfering with normal operations. Second, when digital twins are adopted to provide a faithful replica of the manufac-

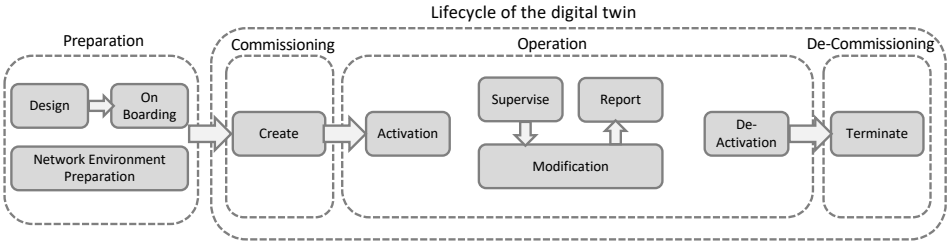

Figure 3.2: Schematic view of the digital twin lifecycle.

turing system, the traffic should mimic a real-life process, including its dynamics and quantitative behavior. Given the application scope, this is not necessary in the proposed work since it focuses on assessing possible weaknesses of the network architecture or protocols, which are reasonably independent of traffic dynamics.

3.2.1 Digital twin lifecycle management

In the proposed approach, the lifecycle management of the digital twin follows the one defined by the 3GPP and ETSI standards for network slices [B23],[B24]. Figure 3.2 shows a schematic representation of the various steps that compose the entire lifecycle management process of the digital twin. Applied to the digital twin case, they can be briefly described as follows.

- The *preparation phase* precedes the instantiation of the digital twin. This step includes the design of the digital twin architecture using MANO descriptors, the onboarding of the descriptor packages in the orchestration platform, and the environment setup (e.g., uploading the needed cloud images and setting up additional elements, such as specific networks for management purposes and connectivity with the outside environment that are not strictly related to the digital twin architecture).

- The *commissioning phase* is the step in which the digital twin is instantiated from the selected descriptors, with all its components deployed in a fully automated way in the underlying infrastructure.

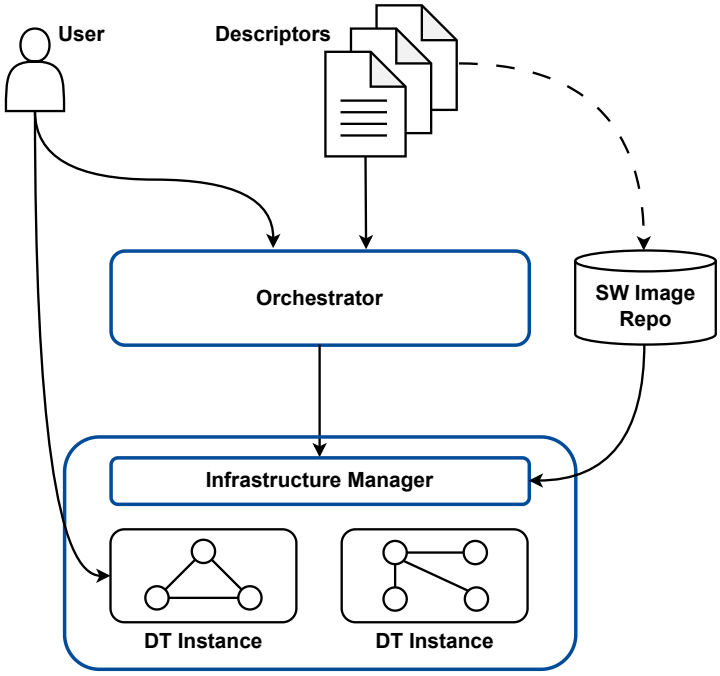

Figure 3.3: Architecture supporting the digital twin implementation

- The *operation phase* starts when the digital twin instance is up and running and involves the run-time monitoring and reconfiguration of its components as needed.
- The *de-commissioning phase* takes place at last when the digital twin is no longer needed and thus involves eliminating all its components and releasing allocated resources.

All these steps have been implemented in the testbed described in this work and some quantitative evaluation of the time needed to perform the commissioning and decommissioning actions will be reported in the remainder of this chapter.

3.2.2 Digital twin provisioning workflow

Figure 3.3 shows a representation of the proposed architecture to support the implementation of the digital twin. The architecture aligns with the NFV-MANO framework [B1] and ensures the management

of the digital twin instances during their entire lifecycle. As traditional cyber range environments, it also provides all the capabilities to experiment with configurations and security tests in the digital twin playground. Users can interact with the orchestration platform and request on-demand the deployment of new digital twin instances on the cloud-based infrastructure with an "as a Service" approach.

As mentioned above, the digital twin is modeled starting from MANO descriptors that specify the characteristics of its "building-block" components, providing any information required to instantiate and configure them in the underlying infrastructure (e.g., virtual resources and software images), and how they are interconnected. In particular, the digital twin architecture is designed using the following two types of MANO descriptors:

- The VNF Descriptor (VNFD) defines the specifications to finalize the instantiation of a VNF (i.e., of a single component of the digital twin). The provided information includes its internal architecture, the exposed connection points, configuration primitives, and the Virtual Deployment Units (VDUs) flavors to run the VNF on one or more virtual machines.

- The Network Service Descriptor (NSD) is a higher-level template that defines the constituent VNFs to be run (recalling the corresponding VNFDs) and the interconnections between them with virtual networks (i.e., Virtual Links in the NFV-MANO terminology). We use this type of template to describe the composition of the overall digital twin architecture.

This modular approach allows the reuse of descriptors and eases the extension with new components. With this approach, a company may flexibly implement digital twins of different industrial network architectures by creating the descriptors and providing the software images to replicate the real-world entities and related connectivity in the virtualized environment.

Once created, the descriptors are passed to the orchestrator, the entity in charge of all orchestration decisions at the service and resource levels. When a user requests a new digital twin instance, the

orchestrator converts the high-level specifications in the descriptors into a precise set of directives that the infrastructure manager executes to deploy and configure all needed virtual resources in the controlled infrastructure. Thanks to the interactions between these entities, the process is fully automated. That also applies to the other lifecycle phases. The user can trigger the modification or deletion of the instance simply by interacting with the orchestration platform.

3.3 Use cases and scenarios

In the following, we present some relevant use cases that can be efficiently developed and integrated over a digital twin instance using the proposed approach. In general, the flexibility of the digital twin lifecycle management allows us to take advantage of the resulting virtualized environment to address security challenges related to deployment, secure provisioning, seamless integration with the Internet, testing, and scalability, among others. As already discussed, in the considered scenario, the main objective is to provide a digital twin of the manufacturing system network to investigate possible weaknesses of the implemented protocols or the deployed network architecture. Therefore, the selected use cases focus on this aspect.

3.3.1 Key management

Public Key Infrastructure (PKI) is the state-of-the-art credential management solution on the Internet to ensure secure end-to-end communications between devices and services. However, PKI is built on a set of protocols that were not designed for industrial environments and presents a large management overhead related to the issuing, storage, distribution, verification, and revocation of certificates. As a result, the necessary public key operations tend to be computationally expensive and many industrial devices lack the required computing resources. On the other hand, symmetric-key cryptography can provide a lightweight solution for Industrial Internet of Things (IIoT) devices. However, both key storage and key management are big issues if using

symmetric-key encryption, especially when considering low-capacity devices [B39]. In any case, secure communications rely on the security of the key management mechanism adopted. Key management consists of the generation, distribution, storage, updating, and revocation of long-term keys on IIoT devices and can be either a centralized, decentralized, or distributed mechanism [B40]. Overall, [B40, B41, B42] show that there is no key management mechanism that can be universally adopted, but it will always be tailored to the needs of the particular deployment (e.g., memory requirement, computational cost, energy requirement, and resource consumption). Thanks to the enhanced flexibility and reconfigurability, simplified deployment procedures, and easy control of network topologies, the proposed industrial network digital twin implementation gives the possibility to easily test different key management mechanisms outside the production environment. Our setup may be adapted to find the one that best fits the requirements of any specific scenario.

3.3.2 Testing of ad-hoc provisioning solutions

Secure device provisioning, also referred to as bootstrapping or onboarding, is a process that provides a device with all the information it needs to connect securely to the network infrastructure and be operational. During secure provisioning, devices are transformed from their manufacturing state to a configured state that enables them to be used in a functional and secure manner. As described by the NIST in [B43], the secure provisioning process of a device is composed of four general phases: pre-onboarding, network-layer onboarding, network connectivity, and application-layer onboarding. Today, a mixture of proprietary, standardized, and academic provisioning solutions exists. In the following, a high-level description of the provisioning process is given in a solution-neutral manner. The pre-onboarding phase occurs before the device is associated with any given network. During this first phase, the device and the network infrastructure are equipped with their bootstrapping credentials (i.e., the information that each entity needs to be identified and authenticated). The bootstrapping

credentials will be then used in the successive onboarding phase to enable the device and the network infrastructure to establish sufficient trust in each other to allow secure provisioning to take place. The goal of the network-layer onboarding phase is to provide new credentials to the device, which will enable it to securely connect to the network in question. Once the device has established a secure network connection, it can use the connectivity to perform the application-layer onboarding if needed. During the application-layer onboarding phase, the device can securely download the application it needs to execute to perform the intended functionality. Once this application is downloaded and executed, the device becomes operational.

Since every device needs provisioning, the effort for device provisioning scales at least linearly with the number of devices. Automated provisioning solutions not only provide the means to scale deployments from single to dozens of devices, but they also have an essential impact on the security of the entire industrial network. Security attributes of the provisioning process must ensure that the network is not jeopardized when new devices are added to it. An inappropriately executed provisioning, or a provisioning solution with an insufficient level of security, can lead to insecure configurations that may enable attackers to eavesdrop or even manipulate communication between devices. By performing such attacks, adversaries can steal confidential data or tamper with industrial processes, causing economic loss, material damage, and even personal injury in some cases [B44].

In addition to these security considerations, another key difference among provisioning solutions dwells on the characteristics of the employed devices. For instance, IIoT devices typically lack screens and keyboards, so trying to provide their credentials can be cumbersome. For consumers, secure provisioning should be easy; for enterprises, it should enable large numbers of devices to be quickly provisioned with unique credentials.

Ideally, the provisioning process should be trusted, efficient, and flexible enough to meet the needs of various use cases. The proposed digital twin can be used to efficiently test ad-hoc provisioning solutions

customized to meet the needs of the specific industrial deployment and, more considerably, the related security requirements. In particular, a new device can be represented by an additional virtual component (e.g., run on a virtual machine) in the virtualized environment of the digital twin. Moreover, the four phases of the provisioning process can be straightforwardly mapped to the four steps of the digital twin lifecycle management reported in Figure 3.2: the pre-onboarding phase occurs in the preparation phase, while the other three phases of the provisioning process take place in the operation phase. The main advantage of testing provisioning solutions in a digital twin environment is therefore related to reducing the manual effort between subsequent testing iterations and saving time between the initial pre-onboarding phase and the final phase of rollback. In fact, testing a provisioning solution in a real industrial scenario would actually require to manually perform the pre-onboarding phase and, in the end, to manually restore the network infrastructure and the new device to the state they were in before the provisioning process took place. By means of the proposed digital twin approach, all the phases of the provisioning process can be performed instead automatically through the four steps of the lifecycle management.

3.3.3 Attack and defense

The proposed digital twin, being the virtual replica of its physical counterpart, shares the expected functional requirements and operational behavior of the corresponding industrial network. Therefore, the virtual environment of the digital twin can be employed practically as a testbed for cybersecurity assessments, implementing attacks against industrial protocols or testing the prevention and detection capabilities of the overall network architecture. By taking this approach, network administrators can focus on optimizing the cybersecurity requirements of the industrial network architecture while leaving the production environment operative. An example scenario of this use case is presented and implemented as a proof of concept in the following section.

3.4 Proof-of-concept implementation

In line with the general framework in Figure 3.3, for the proof-of-concept implementation we set up a testbed integrating Open Source MANO [B26] as an NFV-MANO compliant platform and OpenStack [B25] as a cloud management platform. OSM is in charge of the digital twin orchestration, while OpenStack supports the deployment by providing seamless integration with the cloud infrastructure and dynamic control of compute, storage, and network resources needed for the digital twin instantiation. The digital twin deployment is based on VNFs running on VMs. In particular, the digital twin implementation described hereinafter was tested in a private data center using an OpenStack cluster composed of two physical servers. Each server is equipped with 64 GB of RAM, 40 CPU cores, 1.2 TB of storage, 1 Gbit/sec network interfaces, and Ubuntu 18 LTS as OS. The main objective of the presented experimental evaluation is to demonstrate the potential of the proposed approach and the viability of its implementation even in a limited laboratory infrastructure. The considerations made in the previous chapter about the network slice instance also apply here. A more powerful setup is expected to provide better performance overall.

3.4.1 Digital twin architecture

The digital twin architecture implemented for the experimental validation reported in this work is simple, yet it provides a clear understanding of the operational models we propose, achieving the goal of enabling the execution of a set of cybersecurity tests safely in industrial environments.

Figure 3.4 shows the design of the digital twin architecture. For the design, we considered that a network architecture typical of an industrial scenario (see Figure 3.1) may include:

– one or more industrial network segments of the production plants, where different technologies (e.g., IIoT) may be used for data

collection and connectivity of machinery and other devices populating the OT;

- one or more gateways that manage the incoming and outgoing traffic between the production plants and the rest of the infrastructure (e.g., for monitoring and data analysis);
- one or more Local Area Networks (LANs) with system components dedicated to data analysis, monitoring, and security;
- an access network to interface with the Internet and the outside world.

Therefore, considering the focus on cybersecurity assessment and testing of the proposed use case, some relevant components that may be included are:

- a firewall as a first defense of the infrastructure to filter the network traffic,
- a Virtual Private Network (VPN) server to securely access the enterprise networks,
- an Intrusion Detection System (IDS) and an Intrusion Prevention System (IPS) to monitor the traffic and to detect/counteract cyber-attacks and anomalous behaviors,
- a Deep Packet Inspector (DPI) to examine and manage the network traffic.

As introduced before, the digital twin components outlined in the scheme of Figure 3.4 are run as VNFs on different VMs in the cloud infrastructure. The vertical and horizontal bars represent the virtual networks that interconnect them. The architecture reflects the considerations discussed above. One network of the architecture (the green bar at the bottom) is dedicated to the management of the infrastructure and all digital twin components. A management console connected to this network is provided as a VNF to users and network

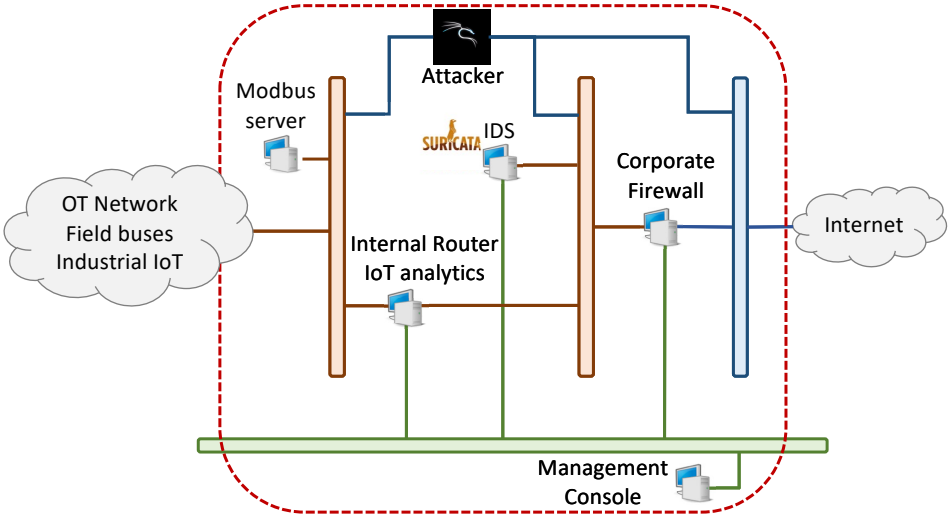

Figure 3.4: Architecture of the industrial network digital twin implemented for the proof of concept. The colored bars represent virtual networks. VNFs are depicted as computers.

administrators so that they can easily access and manage the environment. Another network (the blue bar on the right) is used as an access network to the outside environment and the Internet. Two internal LANs (the orange bars on the left and in the middle) are used respectively as an interface network to the industrial part of the infrastructure and as a corporate network with monitoring and security systems. A firewall is placed between the external access network and the internal LANs. It can also serve as a VPN endpoint or NAT. The traffic is mirrored on an IDS in the corporate network. An internal router interconnects the two LANs and can be complemented by a data analytics system. For the proof of concept and to finalize the experiments reported in this section, a Modbus server is added as a VNF to the industrial LAN. The virtualized environment also includes an attacker machine to perform a set of cybersecurity tests, as will be presented in the following. As shown in the figure, the IDS was implemented with Suricata [B45], a well-known open-source threat detection engine.

3.4.2 Digital twin instantiation

For this experiment, we created the NFV-MANO descriptors needed to deploy the architecture in Figure 3.4, onboarded the resulting packages in the OSM platform, and set up the cloud images for the various building blocks of the digital twin in OpenStack.

We tested the time required to commission and decommission this digital twin architecture. Figure 3.5 shows the time needed to complete the commissioning process, reporting the results of 10 experiments performed on the same infrastructure. Each point in the line chart is the instantiation time measured in one of the runs, while the horizontal dashed line represents the average time calculated on the set of values measured in the 10 runs. The measurements show some variability depending on the specific working conditions of the servers, with an average instantiation time of just over 3 minutes. The time required instead by the decommissioning process is shown in Figure 3.6. As in the previous figure, the graph shows the value measured in each of the 10 experiments and the average termination time of approximately half a minute. As the intuition suggests, the decommissioning phase is much faster. There is no persistence in components, which need to be deployed and configured at each commissioning, but mostly just switched off at decommissioning.

The figures show an overall amount of time for the commissioning and decommissioning of a digital twin instance that is quite acceptable for the purpose, even considering the limited hardware available in the laboratory testbed. In fact, with the proposed approach a company could shift from one digital twin architecture to another in a matter of minutes, assuming that the needed descriptor packages are prepared correctly in advance.

3.4.3 Cybersecurity testing

The use case implemented for the proof of concept is an example of attack and detection on the Modbus protocol [B46]. Modbus is one of the many industrial network protocols adopted for Supervisory Con-

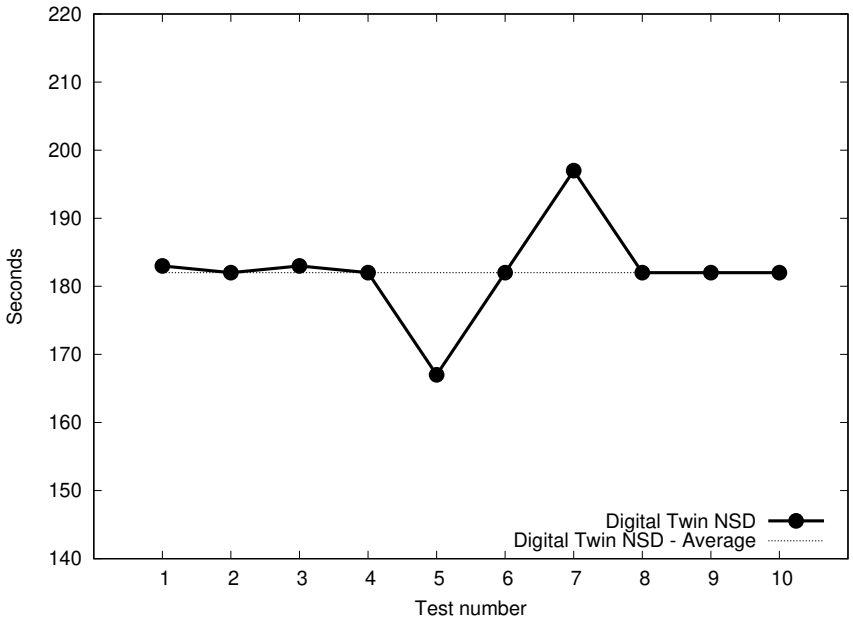

Figure 3.5: Time required for the digital twin commissioning (create an instance). The graph shows the values in seconds measured in ten different runs. The dashed line represents the average instantiation time of approximately 3 minutes (182.2 s).

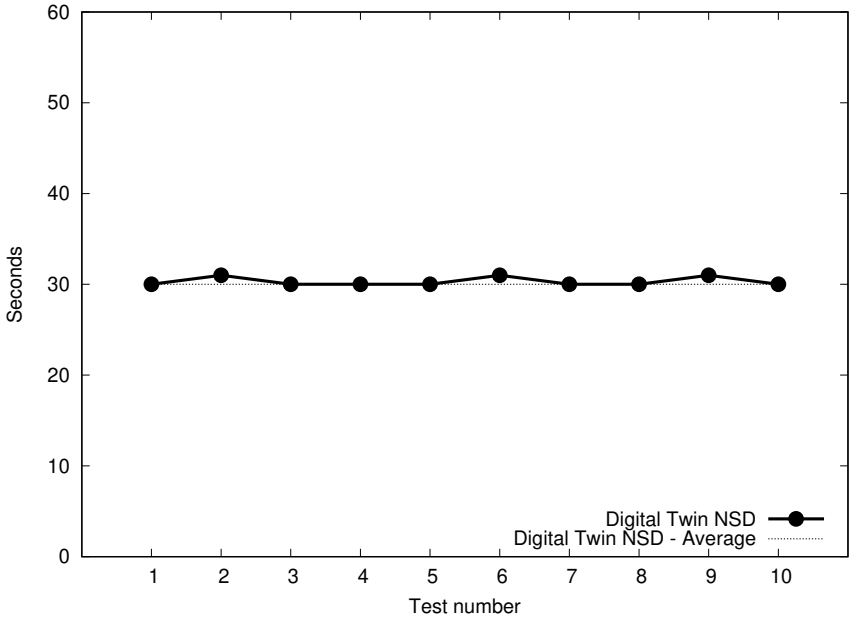

Figure 3.6: Time required for the digital twin decommissioning (terminate an instance). The graph shows the values in seconds measured in ten different runs. The dashed line represents the average termination time of approximately 0.5 minutes (30.3 s).

trol And Data Acquisition (SCADA) systems [B47]. Published originally in 1979 by Modicon for the purpose of using it with its PLCs, it is now one of the most widely used protocols for industrial monitoring and control applications. The original version of the Modbus protocol was designed for serial communication channels, but has been subsequently extended with a variant to provide communications over the TCP/IP stack.

Modbus cybersecurity weakness

All Modbus variants are based on a simple request-response communication mechanism to exchange messages between master and slave units (i.e., controllers and field devices). The messages exchanged contain function codes that indicate requested operations on slaves, diagnostic functions, and error conditions. Modbus was not designed with security in mind. In fact, it has inherent vulnerabilities that attackers can exploit to carry out attacks in industrial networks if appropriate countermeasures are not implemented [B48],[B47].

From a cybersecurity perspective, Modbus has three obvious flaws that can be easily exploited by an attacker.

- Modbus has no encryption mechanisms, which means that all data are transmitted in clear text. As a result, whole messages can be easily intercepted and tampered with.

- The second flaw is related to the Modbus built-in function code mechanism. Function codes are known and some of them may be critical if misused. A simple example is the function code “08”, which means diagnostic mode. If used with the sub-function code “04”, it can force the execution of the slaves in listen-only mode. If a hacker succeeds in broadcasting this function code over the whole network to all slaves, he may be able to completely block the PLCs.

- In serial communication networks, Modbus uses the broadcast transmission to send messages between the master device and

the slave devices. In this scenario, the hacker can easily intercept packets on the broadcast network and launch a man-in-the-middle attack.

Test an attack

As a proof of concept, we first implemented enumeration and disruption attacks on the registers of Modbus devices (on the Modbus server). We still refer to the schematic architecture in Figure 3.4. In practice, the attack takes place in three phases:

1. *identification* of the Modbus server;
2. *enumeration* of the exposed registers, testing for the "read" and "write" operations;
3. *information gathering* by massively reading all the registers.

During the identification phase, the attacker starts scanning the industrial LAN looking for a Modbus server. Once the Modbus server is identified, the register enumeration phase is started. The aim of this phase is to map the attack surface to the devices connected to the server. To achieve that, we tested the possibility of a single reading of each register and then a massive reading of all the 65536 registers. Besides leading to a disclosure of the information exposed by the Modbus server, this massive reading can also cause a slowdown of the service itself, and consequently lead to a denial-of-service attack. After mapping the active registers, we also tried to write on them. This attack may cause two main effects: disrupt the system application or change its behavior to a malicious one.

These are examples of tests that, despite their simplicity, can not be carried out in a real production environment because they introduce enough perturbations to create problems to the production chain.

Test IDS rules for attack detection

Finally, to complete the proof-of-concept experiments, we exploited the digital twin to define and test countermeasures to the possible

attacks. Countermeasures can rely on detection, mitigation, and prevention mechanisms. Whatever mechanism is selected, it is important to test it outside the industrial environment to be sure that it is effective and does not cause any problems to regular operations. In general, this procedure of testing countermeasures on the digital twin can be adopted to test new configurations before their implementation in the real-world environment, as well as to evaluate the operation of mechanisms already in use.

In the implemented example, a set of detection rules for the Suricata IDS was tested in the digital twin. The rules under test are reported in the following and are meant to detect the massive reading attack by monitoring TCP connections on port 502, the one where the Modbus server exposes the registers.

```
alert tcp $EXTERNAL_NET ANY -> $HOME_NET
502 (msg:"snort rule test 1"; sid:14265;
content:"|09|"; offset:9; rev:3;)

alert tcp $EXTERNAL_NET ANY -> $HOME_NET
502 (msg:"snort rule test 2"; sid:42299;
content:" |10 05|"; offset:10; rev:5;)

alert tcp $EXTERNAL_NET ANY -> $HOME_NET
502 (msg:"snort rule test 3"; sid:15076;
content:"|07|"; offset:7; rev:5;)
```

The traffic on the industrial LAN of the digital twin was mirrored to the IDS, and the effectiveness of the rules could then be tested along with the normal operations. The screenshot in Figure 3.7 shows the IDS alerts that notify the malicious activity based on the rules we entered.

Figure 3.7: A screenshot of the IDS command line interface showing the alerts of the Suricata detection rules.

Chapter 4

Orchestration of secure ML pipelines

The previous chapters delved into the implementation of service management and orchestration procedures in two different application domains, showing the level of flexibility and automation offered in service deployment and lifecycle management with the integration of technologies such as NFV and SDN. This chapter shifts the focus to investigating distributed control mechanisms that can be integrated with the above-mentioned technologies to improve network service control.

As will be discussed in the following section, the increasing diversification and dynamicity of services, as well as the growing network heterogeneity and complexity, demand novel approaches to support effective service and network management at different levels of the infrastructure. Novel control mechanisms based on Artificial Intelligence (AI) and Machine Learning (ML) techniques are expected to play a key role in B5G networks. In particular, near-real-time autonomous network operation is required to deal with the expected large traffic dynamicity and guarantee the stringent performance required by future services.

This chapter presents an orchestration system to deploy ML pipelines connecting distributed intelligent agents for near-real-time control of network services. A Machine Learning Function Orchestrator (MLFO) is proposed for deploying the control agents across the network and

configuring them to form the overlay control system with the help of an NFV-MANO platform. The MLFO is an original orchestrator that has been developed in collaboration with the Optical Communications Group of Universitat Politècnica de Catalunya (UPC).

Furthermore, the chapter presents the integration of the MLFO with other components into an architectural framework to orchestrate secure ML pipelines in a dynamic and automated way. Given the distributed nature of the control pipelines, and considering that agents exchange information critical to determining the network behavior over a public infrastructure, security is a key issue to be addressed. Two approaches are proposed to secure communications between the agents. In the first explored approach, communications are secured with IPsec tunnels using a blockchain-based Distributed Ledger Technology (DLT) network for key exchange [P6]. The second solution, instead, leverages an in-network encryption mechanism to encrypt/decrypt the traffic exchanged between the agents with P4 programmable switches [P7].

4.1 Secure ML pipelines for near-real-time control of 6G network services

Although the standardization process for 6G is still in its infancy, it is expected that next-generation networks will rely on a pervasive use of AI to improve network and service management [B49]. The vision for 6G sets a high bar for next-generation mobile networks, which shall support increasingly heterogeneous vertical services and do so with outstanding performance [B50]. Therefore, a new level of automation in network control will be required to cope with the diversification of functional and performance requirements, as well as the growing complexity and heterogeneity of the network itself.

Control mechanisms rely on effective data collection and analysis to build knowledge about the state of the network components and make decisions based on that. Advances made in AI and ML techniques in recent years have attracted the attention of the telco

community and paved the way for novel data-driven control mechanisms. AI and ML models have the potential to improve data analysis and decision-making, increasing network automation to the point of unlocking autonomous management operations through "zero-touch" control mechanisms in the long run [B51].

However, the implementation of these mechanisms opens up new challenges not only from the standpoint of AI techniques but also on where and how to integrate this intelligence within the network architecture and with current management and orchestration frameworks. For instance, near-real-time autonomous control will be essential to swiftly adjust the network resources and functions to dynamic traffic changes and ensure stringent requirements of upcoming services. However, near-real-time decision-making is limited by the centralized nature of control loops implemented in controllers and orchestrators due to the latency in data transfer and decision communication [B52]. A possible solution to reduce response time is to deploy a distributed and coordinated control system based on a set of intelligent agents (i.e., AI/ML-based control algorithms) executed in different locations across the network infrastructure, as close as possible to the data sources. The agents are delegated to local autonomous decision-making and coordinate to provide the overall control process of network services. Therefore, the agents participating in the control need to share knowledge and communicate with each other, forming collaborative pipelines.

Nonetheless, this distributed approach could raise security concerns as it makes the control system more vulnerable to some attacks than a centralized one. Attacks can be crafted to undermine the agents and their communications. Therefore, the implementation of this type of control system requires the definition of procedures for orchestrating the agents and their connectivity, as well as addressing security issues arising from such a distributed approach.

4.2 ML Function Orchestrator

Drawing from the work in [B53], a dedicated orchestrator, namely a Machine Learning Function Orchestrator, is proposed to orchestrate the deployment of secure ML pipelines to control network services in near-real-time and provide the needed supervision of the control process. The concept of ML pipeline is extended to a multi-agent system composed of intelligent agents distributed across the network that coordinate to provide the overall control process by exchanging information. The ML pipeline comprises the set of agents and the communication infrastructure connecting them (i.e., the pipeline). The intelligent agents make autonomous decisions under the supervision of the orchestrator, based on measurements collected and processed in different locations with a distributed telemetry process.

However, the distributed nature of agent pipelines makes the control system vulnerable to some malicious attacks. The agents exchange data and other types of information critical to determining the overall network behavior over a public infrastructure. Therefore, countermeasures to secure communications in case of eavesdropping must be an integral part of the ML pipeline.

As will be discussed in more detail in the next section, the MLFO is part of an orchestration framework and interacts with other system components, such as a Service Management and Orchestration (SMO) platform, to perform the dynamic and automated deployment of the resources needed to run the agents, as well as the configuration of the connectivity between them, also fulfilling the security requirements. It is assumed that an ML pipeline per Network Service (NS) is activated. Agents participating in the control are deployed dynamically in the network infrastructure along with the required connectivity once the NS is created. The MLFO is responsible for deciding the ML pipeline composition, including the locations where agents need to be deployed and how they need to be connected. The orchestration process takes place in two subsequent phases:

– the deployment of the agents in the selected locations, orchestrating the needed resources to run them, and establishing the

required network connectivity;

– the configuration of the agents so that they can run the control algorithms, establish secure communications, and coordinate to form the overall overseen control system. They are also configured to collect telemetry data from the targeted sources.

4.3 Deployment of secure ML pipelines

The deployment of the ML pipelines involves orchestrating both the resources to run the agents and the required communication infrastructure to connect them. VXLANs can be configured to extend local network segments across different clusters, providing a virtual overlay network and logical isolation over the underlying infrastructure for the control pipelines. Nonetheless, as discussed above, communications between agents of an ML pipeline should be secured to prevent malicious attacks in case of eavesdropping. VXLAN encapsulation does not provide any intrinsic security feature, as ensuring secure end-to-end connectivity is beyond the scope of the protocol.

4.3.1 Secure inter-agent communications

We rely on encryption to improve the security of inter-agent communications. Encryption protocols, such as the IPsec suite, allow the setup of secure, authenticated communication channels at the network layer, reducing rogue risk. Pre-shared keys can be used for peer authentication to streamline the IPsec secure channel configuration without requiring the generation of certificates for each agent involved in the control. Such a solution requires an authentication infrastructure to distribute keys among authorized agents.

Distributed Ledger Technologies (DLTs) offer a promising solution by providing a secure, shared, immutable, and decentralized ledger of transactions. DLTs avoid the need for a trusted centralized authority to manage transactions between parties. To add new data to the ledger, the nodes participating in the DLT network have to validate the

transactions (e.g., agree with their order and existence on the ledger) with a consensus mechanism. In recent years, applications of DLTs in the context of 5G services have been proposed and have showed that the consensus mechanism has an impact on the overall performance [B54]. Even with the simplest consensus mechanism, data exchange can be slow for near-real-time applications. However, in the proposed scenario, the impact of the consensus mechanism is limited to the setup phase of the pipelines. It does not affect the actual exchange of data between agents, which takes place in the secure communication channel once established.

In particular, the proposed system uses Smart Contracts (SCs) on the DLT network to facilitate the coordination and collaboration between the agents and the MLFO. The MLFO handles the dynamic association of the agents to the DLT. Smart contracts are then used to exchange keys between the MLFO and the authorized agents.

4.3.2 Orchestration and control system

Figure 4.1 shows an overview of the orchestration and control system. The MLFO coordinates the deployment and configuration of the ML pipelines by interacting with the other entities. The SMO is an NFV-MANO platform that manages the NSs and orchestrates the components needed for the ML pipelines. The SMO system interacts with several VIMs that manage clusters located in different locations, providing local networking, computing, and storage resources to run the agents. An SDN controller is then on top of the packet network and manages the connectivity. The MLFO computes the ML pipeline design according to information gathered from the SMO platform and a topology server based on ALTO [B55], fed with topological and networking data. Finally, a DLT infrastructure comprises different distributed nodes participating in the network and supports the exchange of the keys with a smart contract.

We implemented a testbed based on this architecture. The MLFO, the algorithms and interfaces in the agents, and the ALTO server have been developed in Python 3.10.4 and run in Docker containers.

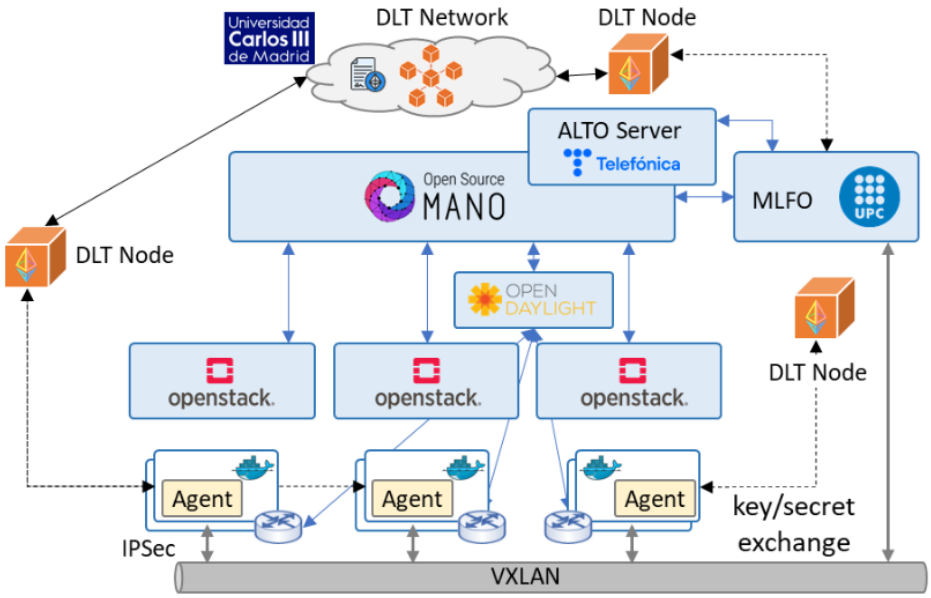

Figure 4.1: Overall orchestration and control system

The SMO platform was implemented with Open Source MANO v.14. The MLFO interacts with OSM via the Northbound Interface (NBI) using the OSM client library [B56]. Three OpenStack clusters (release 2013.1 Antelope) were set up to provide the resources for the deployment. OpenDaylight (ODL) (release 16.0 Sulfur) was used as the SDN controller. The DLT network was set up with four container-based DLT nodes, based on the Geth implementation of the Ethereum blockchain. The smart contract for the key exchange has been written in the Solidity programming language.

4.3.3 Orchestration workflow

Figure 4.2 shows the workflow of the ML pipeline deployment. It is assumed that the MLFO has been previously configured with the needed credentials for the DLT network. The workflow consists of two phases: *i)* the ML pipeline deployment and *ii)* the agents' configuration and key exchange.

The orchestration process is initiated by the SMO platform (OSM) after the deployment of a new NS (step 0 in Fig.4.2). The SMO trig-

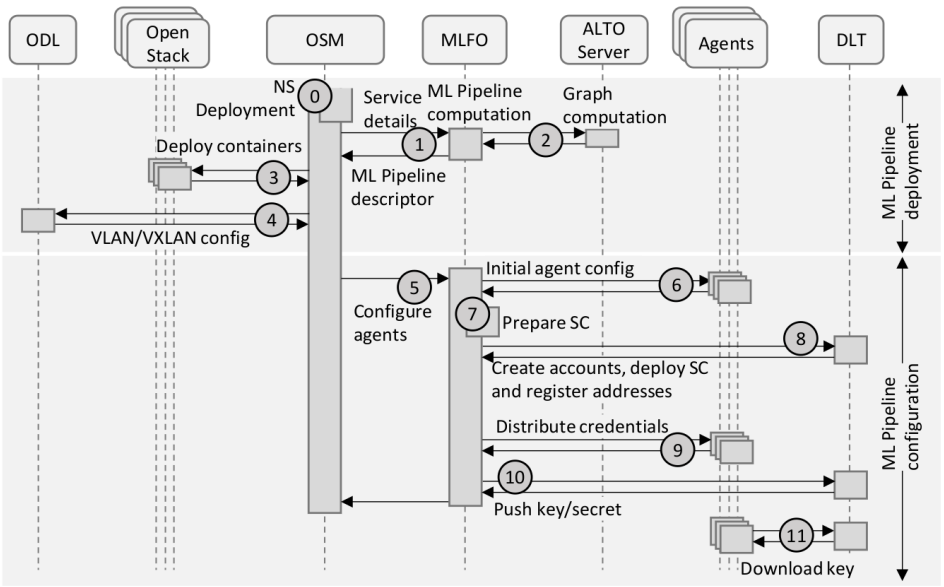

Figure 4.2: ML Pipeline orchestration workflow

gers the ML pipeline deployment and requests the definition of an ML pipeline for the NS given its relevant details, including the location of the VNFs **(1)**. Based on the NS details and the requirements of the ML pipeline in terms of delay and throughput among the agents, as well as the required resources for the agents, the MLFO requests the ALTO server to compute a graph with the resources in the network infrastructure that meet the requirements and can be used to support the ML pipeline **(2)**. Based on all the information, the MLFO computes the optimal ML pipeline design and sends back MANO descriptors. The descriptors define the deployment characteristics of the ML pipeline, including information such as the locations where the agents need to be deployed (clusters), the resources needed to run them, and the connectivity to be created. A list of iterations is generated that includes the communication of the SMO system with the VIMs (OpenStack) for the deployment of the agents encapsulated into VMs **(3)**, and with the SDN controller (ODL) for managing the connectivity **(4)**. Once the agents are running and the connectivity is available, the ML pipeline is deployed and the configuration phase starts **(5)**. When the MLFO receives the request to configure the

agents, it sends the initial configuration that includes the addresses of the VNFs and that of the other agents, as well as the algorithms that every agent runs **(6)**. After that, the MLFO compiles the smart contract that will be used to store the key for the ML pipeline **(7)**. Next, the MLFO creates the DLT accounts for the agents, deploys the smart contract and registers the addresses with credentials to interact with the smart contract **(8)**. In this way, the MLFO can control the access to the smart contract and add or revoke permissions if needed in case of ML pipeline reconfiguration. Once the addresses are registered, the MLFO distributes the credentials to interact with the smart contract to the agents involved in the ML pipeline **(9)**. The agents connect to the smart contract through the local DLT node. The MLFO generates a random key that the agents can use to initialize the IPSec secure communication channel and pushes the key to the smart contract **(10)**. At this time, the deployment of the NS ends from the viewpoint of OSM. Once the transaction is validated by the DLT network, agents receive a notification, download the key and use it for setup a secure communication channel with other entities in the ML pipeline for near-real-time control of the NS **(11)**.

The deployment of ML pipelines following this workflow with the proposed orchestration system was demonstrated with a proof-of-concept demo presented in [B53]. Figure 4.3 shows the reference scenario considered in the demo, with an ML pipeline deployed in distant locations over an optical network segment. For the demonstration, agents were also configured to collect and exchange telemetry measurements from the components to which they were assigned.

4.4 In-network encryption

This section extends the work presented in the previous part of the chapter, reporting a second solution to secure the message exchange between distributed control agents that leverages a programmable data plane.

Data Plane Programmability (DPP) offers unprecedented flexibil-

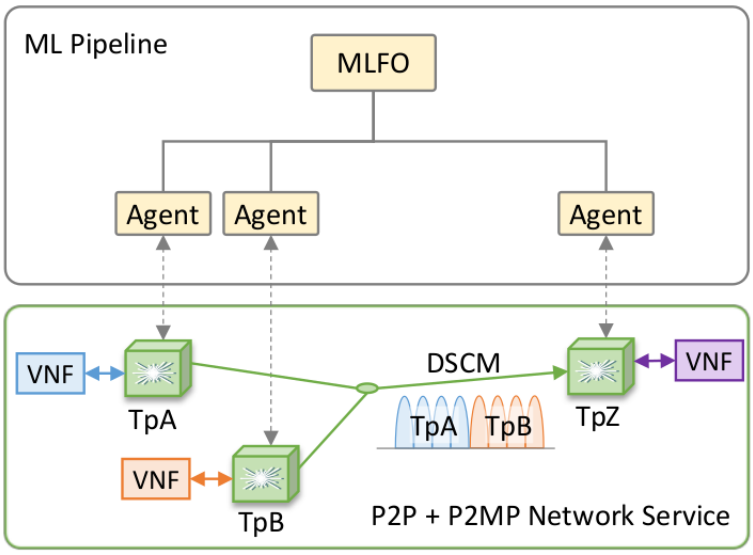

Figure 4.3: Schematic representation of an ML pipeline with agents deployed in different locations close to the VNFs they have to monitor

ity in performing customized in-line packet processing on top of regular forwarding. This has paved the way for the implementation of many functionalities beyond the set of protocols supported by the fixed function pipelines of traditional network appliances [B4]. Therefore, DPP creates the opportunity to offload additional services from other nodes to network devices for in-network computation. P4 (Programming Protocol-independent Packet Processors) is the most widely used domain-specific language for programming the behavior of network devices, based on a flexible parsing mechanism and a multi-stage match-action pipeline [B57].

The work presented in this section takes advantage of the capability offered by DPP to secure the communications between the agents by performing in-network encryption on P4 programmable switches. The switches will be able to identify the traffic related to agents of the same control pipeline and cipher it properly while leaving the other traffic unchanged. In particular, this section reports the implementation and related experimental validation of a custom P4 Parsing Pipeline and an *extern* function in P4 switches to encrypt/decrypt the payload of the traffic sent between the agents. The developed solu-

tion leverages a standard implementation of the Advanced Encryption Standard (AES) symmetric encryption algorithm [B58].

4.4.1 System architecture

Figure 4.4 depicts the envisioned system architecture in this second scenario. The architecture represents an alternative to the orchestration framework presented in the first part of the chapter and shows how the proposed in-network encryption solution can be integrated with the MLFO to deploy secure ML pipelines across the network infrastructure. The deployment workflow overall aligns with what has been discussed in the previous sections.

In this scenario, the MLFO coordinates the whole deployment process and manages the pipeline configuration by gathering and sending information directly to a Service Management and Orchestration platform and a network controller via the exposed Northbound Interfaces. Local Virtualized Infrastructure Managers control then the resources in each computing cluster. As in the previous framework, the SMO platform supports the orchestration process and is in charge of the deployment of the control agents. The agents of the same control pipeline can be distributed in different locations across the network depending on the network services characteristics and based on the MLFO requests. Other entities, as in the previous scenario, can provide additional information for the computation of the optimal ML pipeline composition. Alongside the orchestrator, the network controller manages P4 programmable switches placed at the edge of the computing clusters to handle the connectivity and secure the communications between the agents.

When the deployment of a new control pipeline is triggered and the MLFO creates the ML pipeline descriptors, the SMO platform translates the high-level descriptors into lower-level implementation directives and sends them to the VIMs to run the agents in containers in the underlying infrastructure. Once the distributed agents are running, the network controller installs rules in the programmable switches to match the traffic flows exchanged between the entities of

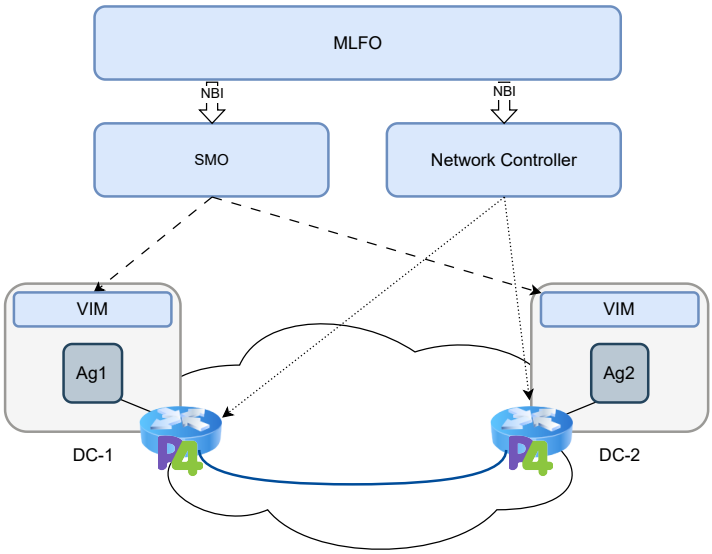

Figure 4.4: Schematic representation of the system architecture and its main entities. Agents deployed in different computing clusters that belong to the same control pipeline exchange data through a secure channel created with in-network encryption.

each pipeline and encrypt/decrypt the data.

This is just an example of integration and other scenarios involving, for instance, multiple local network controllers are possible. Moreover, the proposed in-network encryption can be generally extended to other application scenarios beyond the case under consideration in this chapter.

4.4.2 Encryption/decryption function implementation with externs

Several hardware-based (ASICs, NICs, FPGAs) and software P4 programmable targets (i.e., devices that can execute P4 programs) are available these days, supporting different P4 architectures. These architectures model variants of the processing pipeline and essentially define the set of blocks, capabilities, and interfaces made available to

the programmer. Along with the core language constructs, P4 targets can support, to different extents, additional built-in objects and functions to provide features that may be useful for packet processing in many P4 programs (e.g., checksum generation, registers, and counters). Such capabilities are made available through the concept of P4 *extern*. Externs are part of the $P4_{16}$ language specification [B57] and provide a way to define interfaces between P4 programs and functionalities implemented outside them. In principle, this approach also allows the built-in capabilities to be extended with other custom features that might be difficult to implement efficiently in P4 code.

In this work, we propose a proof-of-concept implementation of the in-network encryption mechanism for the P4 software switch Behavioral Model version 2 (bmv2) [B59]. For example, the source code of bmv2 implements some additional primitives to execute hash functions for checksum calculations and cyclic redundancy checks, which are commonly employed to detect accidental errors in data transmissions. However, bmv2 does not provide built-in cryptographic capabilities to encrypt/decrypt data.

Drawing from the work concerning Hash functions in [B60], we exploit the concept of P4 *extern* to provide the missing encryption and decryption functions, similar to the other features already provided by the programmable software switch. The developed functions are based on a standard C++ implementation of the AES algorithm and, after import and declaration of the extern, can be used in control blocks of the parsing pipeline just like any other P4 function. More details will be provided in the testbed description in the following subsection.

4.4.3 Testbed implementation

The reported implementation focuses on evaluating the proposed in-network encryption within a virtualized environment, wherein each constituent component is realized as a Docker container. We developed specialized Docker container images to implement the functionalities of the agents. In turn, the agents are connected by P4 switches implemented by means of two other containers. They are equipped

with a simple level 2 forwarding pipeline and configured to only forward traffic to the end hosts: the P4 code is compiled for the reference virtual target bmv2. Furthermore, we developed a customized implementation of the control plane, responsible for making high-level decisions including switch pipeline configuration among other tasks. The controller, which executes the control plane functionalities, communicates with the switches through the *P4Runtime* specification [B61], allowing P4 pipeline reconfiguration at runtime. P4Runtime is a well-established API for controlling the data plane elements of a device whose behavior is specified by a P4 program.

Taking into account that the communication between the agents occurs through HTTP REST API, with data exchanged in JSON format, we developed a specific P4 Parsing Pipeline to handle the variable length of the HTTP message. The steps performed are:

1. The parser extracts the Ethernet and the IP headers from each packet;

2. If the *Protocol* field of the IPv4 header is equal to 6 (i.e. the TCP protocol), the header is extracted;

3. Knowing the field *Total Length* of the IPv4 header field and the field *Data Offset* of the TCP header, the P4 parser calculates the actual length of the TCP header, including the TCP options, and the total payload size;

4. Finally the HTTP message is extracted depending on the value of the TCP payload size.

This procedure facilitates subsequent actions by providing the necessary groundwork. It enables the switch to identify both the source and destination endpoints, which is paramount in determining the application of the encryption and decryption actions.

To secure the in-network channel, 128-bit key AES is the chosen algorithm. Symmetric AES keys can be pre-installed in the P4 switches at configuration time or can be exchanged between the controllers

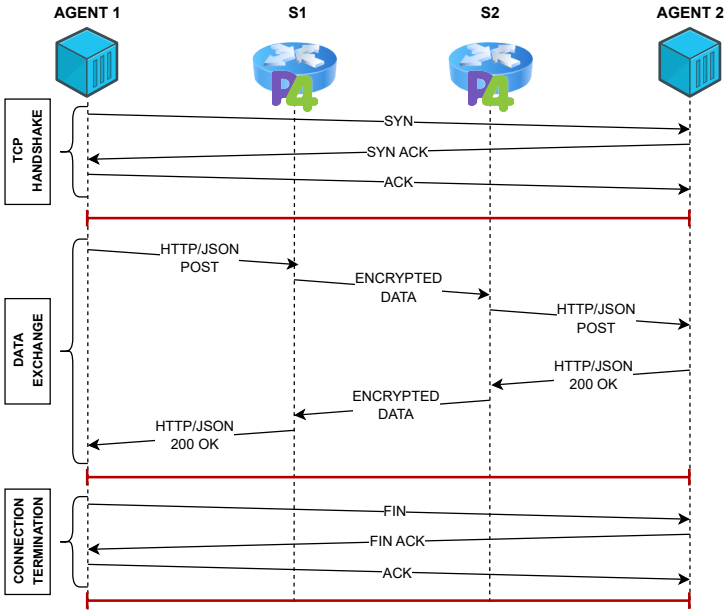

Figure 4.5: Example of a connection flow between two agents exchanging data via REST API in JSON format. The two P4 programmable switches encrypt and decrypt the TCP packet payload containing the HTTP and JSON data.

via Diffie-Hellman (DH) key agreement [B62] at runtime and then installed on the switches. From an operational perspective, generally, the establishment of the secure channel between the two bmv2 switches can be accomplished in two ways. It can be configured permanently at system start-up, routing all communication requiring encryption through this channel. Alternatively, it can be established dynamically for specific traffic flows, with each flow having its dedicated encrypted channel. The decision regarding which strategy to employ is a matter for the control plane.

As introduced before, we exploited the concept of P4 *extern* to develop the AES encryption and decryption functions. An *extern* is an API that uses an external dependency, which can be queried by the target. In this case, the implemented extern leverages standard C++ implementations of the AES 128-bit encryption and decryption

functions. Each switch performs both encryption and decryption, depending on the flow direction, as shown in Figure 4.5. Subsequently, each pipeline encrypts and decrypts the payload in the ingress queue control by calling the extern and emitting the encrypted/decrypted payload in the Deparser pipeline, as summarized in Alg. 1 and Alg. 2.

Algorithm 1: Encrypt action in the bmv2 Ingress Pipeline

Input : The HTTP message m of the input packet.
Output: The encrypted data *enc* correspondent to the HTTP message m.

1 $enc \leftarrow Encrypt(m)$: call P4 extern AES Encrypt function;
2 Update the value of the IPv4 header field *Total Length*;
3 $m.setInvalid()$: do not emit the original HTTP message in the Deparser;
4 $enc.setValid()$: emit the encrypted message in the Deparser.

Algorithm 2: Decrypt action in the bmv2 Ingress Pipeline

Input : The encrypted HTTP data *enc* of the input packet.
Output: The decrypted HTTP message m.

1 $m \leftarrow Decrypt(enc)$: call P4 extern AES Decrypt function;
2 Update the value of the IPv4 header field *Total Length*;
3 $enc.setInvalid()$: do not emit the encrypted message in the Deparser;
4 $p.setValid()$: emit the decrypted HTTP message in the Deparser.

4.4.4 Experimental validation

This subsection reports the results obtained with the testbed described above. The bar chart in Figure 4.6 shows the average round-trip time calculated by exchanging 100000 HTTP messages between the agents. The error bars in red represent the related standard deviation. For comparison, as a communication time, we considered the time interval between sending an HTTP request and receiving the subsequent

response when the in-network encryption or a traditional IPsec secure channel is used. For the IPsec case, we considered different configurations by setting up a secure channel between the agents or the switches using strongSwan [B63]. In tunnel mode, the entire IP packet is encrypted whereas only its payload is ciphered in transport mode. The performance exhibited by our proposed solution, utilizing in-network encryption, is comparable in order of magnitude to that of the various IPsec configurations being evaluated. Given that the tests were performed in a virtualized environment with software programmable switches, mapping our approach on a P4 hardware pipeline should improve performance.

Figure 4.7, instead, shows the average time needed to set up the secure channel and start the connection in three different scenarios. In the first two cases, the P4 programmable switches use pre-shared keys or the Diffie–Hellman (DH) protocol to negotiate a shared secret key over the insecure channel, respectively. In the third scenario, as a comparison, we used the orchestrator to establish an IPsec Host-to-Host tunnel between the agents without in-network encryption. To test the latter scenario, we used Open Source MANO [B26] as an orchestration platform to deploy the agents running strongSwan in two computing clusters with OpenStack [B25] as VIM and to configure the IPsec tunnel at runtime with Day 2 operations defined in the descriptor packages. Day 2 primitives allow the execution of actions on VNFs at runtime. We exploited the Juju python framework to build proxy charms [B64] as a set of scripts to configure the IPsec tunnel on both agents with a shared secret key, start the secure connection, and retrieve the entire process timestamps. Most notably, the average secure connection establishment time with the DH exchange is far less than the time needed to set up and establish an IPsec connection with the orchestrator. Finally, it is important to highlight that if the symmetric keys are pre-installed in the P4 switches, then the encrypted channel establishment time is reduced to the connection time.

The achieved results prove the feasibility of the proposed approach and show that in-network encryption through P4 programmable switches

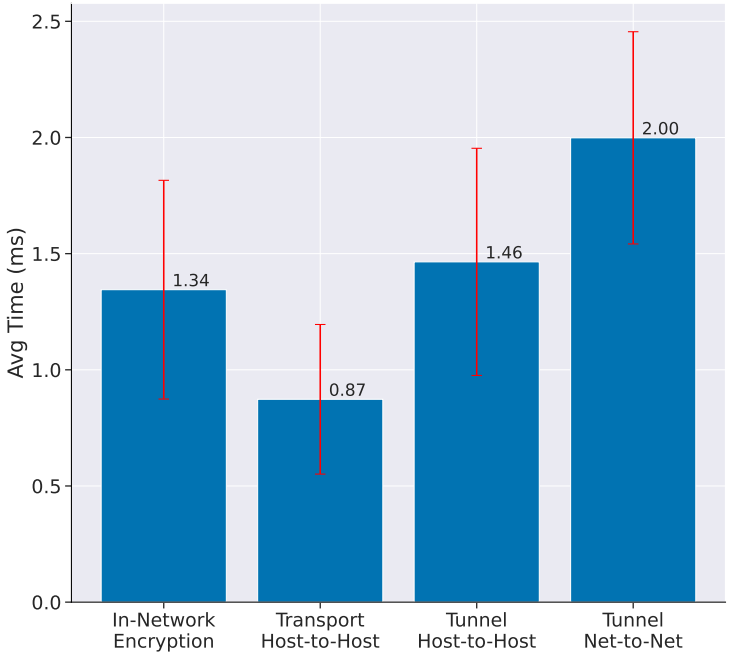

Figure 4.6: Average communication time between agents, comparing in-network encryption with traditional IPsec. The IPsec is configured between the agents (Host-to-Host) in transport and tunnel mode or between the switches (Net-to-Net).

is comparable in order of magnitude with the performance achieved in the testbed with traditional solutions such as IPsec while providing more flexibility in the setup phase.

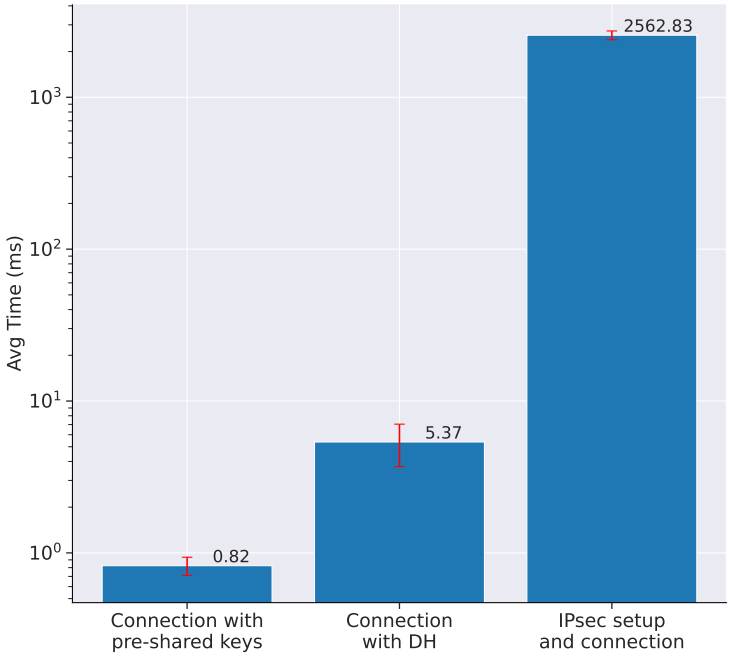

Figure 4.7: Secure communication average set-up time. The y-axis is represented in a logarithmic scale. The first two bars refer to agents communication via the in-network secure channel. The third bar refers to a traditional IPsec configuration using the orchestrator.

Chapter 5

Conclusions

This thesis addressed the management and orchestration of network services in different application domains. The work presented architectural models and practical instruments based on NFV and SDN paradigms that can be employed to orchestrate network components to provide the infrastructure needed to meet the functional and performance requirements of each scenario. Specifically, Chapter 2 presented the design and implementation of a network slice for MC communications. The NFV-MANO approach is applied to orchestrate the slice components on a cloud-based infrastructure spanning different data centers. The experimental validation showed that the network slice can be instantiated in a few minutes, thus allowing maximum flexibility for infrastructure deployment in case of emergency events. Moreover, it showed that a complete separation between the control and data planes can be achieved at both the network (5G) and the service (MC) levels. Deploying media servers and 5G user plane functionalities at the edge provides faster access to mission-critical services and lowers communication latency. Finally, it also showed that the QoS in the transport segment interconnecting the network slice sections can be managed in an integrated way with the slice management thanks to an SDN control. Chapter 3 presented the application of the NFV-MANO principles in a different domain, proposing the orchestration of an industrial network digital twin that provides a realistic virtualized environment to perform cybersecurity testing without in-

terfering with the real industrial plant. The work showed that the digital twin architecture can be designed in a modular and flexible way. The experimental validation demonstrated the automated on-demand provisioning with a proof-of-concept digital twin, reporting instantiation and elimination times. It also showed the potential of using the digital twin to perform security tests and what-if scenario validation. Finally, Chapter 4 presented an orchestration system to deploy secure ML pipelines for near-real-time control of network services. In this work, a novel ML Function Orchestrator performs the deployment and configuration of the ML pipelines with the help of an NFV-MANO platform and other entities. The chapter presented two solutions to secure inter-agent communications. The former exploits IPsec secure channels using a DLT network for key exchange. The latter proposes an in-network encryption mechanism to encrypt/decrypt the traffic with P4 programmable switches. In this second case, the chapter reported results comparing the in-network encryption performance with IPsec.

Acronyms

3GPP 3rd Generation Partnership Project.

AES Advanced Encryption Standard.

AI Artificial Intelligence.

API Application Programming Interface.

CNTT Cloud iNfrastructure Telco Task Force.

COTS Commercial Off-The-Shelf.

CUPS Control User Plane Separation.

CV Coefficient of Variation.

DC Data Center.

DH Diffie–Hellman.

DLT Distributed Ledger Technology.

DMR Digital Mobile Radio.

DPI Deep Packet Inspector.

DPP Data Plane Programmability.

ETSI European Telecommunications Standards Institute.

GST Generic Slice Template.

IDS Intrusion Detection System.

IIoT Industrial Internet of Things.

IMS IP Multimedia Subsystem.

IPS Intrusion Prevention System.

IT Information Technology.

LAN Local Area Network.

MANO Management and Orchestration.

MC Mission Critical.

MCX Mission Critical Everything.

ML Machine Learning.

MLFO Machine Learning Function Orchestrator.

NEST NEtwork Slice Type.

NFV Network Function Virtualization.

NS Network Service.

NSD Network Service Descriptor.

NST Network Slice Template.

ODL OpenDaylight.

OSM Open Source MANO.

OT Operational Technology.

P4 Programming Protocol-independent Packet Processors.

PKI Public Key Infrastructure.

PLC Programmable Logic Controller.

PPDR Public Protection and Disaster Relief.

PTT Push-To-Talk.

QoS Quality of Service.

RAN Radio Access Network.

RTP Real-time Transport Protocol.

RTT Round-Trip Time.

SC Smart Contract.

SCADA Supervisory Control And Data Acquisition.

SDN Software Defined Network(ing).

SIP Session Initiation Protocol.

SMO Service Management and Orchestration.

TETRA TErrestrial Trunked RAdio.

UE User Equipment.

URI Uniform Resource Identifier.

VDU Virtual Deployment Unit.

VIM Virtualized Infrastructure Manager.

VLD Virtual Link Descriptor.

VM Virtual Machine.

VNF Virtual Network Function.

VNFD Virtual Network Function Descriptor.

VNFFGD VNF Forwarding Graph Descriptor.

VPN Virtual Private Network.

WAN Wide Area Network.

WIM WAN Infrastructure Manager.

List of Figures

List of Tables

Bibliography

[B1] *Network Functions Virtualisation (NFV); Architectural Framework*. Group Spec. NFV-MAN 002 version 1.2.1. ETSI, Dec. 2014.

[B2] *Cloud iNfrastructure Telco Task Force*. Accessed: Oct. 1, 2024. URL: `https://cntt.readthedocs.io/en/stable-kali/gov/chapters/chapter01.html`.

[B3] T. Zhang, H. Qiu, L. Linguaglossa, W. Cerroni, and P. Giaccone. "NFV Platforms: Taxonomy, Design Choices and Future Challenges". In: *IEEE Transactions on Network and Service Management* 18.1 (2021), pp. 30–48.

[B4] E. F. Kfoury, J. Crichigno, and E. Bou-Harb. "An Exhaustive Survey on P4 Programmable Data Plane Switches: Taxonomy, Applications, Challenges, and Future Trends". In: *IEEE Access* 9 (2021), pp. 87094–87155.

[B5] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck. "Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies, and Solutions". In: *IEEE Communications Surveys & Tutorials* 20.3 (2018), pp. 2429–2453.

[B6] A. A. Barakabitze, A. Ahmad, R. Mijumbi, and A. Hines. "5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges". In: *Computer Networks* 167 (Feb. 2020), p. 106984. URL: `https://doi.org/10.1016/j.comnet.2019.106984`.

[B7] T. Taleb, B. Mada, M.-I. Corici, A. Nakao, and H. Flinck. "PERMIT: Network Slicing for Personalized 5G Mobile Telecommunications". In: *IEEE Communications Magazine* 55.5 (2017), pp. 88–93.

[B8] E. Schiller, N. Nikaein, R. Favraud, K. Kostas, D. Stavropoulos, S. Alyafawi, Z. Zhao, T. Braun, and T. Korakis. "Network Store: Exploring Slicing in Future 5G Networks". In: Sept. 2015.

[B9] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina. "Network Slicing in 5G: Survey and Challenges". In: *IEEE Communications Magazine* 55.5 (2017), pp. 94–100.

[B10] S. Abe, G. Hasegawa, and M. Murata. "Effects of C/U Plane Separation and Bearer Aggregation in Mobile Core Network". In: *IEEE Transactions on Network and Service Management* 15.2 (2018), pp. 611–624.

[B11] J. Pérez-Romero, I. Vilà, O. Sallent, B. Blanco, A. Sanchoyerto, R. Solozábal, and F. Liberal. "Supporting Mission Critical Services through Radio Access Network Slicing". In: *2019 International Conference on Information and Communication Technologies for Disaster Management (ICT-DM)*. 2019, pp. 1–8.

[B12] C. Bektas, S. Bocker, F. Kurtz, and C. Wietfeld. "Reliable Software-Defined RAN Network Slicing for Mission-Critical 5G Communication Networks". In: *2019 IEEE Globecom Workshops (GC Wkshps)*. 2019, pp. 1–6.

[B13] E. Obiodu and M. Giles. *The 5G era: Age of boundless connectivity and intelligent automation*. GSMA, 2017.

[B14] *Framework of network virtualization for future networks*. Standard Series Y: Global information infrastructure, Internet protocol aspects and next-generation networks. ITU-T, Jan. 2012.

[B15] *Description of Network Slicing Concept*. Deliverable version 1.0.8. NGMN Alliance, Sept. 2016.

[B16] *An Introduction to Network Slicing*. GSMA, Sept. 2020.

[B17] *5G; System architecture for the 5G System (5GS)*. Tech. Spec. 23.501 version 16.6.0. 3GPP, Oct. 2020.

[B18] G. Baldini, S. Karanasios, D. Allen, and F. Vergari. "Survey of Wireless Communication Technologies for Public Safety". In: *IEEE Communications Surveys & Tutorials* 16.2 (2014), pp. 619–641.

[B19] R. Fantacci, F. Gei, D. Marabissi, and L. Micciullo. "Public safety networks evolution toward broadband: sharing infrastructures and spectrum with commercial systems". In: *IEEE Communications Magazine* 54.4 (2016), pp. 24–30.

[B20] D. Lund (ed.) *EU Interoperable Broadband Communication Applications and Technology for Public Safety: Final Definition of the Transition Roadmap and PCP Specification*. Broadmap project deliverable D5.2. Apr. 2017. URL: `http://www.broadmap.eu`.

[B21] *Mission Critical Services in 3GPP*. Accessed: Oct. 1, 2024. URL: `https://www.3gpp.org/news-events/1875-mc_services`.

[B22] *From Vertical Industry Requirements to Network Slice Characteristics*. GSMA, Aug. 2018.

[B23] *Aspects; Management and orchestration; Concepts, use cases and requirements*. Tech. Spec. 28.530 version 15.3.0. 3GPP, Dec. 2019.

[B24] *Study on management and orchestration of network slicing for next generation network*. Tech. Rep. 28.801 version 15.1.0. 3GPP, Dec. 2020.

[B25] *OpenStack*. Accessed: Oct. 1, 2024. URL: `https://www.openstack.org`.

[B26] *Open Source MANO*. Accessed: Oct. 1, 2024. URL: `https://osm.etsi.org/`.

[B27] *Network Functions Virtualisation (NFV) Release 3; Protocols and Data Models; NFV descriptors based on YANG Specification*. Group Spec. NFV-SOL 006 V3.6.1. ETSI, Mar. 2022.

[B28] *Open vSwitch*. Accessed: Jun. 28, 2022. URL: https://www.openvswitch.org.

[B29] *ONOS*. Accessed: Jun. 28, 2022. URL: https://wiki.onosproject.org/display/ONOS/Downloads.

[B30] *Open5GS*. Accessed: Jun. 28, 2022. URL: https://open5gs.org/.

[B31] *LTE broadband solutions*. Accessed: Oct. 1, 2024. URL: https://www.leonardo.com/documents/15646808/16735768/CSP-MCX+broadband+MCC+platform+LQ+(mm09006).pdf.

[B32] *LTE; Mission Critical Communication Interworking with Land Mobile Radio Systems*. Tech. Spec. 123 283 version 15.1.0. ETSI, July 2018.

[B33] *UERANSIM*. Accessed: Jun. 28, 2022. URL: https://github.com/aligungr/UERANSIM.

[B34] *ONOS Intent Framework*. Accessed: Jun. 28, 2022. URL: https://wiki.onosproject.org/display/ONOS/Intent+Framework#IntentFramework-Intents.

[B35] Y. Lu. "Industry 4.0: A survey on technologies, applications and open research issues". In: *Journal of industrial information integration* 6 (2017), pp. 1–10.

[B36] K. E. Hemsley, E. Fisher, et al. *History of industrial control system cyber incidents*. 2018.

[B37] T. Miller, A. Staves, S. Maesschalck, M. Sturdee, and B. Green. "Looking back to look forward: Lessons learnt from cyber-attacks on Industrial Control Systems". In: *International Journal of Critical Infrastructure Protection* 35 (2021), p. 100464.

[B38] R. Piggin. "Development of industrial cyber security standards: IEC 62443 for SCADA and Industrial Control System security". In: *IET conference on control and automation 2013: Uniting problems and solutions*. IET. 2013, pp. 1–6.

[B39] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund. "Industrial Internet of Things: Challenges, Opportunities, and Directions". In: *IEEE Transactions on Industrial Informatics* 14.11 (2018), pp. 4724–4734.

[B40] L. Zhou, K.-H. Yeh, G. Hancke, Z. Liu, and C. Su. "Security and Privacy for the Industrial Internet of Things: An Overview of Approaches to Safeguarding Endpoints". In: *IEEE Signal Processing Magazine* 35.5 (2018), pp. 76–87.

[B41] S. Mantravadi, R. Schnyder, C. Møller, and T. D. Brunoe. "Securing IT/OT Links for Low Power IIoT Devices: Design Considerations for Industry 4.0". In: *IEEE Access* 8 (2020), pp. 200305–200321.

[B42] D. Kelly and M. Hammoudeh. "Optimisation of the Public Key Encryption Infrastructure for the Internet of Things". In: *Proceedings of the 2nd International Conference on Future Networks and Distributed Systems*. ICFNDS '18. Amman, Jordan: Association for Computing Machinery, 2018. URL: `https://doi.org/10.1145/3231053.3231098`.

[B43] S. Symington, W. Polk, and M. Souppaya. *Trusted Internet of Things (IoT) Device Network-Layer Onboarding and Lifecycle Management (Draft)*. Tech. rep. Nist, 2020.

[B44] F. Kohnhäuser, D. Meier, F. Patzer, and S. Finster. "On the Security of IIoT Deployments: An Investigation of Secure Provisioning Solutions for OPC UA". In: *IEEE Access* 9 (2021), pp. 99299–99311.

[B45] *Suricata — Open Source IDS / IPS / NSM engine*. Accessed: Oct. 1, 2024. URL: `https://suricata.io/`.

[B46] *Modbus protocol specification*. Accessed: Oct. 12, 2024. URL: `https://modbus.org/specs.php`.

[B47] D. Pliatsios, P. Sarigiannidis, T. Lagkas, and A. G. Sarigiannidis. "A Survey on SCADA Systems: Secure Protocols, Incidents, Threats and Tactics". In: *IEEE Communications Surveys Tutorials* 22.3 (2020), pp. 1942–1976.

[B48] P. Huitsing, R. Chandia, M. Papa, and S. Shenoi. "Attack taxonomies for the Modbus protocols". In: *International Journal of Critical Infrastructure Protection* 1 (2008), pp. 37–44. URL: https://www.sciencedirect.com/science/article/pii/S187454820800005X.

[B49] M. K. Bahare, A. Gavras, M. Gramaglia, J. Cosmas, X. Li, Ö. Bulakci, A. Rahman, A. Kostopoulos, A. Mesodiakaki, D. Tsolkas, M. Ericson, M. Boldi, M. Uusitalo, M. Ghoraishi, and P. Rugeland. *The 6G Architecture Landscape - European perspective*. Feb. 2023. URL: https://doi.org/10.5281/zenodo.7313232.

[B50] M. Giordani, M. Polese, M. Mezzavilla, S. Rangan, and M. Zorzi. "Toward 6G Networks: Use Cases and Technologies". In: *IEEE Communications Magazine* 58.3 (2020), pp. 55–61.

[B51] *Unlocking Digital Transformation with Autonomous Networks*. White Paper No. 56. ETSI, Mar. 2023.

[B52] A. Banchs, M. Fiore, A. Garcia-Saavedra, and M. Gramaglia. "Network intelligence in 6G: challenges and opportunities". In: *Proceedings of the 16th ACM Workshop on Mobility in the Evolving Internet Architecture*. MobiArch '21. New Orleans, Louisiana: Association for Computing Machinery, 2021, pp. 7–12. URL: https://doi.org/10.1145/3477091.3482761.

[B53] A. Wassington, L. Velasco, L. Gifre, and M. Ruiz. "Implementing a Machine Learning Function Orchestration". In: *2021 European Conference on Optical Communication (ECOC)*. 2021, pp. 1–4.

[B54] K. Antevski and C. J. Bernardos. "Federation of 5G services using distributed ledger technologies". In: *Internet Technol-*

ogy Letters 3.6 (2020), e193. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/itl2.193.

[B55] S. Shalunov, W. Roome, R. Woundy, S. Previdi, S. Kiesel, R. Alimi, R. Penno, and Y. R. Yang. *Application-Layer Traffic Optimization (ALTO) Protocol*. RFC 7285. Sept. 2014. URL: https://www.rfc-editor.org/info/rfc7285.

[B56] *OSM client library*. Accessed: Oct. 1, 2024. URL: https://osm.etsi.org/gitlab/osm/osmclient.

[B57] *P4$_{16}$ Language Specification*. Accessed: Oct. 20, 2024. URL: https://p4.org/wp-content/uploads/2024/10/P4-16-spec-v1.2.5.html.

[B58] *Advanced Encryption Standard (AES)*. NIST FIPS 197-upd1. National Institute of Standards and Technology, May 2023.

[B59] *P4 software switch bmv2*. Accessed: Oct. 1, 2024. URL: https://github.com/p4lang/behavioral-model.

[B60] G. F. Pittalà, L. Rinieri, A. Al Sadi, G. Davoli, A. Melis, M. Prandini, and W. Cerroni. "Leveraging Data Plane Programmability to enhance service orchestration at the edge: A focus on industrial security". In: *Computer Networks* (2024), p. 110397.

[B61] *P4Runtime Specification*. Accessed: Oct. 1, 2024. URL: https://p4.org/p4-spec/p4runtime/v1.3.0/P4Runtime-Spec.html.

[B62] D. K. Gillmor. *Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS)*. RFC 7919. Aug. 2016. URL: https://www.rfc-editor.org/info/rfc7919.

[B63] *strongSwan*. Accessed: Oct. 1, 2024. URL: https://www.strongswan.org/.

[B64] *Charms.osm: a python library to develop charms for Open Source MANO*. Accessed: Oct. 1, 2024. URL: https://github.com/charmed-osm/charms.osm.

Publications

[P1] C. Grasselli, A. Melis, L. Rinieri, D. Berardi, G. Gori, and A. A. Sadi. "An Industrial Network Digital Twin for enhanced security of Cyber-Physical Systems". In: *2022 International Symposium on Networks, Computers and Communications (ISNCC)*. 2022, pp. 1–7.

[P2] D. Borsatti, C. Grasselli, C. Contoli, L. Micciullo, L. Spinacci, M. Settembre, W. Cerroni, and F. Callegati. "Mission Critical Communications Support With 5G and Network Slicing". In: *IEEE Transactions on Network and Service Management* 20.1 (2023), pp. 595–607.

[P3] N. Di Cicco, A. Al Sadi, C. Grasselli, A. Melis, G. Antichi, and M. Tornatore. "Poster: Continual Network Learning". In: *Proceedings of the ACM SIGCOMM 2023 Conference*. ACM SIGCOMM '23. New York, NY, USA: Association for Computing Machinery, 2023, pp. 1096–1098. URL: https://doi.org/10.1145/3603269.3610855.

[P4] C. Grasselli, A. Melis, R. Girau, and F. Callegati. "A Digital Twin for Enhanced Cybersecurity in Connected Vehicles". In: *2023 23rd International Conference on Transparent Optical Networks (ICTON)*. 2023, pp. 1–4.

[P5] R. Bacca, C. Grasselli, G. Tripi, A. Melis, L. H. Bonani, and F. Callegati. "Software-Defined and Secure Industrial Networks for the Industry 4.0". In: *2024 24th International Conference on Transparent Optical Networks (ICTON)*. 2024, pp. 1–4.

[P6] P. González, A. Zahir, C. Grasselli, A. Muñiz, M. Groshev, S. Barzegar, F. Callegati, D. Careglio, M. Ruiz, and L. Velasco. "Deployment of Secure Machine Learning Pipelines for Near-Real-Time Control of 6G Network Services". In: *Optical Fiber Communication Conference (OFC) 2024*. Optica Publishing Group, 2024, M3Z.8. URL: https://opg.optica.org/abstract.cfm?URI=OFC-2024-M3Z.8.

[P7] C. Grasselli, L. Rinieri, P. González, L. Velasco, D. Careglio, and F. Callegati. "In-network Computing for Secure Distributed AI". In: *2024 24th International Conference on Transparent Optical Networks (ICTON)*. 2024, pp. 1–4.