Rheinland-Pfälzische
Technische Universität
Kaiserslautern
Landau

Department of Electrical and Computer Engineering
Division of Wireless Communications and Radio Navigation

Intelligent Management and Orchestration Solutions for Network Slicing in Radio Access Network Architecture

Doctoral Dissertation approved by the Department of Electrical and Computer Engineering at Rheinland-Pfälzische Technische Universität Kaiserslautern-Landau for the award of the degree of Doctor of Engineering (Dr.-Ing.)

Written By: Mohammad Asif Habibi (Born in Baghlan, Afghanistan)

Advisor: Prof. Dr.-Ing. Hans Dieter Schotten[1, 2]

[1]Rheinland-Pfälzische Technische Universität (RPTU) Kaiserslautern-Landau

[2]German Research Center for Artificial Intelligence (DFKI)

Co-advisor: Prof. Dr. Xavier Costa-Pérez[3, 4]

[3]Universitat Politécnica de Catalunya (UPC)

[4]NEC Laboratories Europe

Date of Defense: 2 December 2025

DE 386

Kaiserslautern, December 2025

To my parents, Abo and Kaka.

Declaration of Authorship

I, **Mohammad Asif Habibi**, hereby declare that the work presented in this thesis, entitled "Intelligent Management and Orchestration Solutions for Network Slicing in Radio Access Network Architecture," is entirely my own. I confirm that:

- This work was completed wholly or predominantly during my candidature for the degree of Doctor of Engineering (Dr.-Ing.) at the Rheinland-Pfälzische Technische Universität (RPTU) Kaiserslautern-Landau.
- Where any part of this thesis has previously been submitted for a degree or other qualification at the Rheinland-Pfälzische Technische Universität Kaiserslautern-Landau or any other institution, this has been clearly stated.
- Wherever I have consulted the published work of others, it has been properly and clearly acknowledged.
- Wherever I have quoted from the work of others, the source has been explicitly cited. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all significant sources of assistance.
- Parts of the research presented in this thesis have previously been published in IEEE conferences, journals, and magazines. These works have been appropriately revised, proofread, and incorporated into this thesis.
- This thesis includes only research for which I was the primary author.

Signed: Mohammad Asif Habibi

Final Submission Date: 23 June 2025

Acknowledgements

Firstly, it is my great pleasure to express my sincere gratitude and deepest appreciation to my primary advisor, Prof. Dr.-Ing. Hans D. Schotten, for his invaluable guidance, continuous support, patience, and keen interest in my research throughout my Dr.-Ing. studies and beyond. His insightful advice, encouragement, and unwavering support have been instrumental in the successful completion of this thesis.
I would also like to extend my sincere appreciation to my secondary supervisor, Prof. Dr. Xavier Costa-Pérez, for his valuable feedback, constructive discussions, and insightful suggestions, which have significantly contributed to this work. I am especially grateful for his support in shaping and reinforcing the direction of this research.
Secondly, I would like to acknowledge the European Commission and the German Research Foundation for their financial support over the past years, which made this work possible. I am particularly grateful to Prof. Dr.-Ing. Hans D. Schotten for his continuous support in securing and facilitating this funding. His encouragement and trust have played a crucial role in my academic journey at the RPTU.
Finally, I would like to express my heartfelt gratitude to my family, relatives, friends, and colleagues for their unwavering support, encouragement, and love. Their belief in me has been a constant source of strength and motivation throughout this journey. This achievement would not have been possible without them, and I hope it makes them proud.

Abstract

The next-generation radio access network (NG-RAN) architecture has been standardized by the Third Generation Partnership Project (3GPP) as the radio access network (RAN) architecture for fifth-generation (5G) mobile networks. It consists of a set of next-generation NodeBs (gNBs). Each gNB is composed of a centralized unit (CU), at least one distributed unit (DU), and at least one radio unit (RU). These components of a gNB can be deployed as virtual network functions (VNFs) and/or physical network functions (PNFs). In this thesis, we consider the CU and the DU as VNFs and the RU as a PNF. The CU, DU, and RU can be mapped onto the underlying aggregation data center, edge data center, and cellular network site in the NG-RAN architecture, respectively.

The Open Radio Access Network (O-RAN) Alliance has redefined the NG-RAN architecture by establishing open and standard-compliant interfaces. The architecture defined by the O-RAN Alliance, known as the O-RAN architecture, comprises several components that interoperate to create a flexible, cloud-based RAN for 5G and beyond. In addition to the components specified by 3GPP for NG-RAN, the O-RAN Alliance introduces a new component for the management and orchestration of O-RAN elements. This component is referred to as the service management and orchestration (SMO) framework. The SMO framework is responsible for deploying and operating RAN services and coordinating the various O-RAN components. It includes a non-real-time RAN intelligent controller (Non-RT RIC) and can incorporate components from other standards-developing organizations (SDOs).

Moreover, the O-RAN architecture features the near-real-time RAN intelligent controller (Near-RT RIC), which performs tasks within a near-real-time time frame. The Near-RT RIC, along with other gNB functions, can be mapped onto the underlying infrastructure. This infrastructure includes open-cloud (O-Cloud) sites, which provide the cloud environment for hosting these functions. The O-Cloud also features a notification interface for receiving relevant events. Overall, the O-RAN architecture aims to enhance flexibility, interoperability, and cloud-native capabilities in the deployment and operation of RAN systems, thereby supporting the evolution of 5G and beyond.

The mapping of VNFs onto the underlying physical network infrastructure at the edge of a cellular network is a challenging task due to the joint allocation of virtual compute, storage, and networking resources across nodes and links, the diverse technical requirements of end users, and the need for coordination across multiple host domains. This issue is further complicated by the provisioning of RAN slicing, given the varying characteristics of wireless communication channels. To this end, this thesis addresses the mapping and virtual resource allocation problems of the VNFs of RAN slice subnets onto the underlying intelligent network infrastructure in NG-RAN. In this context, unlike most prior proposals that often fail to meet performance objectives and overlook resource allocation constraints, this thesis introduces and employs automation and intelligent techniques to map VNFs onto their corresponding physical nodes, with the aim of achieving improved efficiency in virtual resource utilization while ensuring the performance of RAN slice subnets in NG-RAN.

Adopting a top-down approach, the key contributions of this thesis are as follows:

- extend the framework of network slicing, as defined by the Next Generation Mobile Networks (NGMN) Alliance, to the NG-RAN architecture, and provide a critical analysis and overview of the components and functionalities of different types of RAN slices;

- integrate the Experiential Network Intelligence (ENI) framework, as proposed by the European Telecommunications Standards Institute (ETSI), into a joint architecture of network functions virtualization–management and orchestration (NFV–MANO), also proposed by ETSI, and the Third Generation Partnership Project network slicing management system (3GPP-NSMS), in order to introduce automation and intelligence into the management and orchestration of different types of RAN slices in NG-RAN;

- propose a learning-assisted solution (consisting of three phases: virtual resource automation, virtual resource management, and virtual resource allocation) for mapping the VNFs of a RAN slice subnet onto the underlying intelligent data centers in NG-RAN;

- unify the management and orchestration components of 3GPP, ETSI, and the O-RAN Alliance within the SMO framework to enhance the unification and interoperability of various standard-compliant interfaces within the O-RAN architecture;

- integrate management data analytics (MDA) into the management systems of 3GPP, ETSI, and the O-RAN Alliance within the SMO framework, with the goal of introducing intelligence and automation into the functionalities of the SMO framework for managing and orchestrating O-RAN components;

- explore several deployment scenarios for integrating MDA and automation into the SMO framework, thereby providing network operators with multiple deployment options for enabling intelligence within the SMO framework;

- present a comprehensive system model based on which the mapping problem of the CU and DU, internal and external VLs, and the VNFs of a RAN slice subnet onto the underlying infrastructure is mathematically formulated;

- discuss various types of machine learning (ML)-assisted algorithms, such as supervised, unsupervised, and reinforcement learning, with a particular focus on the mathematical background and application of reinforcement learning, and select the Q-learning algorithm to solve the VNF mapping problem;

- describe the simulation environment (including the simulation setup, network topology, and simulation parameters) considered for simulating the virtual components involved in mapping different types of RAN slices in NG-RAN;

- obtain simulation results using the Q-learning algorithm for the mapping of RAN slice subnets, demonstrating significant performance improvements in mapping various types of RAN slice subnets onto the underlying infrastructure under different conditions;

- evaluate the performance objectives achieved using the Q-learning algorithm against the service level agreement (SLA); and

- provide findings on the global resource allocation required to host a large number of RAN slice subnets in the underlying infrastructure, as well as highlight the advantages of employing Q-learning for VNF mapping compared to other state-of-the-art algorithms.

Keywords – 5G, automation and intelligence, beyond 5G, machine learning, management and orchestration, mapping the virtual services, network slice, network slicing, RAN slicing, vertical industries, virtual resource allocation, virtualization

Zusammenfassung

Die Architektur des Next-Generation Radio Access Network (NG-RAN) wurde vom Third Generation Partnership Project (3GPP) als die Funkzugangsnetzarchitektur (RAN) für Mobilfunknetze der fünften Generation (5G) standardisiert. Sie besteht aus einer Menge von Next-Generation NodeBs (gNBs). Jede gNB setzt sich aus einer zentralisierten Einheit (Centralized Unit, CU), mindestens einer verteilten Einheit (Distributed Unit, DU) und mindestens einer Funkeinheit (Radio Unit, RU) zusammen. Diese Komponenten einer gNB können als virtuelle Netzwerkfunktionen (Virtual Network Functions, VNFs) und/oder physische Netzwerkfunktionen (Physical Network Functions, PNFs) realisiert werden. In dieser Arbeit werden die CU und die DU als VNFs und die RU als PNF betrachtet. Die CU, DU und RU können jeweils auf das zugrunde liegende Aggregationsrechenzentrum, das Edge-Rechenzentrum sowie den Mobilfunkstandort innerhalb der NG-RAN-Architektur abgebildet werden.

Die Open Radio Access Network (O-RAN) Alliance hat die NG-RAN-Architektur durch die Einführung offener und standardkonformer Schnittstellen neu definiert. Die von der O-RAN Alliance definierte Architektur, bekannt als O-RAN-Architektur, umfasst mehrere Komponenten, die zusammenwirken, um ein flexibles, cloudbasiertes RAN für 5G und darüber hinaus zu ermöglichen. Zusätzlich zu den von 3GPP für NG-RAN spezifizierten Komponenten führt die O-RAN Alliance eine neue Komponente für das Management und die Orchestrierung von O-RAN-Elementen ein. Diese Komponente wird als Service Management and Orchestration (SMO)-Framework bezeichnet. Das SMO-Framework ist für die Bereitstellung und den Betrieb von RAN-Diensten sowie für die Koordination der verschiedenen O-RAN-Komponenten verantwortlich. Es umfasst einen Non-Real-Time RAN Intelligent Controller (Non-RT RIC) und kann Komponenten anderer Standardisierungsorganisationen (Standards Developing Organizations, SDOs) integrieren.

Darüber hinaus umfasst die O-RAN-Architektur den Near-Real-Time RAN Intelligent Controller (Near-RT RIC), der Aufgaben innerhalb eines nahezu echtzeitnahen Zeitrahmens ausführt. Der Near-RT RIC kann zusammen mit weiteren gNB-Funktionen auf die zugrunde liegende Infrastruktur abgebildet werden. Diese Infrastruktur umfasst Open-Cloud (O-Cloud)-Standorte, die die Cloud-Umgebung zur Ausführung dieser Funktionen bereitstellen. Die O-Cloud verfügt außerdem über eine Benachrichtigungsschnittstelle zum Empfang relevanter Ereignisse. Insgesamt zielt die O-RAN-Architektur darauf ab, Flexibilität, Interoperabilität und cloud-native Fähigkeiten bei der Bereitstellung und dem Betrieb von RAN-Systemen zu verbessern und damit die Weiterentwicklung von 5G und darüber hinaus zu unterstützen.

Die Abbildung von VNFs auf die zugrunde liegende physische Netzwerkinfrastruktur am Rand eines Mobilfunknetzes stellt eine anspruchsvolle Aufgabe dar. Dies liegt an der gemeinsamen Zuweisung virtueller Rechen-, Speicher- und Netzwerkressourcen über Knoten und Verbindungen hinweg, an den vielfältigen technischen Anforderungen der Endnutzer sowie an der notwendigen Koordination über mehrere Host-Domänen hinweg. Diese Problematik wird durch die Bereitstellung von RAN-Slicing zusätzlich erschwert, da drahtlose Kommunikationskanäle unterschiedliche Eigenschaften aufweisen. Vor diesem Hintergrund

behandelt diese Arbeit die Probleme der Abbildung und der Zuweisung virtueller Ressourcen für die VNFs von RAN-Slice-Subnetzen auf die zugrunde liegende intelligente Netzwerkinfrastruktur in NG-RAN. Im Gegensatz zu den meisten bisherigen Ansätzen, die häufig die Leistungsziele nicht erreichen und Ressourcenrestriktionen vernachlässigen, werden in dieser Arbeit Automatisierungs- und intelligente Verfahren eingeführt und eingesetzt, um VNFs auf ihre entsprechenden physischen Knoten abzubilden. Ziel ist es, die Effizienz der Nutzung virtueller Ressourcen zu verbessern und gleichzeitig die Leistungsanforderungen der RAN-Slice-Subnetze in NG-RAN sicherzustellen.
Unter Anwendung eines Top-Down-Ansatzes lassen sich die wesentlichen Beiträge dieser Arbeit wie folgt zusammenfassen:

- Erweiterung des von der Next Generation Mobile Networks (NGMN) Alliance definierten Network-Slicing-Rahmens auf die NG-RAN-Architektur sowie Bereitstellung einer kritischen Analyse und eines Überblicks über die Komponenten und Funktionalitäten verschiedener Typen von RAN-Slices;

- Integration des vom European Telecommunications Standards Institute (ETSI) vorgeschlagenen Experiential Network Intelligence (ENI)-Frameworks in eine gemeinsame Architektur aus Network Functions Virtualization–Management and Orchestration (NFV–MANO), ebenfalls von ETSI vorgeschlagen, und dem Third Generation Partnership Project Network Slicing Management System (3GPP-NSMS), mit dem Ziel, Automatisierung und Intelligenz in das Management und die Orchestrierung verschiedener Typen von RAN-Slices in NG-RAN einzubringen;

- Vorschlag einer lernunterstützten Lösung (bestehend aus drei Phasen: virtuelle Ressourcenautomatisierung, virtuelles Ressourcenmanagement und virtuelle Ressourcenallokation) zur Abbildung der VNFs eines RAN-Slice-Subnetzes auf die zugrunde liegenden intelligenten Rechenzentren in NG-RAN;

- Vereinheitlichung der Management- und Orchestrierungskomponenten von 3GPP, ETSI und der O-RAN Alliance innerhalb des SMO-Frameworks zur Verbesserung der Vereinheitlichung und Interoperabilität verschiedener standardkonformer Schnittstellen innerhalb der O-RAN-Architektur;

- Integration von Management Data Analytics (MDA) in die Managementsysteme von 3GPP, ETSI und der O-RAN Alliance innerhalb des SMO-Frameworks mit dem Ziel, Intelligenz und Automatisierung in die Funktionalitäten des SMO-Frameworks zur Verwaltung und Orchestrierung von O-RAN-Komponenten einzubringen;

- Untersuchung mehrerer Einsatzszenarien zur Integration von MDA und Automatisierung in das SMO-Framework, um Netzbetreibern verschiedene Implementierungsoptionen zur Intelligenzsteigerung des SMO-Frameworks bereitzustellen;

- Entwicklung eines umfassenden Systemmodells, auf dessen Grundlage das Abbildungsproblem der CU und DU, der internen und externen virtuellen Verbindungen (VLs) sowie der VNFs eines RAN-Slice-Subnetzes auf die zugrunde liegende Infrastruktur mathematisch formuliert wird;

- Diskussion verschiedener Arten von maschinellen Lernverfahren (ML), wie überwachtes, unüberwachtes und bestärkendes Lernen, mit besonderem Fokus auf die mathematischen Grundlagen und die Anwendung des bestärkenden Lernens, sowie Auswahl des Q-Learning-Algorithmus zur Lösung des VNF-Abbildungsproblems;

- Beschreibung der Simulationsumgebung (einschließlich Simulationsaufbau, Netzwerktopologie und Simulationsparameter), die zur Simulation der virtuellen Komponenten für die Abbildung verschiedener Typen von RAN-Slices in NG-RAN verwendet wird;
- Gewinnung von Simulationsergebnissen unter Verwendung des Q-Learning-Algorithmus für die Abbildung von RAN-Slice-Subnetzen, die eine signifikante Leistungsverbesserung bei der Abbildung verschiedener Typen von RAN-Slice-Subnetzen auf die zugrunde liegende Infrastruktur unter unterschiedlichen Bedingungen zeigen;
- Bewertung der erreichten Leistungsziele durch den Einsatz des Q-Learning-Algorithmus im Vergleich zu den Service Level Agreements (SLAs); sowie
- Darstellung von Ergebnissen zur globalen Ressourcenallokation für das Hosting einer großen Anzahl von RAN-Slice-Subnetzen auf der zugrunde liegenden Infrastruktur sowie Hervorhebung der Vorteile des Einsatzes von Q-Learning für die VNF-Abbildung im Vergleich zu anderen Verfahren auf dem neuesten Stand der Technik.

Schlüsselwörter – 5G, Automatisierung und Intelligenz, Beyond 5G, maschinelles Lernen, Management und Orchestrierung, Abbildung virtueller Dienste, Network Slice, Network Slicing, RAN Slicing, vertikale Industrien, virtuelle Ressourcenallokation, Virtualisierung

Contents

List of Figures

List of Tables

List of Abbreviations

Acronyms

3GPP Third Generation Partnership Project

3GPP-NSMS 3GPP-network slicing management system

5G fifth generation

5GC 5G core network

5GPPP Fifth Generation Public-Private Partnership

6G sixth generation

AAF Authentication and Authorization Function

AI artificial intelligence

AIaaS AI as a service

AMF access and mobility management function

API application programming interface

AUC area under the curve

C-RAN cloud RAN

CAPEX capital expenditure

CCM container infrastructure service cluster management

CCPA California Consumer Privacy Act

CIR container image registry

CIS container infrastructure service

CISM container infrastructure service management

CLA closed loop automation

CN core network

CNF containerized network function

CP control plane

CPU central processing unit

CS communication service

CSC CS customer

CSMF communication service management function

CU centralized unit

CVM container in VM

D2D device-to-device

DAS direct attached storage

DataOps data operations

DDPG Deep Deterministic Policy Gradient

DL deep learning

DMEF Data Management and Exposure Function

DQN Deep Q Network

DU distributed unit

E2E end-to-end

eCPRI evolved common public radio interface

EM element manager

eMBB enhanced mobile broadband

ENI Experiential Network Intelligence

ETSI European Telecommunication Standard Institute

F-RAN fog RAN

FB functional block

FCAPS fault, configuration, accounting, performance, and security

FL federated learning

GDPR General Data Protection Regulation

GN-NSI generic NSI

GN-NSIT GN-NSI template

gNB next generation Node B

GSMA Global System for Mobile communications Association

GST Generic Network Slice Template

H-CRAN heterogeneous cloud RAN

HMEE hardware-mediated execution enclave

HMTC high-performance machine-type communications

I-PoP Intelligent PoP

ID identification number

IETF Internet Engineering Task Force

IoT internet of things

ISG Industry Specification Group

ITU International Telecommunication Union

KB Kilobyte

Kbps kilo bits per second

KPI key performance indicator

LCM life cycle management

LTE long term evolution

MAC medium access control

MANO management and orchestration

MCS modulation and coding scheme

MDA management data analytics

MDAF network data and analytics function

MDAS management data analytics service

MF management function

ML machine learning

MLA management level agreement

MLOps machine learning operations

mMTC massive machine type communications

MSE mean square error

MVNO mobile virtual network operator

NAS network attached storage

NC network controller

Near-RT RIC Near-Real-Time RAN intelligence controller

NEF network exposure function

NEST NEtwork Slice Type

NF network function

NFMF network function management function

NFV network function virtualization

NFV-MANO network function virtualization-management and orchestration

NFVI network function virtualized infrastructure

NFVI-PoP network function virtualization infrastructure point of presence

NFVO network function virtualization orchestration

NG next generation

NG-RAN next-generation-RAN

NGMN Next Generation Mobile Networks

NGP Next Generation Protocols

NIC network interface card

NMS network management system

Non-RT RIC Non-Real-Time RAN intelligence controller

NR new radio

NS network slice

NSaaS network slice as a service

NSB network slice broker

NSC network slice catalogue

NSD network service descriptor

NSI network slice instance

NSMF network slice management function

NSMS network slicing management system

NSS network slice subnet

NSSAI network slice selection assistance information

NSSI network subnet slice instance

NSSMF network slice subnet management function

NSST network slice subnet template

NST network slice template

O-BH open-backhaul

O-Cloud open-cloud

O-CU open centralized unit

O-DU open distributed unit

O-FH open-fronthaul

O-gNB open next generation node B

O-MH open midhaul

O-RAN Open-RAN

O-RU open radio unit

ONF Open Networking Foundation

OPEX operating expenditure

OSS/BSS operations support system/business support system

PCF policy control function

PDCP packet data convergence protocol

PHY physical

PL physical link

PM physical machine

PNF physical network function

PNFD PNF descriptor

QoE quality of experience

QoS quality of service

RAM random-access memory

RAN radio access network

RATs radio access technologies

RB resource block

RF radio frequency

RL reinforcement learning

RLC radio link control

RRC radio resource control

RU radio unit

S-NSI sub network slice instance

S-NSIT S-NSI template

S-NSSAI single-NSSAI

SA5 Service and System Aspects Working Group 5

SAC Soft Actor-Critic

SAN storage area network

SARSA state-action-reward-state-action

SCEF service capability exposure function

SCF Small Cell Forum

SCS/AS service capability server/application server

SD slice differentiator

SDAP service data adaptation protocol

SDC software-defined compute

SDN software-defined networking

SDO standards development organization

SDR software-defined radio

SDS software-defined storage

SDx software-defined everything

SFC service function chain

SGD Stochastic Gradient Descent

SLA service level agreement

SMEF Service Management and Exposure Function

SMF session management function

SMO service management and orchestration

SON self-organizing network

SRDF Service Registration and Discovery Function

SSD solid state drive

SST slice/service type

TIP Telecom Infra Project

TN transport network

TSG Technical Specification Group

UE user equipment

UP user plane

UPF user-plane function

URLLC ultra-reliable low latency communication

uRLLC ultra-reliable low latency communications

US-NSI use case-specific NSI

US-NSIT US-NSI template

V2I vehicle to infrastructure

V2N vehicle to network

V2P vehicle to pedestrian

V2V vehicle to vehicle

V2X vehicle to everything

VCD virtual computing descriptor

vCPU virtual central processing unit

vCU virtual CU

VDU virtualization deployment unit

vDU virtual DU

VIM virtualized infrastructure manager

VL virtual link

VLD virtual link descriptor

VM virtual machine

vMemory virtual memory

VN virtual networking

VNF virtual network function

VNFC virtual network function component

VNFD VNF descriptor

VNFFG VNF forwarding graph

VNFFGD VNF forwarding graph descriptor

VNFM virtual network function manager

vNIC virtual network interface card

vRAM virtual random-access memory

VSD virtual storage descriptor

vSwitch virtual switch

WIM Wide area network Infrastructure Manager

Chapter 1

Introduction and Thesis Layout

1.1 Motivation and Background

As of this writing, the definition of the fundamental concepts and the design of the architectural framework for fifth generation (5G) mobile communication systems have successfully been concluded, the standardization activities are almost over, and the commercial deployment has already commenced worldwide [3]. The main objectives of the 5G communication systems are to provide significant advances in all aspects of the performance of cellular networks. The performance requirements of 5G cellular networks include – but are not limited to – 1000-fold growth in system capacity; enhanced connectivity to at least 100 billion devices; 10 Gbps maximum and 100 Mbps average individual user experience; prolonged battery life with 1000-fold lower energy consumption per bit; support for 250 km/h mobility for high-speed users; a 3-fold increase in spectrum efficiency; a perception of 99.99% availability; 100% coverage; and latency from 1 to 10 milliseconds [4].

Notwithstanding, the 5G communication system, in contrast to its predecessors, is also aimed at enabling new use cases and applications for various vertical industries [5] for the first time in the history of telecommunication (see Fig. 1.1). These use cases include automobiles, public transportation, medical care, public safety, agriculture, entertainment, manufacturing, and many others. In order to efficiently accommodate such vertical use cases along with increased user demands over a network infrastructure, the 5G system requires architectural optimization and reconstruction with respect to current deployment [4, 6].

Network slicing, proposed by the Next Generation Mobile Networks (NGMN) Alliance in [7], can be one of the appropriate solutions to meet the above-mentioned requirements. It is one of the most prominent features and a revolutionary architectural solution of the 5G system that facilitates the operation of heterogeneous services, myriad broadband consumers, and innumerable vertical industries and use cases over a common telecommunication infrastructure in an elastic, scalable, and dynamic manner [8]. With network slicing, the underlying telecommunication infrastructure is partitioned into a number of virtualized, independent, and mutually isolated logical networks – each referred to as a network slice (NS) – employing network function virtualization (NFV), software-defined networking (SDN), and mobile and/or edge computing technologies.

Every NS is tailored to a single use case – such as NS for holographic telepresence, NS for telemedicine, NS for flying vehicles, and many others – and must efficiently embrace one or more communication services. Each NS has its own architecture, requirements, template, network functions (NFs), and packet and signal processing capacity. It must be designed to provision specific applications and services to specific types of end users. Every NS can consist of VNFs and/or physical network functions (PNFs), which are appropriately composed to

support and build up services that are expected to fulfill the business objectives of end users.

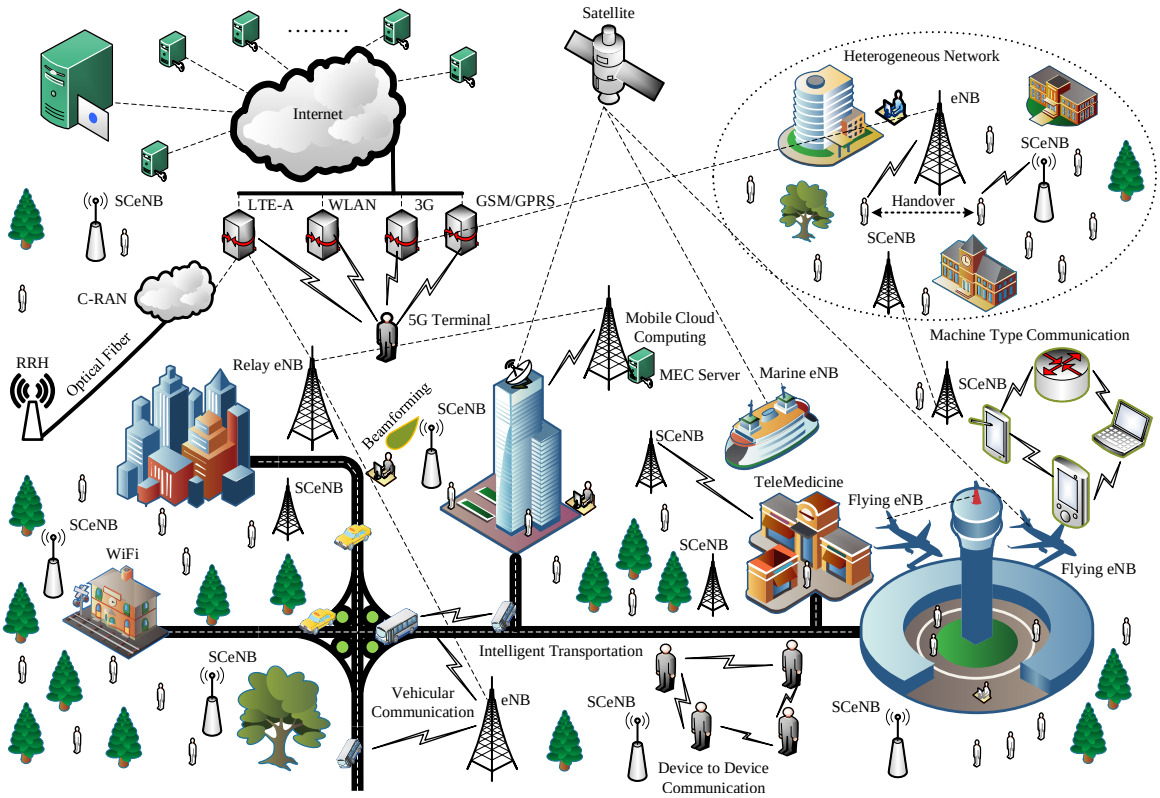

Figure 1.1: An illustration of a number of vertical industries and service sectors that future communication networks are expected to support by means of network slicing.

Each NS is dedicated to deliver one or more communication services (CSs) to one or more CS customers (CSCs) – or tenant – in a structured, elastic, scalable, and automated manner [4]. The performance of NSs is not impacted by any means thanks to the logical and physical isolation between the NSs. Moreover, the CSC is offered with the capability of semi/full control on leased NS via its own set of dedicated northbound application programming interface (API). By means of an API, the tenant can create, activate, modify, deactivate, and decommission the CSs of an NS [9].

1.2 The Types of NSs

Using the quality of service (QoS) attributes as the criteria of classification, the NSs are classified into three types [4]: the enhanced mobile broadband (eMBB) type NS, the ultra-reliable low latency communication (URLLC) type NS, and the massive machine type communications (mMTC) type NS. They are discussed in the following.

- **eMBB type NS:** supports a stable high data-rate connection with high reliability, moderate mobility, high device density, low packet error rate on the order of 10^3. The eMBB type devices send large data payloads in the uplink direction to a given next generation Node B (gNB) at a given time. For example, NS for infotainment applications is an eMBB type NS, which must be configured to offer a short-lived high data rate connection of up to 100 Mbps (for mobile broadband in vehicles) between a vehicle and a nearby gNB.

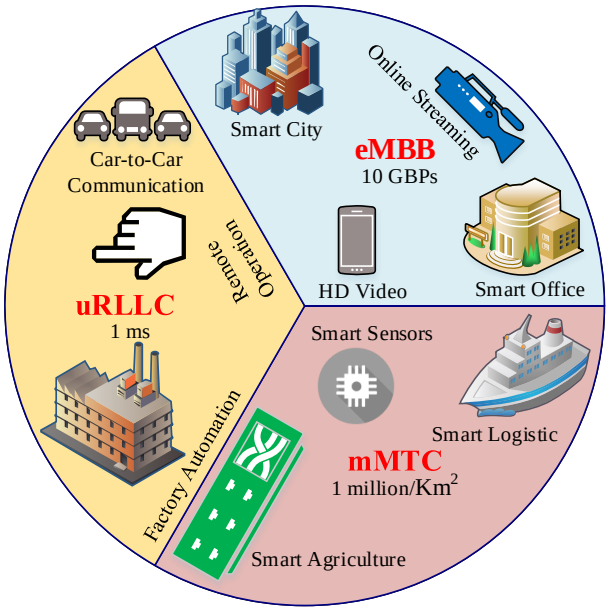

Figure 1.2: The three types of NSs that are supported by the 5G communication system. Each type of NS is individually deployed with configurable characteristics and different levels of isolation and security for only one type of use case.

- **URLLC type NS:** supports data traffic with very high reliability (e.g., packet error rate of around or lower than 10^{-5}) and very low latency transmission of small payloads. Therefore, URLLC traffic is suggested to be processed at the edge of a cellular network. In addition, the URLLC traffic is sporadic; however, the number of transmitters is usually higher than for eMBB. For instance, NS supporting autonomous driving requires ultra-high reliability (99.999%) and ultra-low latency (below 10 ms) in V2V mode over sidelink air-interface (PC5), which could be an URLLC type NS.

- **mMTC type NS:** must be designed to provide connections to a massive number of devices, which are intermittently active and send a small size of data payloads to a gNB. Such devices require no mobility, a low data rate, and a low packet error rate on the order of 10^{-1}. For example, an NS designed for vehicle management and remote diagnostics is an mMTC type NS, which should provide connectivity between a massive number of vehicles and a remote server outside of a communication network.

Each of the aforementioned types of an NS is instantiated based on a template (sometimes also referred to as a blueprint, service profile, or descriptor in various research articles and technical reports), which is a set of attributes with values and ranges, known as the network slice template (NST). The NST defines network resources and functions and their interconnections and configurations to meet the diverse technical requirements of a use case [7]. These attributes are the properties of an NST that could be set to different values [10]. Every attribute plays a significant role in the characterization of an NST. It has its own application and intensity within an NST. For example, latency is the crucial attribute (out of a number of attributes) of an URLLC type NST. Its intensity is extremely vital for the CSs to be provided over an URLLC type NS. However, it is not a critical attribute of an mMTC type NST. Therefore, every attribute should be precisely defined, analyzed, and quantified according to the type of the NST. This, on the one hand, leads to a transparent and constructive bilateral agreement between the tenant and operator, and on the other hand, to an efficient allocation of network resources and the management of NSs.

There are two classes of attributes in a NST: the technical attributes and the business attributes. The former class focuses on the technical aspects of the CSs of an NS, such as bandwidth, latency, device density, security, etc. The latter class includes all the attributes that are related to the service level agreement (SLA) and business aspects of the CSs of an NS such as the start date of the contract, the end date of the contract, the duties of the tenant, the duties of the operator, price, penalty, etc.

1.3 Key Enabling Technologies for Network Slicing

NFV, SDN, and edge/cloud computing are the three key enabling technologies that gave birth to network slicing in 5G communication networks. In this section, we intend to focus on these three core technologies associated with network slicing.

1.3.1 Intelligent Network Function Virtualization

NFV emerged as one of the key technologies, enabling network slicing in 5G. It continues to be an integral part of the beyond 5G and sixth generation (6G) slicing framework [3]. It creates virtual resources, enhances deployment flexibility, and fosters diverse service integration. However, with the deployment of NFV in the early stages of 5G, a large set of diverging VNFs with new considerations - such as sensitivity, priority, transferability, the characteristics of a radio channel, etc. - have been identified, which call for novel intelligent optimization algorithms aimed at coping with network complexity and connection density at the core, edge, and extreme edge of futuristic networks [11].
One such solution is ML, which has already been introduced or employed in various domains (or entities) of virtualized networks, most notably in network automation, resource orchestration, and service management [11, 12]. A large number of such state-of-the-art ML techniques are based on supervised learning, which brings a significant improvement in operational efficiency, high-volume data processing, and capital and operating expenditures. Nevertheless, the implementation of such algorithms has raised many concerns with respect to privacy, security, scalability, and other metrics [13, 12]. Additionally, due to constant changes in network topologies and configurations, it is time-consuming to update the dataset and adapt the supervised ML algorithms to a changing environment [11, 14].
To this end, the integration of reinforcement learning (RL), federated learning (FL), deep learning (DL), etc. into NFV-enabled networks can significantly optimize a wide range of operations and achieve the required level of intelligence in resource scheduling, network elasticity, troubleshooting, etc. while guaranteeing data privacy, network security, and users' secrecy [12]. We provide a list of applications of such learning-assisted algorithms into several operations in Fig. 1.3 aiming to design an intelligent, credible, reliable, and sustainable NFV paradigm for the beyond 5G and foreseen 6G slicing framework. It is worth noting that the integration of ML algorithms has implications for various operations of NFV-enabled networks, which are listed in Fig. 1.3.

1.3.2 Intelligent Software-defined Networking

A large fraction of 5G network technologies has softwarization as core to their realization through software-defined everything (SDx). Softwarization refers to the realization of hardware-based functions using software in order to enable a flexible, reconfigurable, and upgradable implementation of said functions [15]. Softwarization goes hand in hand with virtualization, as only with the application of the latter it is possible to achieve an

abstraction of the underlying hardware into virtual instances. Thus, allowing the utilization of software with virtual machines (VMs) on general-purpose hardware. In SDx-based systems, an abstraction of the underlying hardware into virtual instances coupled with a separation of the control and data planes facilitates management, flexibility, and reconfigurability of the overall network environment. The SDN, software-defined storage (SDS), and software-defined radio (SDR) are a few examples of the technologies falling under the umbrella of SDx. The SDN paradigm has been indispensable for the operation of NS in 5G by decoupling the network into application, control, and data forwarding planes [16]. The SDRs enable an agile front-haul as well as efficient use of the scarce spectrum. The SDS frameworks simplify the administration, provisioning, and maintenance of cloud storage environments.

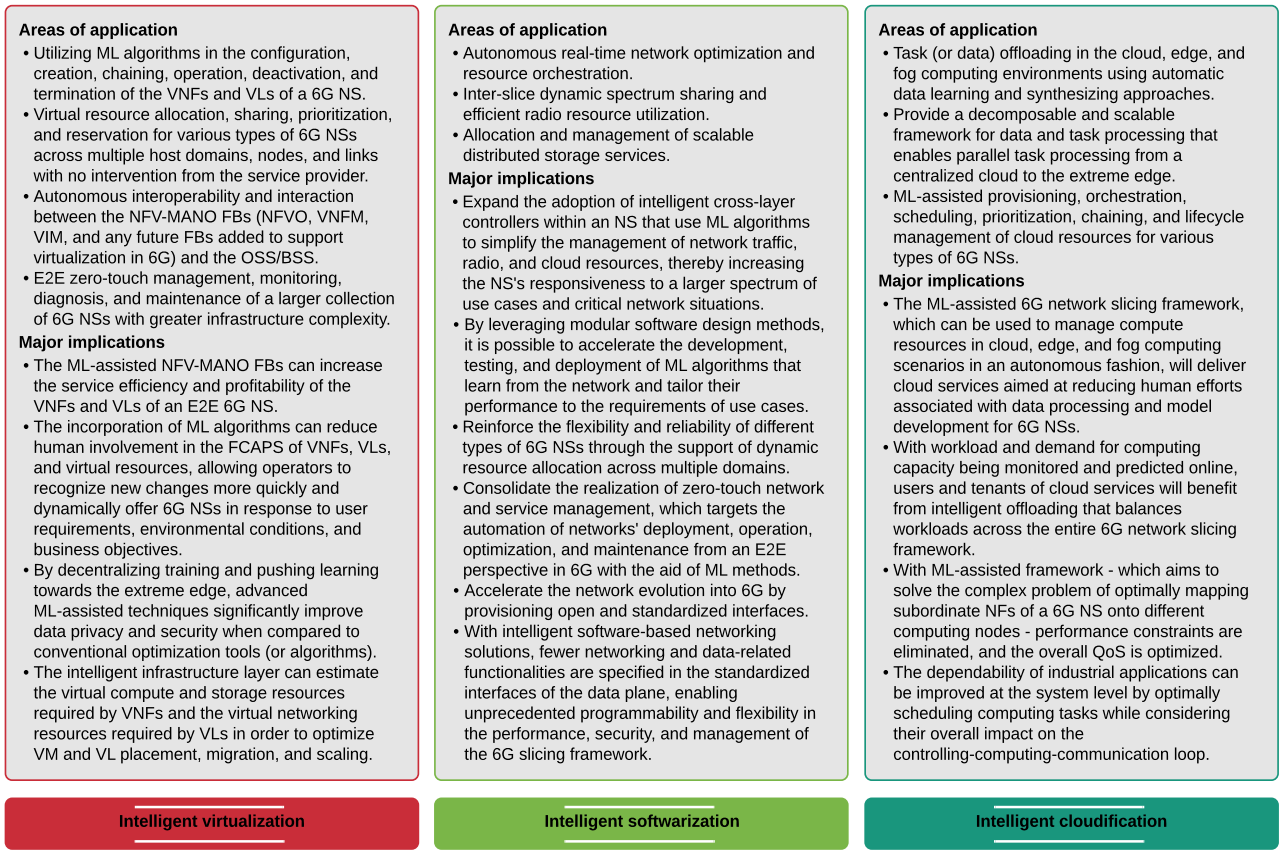

Figure 1.3: The implications and application areas of incorporating automation and intelligence into the three key enabling technologies that facilitate the deployment of open and intelligent network slicing in 5G, beyond 5G, and even 6G network.

The common factor among the previously mentioned SDx solutions is the split between the control layer and data-handling hardware [16]. This split empowers the anticipated evolution of the radio access network (RAN) in beyond 5G and 6G into a cognitive network by leveraging advanced ML algorithms at the controller with a centralized view of the network to manage, maintain, and optimize the underlying infrastructure. The SDx solutions coupled with ML, will also help with the development of intelligent routing protocols with better latency and reliable performance, as well as with the integration and deployment of more secure APIs.
Given this intelligence, slicing framework in beyond 5G and 6G will continue to learn from the traffic and data it is carrying, adapt to the ever-demanding and heterogeneous use-case requirements, and enhance its performance to meet the tenants' expected QoS and quality of experience (QoE) needs. Additionally, it can be integrated into various domains of the

softwarization aspect of a slicing framework, which are listed in Fig. 1.3 along with any possible implications it has on the future slicing framework.

1.3.3 Intelligent Edge and/or Cloud Computing

For more than half a century, mobile communication networks have been developed purely to deliver voice and data services to end-users. Nevertheless, the cloudification of these services, networks, and devices enables operators to develop an agile, flexible, and scalable network slicing framework for 5G and beyond communication systems. This classical idea, however, has been recently shaken by the latest pioneering efforts, which attempt to widely integrate advanced ML algorithms into all kinds of modern digital services aimed at significantly improving the overall performance of a mobile network (see Fig. 1.3). It is now generally agreed that pervasive intelligence will be an indispensable part of 6G networks, and efficient computing will be essential to enable most – if not all – beyond 5G and emerging 6G applications [13, 17]. To put it more radically, 6G will be an intelligence-delivering infrastructure rather than solely an information-transmitting network.
This new role of mobile networks calls for a critical change in the way to evaluate them. As controlling, computing, and communication will converge in 6G more tightly than ever, the 6G QoS will be challenged not only by the communication performance, but also by the computing performance [18]. Moreover, various use cases and applications may also have significantly different requirements on the computing service performance. It demands us to extend the service heterogeneity from the data networking domain, where it was addressed by 5G, into the computing domain. Therefore, use-case-specific heterogeneous management of computing resources and tasks shall be implemented in an efficient, agile and flexible fashion, and merged into the 6G network slicing framework.
Similar to the development of network slicing in 5G, upcoming efforts towards intelligent computing in 6G shall begin with the definition of computing service types, with the QoS requirements clearly defined for each type. Thereafter, it calls for novel methods to address the most significant computing problems – such as ML-assisted resource management, load splitting, task scheduling, and service chain management – of the heterogeneous computing service types [18]. We foresee this task to be coupled with the challenges of intelligent virtualization & softwarization, where slice-specific approaches shall be developed to provision, orchestrate, and manage computing resources at both aggregated data centers, i.e., local and regional cloud servers, and distributed computing units, i.e., edge computing servers and fog computing nodes [18].

1.4 Network Slicing in Next Generation Radio Access Network

The Third Generation Partnership Project (3GPP) has defined that an E2E NS comprises of two network slice subnets (NSSs): the core network (CN) NSS and the RAN NSS [19]. Both NSSs must be joined through a highly reliable and large-capacity transport network (TN), which goes beyond the scope of 3GPP. However, within the scope of the Fifth Generation Public-Private Partnership (5GPPP), projects like the 5G-Crosshaul [20], 5G-NORMA [21], 5G-ESSENCE [22], and 5G-CHARISMA [23] have addressed network slicing in TN. Specifically, the 5G-Crosshaul has proposed the TN NSS in [24]. By chaining the NFs, pairing the network links, and harmonizing the allocation of network resources of the CN, TN, and RAN NSSs; an E2E NS is realized [5]. The E2E NS, throughout its lifetime, must

fulfill the technical and business requirements of the requested communication services of a communication service customer (or tenant). A tenant could be a mobile virtual network operator (MVNO) or the owner of a service sector such as automotive, health care, agriculture, and many others.

Each RAN NSS (hereinafter referred to as a RAN slice) in the 3GPP-defined NG-RAN architecture could be composed of VNFs and PNFs. The NG-RAN consists of a number of gNBs. Each gNB is composed of a single centralized unit (CU), at least a distributed unit (DU), and at least a radio unit (RU) [25]. These components accommodate and distribute the VNFs and PNFs of a RAN slice in a flexible manner [26]. The PNFs - executed on top of dedicated hardware units that require physical resources such as radiofrequency spectrum, antennas, and others - are allocated, configured, and managed by the 3GPP-NSMS entities [19]. The VNFs - deployed on top of general-purpose hardware units that require virtual resources such as virtual compute, networking, and storage - are configured and managed by the NFV-MANO functional blocks (FBs) [27], defined by the European Telecommunication Standard Institute (ETSI). To effectively manage and orchestrate the VNFs, PNFs, virtual resources, and physical resources of an NS, the ETSI and 3GPP jointly proposed a unified framework of the NFV-MANO FBs and 3GPP-NSMS entities in [27] and [28], respectively. This proposed framework can be extended towards the NG-RAN, designing an interoperable functioning architecture to manage and orchestrate the physical part and virtual part of a RAN slice.

To utilize a joint framework of the 3GPP-NSMS and NFV-MANO in the context of RAN slicing in NG-RAN, the VNFs and PNFs of a gNB must be defined in the first place. To that aim, we assume the CU and DU as VNFs and the RU as a PNF in a RAN slice. Based on such assumptions, which are also in compliance with the Open-RAN (O-RAN) Alliance's RAN architecture context [29], we refer to CU and DU as virtual CU (vCU) and virtual DU (vDU), respectively. Subsequently, the vCU, vDU, and RU shall be dynamically mapped - based on an efficient mapping algorithm - onto the aggregation data center, edge data center, and cellular network site, respectively [30].

Each data center, regardless of its location on the edge or in the core of a cellular network, is also referred to as the network function virtualization infrastructure point of presence (NFVI-PoP) in ETSI terminology. The NFVI-PoPs are distributed geographically across the underlying cloud infrastructure, which is owned by a network infrastructure provider and could be managed by multiple service providers. Each NFVI-PoP is made up of physical nodes that can be virtualized to create a number of virtual nodes. To meet the low latency and high bandwidth requirements of the end-users, the NFVI-PoPs are typically interconnected via optical fiber, forming an E2E NFVI-PoP interconnection network. The physical paths in the underlying optical networks can also be virtualized into numerous virtual paths. The cellular network site is defined as a collection of nodes and networking equipment that are used to host the functionalities and links of the RU in a physical manner.

The main objective of the mapping process is to deploy the VNFs and PNF of a RAN slice on suitable virtual and physical nodes that are capable of fulfilling their resource requirements. The vCU and vDU require virtual resources abstracted from underlying general-purpose hardware units in aggregation and edge data centers, respectively. The RU requires physical resources, which are existed in cellular network sites. To compose a complete RAN slice, the virtual resources of vCU and vDU, and the physical resources of the RU must be dynamically chained in order, ensuring the transport of the network traffic in upstream and downstream directions [31]. In addition, such a service chain also includes the internal VL and external VL, and the internal physical link (PL) and external PL, which are used to connect the vCU, vDU, and RU in a RAN slice. The VLs and PLs must also be efficiently mapped onto their

corresponding virtual and physical environments, respectively.
The mapping of the VNFs and VLs of a RAN slice, which is the main focus of this thesis, is accomplished in several operational stages. In the beginning, the service chain of a RAN slice must be designed. To delve deeper into the order of its components, the internal functionalities of vCU and vDU, and the internal and external VLs of a gNB, presented in every RAN slice, must be specified and linked sequentially. Thereafter, the number of virtual resources - required by the vCU, vDU, and VLs - shall be dynamically quantified. Subsequently, the appropriate virtualized environments, which can fulfill the virtual resource requirements of a RAN slice, must be selected. Afterward, the VNFs and VLs are mapped onto the virtualized environments. Finally, the allocated virtual resources shall be constantly optimized - using cutting-edge optimization models and resource prediction techniques - to improve resource allocation, reduce energy consumption, and decrease capital expenditure (CAPEX) and operating expenditure (OPEX).

1.5 State-of-the-Art Related Works

1.5.1 The Foundation for Virtualization and Network Slicing in NG-RAN

To this end, a significant number of standardization efforts, industrial experiments, and theoretical research have been conducted to explore the virtualization of NFs and network links; the allocation of virtual and physical resources; and the management and orchestration of the VNFs and PNFs from the core network down to the RAN architecture. Specifically, following the initiative by the NGMN Alliance for the foundation and definition of network slicing in 5G systems [7], several standards development organizations (SDOs) - such as the 3GPP, the ETSI, the Global System for Mobile communications Association (GSMA), the O-RAN Alliance, and the Internet Engineering Task Force (IETF) - and their respective members from the industry have defined their own visions and provided guidelines to concretize the concept of network slicing towards its realization in NG-RAN.
The scope of each SDO is varied. Among them, the NGMN has mainly been focusing on the definition of key concepts for E2E network slicing that could also apply to slicing the NG-RAN [7], the requirements of RAN slices, and the architecture that hosts the resources of a RAN slice [32]. To delve deeper into RAN slicing, the NGMN has recently studied the lower and higher layer split options of the functionalities of a RAN slice that could be hosted by the underlying infrastructure in a centralized or distributed manner [33]. Such multiple options for functional split require connectivity over the NG-RAN. To that aim, the NGMN has additionally investigated several deployment options for the transportation network to connect the components of a RAN slice [34].
The 3GPP has been playing a leading role in the standardization of RAN architecture for several decades. However, with the extension of virtualization to the NG-RAN and the deployment of RAN slicing, several new aspects have emerged that fall outside its scope. The specifications of the 3GPP have, therefore, been focusing on *(i)* the management and orchestration of physical resources of a RAN slice [19, 28]; *(ii)* the requirements of the communication services that categorize the type of a RAN slice [35]; and *(iii)* the distribution of the functionalities of a RAN slice over the NG-RAN [36].
In contrast to the 3GPP, the Industry Specification Groups (ISGs) related to the ETSI - mainly the NFV and Next Generation Protocols (NGP) - have been merely focusing on the virtualization aspects of RAN slicing [37, 38]. Among others, such ISGs have specified

the management and orchestration of the VNFs and VLs, and the allocation of underlying virtual resources for RAN slices [39]. In addition, an architectural framework has been proposed by ISG NFV that could also be integrated with a standardized operations support system/business support system (OSS/BSS) [27, 40, 41]. To effectively manage the underlying resources and infrastructure [42, 43, 44, 45], several FBs have been specified, which utilize machine-processable description files to automate the operations of the proposed architectural framework [46].
Although the ISG NFV promises to automate the functionalities (during the lifetime) of a RAN slice, many of the processes are still executed manually by the network administrator or tenant. Therefore, full dynamism, E2E automation, and high scalability would be extremely important features to the success of NFV in the next generation of mobile networks. To that purpose, the ETSI has recently created the Experiential Network Intelligence (ENI) ISG, which focuses on the adoption of AI techniques, specifically the ML algorithms, in NFV architecture and OSS/BSS for 5G and beyond communication systems [47]. The ENI System automatically collects several metrics from the NFV-MANO and OSS/BSS, understands their configuration and operating status in real-time, and then utilizes ML algorithms to enable intelligent service deployment, resources management, monitoring, predictions, etc. The grand objective of the integration of intelligence and automation into the aforementioned systems is to improve efficiency in network operation, enhance the performance of the RAN slice, automate complex human-dependent decisions and processes, and many others.
The O-RAN Alliance has been tackling both the physical and virtual aspects of NG-RAN with the goal of transforming the traditional RAN (also referred to as vendor lock-in RAN) into an intelligent, virtualized, slicing-aware, and multivendor interoperable architecture that must operate based on open and disaggregated interfaces [48]. The key innovations introduced by the O-RAN Alliance are: (a) standardizing an open interface between the DU and RU, which is critical for lowering the total cost of RAN deployment and eliminating proprietary lock-in; (b) proposing the Near-Real-Time RAN intelligence controller (Near-RT RIC) network optimization entity for controlling the components and resources of a RAN slice in NFVI-PoPs; and (c) introducing a Non-RT RIC intent-based management entity for realizing automation and intelligence (notably ML algorithms) in all levels of the management and orchestration aspects of the open NG-RAN architecture [29, 49].
The GSMA has identified a large number of use cases - along with their functional, operational, and performance requirements - of vertical industries, which demand network slicing solution [50]. It has also proposed the Generic Network Slice Template (GST) and NEtwork Slice Type (NEST), which are used to quantitatively and qualitatively describe the requirements of an E2E network slice [51, 52]. Last but not least, the IETF has also been active in the high-level specification of virtualization and network slicing in cellular networks such as the architectural framework, operation, management and orchestration, and others [53, 54, 55, 56, 57].

1.5.2 Conceptual and Analytical Models for RAN Slicing in NG-RAN

In parallel with the SDOs and industry, a considerable amount of efforts in academia have also been made aiming to extend virtualization to the RAN architecture and the slicing of its relevant resources. The vast majority of the available studies focuses on *(i)* the slicing of radio resources [58, 24, 59, 60, 61, 62, 63, 64, 65]; *(ii)* the allocation of virtual resources [66, 67, 68]; *(iii)* the energy consumption of various types of RAN slices [69, 70, 71]; and *(iv)* the application of AI techniques and ML-assisted algorithms in the operations of RAN architecture [72, 73]; to name a few.

The above works provided the first attempt to extend virtualization to the network's edge and are considered the theoretical foundation for RAN slicing. However, most of these models are proposed in the context of cloud RAN (C-RAN), fog RAN (F-RAN), and heterogeneous cloud RAN (H-CRAN), which were standardized by the 3GPP before Release 15. The NG-RAN is defined in Release 15 and further developed in Release 16 and Release 17. Therefore, the utilization of the above-cited models might not accomplish the desired performance objectives in the NG-RAN architecture.
Furthermore, while these studies addressed virtual resource allocation, they neglected the most critical stages that are prerequisites for RAN slicing, such as mapping VNFs and VLs onto the underlying infrastructure, designing the service function chain (SFC), and placing VMs and VLs tailored to different types of RAN slices in the NG-RAN architecture. Last but not least, the aforementioned proposals are formulated based on traditional optimization models, which may not comply (due to their time-complexity) with E2E automation, zero-touch network and service management, and fully self-learning requirements of 5G and beyond mobile networks. Therefore, cutting-edge automatic data learning and synthesizing optimization tools are required to map, scale, configure, and allocate the virtual components of a RAN slice in a timely and resource-efficient manner.
Recently, RAN slicing in the context of the NG-RAN architecture has sparked a lot of interest. Among them, the authors of [25] experimentally evaluated the impact of virtualization on middlehaul latency in the NG-RAN architecture. The work of [26] was the first to propose a framework which dynamically provides RAN slice specific functional split options for eMBB, URLLC, and mMTC types of services in the NG-RAN architecture. Other studies have examined the deployment of VNFs and allocation of virtual compute resources [68], the analysis of various functional split options under energy and cost constraints [74], the mapping of RAN slices with a special emphasis on isolation and resource allocation [75], the placement of VNFs [76], the scheduling of VNFs for RAN slices [77], the impact of functional split on fronthaul delay in a RAN slice [78], and the sharing of VNFs across multiple network slice instances (NSIs) [79].

1.5.3 Mapping the VNFs and VLs onto Legacy Computing Data Centers

In addition to the above, there exists a large amount of literature focusing on the mapping of the VNFs and VLs onto the traditional cloud and edge data centers. Among the most recent state-of-the-art solutions, for instance, in [31], a genetic algorithm is proposed to address the mapping of the VNFs and SFCs onto optical networks. In [80], the authors addressed traffic routing and service scheduling using column generation. Meanwhile, the placement of SFC in a federated multi-domain scenario has been explored in [81]. In [82], the authors solved the admission control and SFC problems of the VNFs in virtualized networks by employing relaxation, reformulation, and successive convex approximation methods. Focusing on the scheduling in [83], the authors provided a genetic algorithm-assisted solution scheduling the non-uniform incoming of VNFs requests. In [84], the instantiation of the VNFs and the mapping of SFC have been studied in wide area networks. Lastly, reference [85] proposed a joint algorithm, which constructs the SFC and subsequently maps the VNFs onto cloud data centers to minimize bandwidth.
All the above works and the references therein provided sufficient details related to virtualization and softwarization, discovered various aspects of the mapping of the VNFs and VLs onto the underlying infrastructure, and showed many advantages that virtualization brings to computing data centers. However, the models proposed in these works have been evaluated

in the context of traditional virtualized networks, neglecting the characteristics tailored to a wireless communication channel in both upstream and downstream directions in NG-RAN. Therefore, the deployment of such proposals might not efficiently accommodate the diverging requirements of the VNFs of the bandwidth-devouring eMBB, latency-aware URLLC, and unlimited-things-centric mMTC types of RAN slices in 5G and beyond mobile networks. Lastly, most of the above works are based on a single-objective optimization problem, solving either the mapping of the VNFs or VLs at a given time. However, due to the nature of the mapping problem, both the VNFs and VLs are highly independent and likely conflicting with each other. Based on this, there is a need for a multiple-objective optimization solution, which must jointly map the VNFs and VLs efficiently.

1.6 The Key Research Problems

Despite widespread interest in virtualizing and intelligently managing the NG-RAN architecture as well as the slicing of its underlying resources, the mapping process of the VNFs and VLs of a RAN slice – which is considered one of the most critical steps towards a virtualized, slicing-aware, and autonomous NG-RAN architecture – has not been fully addressed. This is due to the fact that (a) the development and deployment of virtualization, cloudification, and network slicing in the NG-RAN architecture are still in their early stages; (b) the state-of-the-art human-machine oriented interoperability between NFV-MANO, 3GPP-NSMS, and the underlying infrastructure is error-prone, slow, and cumbersome during the execution of the mapping process of the VNFs and VLs of a RAN slice; and (c) the mapping process of the virtual part of a RAN slice is inherently challenging because of the distinctive characteristics of the wireless communication channels (such as de[modulation], de[coding], de[multiplexing], de[ciphering], among others) and dynamic changes in the amount of virtual resources required by each VNF and VL.
Notwithstanding, the mapping process of the VNFs and VLs is also challenged by the density and behavior of the user equipments (UEs) of a RAN slice in a specific geographical region. Furthermore, the virtual resource allocation in the NG-RAN architecture, with the deployment of RAN slicing, has become more complicated due to new considerations such as the trade-off between utilization ratio and isolation level; the harmonization of inter-RAN and intra-RAN slice resource allocation algorithms; and the management of inter-RAN and intra-RAN slice priority. Therefore, the mapping of the virtual part of a RAN slice onto computing data centers and transportation networks in the NG-RAN architecture is regarded as a valid research problem. Proposing an autonomous and intelligent solution to such a problem at an architectural level is the ultimate goal of this thesis.

1.7 The Major Goals and Contributions of the Thesis

To contribute to filling the solution gap identified above, we propose an optimal architectural solution that is both autonomous and intelligent for mapping the virtual components of a RAN slice onto the underlying infrastructure in the NG-RAN architecture. To the extent of our knowledge, this is the first work addressing the mapping problem of a RAN slice in 5G and beyond mobile communication systems. The proposed architectural solution is distinct from the state-of-the-art alternative proposals discussed in the preceding sections in terms of:

- **Application.** The proposed mapping solution is customized to the NG-RAN architecture and is specifically applicable to the mapping process of the virtual components of a single RAN slice.

- **Tool.** The proposed mapping solution employs an ML-assisted functioning architecture to execute the mapping process of a RAN slice, namely the ENI framework, which was recently introduced by the ETSI in order to integrate automation and intelligence into 5G and beyond mobile communication networks.

- **Architecture.** In contrast to the state-of-the-art alternative models, which assume solely the NFV-MANO framework, the proposed mapping solution in this thesis is executed on top of the unified architectural framework of the 3GPP-NSMS, NFV-MANO, ENI System, and the underlying infrastructure.

- **Scope.** The majority of the previous studies have concentrated on finding the optimal candidate NFVI-PoPs and transportation links for mapping the VNFs and VLs of a network service. The focus of this study, however, goes beyond such a dilemma. This thesis investigates the components of the VNFs and the VLs of a RAN slice in greater detail, as well as their mapping onto their respective virtual environments in the underlying NFVI-PoPs and transportation networks, aimed at optimizing a set of predefined optimization goals. These performance objectives could include decreasing the number of active physical components in an NFVI-PoP, minimizing the total bandwidth utilization of the PLs in the transportation networks, reducing energy consumption both in the NFVI-PoP and transportation networks, and so on.

In the light of foregoing goals, the main contributions of this thesis are thus:

- To extend the NGMN Alliance-defined functional architecture for network slicing towards the NG-RAN architecture, as well as to provide a comprehensive overview and critical analysis of the VNFs, PNFs, VLs, PLs, virtual and physical resources, and underlying compute and transportation infrastructures that host the components of a RAN slice at the edge of a cellular network beyond the 5G communication system;

- To propose a unified architectural framework of the NFV-MANO FBs, the 3GPP-NSMS entities, and the underlying compute and networking infrastructures for managing and orchestrating the virtual and physical components of a RAN slice in the NG-RAN architecture;

- To adopt automation and intelligence into the management and orchestration of a RAN slice by integrating the ENI framework of the ETSI into the unified functioning architecture of the 3GPP-NSMS, NFV-MANO, and the underlying physical and virtual infrastructure in the NG-RAN architecture;

- To thoroughly define and model the mapping process of the components of the vCU and vDU, the VLs, and all the virtual aspects of a RAN slice at an architectural level, as well as to explore the internal domains and components of an NFVI-PoP and the underlying transportation networks, based on a number of constraints that are critical for consideration in such a process;

- To propose a learning-assisted solution at an architectural level – leveraging ML techniques and AI tools in the joint framework of ENI, NFV-MANO, 3GPP-NSMS, and the underlying infrastructure – for mapping the components of the vCU and vDU, as well as the VLs, of a RAN slice onto their respective virtualized environments in the underlying edge data centers and transportation networks in the NG-RAN architecture.

1.8 The Structure of the Thesis

The rest of this thesis is structured in the following manner. We commence by extending the architectural framework of network slicing for various types of RAN slices towards the NG-RAN architecture in Chapter 2. In this chapter, we delve deeper into the management and orchestration of the virtual components of a RAN slice, as well as integrate the ENI framework into the unified architecture of the NFV-MANO, 3GPP-NSMS, and underlying NFVI-PoPs aimed at bringing automation and intelligence to the management and orchestration of RAN slices. In Chapter 3, we define and model the problem of mapping the virtual components of a RAN slice onto the virtual resources in the underlying infrastructure. We also propose an ML-assisted solution, at an architectural level, for mapping the internal and external VLs, as well as the internal components of vCU and vDU, of a RAN slice onto the underlying NFVI-PoPs and transportation networks. In chapter 4, we discuss the unification of the 3GPP, O-RAN Alliance, and ETSI management systems within the SMO framework. We also define the integration of ML algorithms into this framework from an architectural point of view. Chapter 5 covers the application of ML assisted algorithms in the NG-RAN architecture. We precisely focus on the theoretical and practical aspects of the Q-learning in the mapping process of a RAN slice. In Chapter 6, we discuss the results we obtained thorough the use of Q-learning in the mapping process of the RAN slices. Lastly, Chapter 7 summarizes the main conclusions of the thesis and provides future research directions in this area.

Chapter 2

Background on Key Concepts for Slicing the NG-RAN Architecture

Summary – In this chapter, we extend the architectural framework of network slicing, defined by the NGMN Alliance in [7], towards the NG-RAN architecture based on the needs for a common, economically sustainable, slicing-aware, and energy-efficient RAN architecture in order to accommodate three well-known categories of 5G communication services [4] - the eMBB, the mMTC, and the URLLC - to the NG-RAN. The proposed architecture, managed by the RAN management and orchestration plane, is shown in Fig. 2.1. It consists of three layers: the communication service layer, the network functions layer, and the infrastructure layer. These three layers and their management and orchestration plane are described in the following subsections, respectively. Additionally, we provide a deeper insight into the management and orchestration plane that is used to manage the VNFs and VLs of RAN slices and orchestrate their respective virtual resources in the underlying infrastructure. To do so, we first address a joint framework of the NFV-MANO and 3GPP-NSMS, which is used for the management and orchestration of RAN slices in NG-RAN. Subsequently, we integrate the ENI into the 3GPP-NSMS and NFV-MANO to bring intelligence to the edge of cellular networks. The goal of the proposed unified framework is to enable the NG-RAN to be more reliable and maintainable and to provide context-aware services in order to meet the business objectives and technical requirements of the tenants and/or MVNOs.

2.1 The Architectural Framework of RAN Slicing in NG-RAN

2.1.1 The Communication Service Layer

This layer represents the types, behavior, and characteristics of the communication services that are provided by the RAN slices to the tenants or MVNOs. The characteristics of the communication services provided by a RAN slice, the isolation and allocation of its resources, the management and orchestration, and other requirements are defined in a deployment descriptor file, called the RAN network slice subnet template (NSST), see Fig. 2.1. The RAN NSST is identified based on network slice selection assistance information (NSSAI), which is specified by the 3GPP in [86]. To assist the 5G network in selecting a RAN slice, the UE sends NSSAI in a signaling message towards the NG-RAN. According to 3GPP specifications [86], the NSSAI contains a set of 8 single-NSSAIs (S-NSSAIs), where each S-NSSAI identifies a RAN slice. This means that a UE can potentially be served by a maximum of 8 RAN slices at a given time, each with its customized RAN NSST.

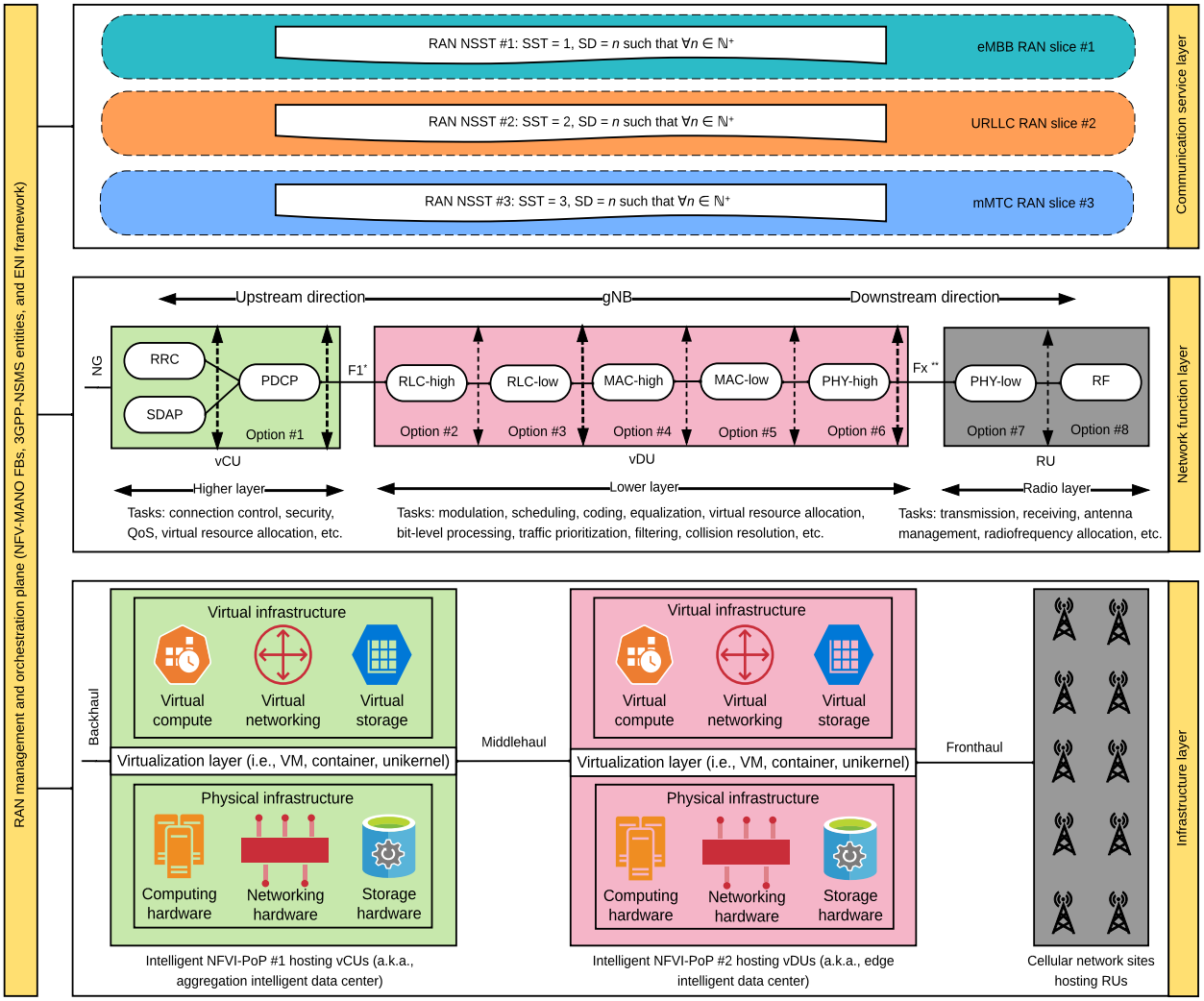

Figure 2.1: The proposed architectural framework of RAN slicing in NG-RAN. Do note that (∗) F1 is a standard interface between the vCU and vDU whereas (∗∗) Fx is used as a generic notation for functional split between the vDU and RU.

Each S-NSSAI is a 32-bit parameter that is comprised of two components (also called fields): the slice/service type (SST) and the slice differentiator (SD) [35].

- The SST is a mandatory 8-bit component that defines the type of communication services, features, and behaviors of a RAN slice. The SST field contains a specific value that can be either standardized or non-standardized (operator-specific). The standardized SST range is from 0 to 127, while the operator-specific SST range includes values from 128 to 255 [86]. For eMBB the SST = 1, for URLLC the SST = 2, and for mMTC the SST = 3. In addition to the three SSTs mentioned previously, two further SSTs have been standardized by the 3GPP. They are SST = 4 for vehicle to everything (V2X) and SST = 5 for high-performance machine-type communications (HMTC). For the sake of simplicity, we consider only the SSTs that belong to the eMBB, URLLC, and mMTC types of RAN slices in this thesis. We assume that the remaining two SSTs are managed in the same manner by the proposed functioning architecture.

- The SD is an optional 24-bit component, complementing the SST. It is used to identify (or differentiate) multiple homogeneous RAN slices offered to the same or different tenant(s). This differentiation could be made on the basis of features, applications,

network coverage zones, and priority, among other criteria. The SD field is also provided with a value, ranging from 1 to $n \in \mathbb{N}^+$. For example, a tenant (or MVNO) may request two eMBB RAN slices and one URLLC RAN slice from a network operator. In this case, the first eMBB RAN slice is identified with SST = 1 and SD = 1, the second eMBB RAN slice is identified with SST = 1 and SD = 2, while the URLLC RAN slice is identified with SST = 2 and SD = 1.

2.1.2 The Network Functions Layer

This layer defines the radio processing functionalities, implemented as VNFs and/or PNFs, that constitute the eMBB, URLLC, and mMTC types of RAN slices (see Fig. 2.1). These functionalities are distributed in NG-RAN in a flexible manner. The NG-RAN is composed of a set of $n \in \mathbb{N}^+$ gNBs. The gNB incorporates three functional modules: the vCU, the vDU, and the RU; which could be deployed at different geographical locations [36]. Each gNB consists of at least a vCU. Depending on network topology and the density of UEs in a specific geographical region, each vCU may correspond to multiple vDUs, and each vDU may correspond to multiple RUs [69]. For the sake of simplicity, we assume a vCU, a vDU, and an RU belonging to a gNB in this thesis, as shown in Fig. 2.1.

These three logical functional modules of a gNB may be provided by the same vendor or by multiple vendors. Thus, the NG-RAN architecture must support seamless interoperability between the hardware and software components from multiple vendors while participating in the creation of a RAN slice [87]. Such an openness and interoperability in the NG-RAN architecture improves several features, including network malleability, programmability, intelligence, and security [88, 89]. It also reduces total network expenditure, extends virtualization to the extreme edge, strengthens the management and orchestration framework, brings versatility to the market, and improves underlying resource allocation. To facilitate open implementation of the components of a gNB, the O-RAN Alliance and the Telecom Infra Project (TIP) have been developing specifications over the last several years [29]. These components of a gNB are denoted as Open-CU, Open-DU, and Open-RU in the O-RAN Alliance's specifications [49, 88].

The radio processing functionalities of a gNB are dynamically distributed in eight split options (option #1 – option #8) over the vCU, vDU, and RU in order to enable cloudification, virtualization, and softwarization of eMBB, URLLC, and mMTC types of RAN slices over a common NG-RAN infrastructure effectively and allocate their required virtual and physical resources efficiently [90]. The distribution of the gNB functionalities among its components, also called the functional split, determines which function to be processed in the vCU, which to be located in the vDU, and which to be moved towards the RU in both downstream and upstream directions.

The 3GPP initially considered option #2 as a functional split between the vCU and vDU and option #7 or option #8 between the vDU and RU. However, other standardization bodies, such as Small Cell Forum (SCF), evolved common public radio interface (eCPRI) Cooperation Group, O-RAN Alliance, and xRAN Cooperation have moved further to identify new options for functional-splits in NG-RAN architecture in consideration of user density, bandwidth, and latency requirements of RAN slices; such as option #1 between vCU and vDU, and option #6, option #7-1, option #7-2a, option #7-2, and option #7-3 between the vDU and RU. These split options are analyzed and compared in [91, 33, 34, 48, 92]. In this thesis, our objective is not to touch the distribution of gNB functionalities, their respective architectures, and implications on the performance of RAN slices. For simplicity, we thus only assume (i) three aforementioned components of a gNB that accommodate these

functionalities physically and virtually, (*ii*) option #2 for the vCU and vDU split which is specified over a well-defined logical interface (F1), (*iii*) option #7 for the vDU and RU split over a standard interface (Fx), and (*iv*) both F1 and Fx have the capabilities of dynamically supporting different requirements (such as the data rate, latency constraints, users density, high mobility, and others) of eMBB, URLLC, and mMTC RAN slices in the NG-RAN.
The NFs of the vCU and vDU are either already virtualized or being candidate for virtualization. Those which are yet to be virtualized require further research both in terms of their softwarization and cloudification. However, there is no doubt that they will be fully virtualized in the near future or in the long term. Based on this, the NFs that are accommodated by the vCU (a.k.a., higher layers functionalities) and vDU (a.k.a., lower layers functionalities) are assumed to be fully virtualized and could be instantiated as VNFs in aggregation data center and edge data center, respectively. The higher layers consist of three major functionalities: namely the radio resource control (RRC), the service data adaptation protocol (SDAP), and the packet data convergence protocol (PDCP). The lower layers consist of five major functionalities: namely the high radio link control (RLC), the low RLC, the high medium access control (MAC), the low MAC, and the high physical (PHY). To get further insights into aforementioned functionalities, the interested readers are suggested to refer to the relevant publications, such as [69, 91, 92], and the references therein. These eight virtual functionalities of the vCU and vDU - and the virtual links connecting them - running in aggregation and edge data centers require virtual resources such as virtual compute, networking, and storage [93].
The NFs of the RU, which are also called the radio layers functionalities, include the low PHY and the radio frequency (RF). These two functionalities are assumed to be the PNFs of a gNB. Both of these physical functionalities and the physical links connecting them are implemented on top of customized equipment and physical resources, such as antenna, radio frequency spectrum, Ethernet cable, optical fiber, and other dedicated hardware installed on Macro, Micro, Pico, and Nano cellular network sites in NG-RAN.
To improve resource utilization, enhance the energy efficiency, decrease CAPEX and OPEX, and increase the number of served tenants, it is expected that the aggregation data center, edge data center, and the cellular site should have the capabilities to host, configure, and allocate the resources of the vCU, vDU, and RU functionalities of various eMBB, URLLC, and mMTC types of RAN slices, concurrently. It specifically means that a gNB could provide $n \in \mathbb{N}^+$ RAN slices of different types at the same time. Based on this, in this thesis, we assume that the virtual and physical resources of the gNB's components are dynamically shared among the three aforementioned types of RAN slices in NG-RAN.

2.1.3 The Infrastructure Layer

This layer allocates the virtual and physical resources of RAN slices in virtualized and physical sites, respectively (see Fig. 2.1). The physical sites are the cellular network infrastructure, which are divided into Macro, Micro, Pico, and Nano sites. They are uniformly or non-uniformly distributed over a specific geographical location based on the network topology, user density, and other requirements associated with network design. Each cellular site consists of an antenna, tower, and other communication equipment, which are used to deploy the PNFs of the RU (namely the PHY-low and RF layers) for the eMBB, URLLC, and mMTC types of RAN slices. The virtualized sites are the cloud sites, namely the aggregation data center and edge data center. Each data center is also called the NFVI-PoP. The NFVI-PoP consists of general-purpose hardware and software components, which are used to host, manage, and execute the functionalities of the vCU (namely the RRC, SDAP, and PDCP layers) and vDU

(namely the RLC-high, RLC-low, MAC-high, MAC-low, and PHY-high layers) of the RAN slices [94]. The VNFs in NFVI-PoP #1 (namely the vCUs) and NFVI-PoP #2 (namely the vDUs) are usually hosted in different geographical locations (see Fig. 2.1). Depending on the deployment scenario, the distance between NFVI-PoP #1 and NFVI-PoP #2, and between NFVI-PoP #2 and cellular sites could be up to tens of kilometers [91]. Therefore, the vCUs and vDUs hosted in NFVI-PoP #1 and NFVI-PoP #2 are interconnected using a reliable F1 Middlehaul link. The vDUs hosted in NFVI-PoP #2 are connected with the RUs hosted in the cellular sites also using a reliable Fx Fronthaul link [92].

Targeting the virtualized sites of the infrastructure layer in this thesis, both NFVI-PoP #1 and NFVI-PoP #2 are used to host a large number of vCUs and vDUs of various types of RAN slices with complex and potentially conflicting demands, respectively. The state-of-the-art NFVI-PoPs manage, orchestrate, and allocate the virtual resources of the vCUs and vDUs using conventional methods. However, the performance of such mechanisms is not efficient to accommodate the requirements of a large number of VNFs in a shared and heterogeneous NG-RAN with scarce virtual resources [95]. To adjust the services of RAN slices based on dynamic changes in the tenants' domains, business objectives, and environmental conditions, we extend the utilization of AI techniques - specifically the ML algorithms such as supervised learning, unsupervised learning, reinforcement learning, and deep learning - to NFVI-PoP #1 and NFVI-PoP #2 in order to manage, orchestrate, and allocate the virtual resources of the vCUs and vDUs. We refer to such AI empowered NFVI-PoPs as Intelligent PoPs (I-PoPs). Each I-PoP uses ML algorithms to optimize the performance of RAN slices resulting in the overall operational enhancement of the cellular network.

In addition to the above, the state-of-the-art NFVI-PoPs are based on human-machine interaction models, which are slow, expensive, error-prone, and cumbersome [96]. For example, each NFVI-PoP in beyond 5G is expected to be a complex platform built of different vendors' software and hardware components and could host a large number of RAN slices with customized services that shall be dynamically scaled up/down or scaled in/out. The conventional human-machine based interaction models - between vendors' components and operator's management system, and between tenants and operator - in such a heterogeneous NFVI-PoP are not adaptable to the dynamic changes in vCUs and vDUs. Moreover, these and other related factors could also lead to a very high CAPEX and OPEX for resource deployment and management in NFVI-PoPs. To reduce the CAPEX, OPEX, and human-dependent decision making processes, the operators need the ability to automate the operations such as the deployment, configuration, optimization, monitoring, management, and orchestration in I-PoPs. Therefore, the utilization of cutting-edge ML algorithms in I-PoPs has profound implications on reducing the CAPEX and OPEX, decreasing energy consumption, and lowering the management and network complexity in NG-RAN.

The bottom side of Fig. 2.1 also depicts that each I-PoP is composed of physical infrastructure, virtualization layer, and virtual infrastructure [97].

The physical infrastructure is characterized by compute, storage, and networking hardware (such as nodes, devices, and links, respectively) that provide compute resources, storage capacities, and internal/external connectivities for the vCU and vDU [97, 98]. The compute node is a general-purpose computing hardware, which is managed by its internal instruction set and realized in a single or multi central processing unit (CPU) servers, such as Tower server, Blade server, Rack server, and others. The storage device is capable of storing a large amount of data related to the VNFs and VLs of a RAN slice, temporally or permanently, which could be used in the form of direct attached storage (DAS), network attached storage (NAS), and storage area network (SAN). The networking links facilitate communication between the compute nodes of the I-PoPs and other elements (including the storage device)

in the form of layer-2/layer-3 (L2/L3) or bare-metal switches.

The virtualization layer is a software platform, which is placed between the physical infrastructure and virtual infrastructure in an I-PoP (see Fig. 2.1). In this layer, the underlying physical resources are abstracted into a number of isolated virtual environments of the virtual compute, networking, and storage resources [97]. The virtual environments of the virtual networking and storage resources are the virtual networking (VN) and the virtual memory, respectively [98]. The former, commonly used interchangeably with VL, virtually connects the vCU, vDU, and the functionalities therein to form a RAN slice. The latter virtually stores the data related to the vCU, vDU, and their respective VLs throughout the life cycle of a RAN slice. The VL and virtual memory are discussed later in the thesis.

The virtual environments of the virtual compute resources are VMs, containers, and unikernels [97]. The VM provides an isolated duplicate virtual environment of a real compute node for hosting VNFs. The container includes only necessary elements for hosting VNFs. The unikernel is an ultra-lightweight single-purpose virtual environment that runs only a single VNF or a single application of a VNF [97]. The VM and unikernel are achieved using Hypervisor (described later), such as Hyper-V, ESXi, Xen, KVN, etc. The container is abstracted by container technology, such as FlowN, Docker, LXC, etc. In each scenario, the performance of the virtual compute environment must be functionally equivalent to the performance of the physical compute environment. Each aforementioned virtual environment has its own characteristics and is suitable for specific virtual compute scenarios. Interested readers may refer to [97] for more detailed information.

In addition, to provide underlying virtual compute resources with E2E automation and full isolation, the usage of container in VM (CVM) has recently emerged as a virtual environment [99], which can also be used for RAN slices in both I-PoP #1 and I-PoP #2. The main objective of this approach is to achieve benefits from container and VM at the same time. For example, the VM guarantees the isolation of virtual resources and the container simplifies the execution of the VNF that requires these resources. The CVMs can also be manifested by Hypervisor. The Hypervisor (*i*) manages the CVMs (and VMs) and the allocation/reallocation of their respective virtual resources; (*ii*) provides isolation among the virtual resources of CVMs; (*iii*) maps the relationship between physical and virtual resources allocated to the CVMs; (*v*) schedules the virtual resources of the CVMs; and (*iv*) emulates the hardware components in such a way that would make the CVMs appear as to be running on real dedicate devices through specific APIs [97].

Considering the isolation of the virtual compute resources, which is essential for RAN slicing as well as supporting multi-tenancy by underlying I-PoPs, none of the aforementioned virtual environments standalone can fully and/or efficiently address the requirements of the eMBB, URLLC, and mMTC types of RAN slices due to diverse resource demands, service characteristics, and others. Therefore, there is a need for further research to specify which type of virtual environment is suitable for which type of RAN slices in NG-RAN. For simplicity, we assume VM as a virtual environment that is used to host the virtual compute resource in this thesis.

The virtual infrastructure is the third layer of an I-PoP, which is composed of virtual compute, networking, and storage resources. As discussed earlier, the virtual resources are abstracted, using Hypervisor, from the underlying physical resources such as compute nodes, physical links, and storage devices, respectively. These virtual resources are discussed in details in the following.

The virtual compute resources are achieved by the virtualization of physical processing nodes or elements (namely the CPU) using a certain compute virtualization technology, such as software-defined compute (SDC), vCenter Converter, Libvirt, etc. These technologies also

move the compute resources to a resource pool, which are then assigned based on a dynamic compute resource allocation algorithm to the VMs. The VMs and the compute resources are then used to host the VNFs of a RAN slice. From the resource allocation perspective, the amount of the compute resources assigned to a VM is usually categorized by the number of virtual central processing unit (vCPU), clock speed, etc. as their measurement units.
The virtual networking resources are the VLs decoupled from the underlying physical links (such as the Ethernet cables, optical fibers, L2/L3 or bare-metal switches, etc.) using a virtual software application, called the virtual switch (vSwitch). The goals of these VLs are to connect VMs, containers, unikernels, CVMs, virtual servers, and other elements of the virtual infrastructure layer within the same or across different I-PoPs. The virtual links connecting the VMs of the of the vCU and vDU are usually measured by the number of total allocated VLs and the bandwidth of each of them by kilo bits per second (Kbps).
The virtual storage resources are abstracted from the underlying hardware data storage resources (such as memory blocks and storage media) in the form of DAS, NAS, or SAN through a certain storage virtualization technology, such as SDS, and making them appear in a virtual storage resource pool. Based on the requirements of the VMs and VLs, the storage resources are dynamically allocated in order to store their data temporarily or permanently in a virtual storage resource pool. The virtual storage resources of the VM and VL are usually measured by the Kilobyte (KB).

2.1.4 The Management and Orchestration Plane

The management and orchestration plane plays an essential role in the assurance of an efficient utilization of the underlying resources and tight integration of the aforementioned three layers - namely the communication service layer, network functions layer, and infrastructure layer - while fulfilling the performance, operational, and functional requirements of the offered services of the proposed RAN slicing architecture in NG-RAN. Fig. 2.1 portrays that it is composed of 3GPP-NSMS, NFV-MANO, and ENI. The 3GPP-NSMS (described in the next section) is responsible, among others, for all operations related to the physical resources, connectivities, and services of RAN slicing in NG-RAN. On the contrary, the NFV-MANO (also explained in the coming section) is responsible for the virtual resources, connectivities, components, and operations that participate in RAN slicing. The unified integration of the NFV-MANO and 3GPP-NSMS is, therefore, extremely important to manage and orchestrate both physical and virtual parts of RAN slices in NG-RAN.
The state-of-the-art integration of the NFV-MANO and 3GPP-NSMS is based on human-machine type interaction models using standard interfaces. For a more effective control and orchestration on the resources and services, the operators demand an automated arrangement and coordination of both management systems in NG-RAN architecture. An inherent intelligence and implicitly autonomic control of all services, layers, and underlying resources is, therefore, needed at the edge of cellular network to fulfill the zero-touch management, orchestration, and operation requirements of the RAN slices. In response to the industry demand for AI/ML-driven intelligent mobile networks, the ETSI launched the ENI ISG in February 2017 [100]. The ENI, as of this writing, has 44 members and 18 participants involving academia, industry, vendors, and research institutions. The ENI System does not necessarily interfere-(into)/change the management, orchestration, and other functionalities of the 3GPP-NSMS and NFV-MANO, but its objectives are to optimize those functionalities and automate complex human-dependent decision-making process using cutting-edge AI/ML tools and methods, and context-aware, metadata-driven policies [100].
The purpose of the interoperability and integration of ENI into NFV-MANO and 3GPP-

NSMS is to automate the management and orchestration of RAN slices aiming to enhance the performance of the NG-RAN architecture. On one hand, the NFV-MANO and 3GPP-NSMS use a variety of tools and data ingestion formats to interact with each other. On the other hand, the ENI utilizes its own customized modules with their respective exposed APIs and varying degrees of ability. It is thus essential for the ENI ISG to define an architecture that transforms the input of the NFV-MANO and 3GPP-NSMS to a format that is understandable to the ENI System and vice versa [96]. To that aim, the ENI ISG has proposed such a modularized system architecture in [47]. We extend this architecture to the NG-RAN and integrate it into the 3GPP-NSMS and NFV-MANO to automate the management and orchestration of services and resources of RAN slices. The detailed description of the extended architecture, in the context of NG-RAN, is provided in next section.

2.2 The Management and Orchestration of RAN Slices in NG-RAN

2.2.1 3GPP-NSMS for RAN Slices

Fig. 2.2 shows the functioning architecture of the 3GPP-NSMS as part of its joint framework with the NFV-MANO. At the start, the communication service management function (CSMF) receives the communication service related requirements from a tenant or MVNO through a set of APIs, translates them to the NS related technical requirements, and delivers them to the network slice management function (NSMF) in a deployment descriptor file, called the NST. Among others, the NST specifies the required physical and virtual resources and their interconnection and configuration of an E2E NS. To this end, the standardization community has defined the eMBB, URLLC, and mMTC types of NSTs. The NSMF uses the NST to manage its respective type of NS throughout its life cycle and allocates its customized physical and virtual resources in isolation from other NSs. The 3GPP has divided the NST into CN NSST and RAN NSST [19]. Both these NSSTs, amongst others, specify the physical and virtual resources of a CN slice and a RAN slice in 5G core network (5GC) and NG-RAN, respectively. The TN slice, which connects the CN slice and the RAN slice, is also instantiated based on a TN NSST. The TN NSST describes, among others, the physical and virtual resources of the communication channels of an E2E NS.
To this end, the standardization community (more precisely, the 3GPP) has defined the eMBB, URLLC, and mMTC types of NSTs. The NSMF employs the NST to manage its specific type of NS throughout its life cycle and allocate its customized physical and virtual resources in isolation from those of other NSs over a shared infrastructure [101]. The 3GPP has divided the NST into CN NSST and RAN NSST [19]. Both NSSTs, among other attributes, define the required physical and virtual resources for a CN NSS and a RAN slice in 5GC and NG-RAN, respectively. The transportation network NSS, which connects the CN NSS and the RAN slice, is also instantiated based on a transportation network NSST. The transportation network NSST defines, among other attributes, the required physical and virtual networking resources for the communication channels of an E2E NS.
To manage an E2E NS effectively and orchestrate its respective resources efficiently, the NSMF splits the NS related requirements into CN NSS, TN NSS, and RAN NSS related requirements. Subsequently, the physical and virtual resources of every NSS are individually managed along with all the events that occurred during its lifetime by a 3GPP proposed management entity, called the NSSMF. This is also illustrated in Fig. 2.2, where the NSMF delegates the NSSs related requirements to their corresponding NSSMFs. Among them,

the NG-RAN NSSMF is responsible for the allocation, deployment, fault management, and performance management of virtual and physical resources of eMBB, URLLC, and mMTC types of RAN slices in a specific geographical region covered by at least one gNB belonging to that NG-RAN.
The 3GPP further hierarchically splits the tasks of the NG-RAN NSSMF into a number of low level management entities [28], called the network function management functions (NFMFs). The NFMF, which is also called the element manager (EM), is responsible to manage the fault, configuration, accounting, performance, and security (FCAPS) of each type of components of a gNB (i.e., vCU, vDUs, and RUs) or each of the components of a gNB individually. For instance, there could be three NFMFs to manage the FCAPS of a gNB, each manages the vCU, vDUs, and RUs, respectively; or, a number of NFMFs could be assigned to manage the FCAPS of each vCU, vDU, and RU in each gNB. In both scenarios, the NFMF is controlled by the NG-RAN NSSMF (see Fig. 2.2).

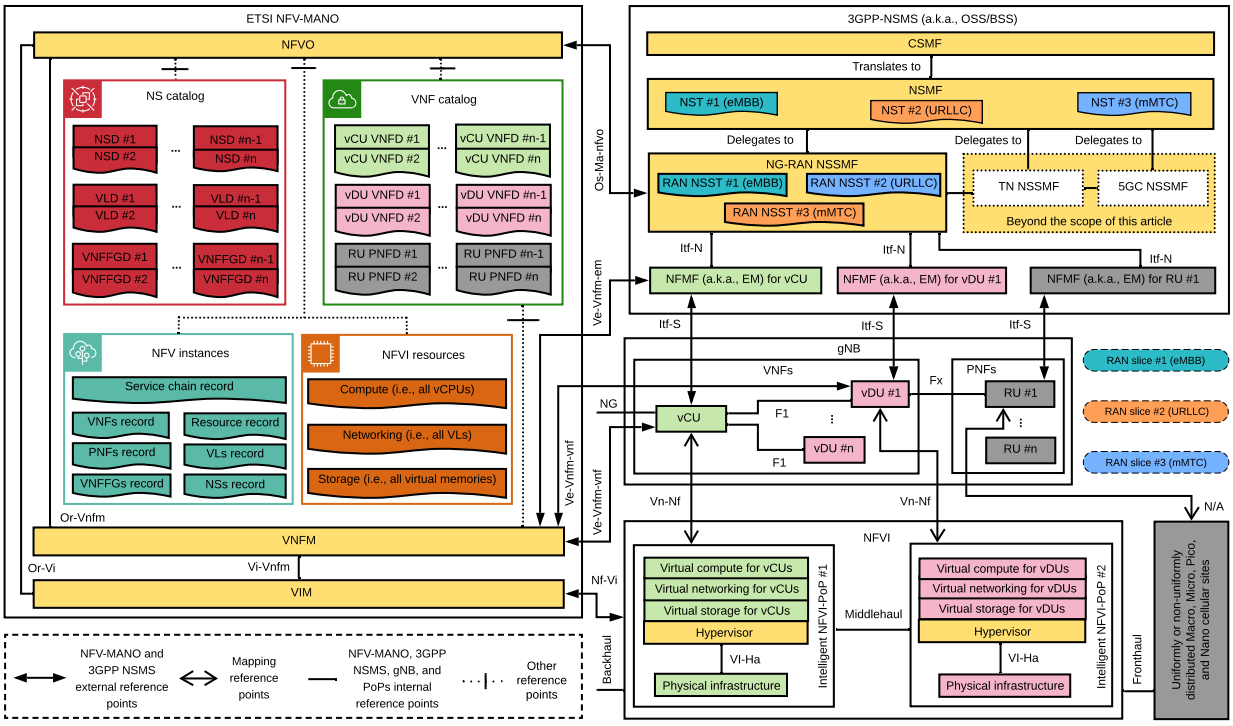

Figure 2.2: The MaO of RAN slices - using the 3GPP-NSMS and NFV-MANO joint framework - in NG-RAN. Note that the VNFM is connected to each of the VNFs (namely the vCU and vDUs) of a gNB, their corresponding NFMFs, and the NFMFs of PNFs (namely the RUs) directly. For simplicity, we show its connection with vCU, vDU #1, and NFMF for vCU. The rest of the NFMFs and VNFs are connected with VNFM in the same vein. Do also note that the detailed description of the 3GPP-NSMS entities and NFV-MANO FBs, and their interactions over standard interfaces are intentionally left abstract in this thesis. The interested readers are advised to refer to the relevant publications. In this figure, NS catalog = network service catalog and NSs = network services.

2.2.2 NFV-MANO for RAN Slices

In previous sections, we assumed the vCU and vDU as VNFs of a RAN slice. The VLs that connect the internal components of these functionalities are said to be the internal VLs. The connection of the vCU and vDU among each other and with the 5GC and RU are

established by external VLs. These VNFs and VLs of a RAN slice require virtual resources such as virtual compute, networking, and storage. The allocation of virtual resources and the management of the VNFs and their connectivity to PNFs are not supported by the 3GPP-NSMS as they fall within the scope of the ETSI NFV-MANO. The NFV-MANO is built of entities, descriptors, repositories, and FBs. These components are connected to the 3GPP-NSMS – which is also called the OSS/BSS – in order to establish a unified functioning framework for the management and orchestration of the virtualized and physical parts of a RAN slice in NG-RAN (see Fig. 2.2).

The entities of the virtualized part of a RAN slice, managed by NFV-MANO, are the RAN functions (i.e., vCUs and vDUs) realized as VNFs, external and internal VLs, the connection points of the VLs, the interconnection of the RAN VNFs to PNFs, and the VNF forwarding graphs (VNFFGs). The vCU, vDU, and their respective internal components are connected by the VLs among each other and with the RU. To specify how the connection pints of the VLs are used to connect the vCU, vDU, and RU, the VNFFG is proposed by the ETSI to describe the topology of the virtualized part of a RAN slice. These elements are present in each type of RAN slices. To manage these entities and their corresponding virtual resources with full automation, great agility, and effective scaling, the NFV-MANO utilizes customized machine-processable descriptor files for each of the given entities.

The descriptors are the deployment templates, which specify the operational, performance, functional, resources, and policy requirements of each of the aforementioned entities of a RAN slice [94]. They are composed using data modelling languages such as YAML, TOSCA, YANG, or others. The information gathered in these model-driven descriptors help the NFV-MANO to manage the vCU, vDU, and their required virtual resources throughout their life cycle with full automation and a great level of control. These descriptors - which might have been previously on-boarded or newly generated followed by on-boarding on the NFVO - are generally related to *(i)* the NFs such as VNF descriptors (VNFDs) for vCUs and vDUs and PNF descriptors (PNFDs) for RUs); *(ii)* the network services namely the network service descriptors (NSDs); *(iii)* the VLs namely the virtual link descriptors (VLDs); and *(iv)* the topology namely the VNF forwarding graph descriptors (VNFFGDs). Among them, the descriptors related to the virtual resources (which are integrated into the VNFDs of the vCU and vDU) and topology (namely the VNFFGDs) are discussed in next chapter. We, therefore, intentionally skip further details on the rest of the description models of the NFV-MANO and the 3GPP-NSMS related entities. However, the interested readers are advised to refer to [52] for more details on the 3GPP related templates, and to [46], and the references therein for more insights into the NFV-MANO related descriptors.

The description models of all entities and other necessary information related to the virtualized parts of eMBB, URLLC, and mMTC types of RAN slices are predefined and stored in four repositories: the network service catalog, the VNF catalog, the NFV instances repository, and the network function virtualized infrastructure (NFVI) resources repository. Fig. 2.2 illustrates that these four repositories hold the (a) NSDs, VLDs, and VNFFGDs; (b) VNFDs for vCUs, VNFDs for vDUs, and PNFD for RUs; (c) information about all VNF instances and network service instances during their life cycles; and (d) information about the NFVI resources (including their status, such as available, allocated, reserved, utilized, and wasted) of all running entities of RAN slices, respectively.

In order to make use of these description models efficiently, operate multiple NSs over a shared NG-RAN architecture dynamically, and orchestrate the virtualized components presented in each types of RAN slices effectively, the NFV-MANO proposes three main management and orchestration FBs: the NFVO, virtual network function manager (VNFM), and virtualized infrastructure manager (VIM).

The NFVO on-boards and creates network service instances, VNFFGs, and VNF instances related to eMBB, URLLC, and mMTC types of RAN slices using their predefined description models in VNFD/NSD. It also validates and authorizes NFVI virtual resource requests of a RAN slice. Moreover, it interacts with NG-RAN NSSMF for a joint instantiation of the virtualized part of the RAN slice. The VNFM manages the life cycle of the VNF instances (namely the vCU and vDU) of a RAN slice. It also governs the overall performance and fault events related to the vCU and vDU that constitute a RAN slice. The VIM allocates, controls, and manages the virtual compute, storage, and networking resources required by each of the vCUs and vDUs of each of the RAN slices in their respective I-PoPs.
The NFV-MANO FBs, the 3GPP-NSMS entities, and the VNFs and PNF of a gNB are interconnected among each other using 3GPP and NFV-MANO standard reference points, as illustrated in Fig. 2.2. Through such an interoperable framework, the physical and virtualized parts of a RAN slice deployed over I-PoPs and cellular network infrastructure are managed effectively and their respective physical and virtual resources are allocated efficiently.

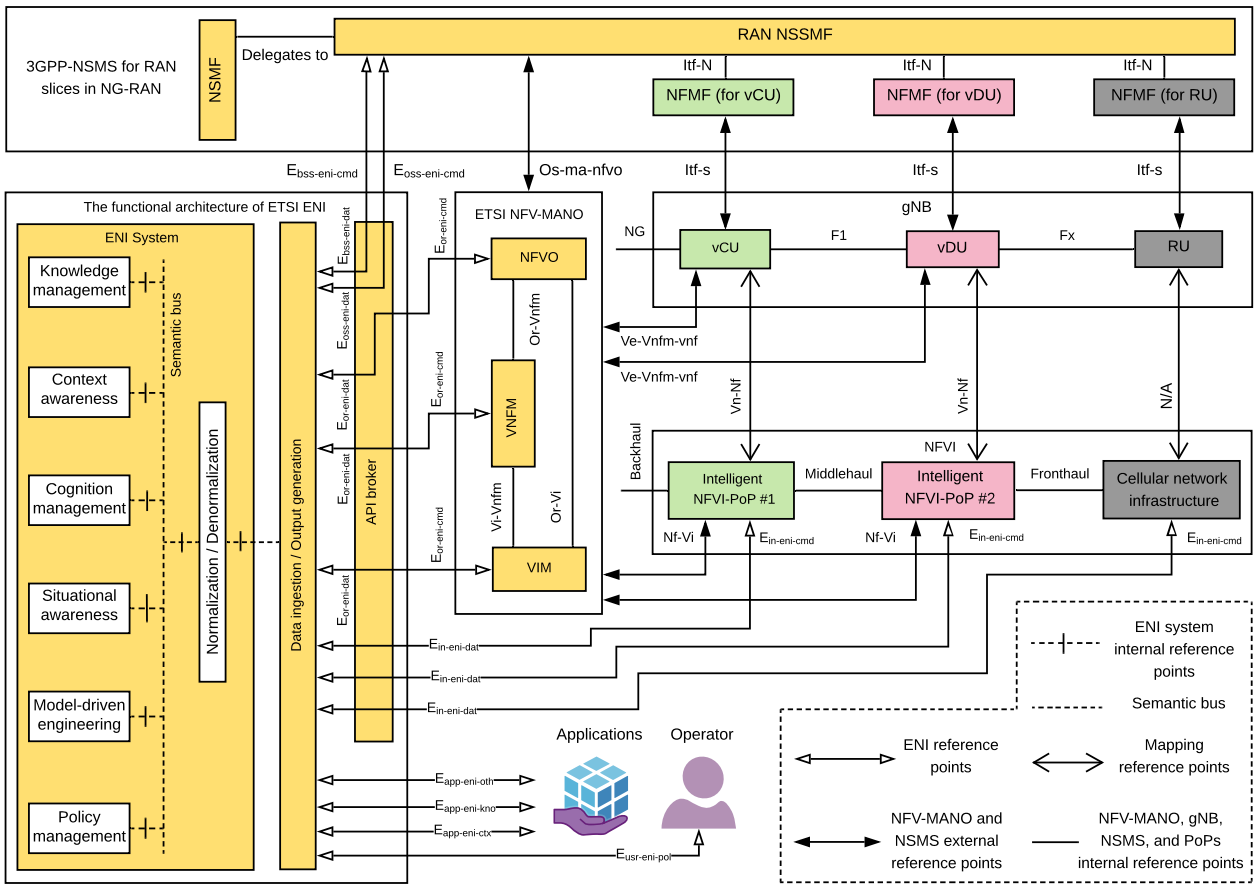

Figure 2.3: The proposed integration framework of the ENI system into the unified architecture of the NFV-MANO and 3GPP-NSMS for the automation of the MaO of RAN slices in NG-RAN.

2.2.3 The Deployment of RAN Slices in I-PoPs

In previous sections, we argued that the virtual resources of vCU and vDU are hosted by the NFVI-PoPs, which are the key components of the ETSI NFV architecture. With the view of our proposal extending intelligence to the NFVI-PoPs, we will refer to them as I-PoPs. Both the I-PoP #1 and I-PoP #2 (see Fig. 2.3), which respectively host the vCU and vDU, are two

major building blocks of the infrastructure layer of our proposed RAN slicing architecture, illustrated in Fig. 2.1 and described in previous section. The management, orchestration, and allocation of virtual compute, storage, and networking resources of the vCU and vDU in I-PoP #1 and I-PoP #2 are accomplished by their respective Hypervisors (see Fig. 2.2). However, the overall management and orchestration of the virtual resources of a RAN slice in I-PoP #1, I-PoP #2, the interconnection of both I-PoPs over Middlehaul interface, and the interconnection of the virtualized part with the cellular infrastructure over Fronthaul interface are executed by the VIM. The physical resources of the RU of a RAN slice are managed and allocated by the 3GPP-NSMS, specifically the NG-RAN NSSMF. To allocate, manage, and orchestrate the physical and virtual resources of the radio functionalities of $n \in \mathbb{N}^+$ eMBB, URLLC, and mMTC types of RAN slices in underlying infrastructure layer, the unified framework of the NFV-MANO and 3GPP-NSMS illustrated in Fig. 2.2 shall dynamically harmonize their interactions aiming to deliver an efficient RAN slicing in the NG-RAN architecture.

The vCUs, vDUs, and their respective VLs of RAN slices demand their customized virtual compute, storage, and networking resources in I-PoPs. These virtual resources of a RAN slice shall be allocated, managed, and orchestrated in complete isolation from the virtual resources of other RAN slices running over the same I-PoPs. In order to simplify the allocation, management, and orchestration processes, the NFV-MANO utilizes machine-processable descriptors related to the virtual compute, storage, and networking resources of the vCU and vDU. These descriptors, which are the main constituents of the VNFD of vCU and VNFD of vDU, are used among the FBs of the NFV-MANO and during the interaction with the 3GPP-NSMS entities. We discuss the description models related to such virtual resources of a RAN slice, in more details, in next chapter.

2.2.4 The Integration of the ENI framework into the unified architecture of the NFV-MANO, 3GPP-NSMS, and I-PoPs

The proposed integration of the functioning architecture of the ENI into the unified architecture of the NFV-MANO and 3GPP-NSMS (see Fig. 2.2), aiming to automate the management and orchestration of RAN slices in NG-RAN, is illustrated in Fig. 2.3. The ENI is integrated through external reference points to 3GPP-NSMS entities, NFV-MANO FBs, I-PoP #1, I-PoP #2, and cellular infrastructure. The main objectives of the integration of the ENI System into the aforementioned assisted systems are to render intelligence and automation to their respective management and orchestration tasks. Moreover, it is also connected to a set of applications, which are required by the ENI to participate in the automation of RAN slices. The ENI System shall also be controlled by an admin, in operator's domain, through an external reference point as shown in Fig. 2.3. The proposed functioning architecture of the ENI, illustrated in Fig. 2.3, is composed of API Broker, Data Ingestion and Output Generation FBs, Normalization and Denormalization FB, internal entities, and internal and external reference points [47].

The Data Ingestion is an input facing FB of the ENI architecture, which is used to ingest structured, semi-structured, and unstructured data – provided by streaming, batch, and/or on-demand mechanisms – coming from 3GPP-NSMS entities (related to the PNFs of RAN slices), NFV-MANO FBs (related to the VNFs of RAN slices), I-PoPs (related to the virtual resources of RAN slices), and cellular infrastructure (related to the radio resources of RAN slices). The Data Ingestion FB performs filtering, correlation, cleansing, anonymization and pseudonymization, augmentation, and labelling operations on raw data collected from the assisted systems. Once the data collection and ingestion processes from three of these

assisted systems are completed, the ingested data is then sent to the Normalization FB in order to interpret and normalize it into a single common and unified data format that is understandable for further processing by the internal entities of the ENI System. The normalization of ingested data related to various types of RAN slices into a standard format, on one hand, enables network operators to quickly learn about the optimal parameters of each of the nodes located in the assisted systems, on the other hand, reduces complex computational problems.
Normalized data coming from Normalization FB is then sent to the Semantic Bus, which is filtered by the Knowledge Management entity. The ENI System is also composed of other five entities - such as Context Awareness, Cognition Management, Situational Awareness, Model-driven Engineering, and Policy Management - which are based on the observe-orient-decide-act process. These six entities (see Fig. 2.3) produce recommendations, commands, and knowledge for the ENI System in order to assist the management and orchestration related operations of NFV-MANO and 3GPP-NSMS for RAN slices in NG-RAN. These internal entities use existing knowledge and/or add new knowledge to enable the ENI System to adapt its behaviour according to the dynamic changes in the assisted systems. Once the commands and recommendations are generated by the ENI entities, they are delivered by the Knowledge Management entity to the Denormalization, which is an output facing FB.
The Denormalization FB executes the inverse operations of the Normalization FB. It specifically means that - once it receives the processed data from the Knowledge Management entity - it is used to translate the processed data coming from the ENI's entities into the format(s) that is(are) understandable by the assisted systems. The translated data is then forwarded to the Output Generation FB to deliver it to the NFV-MANO, 3GPP-NSMS, and I-PoPs. The assisted systems utilize the commands and recommendations of the ENI System to automate, among others , the management and orchestration related operations of RAN slices in NG-RAN. The Denormalization FB communicates with the assisted systems through an API broker in order translate the APIs of the ENI System to the APIs of the assisted systems (see Fig. 2.3). The API Broker shall be compliant with the internal and external reference points of the ENI System. The API Broker also defines a correct way for one assisted system to request services from other assisted system without requiring ENI to understand the details of every APIs of the assisted systems that communicate with each other.
Despite the novel implications - in terms of automation and intelligence - of the ENI on the operations of the 3GPP-NSMS, NFV-MANO, and I-PoPs, there is a number of research problems which are remained unsreolved and need to be addressed in the future reports of the ENI ISG. Among others, the internal reference points between the entities and the FBs of the ENI Systems are not yet defined. There is an urgent need for such a study in order to design a unified ENI framework that is applicable to a wide-range of use-cases including the RAN slicing in NG-RAN. In addition, the integration of the ENI into the assisted systems - especially into the 3GPP-NSMS, NFV-MANO, and I-PoPs - is at an incipient stage. It requires deep study to find out its integration into the FBs of the aforementioned assisted systems, explore the functionalities that are needed to be automated, and define their relevant internal and external reference points.

Chapter 3

Contributions to the Mapping Process of the VNFs and VLs of RAN Slices onto Intelligent-PoPs

Summary – In the previous chapter, we defined the virtual and physical components of different types of RAN slices. We also defined the underlying infrastructure that host such components throughout their lifetime. In this chapter, we address the mapping process of a RAN slice, which is initially a challenging task due to the joint allocation of virtual resources across the nodes and links, the diverse technical requirements of the tenants, the coordination between multiple host domains, and others. This issue is exacerbated further by the extension of virtualization to the NG-RAN architecture and the provisioning of RAN slicing. To that end, this chapter focuses on the mapping problem of the VNFs, as well as their internal and external VLs, of a RAN slice subnet onto I-PoPs and transport networks in the NG-RAN architecture. In this context, in contrast to the majority of the state-of-the-art proposals, which frequently fail to achieve performance objectives and neglect resource allocation constraints, the proposed solution introduces automation and intelligence at an architectural level to map VNFs and VLs onto their corresponding physical nodes and links, with the goal of achieving superior efficiency in virtual resource utilization while granting the performance of a RAN slice subnet. Following these discussions, in this chapter, we model an ENI-enabled functioning framework to define the mapping problem. To accomplish this task, we first present the mapping process of the vCU and vDU. Next, we address the mapping process of the internal and external VLs. Then, we discuss the VNFFGs for the eMBB, URLLC, and mMTC types of RAN slices. We also address the internal domains of an I-PoP and provide an in-depth discussion of their virtual and physical components. Finally, we present several assumptions, main objectives, and a number of constraints that are critical to take into consideration prior to proposing an architectural solution.

3.1 Defining and Modeling the Mapping Problem of the vCU, vDU, and VLs of a RAN Slice onto I-PoPs and Transport Networks in the NG-RAN

3.1.1 Defining the Mapping Problem of the vCU and vDU

In Chapter 2, we assumed that the vCU and vDU are the VNFs of a RAN slice. We also discovered that a vCU and a vDU have three and five internal functionalities, respectively

(see Fig. 2.1). These internal functionalities are referred to as virtual network function components (VNFCs) in this thesis. According to the ETSI ISG NFV, the VNFC is a subset of a VNF that performs a well-defined application within the scope of that VNF [102]. Based on this, the vCU has three VNFCs (i.e., VNFC #1 – VNFC #3) that deploy the RRC, SDAP, and PDCP functionalities, respectively, as illustrated on the upper side of Fig. 3.1. Similarly, the vDU is composed of five VNFCs (i.e., VNFC #1 – VNFC #5), each of which implements the RLC-high, RLC-low, MAC-high, MAC-low, and PHY-high functionalities, respectively. This internal partitioning of vCU and vDU leads to effectively managing the components of a RAN slice, optimally allocating virtual resources to such components, and efficiently enabling and disabling of RAN slice-specific functionalities in a gNB [103]. Additionally, it enables the NFV-MANO to deploy, scale, and upgrade the VNFCs of vCU and vDU independently. Rather than scaling the complete VNF in a RAN slice, only the relevant VNFC within that VNF is scaled.

Furthermore, each VNFC is assumed to be instantiated on a single VM. It is also worth nothing that a VNF might theoretically contain several critical VNFCs that require strict isolation and tight security [104] [105]. To instantiate them, the ETSI proposed the hardware-mediated execution enclave (HMEE) in [106]. Each HMEE is used to host such a critical VNFC in order to protect it from other VNFCs of a VNF in the form of hardware isolation. To ensure compliance with Net Neutrality regulations, we assume that all VNFCs are treated equally in terms of their security and isolation [107]. To that end, we avoid detailed discussion of HMEE in this thesis. Hence, we assume that all VNFCs within the vCU and vDU, without loss of generality, are instantiated on VMs using a one-to-one mapping approach in this thesis. This is also illustrated on the upper side of Fig. 3.1, where the VNFC #1 – VNFC #3 of vCU are mapped onto VM #1 – VM #3 in I-PoP #1, respectively. Similarly, the VNFC #1 – VNFC #5 of vDU are mapped onto VM #1 – VM #5 in I-PoP #2, respectively.

The left side of Fig. 3.1 illustrates the VNFD of a vCU as well as the VNFD of a vDU. The VNFD defines the deployment and behavior requirements of the vCU and vDU. It contains information about the network connectivity, interfaces, resources, and performance requirements of a VNF. The NFV-MANO FBs use the VNFD to make sure that the VNFCs of the vCU and vDU are placed on the right VMs in the I-PoPs. Each VNFD, among other description information, is made up of a virtual computing descriptor (VCD) and a virtual storage descriptor (VSD). The VCD contains information related to the required amount of virtual compute resources of the vCU and vDU. The VSD references the amount of virtual storage resources that are required by the vCU and vDU.

To specify the amount of virtual compute and storage resources at the VNFC level, the ETSI ISG NFV proposed the virtualization deployment unit (VDU) in [40]. The VDU acts as a descriptor file for a VNFC in the proposed architecture. It is used to describe the amount of virtual compute and storage resources, which are required by a VM to host its associated VNFC. The VDU is a critical component of the VNFD. By properly defining the parameters and specifying their values of the VDUs of a VNF, the VNFCs of the vCU and vDU can be placed onto suitable VMs, enabling a more reliable, efficient, and performant mapping process of a RAN slice onto the underlying I-PoPs in the NG-RAN architecture.

3.1.2 Defining the Mapping Problem of the External VLs and Internal VLs

Each VNFC has internal connection points that are used to establish internal virtual connections between other VNFCs within the same VNF[27]. When two connection points of two VNFCs of a given VNF are connected, an internal VL is established. This is also highlighted

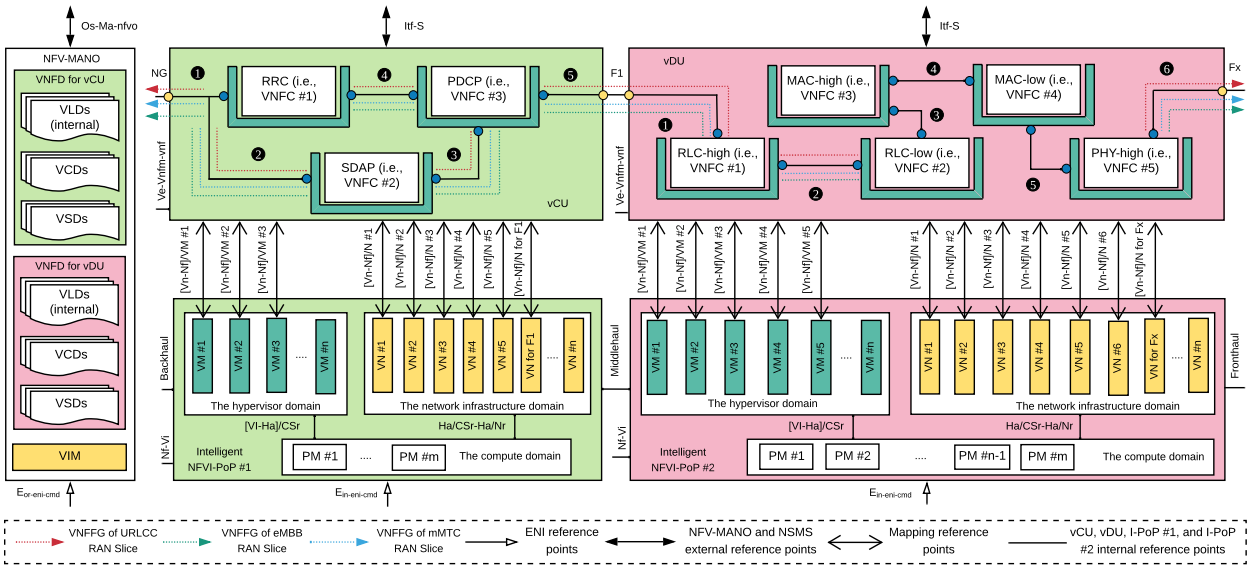

Figure 3.1: The proposed architecture for automating the management and orchestration of RAN slices (see Fig. 2.3) is reduced to a reference point architecture that only shows the mapping of the VNFCs of vCU and vDU, internal VLs, and external VLs in both I-PoPs. The VNFCs and VLs are mapped onto VMs and VNs using a one-to-one mapping approach via standard interfaces. These standard interfaces, connecting the various domains of the NFV-MANO with the vCU and vDU, are illustrated in the figure. Due to space constraints, VL #3, VL #4, and VL #5 of vDU are intentionally left blank in the VNFFGs.

in dark blue color on the upper side of Fig. 3.1, where the three VNFCs of the vCU and the five VNFCs of the vDU are connected using five and six internal VLs, respectively. Additionally, the vCU and vDU have external virtual connection points (highlighted in yellow color) that are used to establish external virtual connections between the vCU and 5GC, the vCU and vDU, and the vDU and RU via external VLs, namely the next generation (NG), F1, and Fx interfaces, respectively. The external VL is created by linking two virtual connection points of two VNFCs belonging to two different VNFs [108].

In both intra-and inter-VNF connectivity scenarios, each VL is assumed to be mapped onto a single VN using a one-to-one mapping approach. On the basis of these assumptions, the five internal VLs of vCU and F1 are mapped onto six VNs in I-PoP #1. Similarly, the six internal VLs of vDU and Fx are mapped onto seven VNs in I-PoP #2 (see Fig. 3.1). The deployment of the internal VLs at each I-PoP, as well as the deployment of the external VLs between I-PoP #1, I-PoP #2, and the cellular network site, should be coordinated in such a way as to deliver an E2E unified RAN slice. The mapping of the NG interface is beyond the scope of this thesis.

It is worth noting that the NFVO maintains information about the logical connectivity (reachability) within an I-PoP, between the I-PoPs, and the overall networking topology of the underlying transport infrastructure. This information enables the NFVO to determine the appropriate I-PoPs (as well as physical paths on middlehaul and fronthaul) for mapping the internal and external VLs while taking into account the underlying infrastructure's placement constraints. This information is also used by the NFVO to help with the networking resource orchestration of the RAN slice. The NFVO provides network connectivity information to the VIM to deploy and manage the virtual networking resources of the internal VLs in an I-PoP. However, the information related to the external VLs is provided to the Wide area network Infrastructure Manager (WIM) in order to deploy and manage the virtual networking

resources over the backhaul and fronthaul links in the NG-RAN architecture. The WIM is a specialized type of VIM that is standardized by the ETSI to manage the virtual networking resources of the external VLs between the I-PoPs and between the I-PoPs and the cellular network sites. For the sake of simplicity, we assume only the VIM for the management and orchestration of the virtual networking resources of both internal and external VLs of a RAN slice in this thesis.
The VNFDs, depicted on the left side of Fig. 3.1, are also composed of VLDs that gather information related to the virtual networking resources of the vCU and vDU, as well as their interconnection with the RU in a gNB. The information contained in a VLD is utilized by the NFVO to determine the optimal mapping of a VL instance. In addition, it is also used by the VIM to determine and manage the required virtual networking resources associated with a VL instance in an I-PoP. Each VLD defines the basic topology of the network connectivity within or between the VNFs of a RAN slice, as well as additional parameters (e.g., bandwidth, latency, QoS class, connectivity type, security, etc.) with their values that describe the required performance of a VL. These performance parameters are included in all VLDs. Their values, however, vary depending on the type of RAN slice. There are two types of VLDs: the external VLD and the internal VLD. The former is used to describe the topology and virtual networking resources that are required to connect the 5GC and vCU, the vCU and vDU, and the vDU and RU of a RAN slice. The latter is used to describe the topology and virtual networking resources that are required to connect the VNFCs whiten the vCU and vDU of a RAN slice. It should be noted that the external VLDs are the components of the NSD (as shown on the left side of Fig. 2.2), whereas the internal VLDs are components of the VNFD (see the left side of Fig. 3.1).

3.1.3 Defining the eMBB, URLLC, and mMTC Types VNFFGs

Once the VNFCs and VLs (internal and external) of a RAN slice have been mapped onto VMs and VNs, the next step is to determine the routes over which the requested traffic should flow – both in downstream and upstream directions. To that end, the VNFFG was proposed by ETSI ISG NFV in [40]. The VNFFG is an effective tool to design, deploy, and manage a RAN slice. To map the VNFFG of a RAN slice successfully, the placement of its VNFCs and VLs onto their respective virtual components should meet the performance requirements while also minimizing the total cost of deployment. It is also used to define the type and amount of requested traffic – matching certain criteria – intended to flow from a source through VNFCs and VLs to a destination in a RAN slice [109]. The traffic flow is controlled by a forwarding path, which is an element of the VNFFG. Each VNFFG may contain at least one forwarding path, which represents the path taken by actual traffic flow on a VL.
Such behavioral and deployment information of a VNFFG is defined in a customized template called the VNFFGD [110]. Similar to other descriptors, the VNFFGD is created in a data modeling language and is subsequently on-boarded into the VNFFG catalog [110]. The VNFFG of a RAN slice is rendered into SFC and a classifier. The former (as standardized by the IETF) creates an ordered list of VNFCs through which traffic should flow from a source to a destination in a forwarding path [111, 112]. The latter is used to classify various types of traffic in accordance with tenant preferences and to filter traffic that should flow through the VNFCs and VLs [113].
There are three types of VNFFGs used to efficiently manage traffic flow in a gNB, as shown in Fig. 3.1: the eMBB type VNFFG, the URLLC type VNFFG, and the mMTC type VNFFG. Each type of VNFFG must satisfy the internal and external connections, traffic

type, and packet flow requirements of its respective RAN slice. It must also clearly illustrate the end-to-end topology, priority dependency between the VNFCs, and overall forwarding rules of the vCU, vDU, and their communication channels with the 5GC and RU in order to meet the demands of the requested traffic [114]. Lastly, the VNFFG must also specify which VNFC (or a feature thereof) of vCU and vDU should be enabled or disabled in order to efficiently meet the performance requirements of a RAN slice. This feature also leads to indicating the exact amount of virtual resources required by a RAN slice in the eMBB, URLLC, and mMTC use case scenarios.
Once the aforementioned information has been specified in the VNFFGDs (of the eMBB, URLLC, and mMTC types of RAN slices) and the VNFFGDs have been uploaded to the network service catalog, the VIM must always use these descriptors to command I-PoP #1 and I-PoP #2 in order to configure and instantiate the VMs and VNs of the respective RAN slice instances [115]. It is worth noting that a RAN slice typically employs a single point-to-point VNFFG, while in some complex use case scenarios, multiple VNFFGs may be deployed to instantiate a RAN slice [116]. In either of the scenarios, if the VNFFG is known ahead of time, the mapping of a RAN slice is a static optimization problem. Conversely, if the VNFFG is unknown, the mapping of the RAN slice becomes a dynamic optimization problem.

3.1.4 The Internal Domains and Components of an I-PoP

We have previously stated that an I-PoP is a geographical location where the vCU and vDU, as well as the VLs, of a RAN slice are hosted by the VMs and VNs, respectively. The required virtual compute and storage resources of the VNFs, as well as the required virtual networking resources of the internal VLs, are abstracted from the underlying physical compute node and networking links of an I-PoP, respectively [42]. The external VLs of a VNF, on the other hand, are abstracted from the underlying transport links (namely the backhaul, middlehaul, and fronthaul). These transport links connect the I-PoPs and are considered the components of the infrastructure layer [117], as shown in Fig. 2.1. Should the need arise, the transport links must migrate data and resources of the VMs and VNs of a RAN slice from one I-PoP to another without recreation, re-entering data descriptions, and significant modification of the application(s) being transported [118].
Each I-PoP is divided into three domains: the compute domain, the hypervisor domain, and the network infrastructure domain. The goal of such a partition of an I-PoP is to effectively control and administer the complexity of the infrastructure layer [42]. Although there is always some functional overlap between these domains, they are largely distinct at functional, structural, and practical levels. The positioning of these domains within our proposed architectural framework is shown in Fig. 3.1 and is discussed further below.

The Compute Domain

The compute domain is composed of a set of industry-standard high-volume physical machines (PMs), network interface cards (NICs), input/output accelerators, and storage building blocks [43].
The PMs, along with storage and peripheral equipment, are housed in a server chassis in the form of rack/blade-servers [119]. Each PM, also referred to as a compute node or physical server, is composed of high-speed single/multi-core(s) CPUs and random-access memory (RAM) [119]. These physical resources are first virtualized into vCPUs and virtual random-access memorys (vRAMs), and then allocated to VMs (based on a dynamic resource

allocation algorithm) in order to execute the codes of the VNFCs [43]. Each VM is thus equipped with a certain number of vCPUs and vRAMs. The virtualization of a PM into a number of VMs is accomplished in conjunction with a hypervisor that is located within the hypervisor domain [42]. To that end, the three VNFCs of vCU and the five VNFCs of vDU require three and five VMs, which are abstracted from underlying PMs and are subsequently allocated through hypervisor domains in I-PoP #1 and I-PoP #2, respectively, as shown in Fig. 3.1.

The NIC and input/output accelerators are used to connect the PMs, provide network input/output functionality to the CPUs of PMs, and connect other equipment within the compute domain [120]. The NIC is also virtualized into a set of virtual network interface cards (vNICs) using a hypervisor. Each vNIC is used by a VM to represent its configuration when communicating with other VMs [121]. A VM can be configured to have a single or multiple vNIC(s), each of which is connected to a different VM. However, the traffic between the VMs is connected by a vSwitch that is abstracted from the underlying physical switch [121]. Furthermore, the NIC is responsible for providing physical connectivity between the compute domain and network domain of an I-PoP. In this context, the compute domain uses this type of interconnection exclusively for its internal communication. It does not necessarily support E2E network connectivity between the components of a RAN slice in the NG-RAN architecture [42].

The storage node, which may be coupled with a PM or deployed separately in a storage chassis, stores permanent and temporary data of the VNFCs in the form of hard disk drives, solid state disks, or hybrid disk drives [42]. Each storage node is distinguished by its capacity and a specific level of latency associated with accessing a state held in storage in order to execute an instruction cycle, as well as security, resiliency, cost, and the volatility of the storage [122, 42]. To meet the virtual storage resource requirements of the VMs, the storage nodes are also virtualized using a hypervisor into virtual memory (vMemory) resources. The vMemory could be defined during the creation phase of a VM. It could also be changed or reconfigured during the operation phase in order to accommodate the dynamic changes in virtual resource requirements of a VNFC.

The Hypervisor Domain

The hypervisor domain is a software environment that is used to abstract virtual compute and storage resources from the underlying physical compute domain, mediate these virtual resources to the VMs, and provide management interfaces to the VIM for the management, orchestration, load-balancing, and monitoring of the VMs [44]. In addition, the hypervisor domain provides an emulated vNIC(s) to each VM allocated to the VNFCs of the vCU and vDU. The vNICs are connected to the vSwitches, located within the hypervisor domain, in order to provide connectivity among the VMs as well as between the VMs and the physical NIC located in the PM.

The VM, vNIC, and vSwitch provided by the hypervisor domain must be equivalent to their original physical environments in terms of performance, functionality, and resource efficiency. Furthermore, the hypervisor domain provides a mechanism for VM migration (running on the same or different PMs or I-PoPs) in order to ensure the atomicity of VM instances [123]. To effectively execute the aforementioned tasks (among others), the hypervisor domain provides the VIM with a list of predefined performance metrics related to the above operations at regular intervals in order to produce real-time and accurate predictions. This significantly improves the performance of VMs, increases their resource utilization ratio, and reduces their energy consumption.

Table 3.1: The list of performance objectives (i.e., metrics) that could be considered for optimization during the mapping process of the virtual components of a RAN slice onto I-PoPs and transport networks in the NG-RAN architecture.

Category	Metric	Category	Metric	Category	Metric
Compute and storage resources	Instantiated VMs of vCU and vDU	Energy consumption	Power consumption of a PM	Service level agreement	Priority
	Active PMs in I-PoP #1 and I-PoP #2		Required power of a VM		Preferences
	VMs resource utilization rate		Power consumption of a VM		Security
	PMs resource utilization rate		VM/PM/I-PoP power wastage		Secracy
	Virtual/physical compute resource usage		Carboon footprint		Privacy
	Virtual/physical storage resource wastage		Thermal dissipation		Network scalibility
	VNFC resource demands prediction		Energy consumption of a VNF and VL		Elasticity
	Intra RAN slices resource allocation	Business	CAPEX		Availability
	VM, PM, and I-PoP stress level		OPEX		Runtime of mapping algorithm
	PM shareability/anti-shareability		Revenue		API authorization level
	Reserved VMs and PMs		Cost		Area of services
	VMs colocation/anti-colocation		Profit		SLA violation
	Virtual and physical resources load balance		Price		Multi-tenancy
Networking resources	VLs resources demand prediction	Migration	VM migration time	RAN slice specific	Downlink per UE/RAN slice
	Jitter, latency, packet loss, and delay		Migration cost		Uplink per UE/RAN slice
	Reserved VLs and PLs		Faiulre rate		Isolation level
	VL and PLs utilization rate		Response time		Maximum number of UEs
	VLs and PLs resource wastage and usage		Number of migration of VMs		Radio spectrum range
	Backhaul/midlehaul/fronthaul distance		Migration overhead		(De)Modulation scheme
	Path redundancy		Inter-VM dependency		(De)Coding scheme
	VL and PL reliability		Interference among VMs		Mobility management
	Transmission cost		Geographical location		Position accuracy
	Path length		Migration acceptance ratio		Mobility interruption time

The Network Infrastructure Domain

The network infrastructure domain contains a large number of high-volume switches that are connected to a reliable transport network in order to provide underlying networking resources for the internal and external VLs of a RAN slice [45]. More specifically, it establishes virtual communication channels between the 5GC and vCU, the VNFCs of the vCU and vDU, the vCU and vDU, and the vDU and RU. The virtual communication channels are abstracted from the underlying physical networking resources by utilizing the resource routing and sharing control layer of the virtualization layer that exists within the domain and are subsequently provided in the form of VNs to their respective VLs. Each VL is obtained by establishing a virtual connection between the vNICs of at least two VNFCs. In order to logically connect the VNFCs within/between the vCU and vDU, their respective vNICs need to be configured and then connected via a vSwitch. The network infrastructure domain provides the VIM with a detailed view and reports on virtual and physical nodes aimed at managing and administering these operations and network equipment. These reports and requests are analyzed by the VIM in order to manage and orchestrate the virtual resources – among other operations – associated with inter and intra RAN slice(s) in both I-PoP #1 and I-PoP #2 in the NG-RAN architecture.

3.1.5 The Assumptions, Objectives, and Constraints of the Mapping Process of a RAN Slice

This subsection makes several assumptions (at the technical and application levels) in order to derive valid conclusions and correct inferences from the mapping process of the virtual components of a RAN slice. Following that, we present the main objectives we are attempting to achieve through the utilization of the unified functioning architecture of the NFV-MANO, 3GPP-NSMS, the underlying infrastructure, and ENI System. Then, we provide a number of constraints that limit the scope of the proposed mapping architectural solution. These aspects of the mapping process must be clearly defined in the SLA and continuously monitored for agreement violations during the lifetime of the requested RAN slice. If these requirements

are not met, both the service provider and the tenant should include an appropriate penalty value in the SLA. When a service provider fails to deliver assured services, the tenant should impose the penalty according to the agreement.

Assumptions

For the sake of clarity, we will first make a number of reasonable assumptions prior to mapping the virtual components of a RAN slice onto the underlying infrastructure. These assumptions are as follows:

- We assume the mapping of a single RAN slice in this thesis. To accomplish this, the RAN NSSMF must first ask the NFVO to instantiate a vCU, a vDU, as well as internal and external VLs, all of which are associated with the requested RAN slice. Notably, we make no assumptions regarding the SST of the requested RAN slice. It could be any of the three SSTs discussed in the previous chapter.

- Following the instantiation of the VNFs and VLs of the requested RAN slice, the associated descriptors (such as VNFDs, VNFFGD, NSD, internal and external VLDs, NSD, and so on) are derived from their respective catalogs and delivered to the NFV-MANO FBs. These descriptors are assumed to comprise (among other attributes) the approximate amount of virtual resources required for each of the VNFCs and VLs based on the SST of the requested RAN slice.

- We assume that the VNFFG of the requested RAN slice has been determined in advance. This specifically means that the VNFCs of the vCU and vDU, as well as the internal and external VLs connecting the VNFCs, are anticipated to be processed sequentially and chained exactly as illustrated in Fig. 3.1.

- Each VNFC requires a fixed number of virtual compute and storage resources. Each VL, whether it is internal or external, utilizes a certain amount of bandwidth. The amount of virtual resources required by these components is assumed to vary over the lifetime of the requested RAN slice. Hence, we assume that it is a dynamic optimization problem from a resource allocation perspective.

- The VNFCs of a VNF can be hosted by the same or different PMs. In the latter case, the bandwidth of each internal VL is limited to a fixed amount, whereas in the former case, each internal VL has an infinite bandwidth [124]. Additionally, the operations management, maintenance, and resource orchestration are not sophisticated in the first scenario. On the basis of these advantages, we assume that the VNFCs of vCU and vDU of the requested RAN slice are placed on the same PM in I-PoP #1 and I-PoP #2, respectively.

Objectives

Based on the aforementioned assumptions, the primary goal of the mapping of the requested RAN slice onto I-PoPs and transport networks is to find an optimal solution aimed at optimizing (maximizing, minimizing, or balancing) a set of predefined performance objectives (i.e., metrics), which are listed in Tab. 3.1. These objectives, which are grouped into seven categories, are used to evaluate the performance of a successful mapping solution of a RAN slice using publicly available analytical and simulation tools. The mapping of a RAN slice can be a single-objective or multiple-objective optimization problem. In the former case, a

single objective can be optimized during a specified time interval; in the latter case, multiple objectives can be optimized during a specified time interval. The following are some common objectives:

- One of the objectives is to keep the number of active PMs in an I-PoP to a minimum. This can be accomplished efficiently by utilizing state-of-the-art ML-assisted algorithms to predict the virtual resources required for the VNFCs. The underlying PMs can then be partitioned into multiple VMs, each of which is configured and allocated autonomously to its respective VNFC based on resource predictions. This results in avoiding under-utilization and over-utilization of virtual resources and minimizing resource wastage in an I-PoP.

- The second common objective is to keep the total bandwidth consumption of the underlying PLs in transport networks to a minimum. To accomplish this objective, the required virtual networking resources for each VL in the requested RAN slice can be predicted in the first stage using ML-assisted algorithms. Following that, the PL must be partitioned in such a way that the total required bandwidth of the VLs does not exceed the total bandwidth of the host PL. Finally, the VL must be mapped autonomously onto the most suitable PL in the transport network.

- Reducing the number of VM and VN migrations from an over-loaded PM and PL to a less-loaded PM and PL is another common objective that can be achieved by the proposed unified functioning architecture. To accomplish this task, a real-time ML-assisted migration algorithm is required that (a) predicts the future resource demands of VMs or VNs based on their historical resource requirements, (b) places VMs and VNs on nodes and links with available resources, and (c) monitors the resource consumption of the nodes and links on a regular basis. By automating these tasks, the number of migrations of virtual components of a RAN slice is expected to decrease to a minimum.

- Decreasing the power consumption of I-PoPs and transport networks is one of the critical performance objectives of the mapping process of a RAN slice. Among other factors, the energy consumption of I-PoPs and transport networks can be increased primarily due to the high volume of communication traffic, the number of active PMs and PLs, inefficient utilization of resources, and cooling equipment. Thus, autonomous management of the components of the underlying infrastructure through the use of ML-assisted algorithms in the context of the proposed solution can significantly reduce energy consumption.

Constraints

The mapping problem of the VNFCs and VLs of vCU and vDU can be limited by a number of intrinsic and use-case-specific constraints that must be stated prior to proposing the architectural solution. Some of the most critical constraints that we believe have a significant impact on the quality of the performance objectives of the mapping process are enumerated below. It should be noted that the performance objectives (i.e., metrics) listed in Tab. 3.1 can also be used as constraints when formulating the mapping problem.

- The virtual compute and storage resource demands of a VNFC must not exceed the virtual compute and storage resource capacity of the host VM. Likewise, the sum of the virtual compute and storage resource demands of all VMs belonging to the VNFCs of a VNF (either vCU or vDU) of the requested RAN slice must not exceed the physical compute and storage resource capacity of the host PM in an I-PoP.

- The virtual networking resource demands of a VL should not exceed the virtual networking resource capacity of the host VN. Similarly, the sum of the virtual networking resource demands of the internal and external VLs belonging to a VNF (either vCU or vDU) of the requested RAN slice should not exceed the total available physical bandwidth of the underlying host PL in the transport network.

- Each VM and VN is limited to hosting a single type of VNFC and VL at a given time, respectively. For example, the VM hosting the MAC-high should be configured exclusively to host this particular VNFC throughout the lifetime of the requested RAN slice. This type of customization of VM and VN results in two advantages: (a) If performance degrades or the RAN slice fails, the root causes of the failure can be identified and resolved quickly and effectively. (b) The VNFCs and VLs can be shared between multiple RAN slices with identical or dissimilar SSTs, thereby optimizing the trade-off between isolation level, resource utilization, and customization.

- To avoid service interruptions and improve survivability, redundant VMs and VNs may be configured in each of the I-PoPs and transport network. The redundant VMs and VNs must be instantiated in the event that any of the activated (or primary) VMs or VNs fails, with the goal of rapidly recovering the requested RAN slice [125]. Due to these advantages, the redundant VM should not be configured on the same PM as the primary VM is hosted. Likewise, the redundant VN must not be configured on the same physical path as the primary VN is hosted.

3.2 The Proposed Architectural Solution for Mapping the vCU, vDU, and VLs of a RAN Slice onto I-PoPs and Transport Networks in the NG-RAN Architecture

Following the definition and modeling of the mapping problem in the previous section, we propose an architectural solution in this section that aims to jointly map the VNFs and VLs of a single RAN slice onto I-PoPs and transport networks in an automated fashion in the NG-RAN architecture. The solution leverages the ENI System in order to optimize the performance of the mapping process by utilizing cutting-edge ML-assisted techniques. The proposed solution executes the mapping process in three distinct phases: (a) the resource automation phase, which applies intelligence and automation to the mapping process of a RAN slice; (b) the resource management phase, which manages and orchestrates virtual resources for inter-and intra-RAN slices; and (c) the resource allocation phase, which allocates VMs to the VNFCs and VNs to the VLs of a RAN slice. These three phases are carried out by three sets of major building blocks, which are connected via standard reference points. The broad design of the proposed solution is depicted in Fig. 3.2. Its three phases, as well as their customized architectures derived from Fig. 3.2, are addressed in detail in the following subsections, respectively.

3.2.1 The Virtual Resource Automation Phase

The functioning architecture of the virtual resource automation phase is depicted in Fig. 3.3, which is derived from the general architecture of the proposed mapping solution in Fig. 3.2. This phase is executed by the ENI System. To begin, the proposed architectural

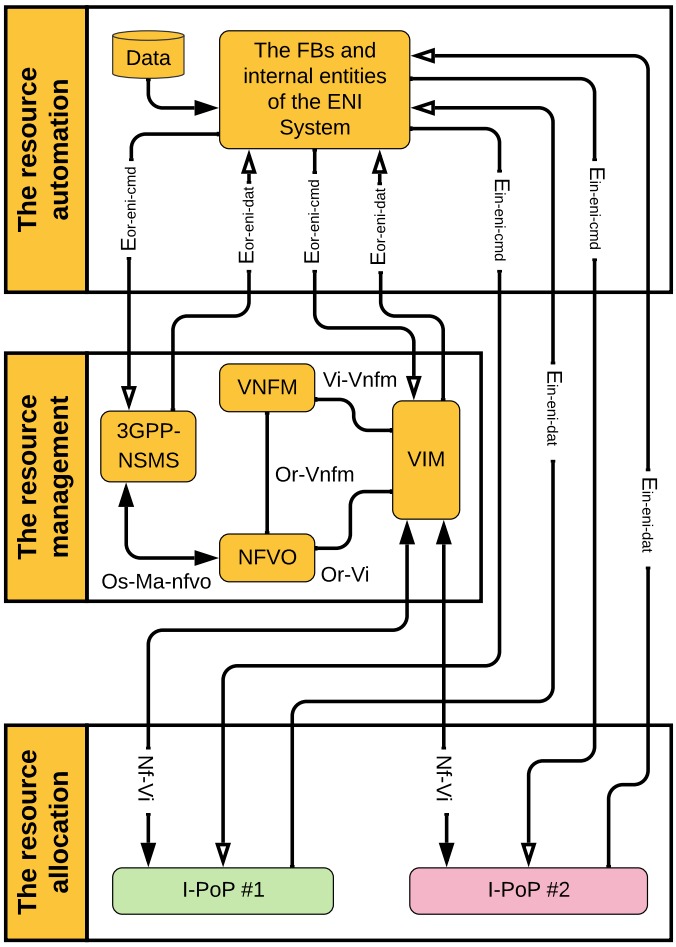

Figure 3.2: The three major sets of building blocks of the proposed architectural framework for mapping the vCU, vDU, and VLs of a RAN slice onto I-PoPs and transport networks in the NG-RAN architecture. The legends contained in this figure have been defined in the previous sections.

framework collects appropriate historical data – at a faster rate, with a higher sense of accuracy, and on a larger scale – pertaining to three categories: (a) the requirements of the eMBB, URLLC, and mMTC types of RAN slices; (b) the design and engineering of the three types of RAN slices; and (c) the calculation of the allocated, available, and reserved virtual resources in I-PoPs and transport networks (see step 1 in the resource automation phase of Fig. 3.3). The ENI System collects these three types of data from the 3GPP-NSMS, NFV-MANO, and the underlying infrastructure. From these three categories of historical data, the proposed framework selects only the data that is relevant to a set of performance objectives we are attempting to optimize during the mapping process of the requested RAN slice. These performance objectives include minimizing the number of active PMs, reducing the number of active PLs, decreasing the number of VM and VN migrations, and lowering energy consumption, among others.

The historical data (or collected data) is then cleaned, randomized, and visualized using a variety of data cleaning, randomization, and visualization techniques. It then partitions the collected data into two distinct sets: the training set and the evaluation set (steps 2-3). The training set is used to train the mapping model data (offline) in order to generate accurate predictions for a number of performance metrics, such as the required amount of virtual resources, the level of isolation, the end-user's (tenant's) preferences, and so on [126]. The proposed framework selects an appropriate ML-assisted algorithm prior to training the

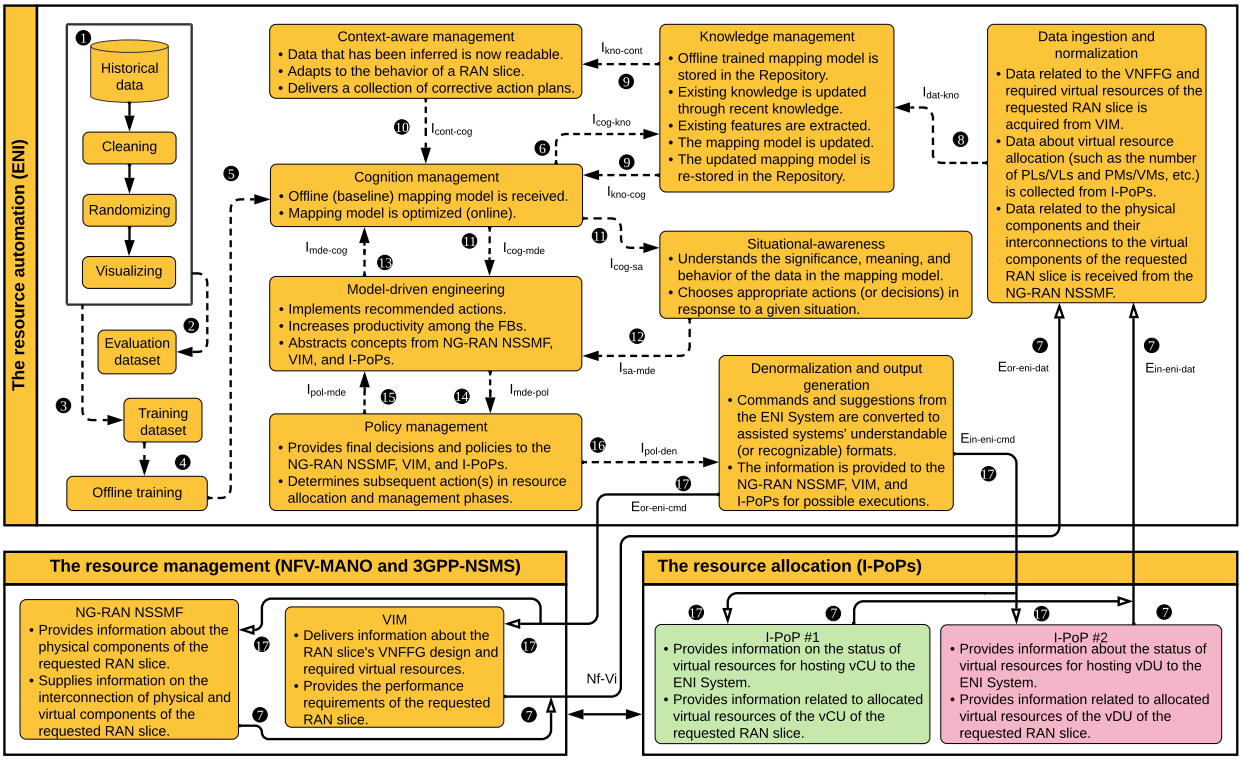

Figure 3.3: The proposed functioning architecture for the virtual resource automation phase of the mapping process of vCU, vDU, and VLs of a RAN slice onto I-PoPs and transport networks in the NG-RAN architecture. It should be noted that the resource automation phase is executed by the ENI System, and its architectural framework is derived from the functioning architecture of the proposed mapping solution illustrated in Fig. 3.2. The legends contained in this figure have been defined in the previous sections.

mapping model, which must analyze the historical data successfully before proceeding with the mapping procedure. The evaluation data set is utilized to assess the performance objectives of the chosen ML-assisted algorithm with respect to the mapping process throughout the lifetime of the requested RAN slice [126].

At this stage of the mapping process, it is assumed that the ML-assisted model has been successfully trained (offline) and is ready to be deployed (online) to the unified architectural framework of the NFV-MANO, 3GPP-NSMS, I-PoPs, and ENI System (step 4). First, the offline trained mapping model is transferred to the Cognition Management FB, which uses it to generate predictions (or recommended actions) about the requirements and design of the requested RAN slice, as well as the status of virtual resources in underlying I-PoPs (step 5). The offline trained mapping model is then transferred to the Knowledge Management FB, where it is kept for periodic (or continuous) learning (step 6). This periodic learning is accomplished through the utilization of real-time data acquired from the assisted systems and is intended for the autonomous incremental development of the respective ML-assisted mapping model [47, 126].

It is worth stating once more that the proposed architectural framework acquires real-time data for a variety of performance metrics associated with the requested RAN slice from 3GPP-NSMS (specifically, the NG-RAN NSSMF), NFV-MANO (specifically, the VIM), and the underlying infrastructure (specifically, the I-PoP #1 and I-PoP #2). The NG-RAN NSSMF provides the ENI System with information about the physical components and their connections to the virtual components of the RAN slice (step 7). The VIM provides

information about the design of VNFFG, the required quantity of virtual (compute, storage, and networking) resources, and the performance requirements of the RAN slice (step 7). The I-PoPs provide information on the status of the virtual resources that are provided to the VNFCs and VLs of the vCU and vDU of the requested RAN slice, such as the number of (available, allocated, and consolidated) PMs (or VMs) and PLs (or VLs), network throughput, and so on (step 7).

The three types of assisted systems described above may provide data to the ENI System's Data Ingestion FB in a variety of formats, with different ranges of features and varying levels of structures. As a result, the Data Ingestion FB first programmatically filters and normalizes this input (raw) data into a uniform (or standardized) data format before passing it on to the other FBs of the ENI System for further processing (step 7). Choosing the appropriate data normalization technique (such as min-max, z-score, mean, and so on) is critical for improving the model's accuracy in mapping the virtual components of the requested RAN slice. The normalized data (a type of data in which the values of numeric columns in a dataset are accurately converted to a common scale) is subsequently transferred to the Knowledge Management FB (step 8).

The ML-assisted mapping model, which has already been developed through offline training on a pre-prepared dataset and is stored in the Knowledge Management FB (see step 6), is updated on a regular (or as needed) basis with newly arrived input data from the Normalization FB. These input data may result in certain changes (modifications) to the existing mapping model, such as feature extraction, editing the current knowledge, adding new knowledge, and so on. Once these modifications have been made, an updated version of the mapping model is created and stored in the Knowledge Repository of the Knowledge Management FB. This process of optimizing (or re-training) the existing mapping model is performed continuously throughout the lifetime of the requested RAN slice, which is tuned by the real-time stream of data coming from the NG-RAN NSSMF, VIM, and I-PoPs. The overarching goal of this optimization is to increase the accuracy of the selected mapping model. The proposed framework then transfers the optimized mapping model (also known as inferred data) to the Context-aware Management FB and Cognition Management FB concurrently (step 9).

The Context-aware Management FB is the first to read the newly arrived inferred data from the Knowledge Management FB (step 9). Furthermore, the Context-aware Management FB must ensure that the inferred data is stored in standardized formats that must be readable by the rest of the FBs of the ENI System. The inferred data, which pertains to the state and environment of the performance objectives (see Tab. 3.1) of the mapping process, may change throughout the lifetime of the requested RAN slice. As a result, the Context-aware Management FB is used to continuously monitor (or measure) the performance objectives and adapt the behavior of a RAN slice in accordance with the SLA [127]. If we assume that an unusual event is detected (for example, a VM allocated to a VNFC is overloaded or a VN assigned to an internal-VL is underutilized), this FB, in conjunction with other FBs of the ENI System, may take corrective actions in accordance with a set of pre-established rules in order to resolve such a hazardous situation. Fig. 3.3 illustrates that the Context-aware Management FB passes context information related to the requested RAN slice to the Cognition Management FB (step 10).

The Cognition Management FB enables the ENI System to analyze normalized ingested data, context, and information associated with the mapping model of the requested RAN slice. Once such an understanding of the mapping model is accomplished, this FB assesses the acquired data and defines the actions that must be taken in order to meet the performance objectives of the mapping process (step 9). The Cognition Management FB is also in charge of making predictions about the QoS and QoE metrics associated with mapping the virtual

components of the RAN slice. These predictions are achieved through the use of offline data (gathered during offline training), inferences (provided by online training), and context information. The probable values generated by the mapping algorithm for each of the predicted parameters enable the network operator to forecast the likelihood of the mapping process of the requested RAN slice. Once the desired predictions have been generated, the Cognition Management FB delivers them to the Situational-awareness FB and Model-driven Engineering FB at the same time (step 11).

Through the use of Situational-awareness FB, the proposed architectural framework is enabled to fully comprehend and analyze the significance, meaning, and behavior of all events that have occurred or will occur during the mapping process of the requested RAN slice in the NG-RAN NSSMF, VIM, and I-PoPs. This FB also enables the proposed framework to understand how the requested RAN slice's behavior influences the performance objectives that the ENI System is attempting to optimize in the short, medium, and long term. Furthermore, in collaboration with the Model-driven Engineering FB, the Situational-awareness FB determines what the ENI System should do in response to the given event(s), makes (or chooses) the most appropriate decisions, and performs the relevant action (or combination of actions). These decisions or recommended set of optimal actions are then forwarded to the Model-driven Engineering FB (step 12).

The main objective of the Model-driven Engineering FB in proposed framework is to make appropriate decisions aimed at implementing recent recommendations or sets of actions produced by the Situational-awareness FB. To achieve that goal, this FB utilizes model-driven engineering mechanisms in order to convert these actions into a form of policy, such as employing machine-readable models rather than code. In addition, the Model-driven Engineering FB employs a collection of algorithms to abstract information and concepts in order to manage the behavior of the requested RAN slice intelligently in the NG-RAN NSSMF, VIM, and I-PoPs. This FB also boosts productivity across the FBs of the ENI System by reusing the standardized models. For example, it assists the Cognition Management FB in making predictions and the Knowledge Management FB in measuring the performance objectives of the requested RAN slice (step 13). Fig. 3.3 shows that the converted form of policies is then passed to the Policy Management FB (step 14).

The Policy Management FB provides scalable and consistent decisions that must be made during the resource management and resource allocation phases in order to guarantee the key performance indicators (KPIs) of the performance metrics of the requested RAN slice, such as PM consolidation, VM allocation, VM migration, virtual resource reservation for emergency cases, and so on. The Policy Management FB also decides on the next action(s) based on the performance of the previous action(s) taken by the NG-RAN NSSMF, I-PoPs, and VIM. If the performance of the action is not satisfactory, this FB adjusts its configuration, redefines (or edits) the relevant policies, and then proceeds with future action(s) under specified conditions (step 15). Fig. 3.3 illustrates that the Policy Management FB then forwards the policy and final decisions of the mapping process to the Denormalization and Output Generation FB (step 16).

Once the predictions, commands, recommendations, and/or suggestions related to the resource management and resource allocation phases are generated by the internal FBs of the ENI System, the final step (during the resource automation phase) is to provide them to the NG-RAN NSSMF, VIM, I-PoP #1, and I-PoP #2 aimed at intelligently optimizing the performance objectives of the mapping process of the requested RAN slice. To accomplish this, such information is first converted by the Denormalization and Output Generation FB into standardized format(s) that must be understandable by the aforementioned assisted systems. Following that, as shown in Fig. 3.3, the ENI System's recommendations, commands,

and predictions are provided to the assisted systems in order to intelligently execute and implement the management, orchestration, and allocation of virtual resources of the virtual components of the RAN slice (step 17). On the basis of these commands, the NG-RAN NSSMF, VIM, and I-PoPs are anticipated to take the necessary course of actions aiming to automate the tasks related to the mapping process of the vCU, vDU, internal VLs, and external VLs of the requested RAN slice onto the underlying physical infrastructure in the NG-RAN architecture.

3.2.2 The Virtual Resource Management Phase

Fig. 3.4 depicts the proposed functioning architecture for the phase of virtual resource management and orchestration. It is derived from the general architectural framework of the mapping process, which is shown in Fig. 3.2. As illustrated in Fig. 3.4, the NG-RAN NSSMF and NFV-MANO FBs are in charge of the virtual resource management and orchestration phase of the proposed mapping solution. The NG-RAN NSSMF, which serves as an input-facing entity to the NFV-MANO FBs, makes use of the RAN NSST to define the required resources for a RAN slice. The NFV-MANO FBs, which are the main building blocks of this phase, rely on a variety of NFV description files in order to instantiate, allocate, and manage virtual resources throughout the lifetime of the requested RAN slice [128].
At the start of this phase, the NG-RAN NSSMF checks the SST and SD of the requested RAN slice (step 1 in the virtual resource management phase of Fig. 3.4). We assume that the tenant requests only a single RAN slice. Therefore, the SD is by default set to one. Once the SST has been determined (step 2), the NG-RAN NSSMF selects the RAN NSST of the requested RAN slice from the repository (step 3). We make no assumptions regarding any particular SST. Hence, the proposed solution is applicable to the mapping process of any of the 3GPP-defined SSTs. The RAN NSST contains all optional and mandatory information related to the configuration parameters, functionalities, and features of the requested RAN slice [129].
The NG-RAN NSSMF then compares the requirements specified in the requested SST to those defined in the RAN NSST to ensure that an appropriate RAN NSST has been selected from the repository (step 4). If we assume that some requirements are not properly defined or the tenant requests the inclusion of additional optional requirements (such as full or partial access to billing and/or charging information, priority, service exposure capabilities, and many others), the RAN NSSMF must re-define the selected RAN NSST (step 5) and upload it to the repository for future use (step 6). However, if all requirements of the requested SST have been defined in the selected RAN NSST in a satisfactory manner (step 7), the RAN NSSMF proceeds to determining the gNB that must cover the desired geographical area of the RAN slice (step 8). The NG-RAN NSSMF selects this gNB from a list of active gNBs in the NG-RAN architecture. It also configures and adjusts the behavior of the vCU, vDU, and RU in accordance with the requirements of the RAN slice specified in the RAN NSST (step 9) [129]. Finally, the RAN NSSMF provides information related to the required virtual resources (among other parameters) to NFVO in order to enable the deployment of the vCU and vDU, as well as to manage and orchestrate their virtual resources throughout the lifetime, of the requested RAN slice (step 10).
Following that, the NFVO prepares and on-boards the required descriptors of vCU and vDU based on the information received from the RAN NSSMF (step 11). They primarily consist of VNFDs for the vCU and vDU, internal and external VLDs, PNFD for the RU, NSD, and VNFFGD. The NFVO shall compare the elements of these descriptors to the requirements associated with the virtual components of the requested RAN slice as provided by the RAN

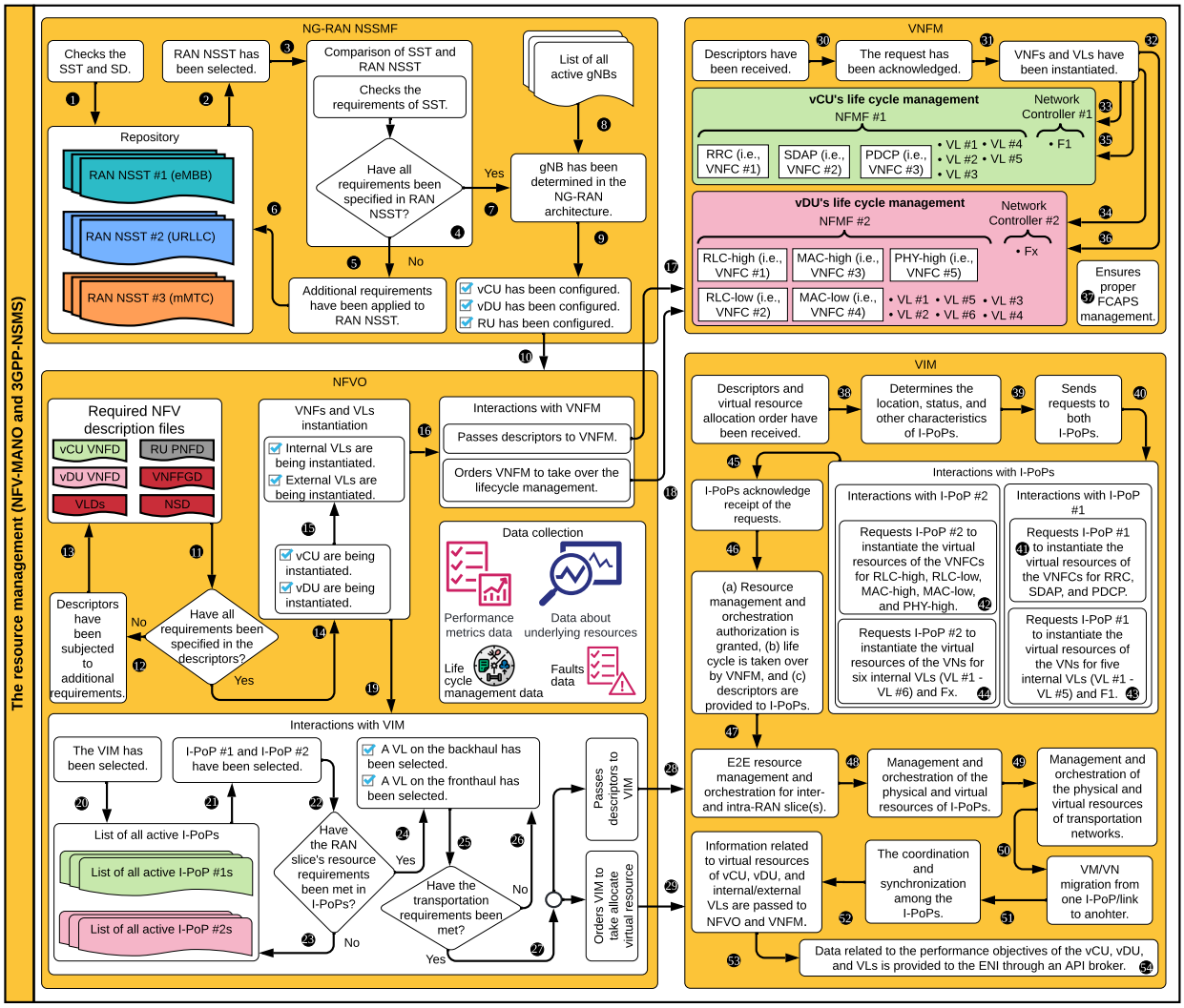

Figure 3.4: The proposed functioning architecture of the virtual resource management and orchestration phase. This framework is derived from the mapping process's overall architectural solution (see Fig. 3.2). The NG-RAN NSSMF and NFV-MANO FBs execute this phase of the mapping framework. The legends contained in this figure have been defined in the previous sections.

NSSMF. In the event that the descriptors do not contain all of the requirements (step 12), the NFVO re-defines the elements or adds new ones and subsequently uploads modified NFV descriptors to their respective repositories (step 13). Nevertheless, if the descriptors contain all the requirements and their elements are appropriately defined, the NFVO proceeds with the instantiation of the virtual components of the requested RAN slice (step 14). During the instantiation process, we let the NFVO begins the instantiation of the vCU and vDU (step 15), followed by the instantiation of their internal and external VLs (step 16). To that end, the NFVO transfers the description files to the VNFM (step 17) and instructs the VNFM to take over the tasks associated with the instantiation and life cycle management for the vCU, vDU, and internal and external VLs of the requested RAN slice (step 18).
The NFVO then determines the VIM that must provide access to both I-PoPs and transport networks in order to manage and orchestrate the underlying virtual compute, storage, and networking resources of inter-RAN and intra-RAN slices (step 19). Once the VIM has been determined (step 20), the NFVO selects the I-PoPs that will accommodate the vCU, vDU, and their associated internal and external VLs of the requested RAN slice. The NFVO selects

two appropriate I-PoPs from a list of active I-PoPs, taking into account a variety of technical, regulatory, and geographic constraints (step 21). Next, the NFVO compares the vCU and vDU resource requirements to the available, reserved, and allocated virtual resources in both I-PoPs (step 22). If the I-PoPs do not meet the resource requirements of the requested RAN slice, the NFVO searches for other appropriate I-PoPs that meet those requirements (step 23). In the event that the selected I-PoPs are able to meet the resource requirements, the NFVO proceeds with the orchestration of connectivity between the I-PoPs as well as their connectivity to the cellular network sites across the transport networks (step 24). Hence, the NFVO selects appropriate VLs on the backhaul and fronthaul in order to connect vCU to vDU and vDU to RU, respectively (step 25). If the transport network requirements of the requested RAN slice have not been met by the selected VLs (step 26), the NFVO searches for and selects other appropriate VLs across the backhaul and fronthaul networks. However, if these requirements are met (step 27), the NFVO prepares to interact with the VIM for the allocation of the required virtual compute, storage, and networking resources. Finally, the NFVO transfers the description files to the VIM (step 28) and orders the VIM to initiate the required virtual resource allocation for the vCU, vDU, and VLs (step 29).

It is also worth noting that the NFVO collects data on performance metrics, resources, faults, and life cycle management from the VIM and VNFM on a periodic basis in order to dynamically scale (both horizontally and vertically) vCU, vDU, and internal and external VLs throughout the lifetime of the requested RAN slice.

Next, we assume that the VNFM received the required description files and an order from the NFVO (step 30) to configure, instantiate, and take over life cycle management of the virtual components of the requested RAN slice. Although the instantiation of VNFs and internal VLs is initially within the scope of VNFM, this procedure begins at the NFVO with the selection of appropriate description files and the execution of several prerequisite tasks, such as assigning VNFs and internal VLs to an associated VNFM, specifying their lifetime, defining specific features and functionalities for their life cycle management, and many others (see steps 15 and 16). It is worth noting that the instantiation and life cycle management of the external VLs are beyond the scope of the VNFM. The WIM and Network Controller are responsible for these two tasks related to the external VLs of a RAN slice over the midlehaul and fronthaul transport links [130].

After the VNFM and WIM acknowledge the request (step 31), the NFVO grants the VNFM and WIM access to data repositories in order to proceed with the instantiation of the VNFs and VLs, as well as manage their life cycles. The VNFM and WIM make use of the description files to complete the remaining tasks associated with the instantiation of VNFs and VLs. First, we let VNFM complete the instantiation process of the VNFCs of vCU (RRC, SDAP, and PDCP), the VNFCs of vDU (RLC-high, RLC-low, MAC-high, MAC-low, and PHY-high), as well as the internal VLs of vCU and vDU. Then, we let WIM complete the instantiation process of the external VLs (i.e., F1 and Fx). We assume that the VNFM and WIM successfully instantiated the vCU and vDU, as well as their internal and external VLs, of the requested RAN slice in I-PoPs and transport links (step 32).

Regarding life cycle management, the VNFM assigns a dedicated NFMF #1 to the vCU and its internal VLs (step 33), as well as a dedicated NFMF #2 to the vDU and its internal VLs (step 34). Likewise, the WIM assigns Network Controller #1 to F1 (step 35) and Network Controller #2 to Fx (step 36). The VNFM assigns a unique identification number (ID) to the life cycle management operations of the requested RAN slice. The NFV-MANO FBs, as well as the I-PoPs and transport links, use this ID to identify all operations and resources pertaining to the requested RAN slice. The VNFM and WIM must also guarantee that the FCAPS of vCU, vDU, and VLs are properly managed and the NFVO is notified on a regular

basis (step 37). Due to the scope of the paper, we skip providing further details on workflow messages exchanged between the NFVO, VNFM, and WIM.

Following the instantiation of the VNFs and VLs, we assume that the VIM selected by the NFVO is fully aware of the SST, description files, life cycle management, and virtual resource requirements of the requested RAN slice (step 38). The VIM also knows the geographical locations, identities, resource status, and connectivity information of both I-PoPs that host the vCU and vDU (step 39). Based on this knowledge, the VIM sends virtual resource allocation requests (in coordination with the NFVO and VNFM) to both I-PoPs (step 40). These requests also include resource management and allocation constraints applicable to the required resources of the vCU and vDU. The VIM utilizes these constraints to determine available resource zones within an I-PoP, as well as to perform optimal partition, reservation, and allocation of the underlying resources. To ensure synchronization between the vCU and vDU of the requested RAN slice, the VIM must send these requests to both I-PoPs at the same time. First, the VIM requests I-PoP #1 and I-PoP #2 to configure, instantiate, and allocate the virtual compute and storage resources for the three VNFCs of vCU (step 41), as well as for the five VNFCs of vDU (step 42). The VIM then requests I-PoP #1 to configure, instantiate, and allocate the virtual networking resources for five internal and one external VLs of vCU (step 43). It also requests I-PoP #2 to configure, instantiate, and allocate the virtual networking resources for six internal and one external VLs of vDU (step 44).

Once the I-PoPs receive the virtual resource allocation requests (step 45) and acknowledge receipt of these requests (step 46), the VIM (a) requests permission from the NFVO to manage and orchestrate the required virtual resources; (b) notifies the VNFM to take over the life cycle management of these resources; and (c) transfers all necessary information related to IDs, lifetime, connectivity, and description files to both I-PoPs in order to proceed with the configuration and allocation of the required resources (step 47).

Despite the fact that the I-PoPs are responsible for configuring and allocating virtual resources, a number of higher-layer tasks are still in the scope of VIM. These tasks, which are performed sequentially after resource instantiation, include the following: (a) E2E management and orchestration of the virtual resources of the requested RAN slice (step 48); (b) management and orchestration of the underlying virtual and physical resources of the I-PoPs (step 49), as well as the transport networks (step 50); (c) migration (if necessary) of the VMs and VNs from I-PoP #1 to I-PoP #2 and vice versa (step 51); and (d) coordination and synchronization among the I-PoPs (step 52) [94]. Furthermore, the VIM collects and periodically passes information regarding the configuration, allocation, and reservation of virtual resources of the vCU, vDU, and internal and external VLs to the NFVO and VNFM (step 53). This information is used by both of these FBs to dynamically optimize the operations associated with the life cycle management and virtual resources of the RAN slice.

The VIM additionally gathers and passes data related to the performance metrics (see Tab. 3.1) of the vCU, vDU, and internal and external VLs to the ENI System through an API broker (step 54). Once the input data from the VIM has been processed and the recommendations (or commands) have been generated by the internal FBs of the ENI System (see the preceding subsection), the ENI System passes such recommendations back to the VIM with the goal of automating and intelligently performing the tasks within the scope of the VIM. The VIM uses these recommendations to automate and intelligently manage, among others, the following four major tasks: coordination among the I-PoPs; management and orchestration of the overall operations of the I-PoPs; management and orchestration of virtual and physical resources of the requested RAN slice; and the migration of the VMs and VL from one I-PoP to another.

3.2.3 The Virtual Resource Allocation Phase

The proposed architectural framework for the virtual resource allocation phase is shown in Fig. 3.5. Similar to the previous two phases, the functioning architecture of this phase is also derived from the general architectural framework of the mapping process, as depicted in Fig. 3.2. The I-PoPs (#1 and #2) and transport networks are in charge of executing the phase of virtual resource allocation. They configure, allocate, and terminate virtual compute and storage resources for the VNFCs of the vCU and vDU, as well as the virtual networking resources of the VLs, of the requested RAN slice.

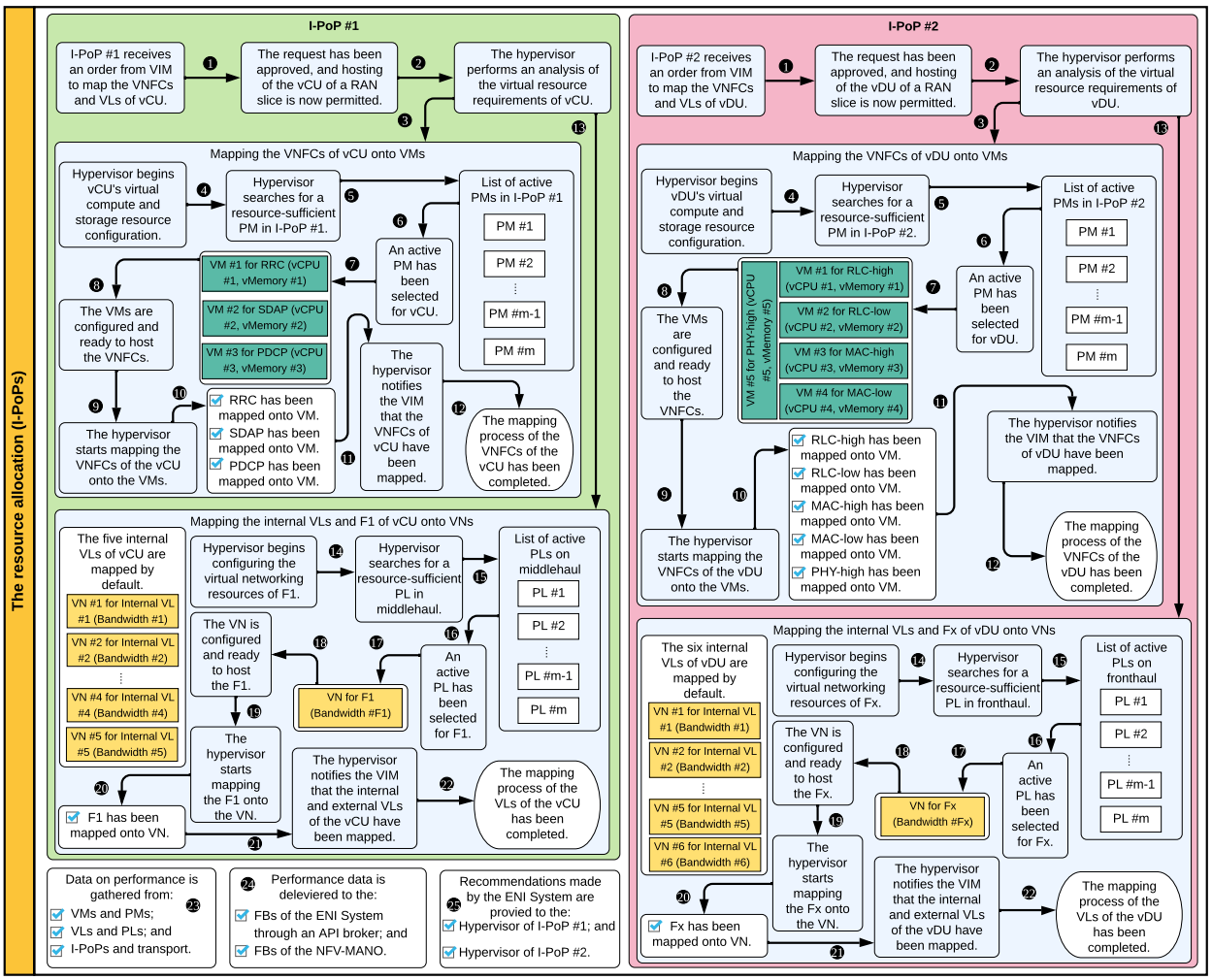

Figure 3.5: The virtual resource allocation phase's proposed functioning architecture. It is derived from the mapping process's general architectural framework, which is illustrated in Fig. 3.2. The I-PoP #1 and I-PoP #2 are in charge of the virtual resource allocation phase. The legends contained in this figure have been defined in the previous sections.

To begin, the I-PoP #1 and I-PoP #2 receive requests from the VIM to prepare, configure, and allocate virtual resources for vCU, vDU, and VLs (step 1 in the virtual resource allocation phase of Fig. 3.5). After processing the requests and granting permission to host the RAN slice (step 2), each I-PoP notifies its hypervisor regarding the mapping of the VNF and its associated VLs. Based on the predictions generated by the ENI System and according to the description files received from the NFVO, the hypervisors analyze the virtual compute and storage resource demands (workloads) of the VNFCs and the virtual networking resource demands (bandwidth) of the VLs in both I-PoPs and transport networks (step 3). We

assume that all PMs and PLs are placed sequentially in I-PoPs and transport networks. The hypervisors then begin the process of configuring virtual compute and storage resources for the VNFCs (step 4).

To that end, the hypervisor (in I-PoP #1 and I-PoP #2) searches for an active resource-sufficient PM whose available physical compute and storage resources must equal or exceed the sum of the virtual compute and storage resource demands of the VNFCs of a VNF (step 5). The hypervisors perform searching for an active PM in a list of sequentially-ordered PMs in I-PoP #1 and I-PoP #2 (step 6). We assume that each hypervisor has selected a PM (step 7). In I-PoP #1, the hypervisor configures the virtual compute and storage resources for three VMs (each for a VNFC of vCU) on the selected PM (step 8). In parallel, the hypervisor configures the virtual compute and storage resources of five VMs (each for a VNFC of vDU) on the selected PM in I-PoP #2 (step 8). It is critical to ensure that the virtual compute and storage resource requirements of a VNFC do not exceed the virtual compute and storage resource capacity of the host VM when configuring the selected PM. We assume that the VMs have been configured and are prepared to host the VNFCs of the vCU and vDU (step 9). The hypervisors thus programmatically begin mapping the VNFCs onto the underlying configured VMs in I-PoP # 1 and I-PoP #2 at the same time (step 10). To accomplish this, it starts with the mapping of the first VNFC and concludes with the mapping of the final VNFC of a VNF (step 11). Such a sequential chaining and processing of the VNFCs of the requested RAN slice must be specified in its corresponding VNFFGD. The hypervisor notifies the VIM of mapping the VNFCs of a VNF (step 12). Finally, the VNFCs have been successfully mapped onto their corresponding VMs in an I-PoP.

Following that, the hypervisors begin mapping the internal and external VLs (step 13). Due to the assumption that the VNFCs of a VNF are hosted on the same PM in an I-PoP, no mapping process is required for the internal VLs of the requested RAN slice. To map F1 and Fx, the hypervisors begin configuring virtual networking resources (step 14). First, a list of active PLs must be searched (step 15) for those with sufficient physical networking resources capable of mapping F1 and Fx onto the middlehaul and fronthaul, respectively (step 16). After selecting a resource-sufficient PL (step 17), the hypervisors configure a VN for F1 and Fx on both the middlehaul and fronthaul, respectively (step 18). When configuring the selected PL, it is critical to ensure that the virtual networking resource requirements of a VL do not exceed the virtual networking resource capacity of the host VN in both middlehaul and fronthaul transport links. The VN is configured and ready to host the VL (step 19). The hypervisors start mapping the F1 and Fx onto their respective VNs sequentially (step 20), as specified in the VNFFGD of the requested RAN slice. The F1 and Fx have been mapped onto VNs (step 21). The VIM is then notified of the F1 and Fx mapping (step 22). Finally, the internal VLs and external VLs of vCU and vDU have been successfully mapped onto I-PoPs, as well as middlehaul and fronthaul transport links, respectively.

The virtual compute and storage resource requirements of the VNFCs and the virtual networking resource requirements of the VLs may change throughout the lifetime of the requested RAN slice. Hence, the resource capacity of the VMs and VNs shall also be reconfigured by the proposed solution in an automated fashion in response to changes in resource demands, in order to meet the virtual resource requirements of the respective virtual components of the RAN slice. In addition, the hypervisors may configure redundant VMs and VNs to avoid service interruption, improve survivability, and increase availability [125]. Providing redundancy, however, may necessitate allocating and reserving extra virtual resources, thereby increasing total costs and complicating resource management. This may require an agreement between the service provider and tenant in the SLA regarding the additional cost and various levels of redundancy [131]. If both parties agree on providing

redundancy, the hypervisors must configure redundant VMs and VNs on different PMs and PLs than those that host the primary VLs and VNs. The configuration of redundant VMs for the VNFCs of vCU and vDU is performed from steps 5 to 8 in I-PoP #1 and I-PoP #2, respectively. The configuration of redundant VNs for F1 and Fx is performed from steps 15 to 19 on middlehaul and fronthaul, respectively.
Once the mapping process of a RAN slice and the configuration of redundant VMs and VNs have been completed, the proposed framework updates the knowledge repositories of the hypervisors in I-PoP #1 and I-PoP #2. Following that, the hypervisors collect data related to the performance objectives (see Tab. 3.1) from VMs and PMs, VLs and PLs, and other components of I-PoPs and transport networks on a regular basis (step 23). The performance data must contain all necessary information about the status of the VMs, VNs, PMs, and PLs. It must also include the number of hosted vCUs, vDUs, VLs, as well as the available, reserved, and allocated resources within I-PoPs and transport networks. Such a collection of performance data assists the I-PoPs and transport networks in managing and orchestrating their resources, as well as deciding whether to accept or reject future RAN slice requests. The hypervisors then pass performance data to the management entities of the I-PoPs.
The I-PoPs construct performance data into two distinct sets. The first data set is transferred to the VIM (step 24). The VIM utilizes this data to manage the operations and resources of the I-PoPs and transport networks. The VIM also shares this information with the NFVO and VNFM in order to effectively manage and orchestrate the life cycle and resources of the requested RAN slice throughout its lifetime. The second data set is routed through an API broker to the ENI System (step 24). It contains information about the metrics associated with the performance objectives of the requested RAN slice. This data set is then processed by the internal entities and FBs of the ENI System in order to generate recommendations regarding the operations and resources of the I-PoPs (step 25).
To accomplish this, the ENI System employs ML-assisted algorithms to (a) predict the virtual resource requirements of the VMs and VNs of the requested RAN slice based on the historical data or learning from an environment in steps 5 and 15; (b) configure the virtual compute and storage resources of VMs, as well as the virtual networking resources of VNs, in steps 8 and 18; and (c) dynamically allocate them to the VNFCs and VLs in steps 11 and 21. In comparison to the state-of-the-art VNF mapping algorithms, the use of ML-assisted algorithms in the preceding steps of the virtual resource allocation phase is critical for obtaining the performance objectives such as optimizing virtual resource allocation performance, minimizing energy consumption, and intelligently scaling the requested RAN slice [132, 88, 89].
The proposed framework also provides ENI's recommendations for physical resource virtualization. Thus, the hypervisors in I-PoPs and transport networks could efficiently abstract the virtual resources of VMs and VNs from the underlying physical resources of the PMs and PLs based on the predictions generated by the relevant ML-assisted algorithm(s) of the ENI System. At this stage of the virtual resource allocation phase (steps 7 to 9 and steps 17 to 19), accurate predictions of the virtual resource requirements of VMs and VNs of the requested RAN slice are critical for the hypervisor to partition the PMs and PLs and efficiently allocate them to the VNFs and VLs of the requested RAN slice. In comparison to the state-of-the-art resource virtualization and abstraction solutions, such intelligent and automated virtualization of physical resources in I-PoPs and transport networks undoubtedly results in achieving the performance objectives set in the previous section such as the reduction of the active number of PMs and PLs, the lowering of VM and VN migration, the maximization of revenue and profit, and the avoidance of over-and under-utilization of physical resources.

Chapter 4

Contributions to the SMO Framework in O-RAN Architecture

Summary – The emergence of the O-RAN architecture offers a paradigm shift in cellular network management and service orchestration, leveraging data-driven, intent-based, autonomous, and intelligent solutions. The SMO framework plays a role in the MANO of applications, services, and components within O-RAN. It comprises the Non-RT RIC and possesses the capability to incorporate management systems and components from multiple SDOs, notably the 3GPP and the ETSI. This multi-SDO-based SMO framework aims to provide a rich set of MANO services in a coherent and unified manner. However, the increasing complexity and scale of O-RANs demand autonomous and intelligent models for optimizing SMO operations. To that end, this chapter provides the following major contributions:

- It unifies the components of the O-RAN Alliance, 3GPP, and ETSI to enhance the capabilities of the SMO framework regarding the MANO of O-RAN slice subnets and their required resources. The objective of this unification is to eliminate ambiguity in the interaction, interworking, and integration of various standards within a unified SMO framework and to enable it to leverage the latest specifications. It facilitates standard-compliant interoperability among the components of the three modules by proposing two logical functions and their interfaces within the SMO framework. This interoperability is intended to enable the components and interfaces of the O-RAN architecture to influence the decisions made by the 3GPP and ETSI components and interfaces concerning O-RAN slice subnets.

- It proposes an intelligence-driven approach by integrating management data analytics (MDA) into the 3GPP and ETSI management systems within SMO. Furthermore, it introduces the AI/ML Function into the Non-RT RIC, consolidating existing intelligent components and introducing novel ones to enhance the intelligence capabilities of the Non-RT RIC. The chapter further proposes an architectural solution that unifies and facilitates interoperability among the intelligent systems of the 3GPP, ETSI, and Non-RT RIC - collectively referred to as three modules - within SMO. It also presents the end-to-end lifecycle workflow of the AI/ML model across the three modules. The chapter, then, outlines key research challenges related to integrating MDA.

- It proposes three scenarios for integrating ML algorithms into SMO. It focuses on exploring one of the scenarios in which the Non-RT RIC plays a major role in data collection, as well as model training, deployment, and refinement. Finally, we identify potential challenges associated with implementing centralized ML within SMO.

4.1 Unifying 3GPP, ETSI, and O-RAN SMO Interfaces

The MANO of O-RAN NSSs, hereinafter referred to as O-RAN slices, in O-RAN architecture are challenging due to their heterogeneous operator- and tenant-specific requirements, the integration complexity of components and systems of various SDOs, and the growing multiplicity of parameters that must be defined and configured to acquire optimality. To address these challenges, the O-RAN Alliance specified the SMO framework [133], which is in charge of intelligently managing and optimizing a large number of O-RAN slices, autonomously orchestrating their required virtual and physical resources, and supporting the FCAPS of the O-RAN slice components via a number of open and standardized interfaces. Figure 4.1 illustrates that the Non-RT RIC is a crucial component of the SMO framework [134] that controls the content carried over the A1 interface and operates on a control loop with a time scale of >= 1 second. It consists primarily of the Non-RT RIC framework and its Applications (rApps). The former provides functionality that terminates the A1 interface, enables the rApps to expose the SMO services via the R1 interface, and manages AI and/or ML workflows to run the rApps [134]. The latter provides value-added services related to the operations of the O-RAN architecture by leveraging the functionality exposed by the Non-RT RIC (or SMO framework) and subsequently applying such services over the O1, O2, and O-FH M-Plane interfaces, thereby enabling the Non-RT intelligent control and optimization of O-RAN slices and their resources, as well as providing policy-based guidance to the Non-RT RIC [134].

To embed the novel capabilities and features of the Non-RT RIC into the elements of an O-RAN slice, the O-RAN Alliance specified the Near-RT RIC, which is deployed at the edge of a cellular network and connected with each element of an O-RAN slice via a southbound interface (i.e., E2) [133, 134]. It allows Near-RT autonomous control and intelligent optimization (in a control loop with a periodicity >= 10 miliseconds and < 1 second) of the O-RAN slice elements by employing fine-grained data collection methods and actions over the E2 interface, as well as managing their AI/ML workflows. Figure 4.1 illustrates that it terminates three open interfaces, incorporates Near-RT RIC Applications (xApps), and consists of components (such as conflict mitigation, subscription management, security, etc.) that support the execution of xApps [133].

Both RICs deploy the non- and near-RT policies and control actions atop 3GPP-defined open next generation node B (O-gNB), which according to Split Option 7.2x can be split into open centralized unit (O-CU), open distributed unit (O-DU), and open radio unit (O-RU) [133, 135]. The O-CU implements the higher layer radio functionalities and can be further split into two components: the control plane (CP) and the user plane (UP). The O-DU deploys the lower layer radio functionalities. The O-RU accommodates the physical layer functionalities and is hosted by a cellular site. The O-CU and O-DU, which terminate E2 interfaces, support a real-time control loop that operates in a timeframe of < 10 miliseconds [133]. The E2 nodes can be deployed on the open-cloud (O-Cloud), which is a cloud platform consisting of virtual/physical resources that host the components of an O-gNB, the Near-RT RIC, the supporting software components, and other MANO functions. The O-Cloud sites are distributed geographically across the cloud infrastructure and are connected among each other and with cellular sites via highly reliable and high-capacity open-backhaul (O-BH), open midhaul (O-MH), and open-fronthaul (O-FH) transport links, respectively [135].

Notwithstanding the O-RAN components and interfaces, the SMO framework must support MANO components of ETSI ISG on NFV and 3GPP Technical Specification Group (TSG)

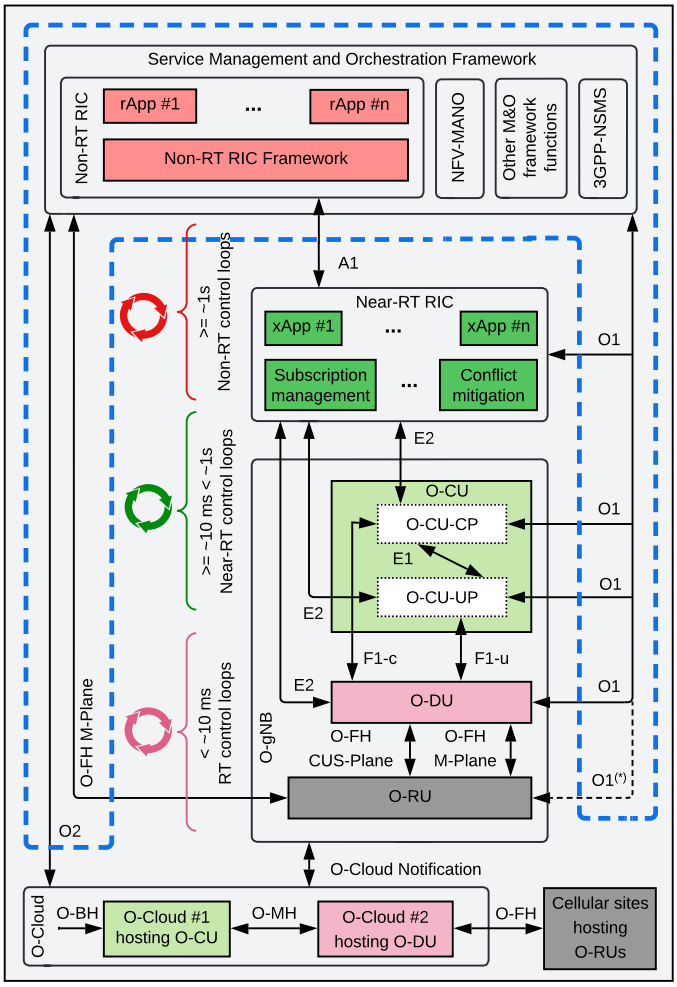

Figure 4.1: O-RAN slicing-aware architecture. This chapter addresses topics within the boundaries of the dashed blue box. (*) The O-RU termination of the O1 interface towards the SMO framework is a subject for future study.

Service and System Aspects Working Group 5 (SA5) to facilitate E2E MANO for O-RAN slices. We refer to the specifications of the two SDOs as NFV-MANO and 3GPP-NSMS, respectively. **To realize a unified SMO framework, interoperability among the Non-RT RIC, NFV-MANO, and 3GPP-NSMS is a prerequisite.** Although the O-RAN Alliance presented multiple deployment options for the components of the NFV-MANO and 3GPP-NSMS within the SMO framework [133], **the corresponding technical report does not provide sufficient detail on how these components can be tightly integrated, smoothly unified, and made to interwork seamlessly for the effective implementation of O-RAN slicing.** Furthermore, at the time of writing this section, **the O-RAN Alliance had not yet defined the functions and interfaces required to achieve interoperability among these modules.**
To address the research challenges described above, we propose the following contributions:

- First, we clarify ambiguity regarding the interworking and interaction between the Non-RT RIC, NFV-MANO, and 3GPP-NSMS by designing a unified architecture for the SMO framework, thereby enabling the Near-RT RIC, the O-gNB NFs, and the O-RAN interfaces to support slicing-aware and slice-dedicated xApps and rApps features and capabilities.

- Second, we propose two functions for the Non-RT RIC and interfaces between the

Non-RT RIC, 3GPP-NSMS, and NFV-MANO within the SMO Framework. The goal is to facilitate standard-compliant interoperability among these modules, allowing the Non-RT RIC to influence the O1, O2, and O-FH M-Plane interfaces and enabling the NFV-MANO and 3GPP-NSMS to access the rApps and Non-RT RIC services.

The rest of this section is organized as follows. We begin by discussing the unification of the three modules and the design of a unified SMO framework. Then we describe the proposed functions and interfaces and their potential contributions to the O-RAN architecture. Finally, we summarize this section by drawing conclusions and outlining future research directions.

4.1.1 Unifying 3GPP-NSMS, NFV-MANO, and Non-RT RIC for O-RAN Slicing within the SMO framework

The SMO framework can consist of the 3GPP-NSMS, the NFV-MANO, and the Non-RT RIC. In this section, we discuss and unify these modules to design a unified and standard-compliant framework for the MANO of O-RAN slices.

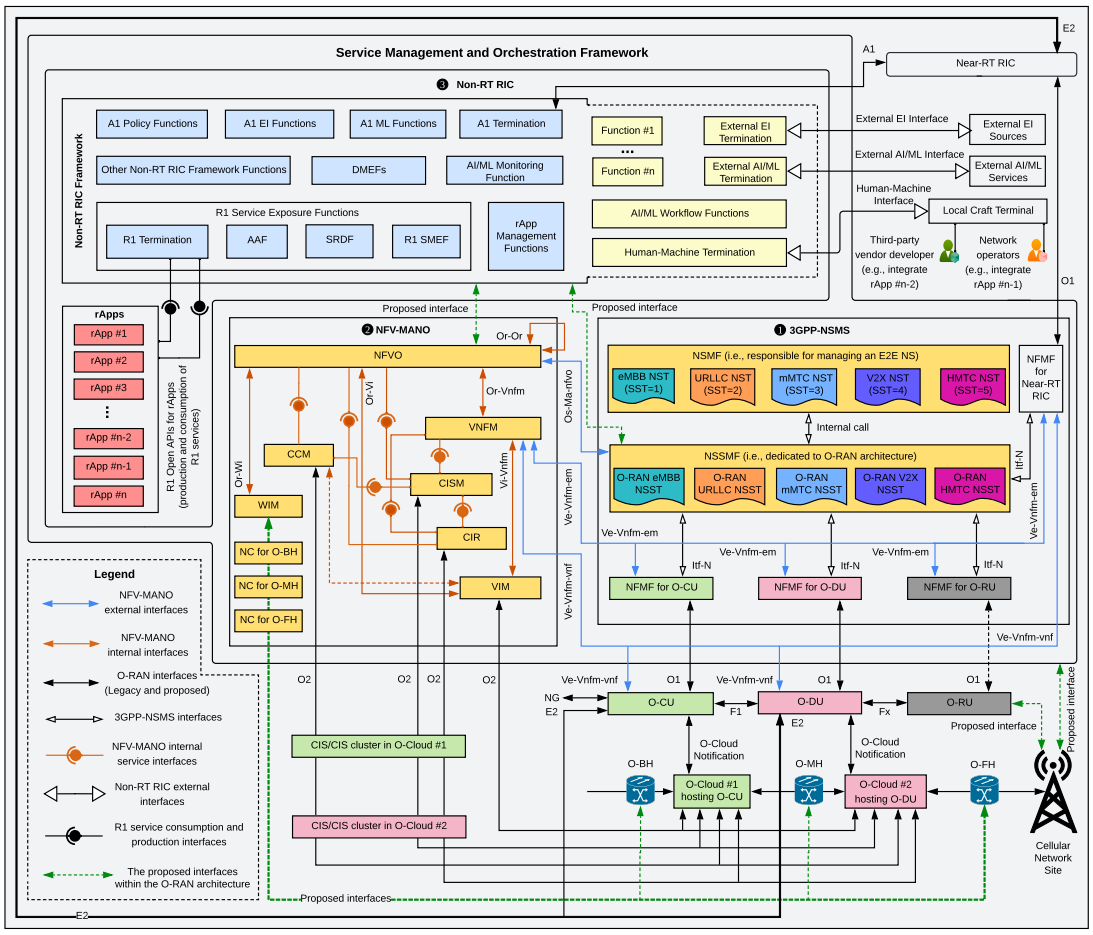

Figure 4.2: The unification of 3GPP-NSMS, NFV-MANO, and Non-RT RIC within the SMO framework in an attempt to design a unified framework for the M&O of O-RAN slices in the O-RAN slicing-aware reference architecture. Note 1: mMTC is also referred to as mIoT in 3GPP terminology. Note 2: The interaction between CCM and VIM within the NFV-MANO architecture is a subject for future study. Note 3: EI stands for enrichment information in this figure. Note 4: To ensure completeness, we have included the management components of transport links (i.e., WIM and NCs) in the figure. However, a comprehensive discussion of their roles and the M&O of transport links within the O-RAN architecture is beyond the scope of our study.

3GPP-NSMS for O-RAN Slicing

The 3GPP-NSMS within the SMO framework includes NSMF, NSSMF, and NFMF, as shown in part (1) of Figure 4.2. These entities are linked via 3GPP-defined interfaces [135]. The NSMF manages an E2E NS using a customized NST. To date, the 3GPP has specified five SSTs, each of which identifies the service type of an NS [136]. They are eMBB, URLLC, mMTC, V2X, and HMTC [135]. Each NS has an SST value of $1-5$, respectively. The NSSMF manages a subnet of an E2E NS. In the O-RAN architecture, it manages the FCAPS of an O-gNB that participates in O-RAN slicing. The NSSMF also translates the requirements of an O-RAN slice into the quantity of virtual, physical, cloud, and/or radio resources that must be allocated to the respective O-RAN slice using a customized NSST. The NFMF is responsible, from an application standpoint, for the FCAPS of a particular type of O-gNB NF via the O1 interface, which is analogous to the Itf-S interface in [135]. It expressly denotes the existence of separate NFMFs for the O-CU, O-DU, and O-RU in an O-RAN slice. The Near-RT RIC is a logical NF; hence, its FCAPS can also be managed by a NFMF.
The O-RU is the PNF. There are three scenarios for managing an O-RU (see Figures 4.1 and 4.2): (i) the O-DU manages the O-RU via the O-FH M-Plane interface; (ii) the SMO framework controls the O-RU via the O-FH M-Plane interface; and (iii) the SMO framework administers the O-RU via the O1 interface. The logical functionalities of an O-RU are hosted by cellular sites. To map these functionalities onto the cellular sites, a logical interface is required between the O-RU and cellular sites. In addition, a logical interface is needed between the SMO framework and cellular sites in order to manage radio and physical resources. Figure 4.2 shows that the O-RAN Alliance has not yet defined these interfaces. To the best of our knowledge, we are the first to propose the two interfaces, which we believe are essential to the realization of a unified O-RAN architecture. Especially, they are critical in the MANO of shared O-RUs and mapping them onto cellular sites [137]. We denote the two interfaces as CellularSite Notification and O2, respectively. Table 4.1 provides additional information on their scope and implications.
The O-CU, O-DU, and Near-RT RIC are provided as VNFs. The lifecycle management of VNFs and their connectivity with PNF(s), as well as the control of the corresponding NFMFs, from a virtualization perspective, are within the scope of the NFV-MANO. The 3GPP-NSMS exclusively manages these NFs from the viewpoint of applications. The Non-RT RIC interacts with them to optimize their operations through the use of appropriate rApps. To jointly manage the PNFs and VNFs of an O-gNB, as well as their required resources, the NSSMF initiates interaction with the NFV-MANO. Once such interoperability is established, the 3GPP-NSMS entities interwork with the NFV-MANO FBs for a unified MANO of O-RAN slices [135], as shown in parts (1) and (2) of Figure 4.2.

NFV-MANO for O-RAN Slicing

Part (2) of Figure 4.2 depicts the architecture of NFV-MANO Release 4, which includes management functions (MFs) for CNFs atop FBs proposed by the ETSI ISG NFV in previous releases. Originally, the NFV-MANO consisted of the NFVO, the VNFM, the VIM, and the WIM [138]. The NFVO instantiates O-CU and O-DU, in addition to validating and authorizing requests for virtual resources of an O-RAN slice. It also interacts with the NSSMF to jointly manage the virtual and physical components of an O-gNB. The VNFM manages the lifecycle and FCAPS of O-CU and O-DU from a virtualization perspective. The VIM manages and orchestrates the virtual and physical resources of O-Cloud sites via the O2 interface. It also lets O-Cloud sites notify the hosted O-CU, O-DU, and Near-RT RIC of critical notifications via the O-Cloud Notification interface. The WIM manages the virtual

Table 4.1: The logical interfaces that have not yet been defined within the O-RAN architecture (see Figure 4.2). We prefer to denote these interfaces as analogous to the existing interfaces standardized for the O-RAN architecture to comply with the terminology and scope of the specifications of the O-RAN Alliance.

Endpoints	Proposed Interface	Scope
SMO framework – Cellular Site	O2	This interface connects the SMO framework to cellular sites to support the MANO capabilities of the cellular infrastructure. The scope of this interface is comparable to that of other O2 interfaces that connect the SMO framework to O-Cloud sites hosting O-CU and O-DU. However, its scope is pertinent to radio resource management at cellular sites. This interface is anticipated to align with the 3GPP specifications for radio resource management in the O-RAN architecture to the extent possible.
O-RU – Cellular Site	CellularSite Notification	The CellularSite Notification interface enables the O-RU logical functions, which are deployed on cellular sites, to subscribe to events and/or status updates from the relevant cellular sites. The cellular sites will include event producers to allow radio resource workloads to receive events and/or status updates that are only known to the cellular infrastructure using the CellularSite Notification interface.
NC – O-BH	O2*	This interface connects the NC and O-BH. It is utilized by the WIM and NC to manage the O-BH transport links between the O-CU and core network and to orchestrate their respective networking resources, devices, and traffic. Employing the NC – O-BH interface, the WIM and NC define and configure the behavior, enforce the policies, and monitor the performance of an O-BH transport link.
NC – O-MH	O2*	This interface links the NC to the O-MH. The WIM and NC use the NC – O-MH interface to manage the O-MH transport links between the O-CU and O-DU, as well as to orchestrate networking resources, components, and traffic that exist in this transport link. The WIM and NC utilize this interface to define and configure the behavior, enforce the policies, and monitor the performance of an O-MH link.
NC – O-FH	O2*	This interface connects the NC and O-FH. The WIM and NC employ the NC – O-FH interface to manage the O-FH transport links between the O-DU and O-RU and to orchestrate their networking resources, devices, and network traffic over the O-FH transport link. The WIM relies on NC to define and configure the behavior, enforce the policies, and monitor the performance of an O-FH link.

Note*: The scope of the chapter precludes further analysis or discussion of the O2 interfaces proposed for transport slicing in this table.

connectivity among O-Cloud sites and between O-Cloud sites and cellular sites through the use of network controllers (NCs). Each NC manages a specific transport network. To this end, the O-RAN Alliance has not yet specified the interfaces between NCs and O-BH, O-MH, and O-FH. We are the first to propose these three interfaces for transport link management. We denote them as O2 interfaces. Further elaboration on their scope and implications is provided in Table 4.1.

The ETSI ISG NFV has recently added several MFs to the NFV-MANO architecture to support the MANO of CNFs (e.g., if O-CU and O-DU are provided as CNFs). Specifically, they are container infrastructure service management (CISM), container image registry (CIR), and container infrastructure service cluster management (CCM) [138]. The CISM manages container-based applications and infrastructure services, including the management of containers, the orchestration of container clusters, and the management of CNFs. The CIR manages the lifecycle of container images securely and effectively, such as image storage, retrieval, modification, and deletion. The CCM provides the tools and procedures to manage the lifecycle of container infrastructure service (CIS) clusters, including deployment, scaling, and monitoring. Part (2) of Figure 4.2 shows that the CNFs MFs expose a series of management services that are invoked by NFV-MANO FBs via internal interfaces and external consumers (e.g., O-Cloud sites in O-RAN architecture) via O2 interfaces. They also consume management services produced by the NFV-MANO FBs.

Non-RT RIC for O-RAN Slicing

Part (3) of Figure 4.2 shows that rApps and the Non-RT RIC framework are the major building blocks of the Non-RT RIC architecture. rApps utilize the capabilities and features of the Non-RT RIC (or SMO) framework to provide value-added services [134], such as data analytics and enrichment information, associated with the operation and optimization of O-RAN slices. The Non-RT RIC framework includes several functions that terminate the A1 interface to the Near-RT RIC, expose or consume a set of R1 services to or from the rApps, and manage various types of rApps. These functions are interconnected with one another and with rApps via Non-RT RIC internal interfaces. The Non-RT RIC framework can also be connected to other MANO frameworks (e.g., NFV-MANO FBs, 3GPP-NSMS entities, etc.) via SMO internal interfaces. Additionally, the Non-RT RIC architecture comprises external interfaces that link the Non-RT RIC framework and rApps to external sources and services situated outside of SMO. Furthermore, the Non-RT RIC architecture must have the ability to integrate future logical functions that may offer extra novel services via the R1 interface or additional enrichment information via the A1 interface.

rApps: rApps are modular software applications that use Non-RT RIC (or SMO) framework capabilities to deliver value-added services for intelligently optimizing and operating O-RAN slices. These value-added services include, but are not limited to: (a) policy-based recommendations and enrichment information via the A1 interface; (b) data analytics, AI/ML model training, and inference for O-RAN slice optimization or other rApps; and (c) recommendations associated with configuration management actions via the O1 interface. An example of an rApp could be a module that provides real-time analytics and predictive maintenance for network components within an O-RAN slice. This module could utilize ML to analyze performance metrics from network components and generate insights to assist operators in optimizing the O-RAN slice based on data-driven decisions. A second example would be a module that provides intelligent load-balancing capabilities for traffic within O-RAN by dynamically routing traffic based on network conditions and congestion levels.

rApps can be developed by third-party developers, network operators, and vendors and

integrated into the Non-RT RIC via the open and standardized R1 interface. This approach fosters greater innovation, competition, and flexibility in the development of O-RAN services and applications. After a successful insertion of an rApp into the Non-RT RIC, the contained policies and recommendations are sent to the R1 termination function via the R1 interface, which then interacts with the Non-RT RIC framework logical functions to utilize the R1 services. The R1 services provided by the Non-RT RIC include service management, data management, exposure services, A1-related services, O1-related services, O2-related services, AI/ML workflow services, etc. The R1 services can be generated by logical functions within the Non-RT RIC (or SMO) framework or by rApps. Additionally, R1 service providers can also be R1 service consumers.

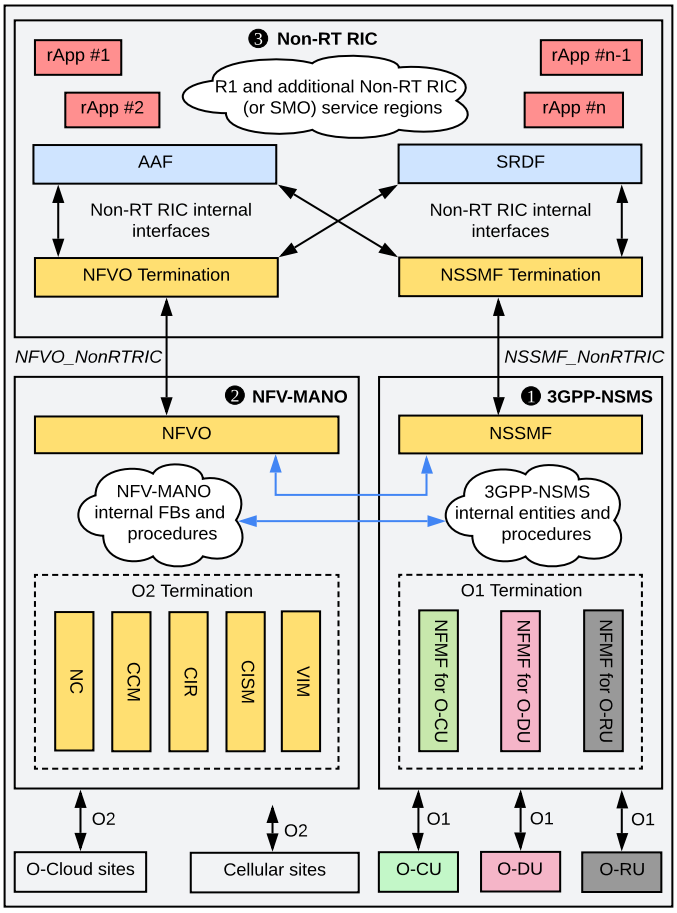

Figure 4.3: The proposed architectural solution for realizing standard-compliant interoperability among the Non-RT RIC, NFVO, and NSSMF.

Non-RT RIC framework: Part (3) of Figure 4.2 shows that the Non-RT RIC architecture consists of functions anchored inside (in Pattens Blue) and outside (in Lemon Chiffon) the Non-RT RIC framework. In certain deployment scenarios, some functions of the two categories may also be regarded as non-anchored functions that may or may not be included in the Non-RT RIC framework. Moreover, there may be functions anchored outside of SMO that can assist the Non-RT RIC framework in tasks such as the ingestion of enrichment information, the training of AI/ML models, and the manual insertion of rApps into the Non-RT RIC by technicians. Each function (of the aforementioned two categories) can interact not only with functions within its own category but also with functions of another category, as well as functions located outside of SMO, via implementation-specific internal and external interfaces. In this chapter, we focus on the structure of SMO, omitting the

Table 4.2: The definitions, scope, and implications of the two proposed functions that facilitate interoperability among the Non-RT RIC, NSSMF, and NFVO.

Proposed Termination	**Scope**
NSSMF Termination	The NSSMF Termination enables the NSSMF (and other 3GPP-NSMS entities used for the MANO of an O-RAN slice) to send and receive messages to and from the Non-RT RIC via the `NSSMF_NonRTRIC` interface, with the goal of subscribing to and utilizing certain services and capabilities of the Non-RT RIC and rApps. In addition, this logical function permits the Non-RT RIC and an rApp to send and receive messages to and from NSSMF in an effort to influence decisions made by the 3GPP-NSMS. The messages exchanged between the NSSMF and the Non-RT RIC primarily consist of the following types: (a) handshaking messages, establishing an authorized connection between the Non-RT RIC and NSSMF; (b) request/response messages, requesting information or services from NSSMF/Non-RT RIC and receiving a response in return; (c) acknowledgment messages, confirming that a particular message or service has been received successfully; (d) control messages, managing and controlling the exchange of messages between the Non-RT RIC and NSSMF; and (e) error messages, indicating and reporting errors or exceptional conditions that occur during the interaction between the Non-RT RIC and NSSMF. Overall, the messages encompass various protocols necessary for seamless communication and interoperability between the Non-RT RIC and NSSMF.
NFVO Termination	The NFVO Termination facilitates the exchange of messages between the NFVO (and other FBs and CNFs MFs used for the MANO of the virtualized parts of an O-RAN slice) and Non-RT RIC via the `NFVO_NonRTRIC` interface. It allows the NFVO to exchange messages with the Non-RT RIC to gain authorized access to its capabilities and features, as well as to the rApps. This logical function also enables the Non-RT RIC and an rApp to send and receive messages to and from the NFVO aimed at influencing the internal procedures and decisions made by the NFV-MANO with respect to the virtualized and cloudified aspects of various types of O-RAN slices. The most common messages exchanged between the NFVO and the Non-RT RIC include: (a) handshaking messages, initiating an authorized interaction between the Non-RT RIC and NFVO; (b) request/response messages, inquiring information or services from the NFVO/Non-RT RIC and receiving a corresponding response; (c) acknowledgment messages, affirming the successful receipt of a specific message; (d) control messages, managing and controlling the exchange of messages between the Non-RT RIC and NFVO; and (e) error messages, indicating and reporting errors or exceptional conditions that occur during the interaction between the Non-RT RIC and NFVO. These messages facilitate effective communication and coordination between the NFVO and Non-RT RIC, ensuring the smooth operation and maintenance of their interoperability.

deployment and functional aspects. Hence, we choose to (a) consider the categorization of functions inside and outside the Non-RT RIC framework as a deployment matter, and (b) omit discussion on the interconnection between functions, leaving it to the discretion of vendors and operators to design the interconnection and deployment of said functions within the Non-RT RIC architecture. Nevertheless, we cover the necessary features and capabilities of anchored and non-anchored functions as per the latest O-RAN Alliance specifications.
The functions anchored inside the Non-RT RIC framework include R1 Service Exposure Functions, rApps Management Functions, A1-related Functions, AI/ML Monitoring Function, Data Management and Exposure Functions (DMEFs), and Other Non-RT RIC framework Functions, as portrayed in part (3) of Figure 4.2. The R1 Service Exposure Functions enable a range of services such as service discovery, registration, notification, authorization, and authentication through Authentication and Authorization Function (AAF), Service Registration and Discovery Function (SRDF), and R1 Service Management and Exposure Function (SMEF). The AAF provides authorization and authentication for rApps prior to granting them access to the R1 services. The SRDF enables rApps to discover and register for the desired R1 services. The R1 SMEF offers management and exposure services to R1 service consumers and providers and enables interaction between the two parties. The R1 services are facilitated by the R1 Termination Function, which allows the exchange of messages between the rApps and the Non-RT RIC framework, thereby empowering the rApps to access the R1 services via the R1 interface. The rApp Management Functions are responsible for resolving any potential conflicts that may emerge between rApps. A1-related Functions (namely Policy, Enrichment Information, and ML) produce A1 policy services, handle A1 enrichment information services, and enable/route A1-related ML services, respectively. The produced A1-related services are exposed to the rApps via the R1 interface and to the Near-RT RIC via the A1 interface, interfacing the R1 Termination Function and the A1 Termination Function, respectively. The AI/ML Monitoring Function ensures that various AI/ML models are functioning normally in multi-vendor scenarios through online monitoring services. The DMEFs provide the data registration, discovery, request, subscription, delivery, offer, and processing services between the data producers and consumers within the Non-RT RIC (or SMO) framework and beyond. Lastly, Other Non-RT RIC framework Functions accommodate any potential future functions identified in later studies.
Part (3) of Figure 4.2 also depicts that a number of functions anchored outside of the Non-RT RIC framework interact with external services and sources placed outside of SMO via their respective external interfaces. The External Enrichment Information Termination Function assists the Non-RT RIC framework in collecting external enrichment information from external sources for improving the performance of rApps and R1 services. There is also an External AI/ML Termination Function that can be used to import trained ML models (along with their metadata) from an external AI/ML server into the Non-RT RIC framework, thus further enhancing the capabilities of the Non-RT RIC architecture. The Human-Machine Termination Function enables technicians to access the Non-RT RIC for injecting O-RAN intents and operating rApps via an API. Moreover, the AI/ML Workflow Functions produce AI/ML workflow services. Finally, Function #1 − Function #n serve as placeholders for potential future functions that could be incorporated into the Non-RT RIC architecture.

4.1.2 Proposal for Enabling Interoperability Among Non-RT RIC, 3GPP-NSMS, and NFV-MANO

This section proposes an architectural solution that facilitates standard-compliant interoperability among 3GPP-NSMS, NFV-MANO, and Non-RT RIC. It also examines the potential

influence of Non-RT RIC on the O1 and O2 interfaces, as well as explores the enhanced capabilities that could result from such interoperability for 3GPP-NSMS and NFV-MANO.

The Interworking of NSSMF, NFVO, and Non-RT RIC

To enable interoperability among the three modules within the SMO framework, we introduce two logical functions (a.k.a. terminations) to the Non-RT RIC architecture, namely the NFVO Termination and the NSSMF Termination, as shown in part (3) of Figure 4.3. The former links the NFVO and Non-RT RIC, while the latter connects the NSSMF with the Non-RT RIC. We provide additional information regarding their definitions, scope, and implications in Table 4.2. Both functions can be situated inside or outside the Non-RT RIC framework. However, they are considered as parts of the Non-RT RIC architecture. The goal of this interoperability is to grant Non-RT RIC access to the NFVO and NSSMF, allowing it to influence the decisions made by their managing components and interfaces. Additionally, this interoperability authorizes the NFVO and NSSMF to access the R1 services and rApps, thereby enhancing the capabilities of their internal procedures and the O1 and O2 interfaces. The proposed interconnection between the three modules shall be made via standard-compliant interfaces. To date, the O-RAN Alliance has not yet defined such interoperability or the interfaces that connect these modules. To address this issue, we propose two interfaces between the three modules, as illustrated in Figure 4.3. We denote the interface between the NFVO and Non-RT RIC as `NFVO_NonRTRIC` and the interface between the NSSMF and Non-RT RIC as `NSSMF_NonRTRIC`. More details on their scope and implications are provided in Table 4.3.

To deploy the proposed interfaces between the three modules, there are two possible deployment options. The first option involves the Non-RT RIC interacting solely with NSSMF. The NSSMF then interacts with NFVO. In our previous work [135], we proposed a framework that facilitates interoperability among the 3GPP-NSMS and NFV-MANO. In the first option, we assume that the joint interactions and workflows between the NSSMF and NFVO are executed in the same manner. Hence, the `NFVO_NonRTRIC` interface can be disabled in the first option. The second deployment option entails the Non-RT RIC interacting directly with NSSMF and NFVO. Thus, the two interfaces are enabled in the second option. Each deployment option has advantages and challenges. For a more in-depth examination of interoperability among the three modules using the proposed functions and interfaces, Figure 4.3 portrays our adoption of the second option.

The NFVO Termination and NSSMF Termination are connected to the AAF and the SRDF via Non-RT RIC internal interfaces. The AAF is responsible for verifying the identities of the NFVO and NSSMF when they access the Non-RT RIC. Additionally, the AAF also verifies the R1 services, which are produced by the Non-RT RIC and rApps, that the NFVO and NSSMF will consume. After completing the authentication of R1 services, as well as service consumers and producers, this function is granting or denying service consumption and production requests based on either vendor-specific algorithms or network operator policies. The access requests and other messages between the three modules are exchanged via the `NFVO_NonRTRIC` and `NSSMF_NonRTRIC` interfaces. The SRDF permits the NFVO and NSSMF to discover, register, and de-register R1 services and rApps within SMO service regions. This function can also include profiles for R1 services and rApps to improve service discovery and registration. When required, both proposed terminations shall have the capabilities to connect with other legacy and future logical functions anchored within and outside the Non-RT RIC framework.

From the perspective of the SMO framework, the NFMFs and VIM (along with CISM, CIR,

Table 4.3: The detailed description of the two logical interfaces that connect the two proposed terminations between the Non-RT RIC, NSSMF, and NFVO.

Endpoints	Proposed Interface	Scope
NSSMF – Non-RT RIC	`NSSMF_NonRTRIC`	The `NSSMF_NonRTRIC` interface connects the NSSMF and NSSMF Termination to enable seamless interoperability and data exchange between the 3GPP-NSMS and the Non-RT RIC. This interface shall perform the most critical functionalities that are required to be executed while connecting the NSSMF and the Non-RT RIC. These functionalities mainly involve protocol conversion, data encapsulation and decapsulation, addressing and routing, data link control, flow control, security and encryption, error handling, synchronization, performance monitoring, and others. The `NSSMF_NonRTRIC` interface performs these functionalities on the messages and data (see Table 4.2) that are exchanged between the two entities, ensuring compatibility, reliability, and security across the entire SMO framework.
NFVO – Non-RT RIC	`NFVO_NonRTRIC`	The `NFVO_NonRTRIC` interface enables smooth interoperability and message exchange between the NFV-MANO and Non-RT RIC by connecting the NFVO and NFVO Termination. This interface plays a crucial role in enabling seamless communication by performing various critical functionalities. The functionalities of the `NFVO_NonRTRIC` interface comprise tasks comparable to those listed for the `NSSMF_NonRTRIC` interface above. By efficiently executing most of these tasks on the exchanged messages and data of various types of O-RAN slices, the `NFVO_NonRTRIC` interface is expected to ensure compatibility, reliability, security, and privacy between the NFV-MANO and Non-RT RIC.

CCM, and NC) are viewed as the O1 Termination and O2 Termination, respectively. These two terminations are anchored outside the Non-RT RIC architecture, as shown in parts (1) and (2) of Figure 4.3. The O1 Termination is connected to the NSSMF via the 3GPP-NSMS internal interfaces and then linked to the Non-RT RIC via the `NSSMF_NonRTRIC` interface. The O2 Termination (except O2 Termination for cellular sites) is connected to NFVO via the NFV-MANO internal interfaces. The NFVO is then linked to the Non-RT RIC via the `NFVO_NonRTRIC` interface.

Influence of Non-RT RIC on the O1 and O2 Interfaces

Once interoperability among the three modules is established, the Non-RT RIC is able to access the NFVO and NSSMF in order to influence their internal procedures and the decisions made via the O1 and O2 interfaces. This specifically means that the Non-RT RIC assists the 3GPP-NSMS entities in optimizing the performance of the O-RAN slice-specific functionalities and operations, such as lifecycle management and FCAPS, according to the O-RAN NSSTs. The Non-RT RIC does so by providing policy-based recommendations and implementing them via the O1 Termination. Furthermore, the Non-RT RIC provides enrichment information and recommendations to the NFV-MANO, directly or indirectly, to enhance its functionalities and operations regarding the MANO of VNFs, CNFs, and the underlying virtualized and cloudified resources via the O2 Termination. The Non-RT RIC can provide these value-added services by developing tailored rApps (which may relate to

the MANO services or O-RAN slice performance optimization) that allow the Non-RT RIC to implement control actions and policies via the O1 and O2 Terminations. It should be noted that the scope of the Non-RT RIC in the proposed solution is limited to improving the performance of the functionalities and operations of the NFV-MANO and 3GPP-NSMS through the addition of new capabilities and features. It does not necessarily interfere with or modify the MANO functionalities of the two modules.
To offer value-added services, the Non-RT RIC collects a significant amount of data containing multiple parameters related to the O-RAN slices from the 3GPP-NSMS and NFV-MANO, utilizing interfaces such as `NFVO_NonRTRIC` and `NSSMF_NonRTRIC`, as well as logical functions such as O1 Termination and O2 Termination. The collected data is related to the MANO services and FCAPS functionalities that are gathered from various sources such as E2 nodes, O-Cloud sites, and cellular network sites. The Non-RT RIC uses the collected data to train its ML model, utilizing either External AI/ML Services or an internal logical function (see Figure 4.2). The trained model is then employed by the Non-RT RIC to improve rApps' performance objectives and enhance the Non-RT RIC services. Finally, the rApps and Non-RT RIC provide the policies and recommendations generated by the trained ML model to 3GPP-NSMS and NFV-MANO.

3GPP-NSMS and NFV-MANO Capabilities Enhancement

The 3GPP-NSMS can access the R1 services and rApps via the `NSSMF_NonRTRIC` interface to enhance the capabilities of its entities and interfaces, as well as to optimize its interaction with NFV-MANO and other managed and managing systems within and beyond the SMO framework. Once authorized access to Non-RT RIC capabilities has been granted, the NSSMF employs R1 services and rApps features and can also expose them to the remaining entities and interfaces of the 3GPP-NSMS. For example, an rApp designed to assist NSSMF in selecting appropriate NFMFs for intelligently and optimally managing the FCAPS of the O-gNB NFs is installed in the Non-RT RIC. The NSSMF leverages the capabilities of this particular rApp to enhance its functionalities and those of NFMFs with respect to the FCAPS of respective O-gNBs in its service region. By leveraging the Non-RT RIC capabilities via the `NSSMF_NonRTRIC` interface and NSSMF Termination, the network operator can optimize the performance of the entities and interfaces of the 3GPP-NSMS.
The NFV-MANO can also enhance its capability by accessing and leveraging authorized rApps and R1 services via the `NFVO_NonRTRIC` interface. Once the NFVO has been granted access to the Non-RT RIC, it utilizes the capabilities of the Non-RT RIC and can also expose them to the remaining NFV-MANO FBs and CNFs MFs. Additionally, the NFV-MANO can optimize its interactions with the underlying infrastructure, 3GPP-NSMS, and other MANO systems by leveraging the Non-RT RIC capabilities. For instance, an rApp is designed and installed in the Non-RT RIC to assist the NFV-MANO in intelligently mapping the VNFs and CNFs of an O-RAN slice onto the underlying cloudified and virtualized O-Cloud sites. The NFVO leverages this rApp and exposes its capabilities to the VIM and O-Cloud sites via standardized interfaces during the placement of the NFs of an O-RAN slice onto the underlying infrastructure. We foresee that leveraging the Non-RT RIC capabilities by the NFV-MANO via the `NFVO_NonRTRIC` and NFVO Termination will enhance the performance of its FBs and CNFs MFs.
Note A: Drawing upon the preceding sections, the O-RAN Alliance considers three functions within the Non-RT RIC architecture, enabling its intelligence and automation. They are the AI/ML Monitoring Function, the External AI/ML Termination Function, and the AI/ML Workflow Functions. We propose their integration into a unified function. The proposed

function is referred to as the AI/ML Function, encompassing the above and any additional functions. Further discussion on the AI/ML Function can be considered in future studies of this work.

Note B: To further illustrate the implementation of our proposed logical interfaces and functions, which are detailed in Tables 4.2 and 4.3, we have provided Swagger API specifications in [Click Here]. These specifications describe the usage of a Unified O-RAN Slice Lifecycle Management API within the proposed SMO framework. The API outlines endpoints for managing `NSSMF_NonRTRIC` entries and `NFVO_NonRTRIC` entries, facilitating operations such as creation, retrieval, update, and deletion. Additionally, the API supports message exchange between NSSMF and Non-RT RIC and NFVO and Non-RT RIC. Descriptions of request parameters and response structures are included, covering O-RAN slice information, service types, quality of service profiles, security levels, etc.

4.2 Enhancing SMO Framework Through Management Data Analytics

The SMO framework is responsible for integrating the Non-RT RIC and the management systems of other SDOs [139]. These management systems primarily encompass those specified by the 3GPP and the ETSI. We refer to the management systems of these two SDOs as the 3GPP-NSMS and the NFV-MANO, respectively. To ensure that the SMO framework can deliver unified MANO services for O-RAN components, seamless interoperability among the three modules is crucial. In [1], we tackled the challenges of unification and interoperability among the three modules by introducing a unified and standard-compliant SMO framework. However, our proposal, while acknowledging the necessity, did not address the integration of AI/ML[1] models into SMO.

The integration of AI/ML has revolutionized network operations management and service optimization, offering unparalleled capabilities in analyzing vast amounts of data, predicting network behavior, and automating decision-making processes [140, 141, 142]. AI/ML algorithms have the potential to enhance the operations and network planning of mobile service providers [143]. These technologies enable swift and effective network optimizations, thereby enhancing efficiency. Moreover, they open avenues for generating additional revenue by integrating big data and networking capabilities, enabling the delivery of tailored customer experiences while maintaining stringent assurance standards [140]. In particular, within the context of O-RAN architecture, the incorporation of AI/ML algorithms holds immense potential for enhancing the efficiency, reliability, and intelligence of network MANO functions [142, 144, 145]. By harnessing the power of data analytics, predictive modeling, and automated decision-making, the integration of AI/ML algorithms is aimed at elevating the capabilities of SMO to new heights, enabling it to proactively identify and mitigate network and service issues, optimize resource allocation, and deliver superior quality of service to end-users [144, 146]. Hence, it is crucial to explore the potential advantages of integrating AI/ML models at an early stage, as their incorporation could enhance the performance of the SMO framework [147], leading to more advanced and intelligent MANO capabilities [140].

[1]The terms “MDA," “intelligence," and “AI/ML models or algorithms" are used interchangeably to refer to the implementation of advanced computational techniques aimed at extracting insights from various management systems, facilitating informed decision-making, and optimizing MANO processes within O-RAN in an intelligent and autonomous manner. Consistently throughout the thesis, these terms are utilized synonymously to underscore the adoption of data-driven approaches, aligned with the latest frameworks of prominent SDOs (e.g., 3GPP, ETSI, and O-RAN Alliance), to enhance managerial intelligence within SMO.

Recently, the O-RAN Alliance has studied the integration of AI/ML into the Non-RT RIC. In [148], the Alliance explores various use cases, outlining requirements and exploring multiple ML-assisted solutions to enhance the capabilities of the Non-RT RIC. Despite its valuable contributions, this study exhibits two limitations: First, it neglects to define the interactions between the intelligent system of the Non-RT RIC and other intelligent systems residing within SMO. Second, this study is based on an AI/ML architecture for the Non-RT RIC, which, in our opinion, demands substantial revision to enable the effective and unified delivery of ML-assisted solutions throughout the O-RAN architecture. The ETSI ISG NFV has also made significant contributions to the intelligentization and automation of the latest NFV-MANO framework, as evidenced in [149]. In addition, the 3GPP TSG SA5 has studied the integration of MDA into the 3GPP-NSMS in [150]. These two documents, [149, 150], establish a comprehensive foundation for understanding the role of AI/ML technologies in the MANO of services, networks, and their associated management systems, providing valuable insights into their potential benefits and challenges. However, to this end, neither the 3GPP nor the ETSI has yet explored the integration of their corresponding intelligent systems into O-RAN. The intelligent systems of these two SDOs may require significant architectural restructuring to seamlessly integrate into SMO. Moreover, NFV-MANO and 3GPP-NSMS may present incompatible deployment requirements, and their integration may not be compatible with the specifications defined by the O-RAN Alliance, demanding a harmonization of performance, functional, and operational requirements across all three modules.

In addition to the contributions made by the three SDOs, academia and industries have also dedicated substantial efforts to researching the integration of AI/ML capabilities into O-RAN. For example, in [151], the authors employed an intelligent master-actor-based architecture for testing AI/ML models within O-RAN, considering aspects such as performance, decision-making, security, and vulnerabilities. In [146], an AI/ML-based architectural solution is proposed to enable hierarchical and distributed optimization and automation for both Near-RT RIC and Non-RT RIC. Although the paper overviews the intelligent systems of other SDOs (including those of ETSI and 3GPP), their integration into and utilization within the O-RAN architecture is not studied. Another notable contribution in [147] explores the ML models' lifecycle workflow within the Non-RT RIC, illuminating the processes and considerations involved. Reference [152] introduces a network telemetry architecture aimed at supporting E2E analytics in O-RAN. It showcases an AI/ML workflow for two ML algorithms using the xApps. In [126], the authors investigated the integration of deep learning into O-RAN and outlined the key steps to embed a deep learning algorithm within O-RAN. Finally, in [153], a network intelligence orchestration solution is introduced as an rApp. This solution computes an optimal set of data-driven algorithms along with their execution locations. Its aim is to prevent conflicts between these algorithms, regulate desired parameters, and meet expected time requirements.

Despite the significant efforts made by the corresponding SDOs, academia, and industry to enhance the intelligentization and automation of the O-RAN architecture, **a major challenge remains in realizing a unified, interoperable, and multi-SDO-based intelligent system within SMO**. While advancements in the intelligentization of individual module of each SDO have been notable, **the lack of seamless collaboration and interoperability among these intelligent systems impedes the delivery of a coherent and unified approach to the MANO of O-RAN components**. This fragmentation poses **a formidable obstacle to realizing the full potential of AI/ML-driven solutions**, hindering the SMO framework from achieving a holistic and synergistic intelligent system. Furthermore, with the integration of intelligent management systems from different SDOs into

SMO, a variety of data types from diverse components are considered as inputs for ML model training. The **interpretation, normalization, and processing of this input data pose complex challenges**, necessitating the design of a tailored intelligent system to manage the ML workflow within the SMO framework. Addressing these research challenges is crucial for unlocking the transformative capabilities of AI/ML algorithms and ensuring their effective integration into the operations of the SMO framework for delivering advanced MANO services in an intelligent, autonomous, and efficient manner. The intelligent automation of MANO services within SMO directly influences the QoS, resource allocation, energy consumption, and the overall performance of the O-RAN architecture.
In response to the outlined research challenges, this section presents the following major contributions:

- First, **we focus on the integration of MDA into 3GPP-NSMS**. The objective of this integration is to enhance the performance of MANO operations from applications standpoint in conjunction with an O-gNB within O-RAN.

- Second, **we integrate MDA into NFV-MANO**. This integration aims to intellegentize operations associated with the MANO of the virtualized and cloudified aspects of an O-gNB, as well as the underlying wireless infrastructure, within the O-RAN architecture.

- Third, **we introduce an architectural solution to support AI/ML Function within the Non-RT RIC**. The AI/ML Function was initially proposed in [1]. In this section, we delve into its capabilities, components, and broader impact on the intelligentization of operations associated with the Non-RT RIC.

- Fourth, **we propose an architecture that unifies the intelligent systems of the three modules and enables interoperability among them** via standard-compliant interfaces. The objective of such a unification is to design a unified, crowd-sourced, interoperable, and multi-SDO-based intelligent system for SMO.

- Fifth, **we present a comprehensive E2E lifecycle workflow for realizing ML models** across the three modules. The goal is to showcase step-by-step operations for the development, deployment, and refinement of ML models across a unified intelligent system proposed for the SMO framework.

- Sixth, **we identify a number of research challenges related to integrating intelligence and MDA within SMO.** These challenges help prioritize areas of focus, guide research efforts, and advance the state-of-the-art methods, ultimately contributing to improved network performance and decision-making capabilities within SMO.

4.2.1 Integrating MDA into 3GPP-NSMS within the Context of SMO Framework

In this subsection, we discuss the MFs of the 3GPP-NSMS. We elaborate on how these MFs can be utilized within the SMO framework to manage the O-RAN components and orchestrate their required resources. Moreover, we explore the integration of the MDA into the MFs of the 3GPP-NSMS. The aim of this integration is to enhance the operations of the 3GPP-NSMS within SMO.

4.2.2 3GPP-NSMS for the M&O of O-RAN

The 3GPP TSG SA5 is among the various working groups within the 3GPP ecosystem, with a specific focus on the MANO aspects of public networks. This group therefore defines an architectural framework for the MANO of 3GPP networks [154], aimed at addressing a long list of desired use cases and requirements for the 5G and beyond communications systems [155]. Notably, among the use cases considered in [155], managing network slices within 5G networks is highlighted. This capability enables 3GPP networks to deliver diverse services by utilizing dedicated slices of a public network, each tailored to different service requirements and featuring ad-hoc performance monitoring and configuration [3]. Such services may include, but are not limited to, V2X, massive internet of things (IoT), and eMBB [3]. The 3GPP TSG SA5 specifications also provide an example of a network slice as a service (NSaaS) provided to customers (i.e., the owners of private networks), demonstrating how different network slice instances can encompass a set of dedicated and/or shared NFs over a common telecommunications infrastructure.
In the architectural framework defined by the 3GPP TSG SA5 [154], two MFs become relevant for the MANO of O-gNBs within O-RAN: the NFMF and the NSSMF. The NFMF is responsible for the lifecycle management and FCAPS of NFs within a network [155]. In O-RAN, the NFMF can manage each distinct NF of an O-gNB, such as the O-CU, O-DU, and O-RU, with dedicated NFMFs for each [135]. The key responsibilities of the NFMF with respect to the O-gNB encompass provisioning (i.e., setting up and configuring NFs to meet requirements), fault management (i.e., detecting and solving issues), and performance management (i.e., monitoring and analyzing performances) [154].
On the contrary, the NSSMF concentrates on the management and FCAPS of a network slice subnet [155], assuming responsibilities like slice subnet creation and configuration (i.e., defining and deploying slice subnets according to the customer requirements), resource allocation (i.e., allocating resources for the slice subnets to meet the requirements), monitoring (i.e., to check if specific service level agreements are respected), and adaptation (e.g., in case of changing demands over time). Within the context of O-RAN, the NSSMF is responsible for the MANO of an entire O-gNB, managing the O-RAN slice subnet and its required resources. As depicted in Fig. 4.4, multiple NFMFs manage distinct NFs for the same slice subnet, which can then be managed by a single NSSMF managing that slice subnet. Furthermore, the NSSMF can interact with other 3GPP MFs, such as the NSMF, to realize an E2E service MANO within the 3GPP TSG SA5 framework [154] and beyond (i.e., transport network MANO).
The NSSMF, NFMF, and NSMF collectively form the 3GPP-NSMS [156], which, in turn, can be regarded as part of the SMO framework within O-RAN [1]. Hence, the NSSMF and NFMF may be anchored within SMO. The inclusion of the NSMF within the scope of SMO is variable and may or may not be considered [1]. We consider the NSMF beyond the scope of the SMO framework in this thesis. Moreover, to comprehensively and efficiently manage a network slice subnet, encompassing NFs and O-RAN aspects, the 3GPP-NSMS must interact (i.e., produce and consume MANO services) with NFV-MANO and with components of the O-RAN Alliance, notable the Non-RT RIC, within SMO.

Integrating MDA into 3GPP-NSMS

Within the research and standardization communities, there is widespread acknowledgment that the performance of the 3GPP-NSMS, whether within or beyond the SMO framework, can be significantly enhanced through the application of automation techniques, data-driven solutions, and MDA-assisted mechanisms [150, 135]. Furthermore, it is also acknowledged

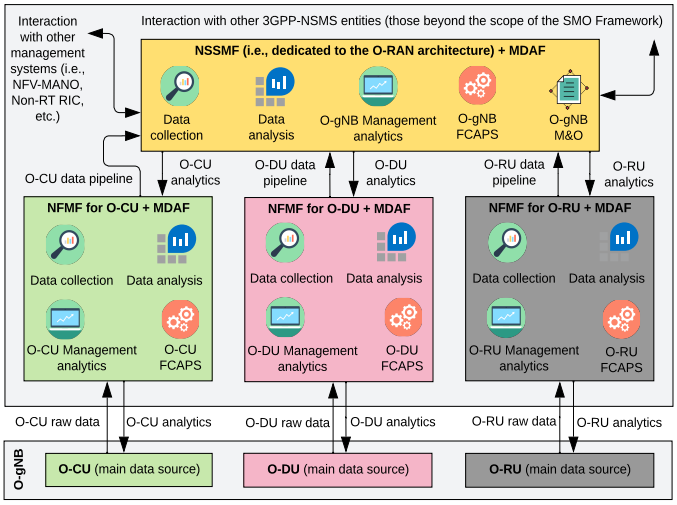

Figure 4.4: Integrating MDAS into the 3GPP-NSMS within the SMO framework

that employing MDA also improves the capabilities of the 3GPP-NSMS during its interaction with other systems (both management and non-management), such as the NFV-MANO and the Non-RT RIC.

The MDA, conceived as the endeavor to derive meaningful insights from unprocessed management data, originated within the management domain of a cellular network [150]. The MDA System is also standardized by the 3GPP TSG SA5 [150]. Its aim is to autonomously and intelligently address specific optimization challenges related to the MANO of networks and services. The MDA System provides MDAS, comprising statistical data from the past or predictive insights into the future. The determination of this depends on whether the requested timeframe is situated in the past or the future, respectively [157]. The output reports generated by the MDAS might incorporate a confidence parameter ranging between 0 and 100 [157]. This parameter communicates the level of certainty associated with the prediction made, potentially influenced by factors such as the volume of data utilized for prediction generation, the age of the AI/ML model, and other relevant considerations.

Within SMO, the MDA System is used to gather management data from diverse management systems and components, train dedicated ML models, and generate insightful analytics. A pivotal obstacle within the MDA System lies in effectively managing the vast volume of management data stemming from the SMO framework and transforming it into valuable insights. This involves understanding data structures and relationships, extracting actionable knowledge, and facilitating well-informed decision-making [150]. The overarching goal, anticipated to materialize in SMO, is to facilitate a highly autonomous management plane for O-RAN, which is capable of self-configuration, self-monitoring, self-healing, and self-optimization with minimal intervention from human users and/or human administrators [150, 158].

To integrate analytics into the functionalities of the NSSMF and NFMF within SMO, the 3GPP-NSMS can leverage the MDA System [150]. The MDA System can collect management data from NSSMF and NFMF, analyze the collected data, and provide MDA back to the corresponding functions [159]. To achieve this capability within the 3GPP-NSMS, the MDA System can employ the network data and analytics functions (MDAFs) [150], as illustrated in Figure 4.4. The MDAFs can be integrated into or deployed alongside the NSSMF and NFMF via standard-compliant interfaces. In either case, each NSSMF and NFMF determines which essential management data needs to be exposed to the MDA System, taking into account privacy and security concerns. In turn, the MDA System must specify the necessary

output required by the NSSMF and NFMF to support the corresponding use case within the 3GPP-NSMS.
When designing the MDA System for the 3GPP-NSMS, the unique characteristics of wireless communication channels within O-RAN must be taken into consideration [135]. It specifically means that the inherent unpredictability linked to factors like traffic conditions and radio-wave propagation, as well as the dynamic nature resulting from interference conditions and user mobility, necessitates thorough attention during data collection and processing, as well as leveraging analytics derived from such data. For example, several supervised ML models operate under the fundamental assumption that the data characteristics utilized for the design of the MDA System of the 3GPP-NSMS may remain consistent post-deployment. Yet, during the deployment phase, this crucial assumption is frequently breached, primarily due to the non-stationarity of the cellular network sites. Consequently, the accuracy of the ML model, deployed within the 3GPP-NSMS, may endure substantial performance degradation. Given the unique characteristics of wireless systems, it is crucial that the MDA System within O-RAN incorporates methods to guarantee resilience against non-stationarity.
The MDAS within the 3GPP-NSMS can be produced and consumed in centralized or distributed manners [160, 157]. In a centralized intelligence paradigm, the NSSMF collects management data and that of its respective NFMFs. Next, it provides the collected data to its associated MDAF. The MDAF cleans and analyzes the collected data, trains the respective AI/ML model, and delivers analytics to the NSSMF and NFMFs. In a distributed intelligence paradigm, the NFMFs collect data and provide it to their corresponding MDAFs. Each local MDAF trains its local model and delivers analytics to the corresponding NFMF. The customization of MDAFs for the NSSMF and NFMF can be such that the MDAF for the NSSMF may possess greater analytical capabilities compared to the MDAF for the NFMF. It is also worth noting that each MDAF and its consumer(s) interact with each other using `Analytic ID`. This ID indicates the nature of information collected by the MDAF from data sources (i.e., NSSMF and NFMF in the case of 3GPP-NSMS) and the type of information furnished in an analytics report to the MDAF consumer.
Fig. 4.4 shows the integration of the 3GPP-NSMS with MDA System within the SMO framework. Different NFMFs manage various NFs of an O-gNB: one for the O-CU, one for the O-DU, and one for the O-RU. Each NFMF is integrated with a MDAF, tasked with collecting and analyzing local analytics pertaining to the NFs managed by the respective NFMF. These analytics from each NFMF are aggregated at a higher level in the NSSMF, providing an overarching view of all NFs and NFMFs. The NSSMF can collect and analyze data through its associated MDAF. The MDAF co-located in the NSSMF can therefore combine the results of the lower-level MDAFs. It then provides analytics to both its corresponding NSSMF and also to local MDAFs.
Finally, it is critical for the MDA System of 3GPP-NSMS to generate and utilize AI/ML services exchanged with the intelligent systems of the management systems belonging to other SDOs within SMO. This approach facilitates the creation of a unified intelligent architecture for SMO, regardless of whether its MDAS paradigm is centralized or distributed.

4.2.3 Leveraging MDA in NFV-MANO within the Context of SMO Framework

In this section, we examine the framework of the NFV-MANO and its major building blocks. We discuss its integration into SMO for the MANO of the virtualized and cloud-native aspects of O-RAN components and resources. In addition, we explore the integration of MDA into NFV-MANO. The goal is to enhance the functionalities of NFV-MANO within SMO.

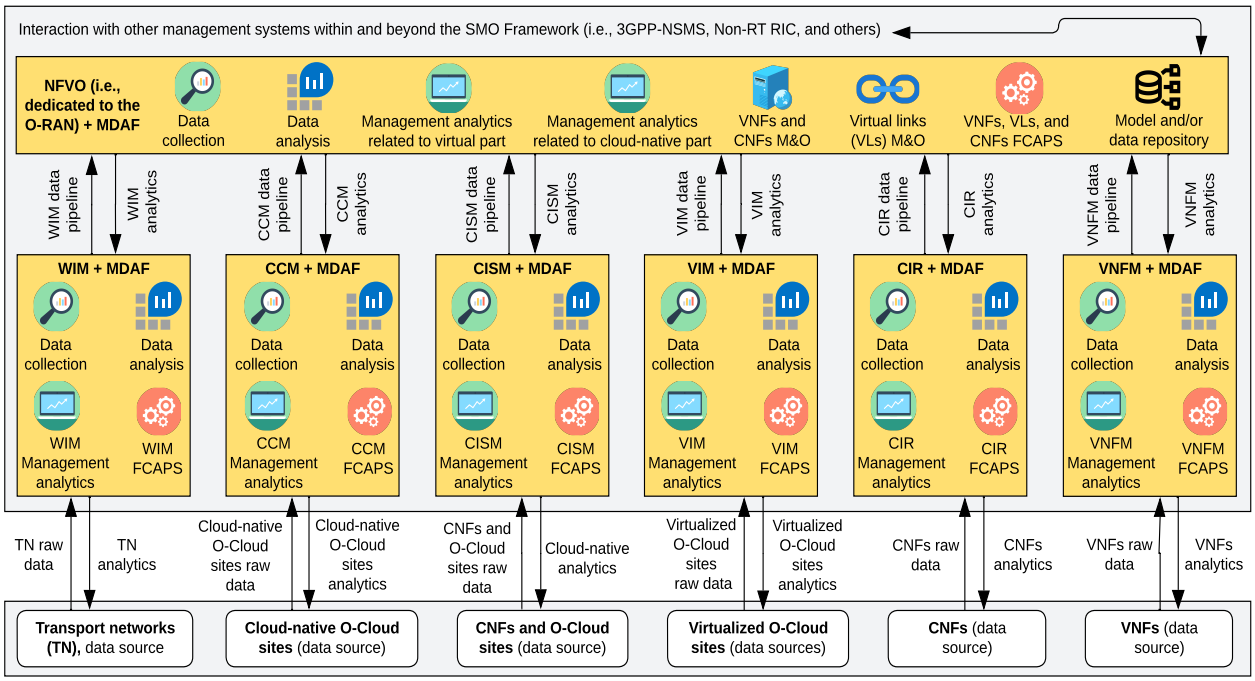

Figure 4.5: Integrating MDAS into the NFV-MANO within SMO aims to intelligentize the MANO of CNFs and VNFs through intelligent processes

NFV-MANO for the M&O of O-RAN

The ETSI NFV is one of the ISGs in the European standardization body. Founded in 2012 to address the increasing need for flexible, scalable, and efficient network infrastructure through virtualization, its scope has since expanded to encompass containerization as an advanced deployment option [161]. Recently, it has broadened its focus to include the physical management of network and computing infrastructure, supporting initiatives such as RAN virtualization [162]. Today, the ISG NFV remains a pivotal reference point for industry, with its standards continuing to be widely utilized and, consequently, remaining highly relevant. The NFV-MANO framework, introduced by the ETSI ISG NFV, initially comprised three primary FBs, which are integral for the MANO aspects of NFV resources and services [163]. These blocks include: (a) the NFVO, tasked with the lifecycle management of VNFs and the allocation of NFVI resources; (b) the VIM, responsible for managing NFVI resources, including compute, storage, and networking; and (c) the VNFM, which manages the lifecycle and FCAPS of individual VNF instances, encompassing tasks such as instantiation, scaling, and termination.

In Release 4 [149], the ETSI ISG NFV introduced several MFs in addition to the existing FBs to the NFV-MANO framework. The latest architecture of the NFV-MANO is illustrated in Figure 4.5. The newly introduced MFs consist of CISM, CIR, CCM, and WIM [1, 149]. The CISM is responsible for managing container-based applications and infrastructure services, including container management, container cluster orchestration, and management of CNFs. The CIR manages the secure and efficient lifecycle of container images, encompassing tasks such as image storage, retrieval, modification, and deletion. The CCM provides tools and procedures for managing the lifecycle of CIS clusters, covering deployment, scaling, and monitoring. Lastly, the WIM is tasked with managing virtual links among O-Cloud sites and between O-Cloud sites and cellular network sites in the underlying NFVI. This MF facilitates the E2E MANO of transport network and resources.

Both the FBs and MFs within the NFV-MANO framework play a key role in the MANO of

VNFs and CNFs [1], as well as their required resource. They can be considered part of the SMO framework within O-RAN. They are also accountable for the MANO of the virtualized and cloudified resources within the underlying wireless network infrastructure. In addition, the NFV-MANO, particularly the NFVO, within the SMO framework must interact with the management systems of other SDOs to enable SMO to deliver unified MANO services [1]. These management systems from other SDOs may encompass the 3GPP-NSMS and the Non-RT RIC [1].

Integrating MDA into NFV-MANO

In the pursuit of achieving unprecedented operational agility and efficiency within SMO, it is crucial to imbue NFV-MANO with automation and intelligent solutions. With this objective in mind, the ETSI ISG NFV identified four deployment areas (options) in Release 4 to facilitate autonomous management and integrate intelligence into NFV-MANO [149]. These deployment areas encompass intent-based management for network services, MDA-assisted management, autonomous container infrastructure management, and hierarchical closed loop automation (CLA). As of this writing, the ETSI ISG NFV has not specified the integration of intelligence into the NFV-MANO within the aforementioned areas. However, there is anticipation that the ISG will introduce two new enhancements within the NFV-MANO in the near future: intent-based management solutions and MDA-assisted mechanisms. While these two solutions have been briefly examined within the context of NFV-MANO in reference [149], the ETSI ISG NFV has not yet undertaken an in-depth exploration of the precise specifications encompassing the potential interfaces, procedures, and functionalities of the aforementioned solutions.

The intent-based management solution enables a third party to send an intent, utilizing standard-compliant interfaces, to the NFV-MANO. The NFV-MANO framework will be required to fulfill the specified intent. The third party can be a management service provider, a Non-RT RIC, a 3GPP-NSMS, etc. The MDA System gathers network and service management data from NFV-MANO, processes the collected data, trains the selected AI/ML model, and subsequently provides management analytics to the corresponding MDA consumers. The MDA-assisted solution is claimed to enhance the performance of interactions among the FBs and CNFs MFs. Additionally, it aims to optimize tasks associated with MANO and FCAPS within the NFV-MANO, as referenced in [150, 149]. The intent-based and MDA-assisted management solutions can be implemented individually, or in specific scenarios, they can be deployed side by side to enhance the capabilities of the NFV-MANO with respect to automation and intelligentization. In this thesis, our focus is solely on the integration of the MDA System within the internal processes of NFV-MANO.

The goal of integrating MDA into NFV-MANO is to achieve unprecedented operational agility and efficiency [164]. Hence, we consider MDA to be part of the SMO framework. According to [149], MDA serves as a key tool for diagnosing ongoing issues affecting the performance, health, or behavior of network services. Furthermore, it enables the prediction of potential issues, such as potential failures or performance degradation, thereby facilitating proactive management and maintenance.

Moreover, in [165], the recommendations previously determined were addressed by the ISG NFV to derive service and interface requirements for the MDA. Based on this, reference [165] introduced the MDAF within NFV-MANO, equipped with AI/ML models aimed at enhancing decision-making processes in automation related to network and service MANO. Another crucial aspect introduced is the MDA-1 interface, which facilitates communication between a MDA consumer (e.g., NFVO) and the MDA producer (i.e., MDAF). This interface

enables the consumer to invoke data analytics operations provided by the producer in a subscription/notification approach, wherein the consumer subscribes to specific data analytics, and the producer delivers results when available. It is important to note that the MDA consumer may be an internal NFV entity such as the NFVO, but also other systems external to NFV-MANO. In the scope of this paper, these external management systems to the NFV-MANO, consuming MDA services, may be the Non-RT RIC or the 3GPP-NSMS.
With the commencement of Release 5 [166], ETSI ISG NFV continued to specify alternative applications of the MDAS, aiming for more automated and integrated data analytic use cases. This initiative holds the potential for increased integration with external frameworks (both management and network). However, despite the progress made in Release 5, ISG NFV has not yet achieved full integration with external frameworks from other SDOs, such as the intelligent systems of the 3GPP-NSMS and Non-RT RIC. As described in [1], such integration is crucial for establishing a unified and standard-compliant SMO framework.
The MDA System can integrate MDAFs into NFV-MANO through two distinct scenarios. One possible scenario is to create FBs and MFs that are inherently compatible with MDAS. An alternative scenario is to create MDAFs as distinct entities and link them to their corresponding FBs and MFs via standard-compliant interfaces, such as MDA-1 interface. In both scenarios, the MDA System collects network and service management data from NFV-MANO, processes the gathered data, trains the AI/ML model, and then furnishes MDAS back to the respective FBs and MFs.
The MDAS within the NFV-MANO can be provisioned in either a centralized or distributed manner.

- In a centralized approach, the NFVO acts as the central data aggregator, collecting data from all the FBs and MFs. It then provides this aggregated data to its customized MDAF, which is responsible for cleaning, analyzing, and training ML models. The MDAF then delivers the generated analytics back to the NFVO, which utilizes them to enhance the MANO capabilities of all FBs and MFs within NFV-MANO. As previously mentioned, the MDA-1 interface facilitates communication between NFVO and its MDAF.

- In a distributed approach, each FB and MF is directly connected to its corresponding MDAF through standard-compliant interfaces, such as the MDA-1 interface. This decentralized approach allows each FB and MF to independently collect, process, and analyze their own data, leading to more localized and targeted analytics. Subsequently, the MDAFs provide their analytics directly to the corresponding FBs and MFs, enabling them to make well-informed decisions.

Regardless of whether the NFV-MANO adopts a centralized or distributed approach, it must have the capability to generate and utilize AI/ML services exchanged with the intelligent systems of management systems belonging to other SDOs within the SMO framework. This capability to exchange data and analytics among various intelligent systems fosters a unified architectural solution for the intelligentization and automation of SMO. The NFV-MANO can facilitate this interaction via the NFVO and through standard-compliant interfaces, as shown in Figure 4.5.
In Figure 4.5, the integration of MDA capabilities into NFV-MANO is illustrated. Each yellow box shown in the figure represents a FB or MF, responsible for managing different resources and services of the NFV infrastructure, as indicated at the bottom line of the figure, e.g., WIM for managing the underlying TN, VNFM for managing the VNFs. Each FB and MF can be linked with a local MDAF, performing local data collection and analysis

of the respective FB and MF and therefore providing local analytics. This matches to the distributed approach mentioned earlier. All the FBs and MFs, as outlined in [163], interact through standardized interfaces with the NFVO. The NFVO, in turn, can be integrated with its customized MDAF, enabling the collection and analysis of statistics at the orchestration level and integrating inputs from local MDAFs. Finally, at the top of Fig. 4.5, an interface is shown, assumed to allow interaction with other systems, such as the 3GPP-NSMS and the Non-RT RIC. This interface serves the purpose of enabling interaction among the intelligent systems of the three modules. More details on such integration are further described in the continuation of this paper.

4.2.4 Enhancing Non-RT RIC Capabilities with AI/ML-assisted Solutions

In [1], we extensively examined the architectural design of the Non-RT RIC. The proposed design incorporates a variety of functions, with three specifically pivotal for fostering its intelligentization and automation capabilities. These functions include the AI/ML Monitoring, External AI/ML Termination, and AI/ML Workflow. We introduced their incorporation into a unified function, collectively referred to as the AI/ML Function. This newly proposed function must have the capability to encompass not only the mentioned three functions (i.e., those introduced by the O-RAN Alliance) but also any additional ones related to AI/ML services in the future. Its primary responsibility is to automate and infuse intelligence into the operations of the Non-RT RIC.

In this section, our attention centers on the AI/ML Function. Here, we explore the architecture underpinning this critical function, delving into both its established functionalities and the introduction of innovative capabilities and components. These enhancements are designed to ensure alignment with the diverse requirements of AI/ML model training, hosting, execution, and monitoring within Non-RT RIC.

Figure 4.6 portrays an illustrative depiction of the architectural framework proposed for the AI/ML Function. This framework is crafted to enhance the ability of the Non-RT RIC to make data-driven decisions, optimize the performance of SMO, and ensure the efficient management of managing objects and managed objects within the O-RAN architecture. By offering real-time insights into Non-RT RIC behavior and AI/ML model performance, the proposed architectural framework plays a pivotal role in realizing the vision of intelligent and automated network and service management within the SMO framework. Through this discussion, we aim to elucidate the different parts, components, interactions, and objectives of the proposed AI/ML Function in a comprehensive manner.

Figure 4.6 illustrates the composition of the AI/ML Function architecture, which consists of four parts: (a) Management Data Collection and Processing; (b) AI/ML Model Preparation and Management; (c) Compulsory and Optional Components; and (d) External Integration Points. The Management Data Collection and Processing part encompasses tasks associated with collecting, processing, storing, and managing management data. It also identifies the sources from which management data needs to be collected. The AI/ML Model Preparation and Management part assumes responsibility for various tasks like training, testing, validating, hosting, management, and monitoring of the AI/ML model. The Compulsory and Optional Components part outlines various elements that may be incorporated into the AI/ML Function. These components can be either mandatory (compulsory) or discretionary (optional), depending on the specific requirements and objectives of the AI/ML Function. The External Integration Points part plays a critical role in facilitating the connection and interoperability of the AI/ML Function with other external AI/ML service-producing

components, both within and outside the SMO framework. This part enables data/model exchange, communication, and collaboration, essential for harnessing the full potential of AI/ML capabilities.

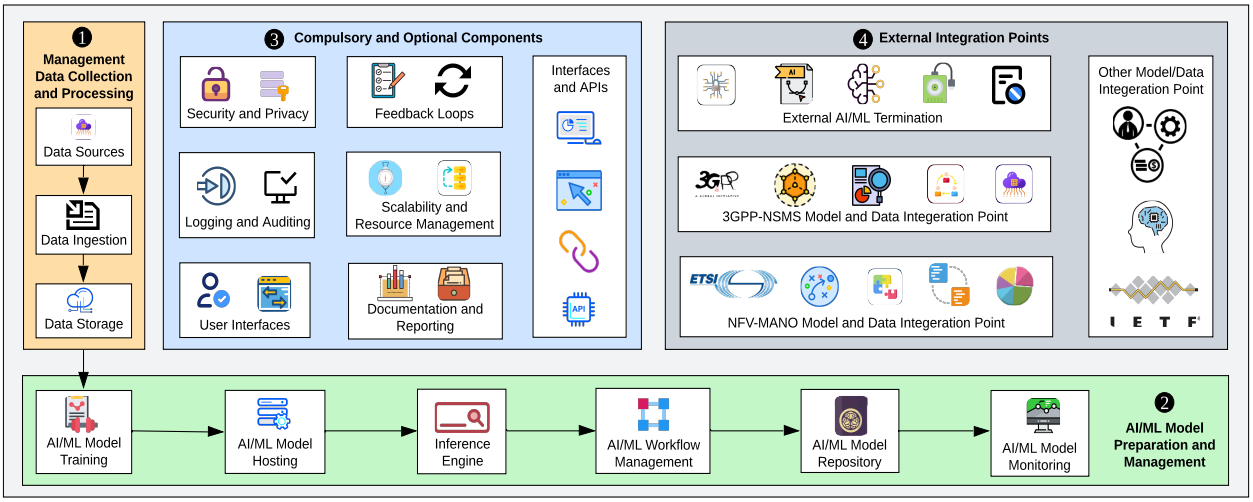

Figure 4.6: The proposed architecture for the AI/ML Function embedded within the Non-RT RIC. This diagram portrays its integral elements, encompassing crucial stages such as data collection and ingestion, model training and management, external reference points, and several optional and mandatory functionalities.

In the subsequent subsections, we provide a detailed breakdown of the components and functionalities within each of the aforementioned parts, offering a comprehensive view of the AI/ML Function and its enhanced capabilities and features within the Non-RT RIC architecture.

Management Data Collection and Processing

Data Sources are the foundational elements of the AI/ML Function, encompassing various management data producers both within and beyond the Non-RT RIC architecture. These sources include the components of the three modules, which generate a wealth of data related to network and service performance. They may also include components beyond the three modules. Their primary role is to identify and collect a diverse range of management data types [167], including network telemetry, user behavior, and service performance metrics, from various sources within and beyond SMO.
Data Sources serve as the initial data touchpoints, providing management data – whether raw or cleansed – to fuel subsequent AI/ML processes. To fulfill this role effectively, Data Sources must ensure data quality, preprocessing, and validation, accommodating various data formats and sources [167]. They play a critical role in maintaining scalability, data privacy, real-time capabilities, and seamless integration within the AI/ML Function. The efficiency of Data Sources directly impacts the quality and effectiveness of AI-driven insights and decision-making processes [167], making them integral components of the Non-RT RIC's intelligent system.
Data Ingestion serves as the crucial bridge connecting Data Sources to subsequent AI/ML processes within the AI/ML Function. Its primary role is to efficiently receive, preprocess, and structure raw data collected from various sources. Data Ingestion ensures data quality, cleanliness, and compatibility by performing tasks such as validation, format conversion, and organization [167, 168]. It plays a pivotal role in preparing data for AI/ML model

training, real-time inference, and monitoring. This contribution enhances the reliability and effectiveness of data-driven insights and decision-making within the AI/ML Function.

Data Storage serves as the foundation for the management and persistent storage of data. It retains historical and real-time data collected from diverse sources [169]. These components are designed to be highly scalable, accommodating the ever-growing volumes of data generated within SMO. They ensure data accessibility for various AI/ML processes, including model training, real-time analytics, and performance monitoring [167]. Security measures, including data encryption and access controls, are integral to safeguard data integrity and confidentiality. Effective data lifecycle management, encompassing retention policies and archiving, optimizes storage resources while maintaining data relevance. By fulfilling these objectives, Data Storage components support the Non-RT RIC in making data-driven decisions, facilitating real-time analysis, and preserving historical data as a valuable resource for trend analysis, compliance, and informed decision-making.

AI/ML Model Preparation and Management

AI/ML Model Training is responsible for the training and refinement of AI/ML models, designed to process, analyze, and extract insights from the management data collected by Data Sources and stored within Data Storage. The primary objective is to craft AI/ML models that effectively capture patterns, make predictions, and provide recommendations based on the data's characteristics [169]. This entails processes like feature engineering and hyperparameter tuning to enhance model accuracy. The quality of training data is paramount, necessitating data preprocessing and validation for optimal results. Rigorous testing against real-time and accurate data ensures the reliability and performance of the AI/ML models. Once trained, these models are deployed to undertake real-time analytics, anomaly detection, and predictive tasks. The AI/ML Model Training component stands as a linchpin in the realization of data-driven decision-making within the AI/ML Function, influencing the precision and effectiveness of its insights and actions across the Non-RT RIC [167, 169].

AI/ML Model Hosting focuses on the hosting, deployment, and real-time execution of trained AI/ML models. Once models are crafted and validated, this component ensures their integration into the AI/ML Function, making them available for inference. Its core objectives encompass deploying models, enabling real-time analysis by accepting data inputs, processing them through the models, and delivering swift predictions or recommendations. In this component, scalability becomes paramount to accommodate varying workloads and dynamic network environments [167], while efficient resource utilization and low-latency execution are critical for timely decision-making [169]. AI/ML Model Hosting also orchestrates version management, ensuring compatibility and facilitating updates. Through these endeavors, it serves as the vital conduit that empowers the Non-RT RIC's intelligent system, delivering actionable insights and enabling data-driven decision-making.

Inference Engine plays a critical role in the AI/ML Function as the real-time execution powerhouse. Its objective is the execution of trained AI/ML models, enabling timely, responsive, and data-driven decision-making [169]. This component accepts input data from diverse sources, preprocesses it as per model requirements, and guides it through the deployed models to produce instant predictions or recommendations [167]. Similar to AI/ML Model Hosting, achieving low latency is a top priority, ensuring that insights are swiftly delivered for real-time monitoring and decision-making. Furthermore, scalability is key, with the Inference Engine designed to accommodate concurrent requests and dynamic network environments. Compatibility with different model versions and resource optimization further enhance its

capabilities. Through these efforts, the Inference Engine stands as a key, enabling the Non-RT RIC's intelligent system to translate data into actionable insights and support informed decision-making [167].

AI/ML Workflow Management is an essential element within the AI/ML Function. The O-RAN Alliance included this component as an internal feature of the Non-RT RIC [1]. We utilize this function as a component of the AI/ML Function. It aims to effectively manage, orchestrate, automate, and optimize the E2E lifecycle of the AI/ML workflow, which is crucial for improving the performance of the AI/ML model. The key functionalities of this component involve designing workflow, handling the lifecycle of an AI/ML model, scalability, integration, and compliance. These aspects collectively strive to improve the efficiency and reliability of SMO by leveraging AI/ML algorithms. To implement the functionalities listed above, it is necessary for the AI/ML Workflow Management to provide interoperability across the various components of the AI/ML Function within the Non-RT RIC.

AI/ML Model Repository serves as a centralized platform for the systematic storage, version control, and management of AI/ML assets, with a specific focus on models, datasets, configurations, and metadata [168]. Its core objectives encompass asset management, collaboration facilitation, reproducibility assurance, security provision, scalability support, and comprehensive documentation. Its key features include meticulous storage combined with version control, metadata management, robust access control, streamlined model deployment, efficient asset discovery, tight integration with AI/ML workflows, data versioning capabilities, and audit trail maintenance [168]. By fulfilling these roles, the AI/ML Model Repository fosters collaboration among AI/ML engineers, ensures experiment reproducibility, simplifies model deployment processes, and provides a secure, well-organized repository for AI/ML assets, ultimately supporting the development, deployment, refinement, and management of AI/ML-assisted solutions within the Non-RT RIC architecture [168].

AI/ML Model Monitoring monitors the operations of the AI/ML Function and ensures its optimal performance. This component plays a diverse role, starting with the meticulous monitoring of every component inside the AI/ML Function, ensuring their well-being and operational effectiveness. It records AI/ML Function events through logging and auditing, enabling anomaly detection and troubleshooting. By efficiently controlling and maintaining the various components of the AI/ML Function, this component ensures the reliability, stability, and responsiveness of the Non-RT RIC, ultimately supporting data-driven decision-making.

Compulsory and Optional Components

Interfaces and APIs orchestrate seamless communication and interaction among the components of the AI/ML Function. The objective of APIs and open Interfaces is to establish channels that enable the smooth exchange of data and commands [164]. This enables the uninterrupted flow of information from Data Source components to trained AI/ML models, through the inference engine, and across different aspects of the AI/ML Function. In addition to internal connections, they also offer standardized interfaces and APIs to external components, allowing external platforms, management systems, and orchestration frameworks to easily connect and utilize the capabilities of the AI/ML Function. These components are designed with scalability, extensibility, and performance optimization in mind, adhering to industry standards and best practices. Through this framework, Interfaces and APIs empower the Non-RT RIC's intelligent system to seamlessly connect, share, and act upon data-driven insights, fostering agile and informed decision-making.

Security and Privacy assume the role of vigilant sentinels, standing guard over the

sanctity of data and operations within the AI/ML Function. Their mandate encompasses a multi-faceted mission, beginning with the protection of data's confidentiality, integrity, and availability. Through encryption, access controls, and data segmentation, they shroud sensitive information in a protective cloak. Access control mechanisms ensure that only trusted individuals and systems gain entry, reinforced by rigorous authentication and authorization processes [164]. Audit trails and logs diligently record every system interaction, serving as watchful eyes that monitor activities and detect anomalies. Compliance with data privacy regulations and standards is non-negotiable, with stringent privacy safeguards in place. A well-honed incident response protocol stands ready to thwart any security breaches. Regulatory compliance, data minimization, security testing, and privacy impact assessments all fall within their purview [164]. In summary, Security and Privacy components play a crucial role in maintaining trust by ensuring that users, administrators, and stakeholders have their data protected, their privacy maintained, and the integrity of the AI/ML Function preserved.

Logging and Auditing act as careful record keepers and attentive guardians within the AI/ML Function. They capture and record a rich spectrum of system events, from user interactions and configuration changes to data accesses and security-related incidents. These records serve as the memory of the AI/ML Function, enabling retrospective analysis, anomaly detection, and forensic investigations. Data retention policies ensure that historical records remain accessible for as long as required, supporting compliance verification with data privacy regulations. Integrity and access control measures safeguard the sanctity of log and audit data, preventing tampering or unauthorized access. In the crucible of log aggregation, data from various system facets converges for centralized analysis. Lastly, Logging and Auditing are the custodians of transparency, accountability, and security, nurturing a culture of vigilance and data-driven insights within the AI/ML Function.

User Interfaces are facilitating communication between administrators and the components of the AI/ML Function. These interfaces serve a diverse range of functions, providing users with user-friendly access to the AI/ML Function while delivering information, analysis, and management options in visually understandable formats. Operators have the ability to effortlessly engage with the AI/ML Function, utilizing configuration options and initiating actions. The utilization of real-time dashboards and alerts serves to provide users with up-to-date information, while the implementation of role-based access restrictions guarantees that users only view the data that is related to their specific duties. The primary focus in interface design is on usability, which aims to facilitate the efficient completion of tasks while minimizing the need for extensive learning. The consideration of scalability ensures that interfaces maintain their responsiveness and functionality when the AI/ML Function undergoes scaling. To sum up, User Interfaces serve as the portal to the AI/ML Function, rendering complex data and operations accessible, comprehensible, and responsive to the human touch, empowering operators and administrators to make informed decisions.

Scalability and Resource Management are key components that orchestrate computational resources to maintain system harmony and responsiveness. The objective of these components is to effectively distribute and orchestrate resources, including the compute, memory, and storage, with accuracy and flexibility. These computing resources are employed for training, validating, and testing the AI/ML models within the Non-RT RIC. The optimal distribution of these resources is crucial for enhancing the speed and effectiveness of training [164]. The components are responsible for implementing auto-scaling techniques that adaptively allocate resources based on fluctuating workloads. These strategies are designed to efficiently handle both high-demand periods and low-activity periods by either scaling up resources or conserving them accordingly.

Load balancing techniques can be effectively utilized to achieve an equitable distribution of

data and workloads across available resources, thereby preventing congestion and ensuring optimal system performance. Continuous resource monitoring maintains a constant state of vigilance, promptly alerting operators when thresholds are reached or surpassed. Capacity planning is a strategic process that involves forecasting and preparing for future resource needs in order to facilitate system expansion. Resource optimization strategies seek to minimize waste and enhance efficiency. The deployment of these techniques can ensure that the AI/ML Function architecture is capable of scaling and orchestrating harmoniously to meet the dynamic performance requirements.

Feedback Loops act as the cognitive foundation of the AI/ML Function, facilitating an ongoing exchange of information between its components. These loops extract data-driven insights, detect anomalies, and orchestrate adaptive responses, ensuring that the AI/ML Function remains agile, responsive, and perpetually improving. By integrating real-time data and historical knowledge, the Feedback Loops unravel hidden patterns and deviations, sounding the alarm when discrepancies are spotted and proactively averting potential disruptions. These loops drive the engine of continuous improvement, channeling feedback into the refinement of AI/ML models, system configurations, and resource allocation strategies. By utilizing predictive foresight, they enhance the resilience of the system by anticipating and proactively resolving forthcoming trends and challenges. In the context of dynamic data landscapes, Feedback Loops monitor and optimize the performance of the AI/ML Function. These loops possess the essential qualities of vigilance, insight, adaptability, and automation, enabling them to effectively coordinate and enhance operational efficiency.

Documentation and Reporting are responsible for recording and communicating information. They rigorously document the progress of the AI/ML Function and convey its insights to the components (within or beyond the Non-RT RIC) interested in understanding its operations. Documentation and Reporting diligently capture and store log data, performance metrics, and noteworthy events. This practice aims to create a comprehensive historical narrative that sheds light on the behavior and evolution of the AI/ML Function. Equipped with an abundance of data, these components generate notifications, dashboards, visual representations, and reports that transform unprocessed data into actionable insights.

The purpose of these reports extends beyond mere documentation, as they serve as coherent narratives that effectively convey the state of the AI/ML Function, including its overall condition, any deviations from normalcy, and prevailing patterns, to administrators and operators for making decisions. By employing real-time monitoring, configurable reporting, and adherence to compliance standards, these components play a crucial role in maintaining transparency, accountability, and the ability to make well-informed decisions and continuously improve the AI/ML Function. In essence, Documentation and Reporting are the chroniclers and interpreters of the AI/ML Function. They provide guidance and insights that contribute to achieving operational excellence and fostering data-driven knowledge and well-informed decision-making.

External Integration Points

External Integration Points connect the AI/ML Function with the AI/ML (or management data) producers anchored within and beyond SMO, facilitating access to a wide range of data, services, and opportunities for cooperation. The AI/ML producers encompass components such as the 3GPP-NSMS MDAFs, the NFV-MANO MDAFs, and other producers of data or models. The major role of the External Integration Points is the efficient acquisition of data from external sources, the sharing of AI/ML models and insights with external services, and the capability to receive event notifications from the wider ecosystem. These integration

points also facilitate data enrichment, which involves incorporating information from external sources to enhance the contextual comprehension of AI/ML analysis. To achieve interoperability, it is envisaged that the External Integration Points will engage in communication using standardized language and protocols while also implementing rigorous security measures and access controls. External Integration Points enable the AI/ML Function to surpass its limitations by handling various data formats, protocols, and compliance requirements. This allows for the integration of external information, enhances the analytical capabilities of the system, and facilitates data-driven intelligence and collaboration.

4.2.5 Unification and Interoperability Among the Intelligent Systems of the Three Modules

To facilitate seamless integration and the effective implementation of intelligent-driven solutions within SMO, it is crucial for the distinct components of the intelligent systems across the three modules to collaborate in a unified fashion. To realize this unification, we propose an architectural solution, as illustrated in Figure 4.7. The proposal leverages two logical functions, also referred to as terminations, namely the NSSMF Termination and the NFVO Termination. It also utilizes their associated interfaces, namely the `NSSMF_NonRTRIC` and `NFVO_NonRTRIC`. The aim of leveraging the two terminations and their associated interfaces is to unify the three modules and facilitate interoperability among them. We conceptualized and detailed the proposed terminations and interfaces in [1]. In this thesis, specifically in this section, we explore the complexities involved in fostering seamless interoperability among components that both generate and consume AI/ML services across the three modules. To further enhance the capabilities of the three modules, the two terminations and their associated interfaces shall expand their support to encompass functionalities related to AI/ML service exchange and delivery. This extension aims to complement the fundamental features and characteristics of the terminations and interfaces, as addressed in [1].
The NFVO Termination and NSSMF Termination serve as pivotal components for the exchange of data (cleaned and raw) and AI/ML models between the MDAFs of the NFVO and NSSMF and the AI/ML Function of the Non-RT RIC. To achieve effective data or model exchange, Data Transformation and Model Translation are crucial features of the two terminations, as shown in Figure 4.7. Data Transformation acts as a data wrangler, ensuring that the data formats and structures between modules align. It performs data cleaning, normalization, encoding, and other necessary operations to harmonize the data flow. On the other hand, Model Translation acts as a linguistic interpreter, translating the models developed in one module into a language understandable by the other module. It encapsulates models as serialized representations, converts algorithms to compatible formats, and ensures parameter compatibility. By seamlessly transforming data and translating models, both terminations enable the seamless exchange of information and expertise between the intelligent systems of the three modules, enhancing the overall performance and capabilities of SMO. Furthermore, the two terminations also establish communication with the AI/ML Function of the Non-RT RIC via internal interfaces. This interaction enables seamless data and model exchange among the three modules within the SMO framework. Further details on the definitions and scope of both terminations, in relation to supporting AI/ML service exchange within the SMO framework, are provided in Table 4.4.
The `NFVO_NonRTRIC` and `NSSMF_NonRTRIC` interfaces enable communication between the MDAFs of NFVO and NSSMF and the NFVO Termination and NSSMF Termination, respectively. Beyond their core functionalities described in [1], the two interfaces can additionally enable signaling, synchronization, and transfer of data and/or models between the intelligent

Table 4.4: The definitions and scope of the two proposed functions that enable interoperability among the intelligent systems of the three modules

Termination Name	**Scope Regarding Support for AI/ML Capabilities**
NSSMF Termination	Building upon its main features (described in [1]), the NSSMF Termination must also support advanced features related to AI/ML and automation within SMO. The newly introduced enhancements aim to leverage the capabilities of AI/ML to improve efficiency, accuracy, security, and performance when the intelligent systems of the Non-RT RIC (i.e., the AI/ML Function) and 3GPP-NSMS (i.e., the MDA System) interact via this termination. These mainly include the exchange of data (both raw and cleansed) as well as AI/ML model between the MDAF of the NSSMF and the AI/ML Function of the Non-RT RIC. In the exchange processes, AI/ML plays a key role by enabling intelligent routing and load balancing through predictive analytics. These analytics anticipate data traffic patterns and adjust routing strategies dynamically to balance the load across two modules, ensuring efficient and timely data/model exchanges. Additionally, real-time monitoring and feedback mechanisms continuously track the performance of data/model exchanges, identifying bottlenecks and inefficiencies. They utilize AI/ML algorithms to adapt and improve future exchanges based on past performance. Finally, AI-driven error handling and recovery processes employ predictive maintenance to foresee potential failures during model/data exchanges. This enables preemptive measures to avoid disruptions and automated recovery protocols to ensure minimal downtime and data loss in case of issues.
NFVO Termination	In addition to the core functionalities outlined in [1] for this termination, it must also support the exchange of AI/ML capabilities between the NFV-MANO and the Non-RT RIC. These capabilities primarily involve the exchange of data (both raw and cleansed) and trained AI/ML models between the AI/ML Function of the Non-RT RIC and the MDAF of the NFVO. This entails robust communication protocols to ensure reliable transmission of diverse data formats, ranging from structured datasets to complex ML models. Moreover, the NFVO Termination boasts comprehensive authentication and authorization mechanisms to safeguard the integrity and confidentiality of the exchanged data and models. Furthermore, it is endowed with scalability features capable of accommodating varying workloads and evolving data requirements over time. Within the NFVO Termination, real-time feedback and monitoring of data and model exchange facilitate the identification of bottlenecks and inefficiencies by both modules. The monitoring mechanism can utilize AI/ML algorithms to adjust and enhance forthcoming data and model exchanges based on insights from previous performance. Finally, during the exchange process, AI/ML takes center stage by facilitating intelligent routing and load balancing through predictive analytics. These analytics forecast data traffic patterns and dynamically adjust routing strategies to evenly distribute the load between the two modules, ensuring the smooth and timely exchange of data/models.

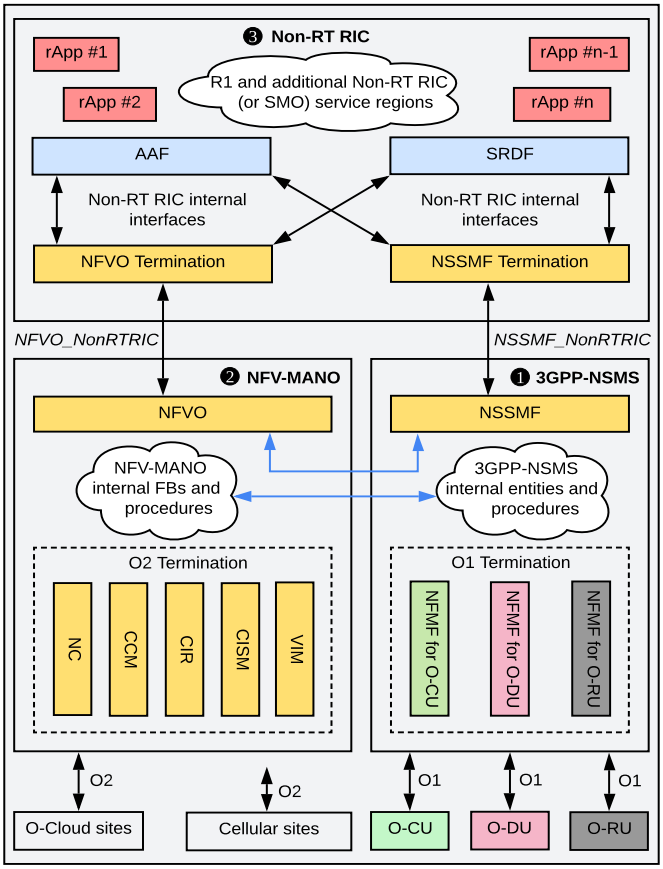

Figure 4.7: Proposal for seamless unification and interoperability among the intelligent systems of the three modules within SMO

systems of the three modules. They can handle high-throughput data transfer, ensuring data accuracy and reliability, even in the presence of network disruptions. Furthermore, they facilitate authentication, authorization, and data encryption, safeguarding sensitive information from unauthorized access. By implementing these functionalities, the two interfaces ensure a secure, reliable, and efficient communication channel between the three intelligent systems. Table 4.5 provides additional information regarding the scope and implications of the two interfaces in supporting AI/ML service exchanges between the two proposed terminations.

Moreover, the MDAFs for the NSSMF and NFVO can have additional capabilities beyond what is shown in Figures 4.4 and 4.5. These supplementary capabilities would enable the two MDAFs to consume and produce AI/ML services both within and outside SMO. These capabilities include, but are not limited to, archiving and exchanging raw data, cleansed data, and trained models. The MDA Systems of the 3GPP-NSMS and NFV-MANO execute these tasks in collaboration with the Non-RT RIC through the defined interaction points depicted in Figure 4.7. The interaction points not only facilitate the creation and provisioning of AI/ML models and data for the Non-RT RIC but also enable NFV-MANO and 3GPP-NSMS to retrieve and extract AI/ML models or data from the Non-RT RIC as required. This capability empowers the Non-RT RIC to influence the decision-making of the MDA Systems in 3GPP-NSMS and NFV-MANO. Furthermore, it grants both MDA Systems access to a wealth of data and models from the AI/ML Function and R1 services within the Non-RT RIC, further bolstering their model development, deployment, and enhancement capabilities.

Introducing this multi-SDOs, unified, interoperable, and crowd-sourced intelligent system across the three modules represents a significant shift within the SMO framework, transcending

Table 4.5: Detailed definitions and descriptions of the two interfaces used to connect the two proposed logical terminations (namely NFVO Termination and NSSMF Termination) within the Non-RT RIC to the intelligent systems of NFV-MANO and 3GPP-NSMS, emphasizing their support for AI/ML capabilities

Endpoints	**Interface Name**	**Scope Regarding Support for AI/ML Capabilities**
MDAF of NSSMF – AI/ML Function	`NSSMF_NonRTRIC`	Initially, the `NSSMF_NonRTRIC` interface is used to connect the NSSMF and NSSMF Termination, aiming to enable interoperability between the 3GPP-NSMS and the Non-RT RIC within the SMO framework. In addition to its fundamental features, as discussed in [1], this interface must also support the exchange of data (both raw and cleansed) and AI/ML models between the AI/ML Function within the Non-RT RIC and the MDAF of the NSSMF. To ensure seamless data/model exchange, the `NSSMF_NonRTRIC` interface should incorporate several key functionalities. Firstly, it should enable efficient data transfer mechanisms, supporting various data formats and protocols. It should also facilitate model serialization and deserialization to ensure AI/ML models can be transmitted in a compact and standardized format. Furthermore, the `NSSMF_NonRTRIC` interface should provide secure authentication and authorization to protect data/model integrity and privacy. Version control features are essential for tracking changes in models and datasets, ensuring compatibility and reproducibility. Moreover, it must support asynchronous communication and include error handling and logging mechanisms to diagnose and resolve issues promptly. Finally, the `NSSMF_NonRTRIC` interface should include monitoring and reporting capabilities to track performance metrics and data flow, ensuring smooth and efficient operations between the intelligent systems of the two modules.
MDAF of NFVO – AI/ML Function	`NFVO_NonRTRIC`	Originally, the `NFVO_NonRTRIC` interface facilitates seamless interoperability between the NFV-MANO and Non-RT RIC by linking the NFVO with the NFVO Termination within the SMO framework. Besides its core functionalities outlined in [1], the `NFVO_NonRTRIC` interface must also enable the transfer of both raw and processed data, as well as AI/ML models, between the AI/ML Function within the Non-RT RIC and the MDAF of the NFVO. For a smooth exchange of data and AI/ML models, the interface should integrate several essential functionalities. First and foremost, this interface should facilitate efficient data transmission mechanisms, accommodating diverse data formats and protocols. Additionally, it should streamline model serialization and deserialization processes to guarantee compact and standardized model transmission. Moreover, the `NFVO_NonRTRIC` interface ought to furnish robust authentication and authorization mechanisms to safeguard the integrity and privacy of data and models. Version control functionalities are indispensable for monitoring alterations in models and datasets, thereby ensuring compatibility and reproducibility. Furthermore, it must facilitate asynchronous communication and incorporate error handling and logging mechanisms to swiftly diagnose and resolve issues. Lastly, the `NFVO_NonRTRIC` interface should encompass monitoring and reporting functionalities to monitor performance metrics and data flow, guaranteeing seamless and efficient operations between the intelligent systems of the NFV-MANO and Non-RT RIC.

siloed limitations and embracing a holistic perspective of the O-RAN architecture. It fosters seamless communication and collaboration among the intelligent systems of the three modules, breaking down data silos and enabling the exchange of knowledge and insights. This interconnectedness empowers SMO with the ability to make well-informed decisions grounded in a wider and more comprehensive understanding of the network's dynamics in an intelligent manner. The crowd-sourcing aspect further strengthens the SMO framework's capabilities by tapping into the collective intelligence of a vast network of components. By incorporating data and insights from diverse sources, the intelligent system of each module can continuously refine its AI/ML models, adapt to evolving network conditions, and provide personalized recommendations to optimize user experiences. In conclusion, the proposed unification and interoperability among the three intelligent systems represent a groundbreaking advancement in the MANO of O-RAN components and interfaces. This transformative approach paves the way for proactive issue identification, intelligent solution deployment, and optimized network performance and resource utilization, propelling O-RAN into a new era of efficiency, resilience, intelligence, automation, and innovation.

4.2.6 E2E Lifecycle Workflow of the AI/ML Model Across the Three Intelligent Systems

Building upon the contributions from the previous sections, we present an E2E workflow for the lifecycle of a ML model across the three intelligent systems in this section. The proposed E2E workflow provides a well-structured plan for ML development within the SMO framework, fostering efficient and effective deployment of trained models. It promotes collaboration and knowledge sharing among the three intelligent systems, ensuring that ML models are enriched by collective insights and expertise. Furthermore, the workflow streamlines seamless integration and exchange of data, enabling the continuous refinement and optimization of ML models based on real-time management and network data. The holistic approach presented in this section aims to maximize the intelligence and adaptability of the SMO framework by leveraging the strengths of each module and fostering a synergistic relationship among them. The comprehensive and integrated workflow for the development, deployment, and optimization of ML models across the three modules within SMO is illustrated in Figure 4.8. This workflow is structured into several distinct phases, each of which is represented visually by an individual box. The components of each module undertake a set of specific ML operations throughout each phase of the workflow. The output of each phase is transmitted to the subsequent phase through a dedicated step, as denoted in Figure 4.8. These phases or steps can be optional or mandatory. The inclusion or exclusion of specific phases and steps within the proposed workflow is determined by the discretion of the network operator and AI/ML model deployment requirements.

In the first phase, data sources and relevant management data are identified and gathered within each module through corresponding interfaces, as shown in Figure 4.8. For example, the MDA System of the 3GPP-NSMS collects management data from NSSMF, NFMFs, O-gNB NFs, and UEs. Likewise, the MDA System of the NFV-MANO and the AI/ML Function of the Non-RT RIC acquire relevant data from their respective sources. Subsequently, the collected data can be archived in a designated data repository within each module (step 1). This repository serves as a central hub for managing and accessing the vast amount of data generated by each module. Following collection, the collected data undergoes thorough storage, management, and organization to ensure seamless access, analysis, and sharing (step 2). Next, the selected data undergoes rigorous cleansing, ingestion, fusion, and preparation for training (step 3). The refined and cleaned data is then meticulously analyzed and prepared

to be utilized to train and develop the desired ML models (step 4).

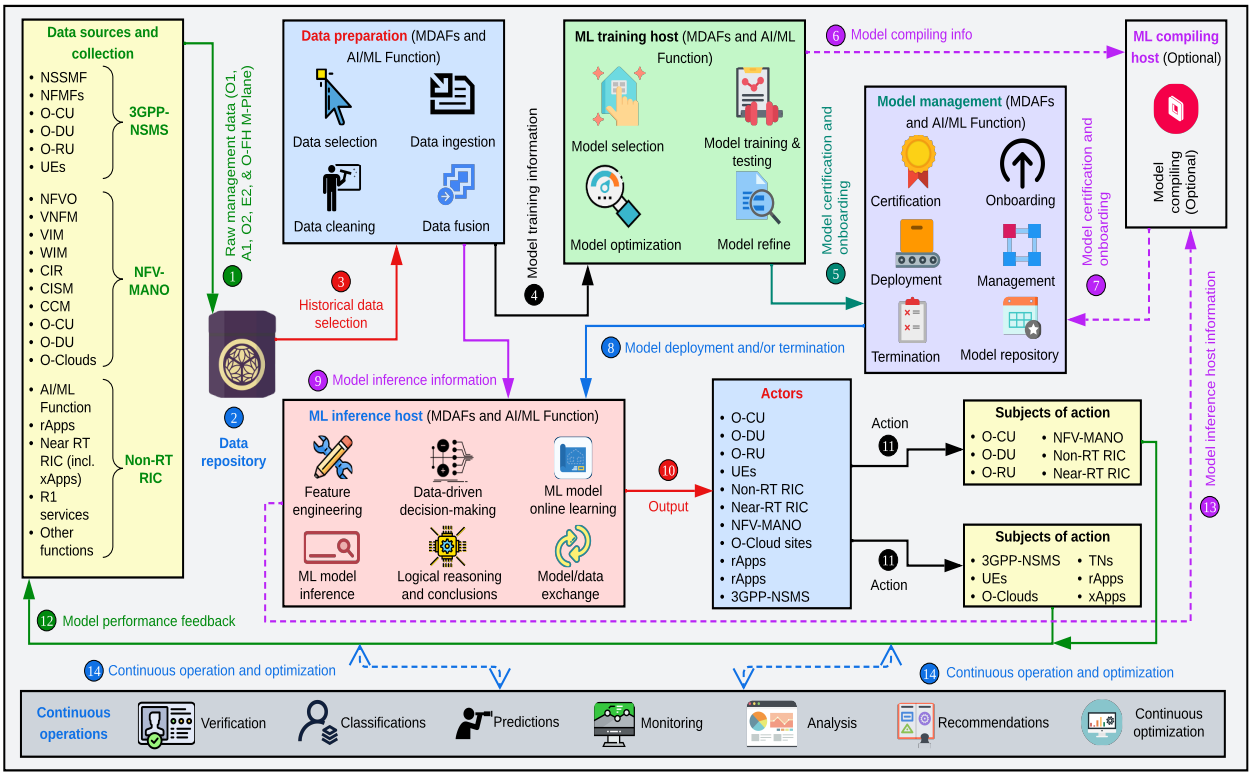

Figure 4.8: E2E workflow for data collection and ML model lifecycle management across the intelligent systems of the three modules within SMO

Within each module, the model training host, such as a MDAF responsible for model training, decides whether to utilize an existing model or initiate the training of a new model. The selection of an appropriate ML model is performed in accordance with the requirements of the use case that demands a ML-assisted solution. Upon selecting a ML model, it undergoes training, testing, and validation using the prepared data as a foundation. Once the model has successfully completed training and validation, the training host will communicate the trained model's availability to the model management phase (e.g., any MDAF responsible for managing ML models), initiating the process of certification, onboarding, and deployment (step 5), as shown in Figure 4.8.

For seamless deployment of the trained ML model across each module, the training host may provide model compilation instructions to the designated compiling host (step 6). The compiling host, if utilized, can be a MDAF or AI/ML Function capability that transforms a trained model into an optimized executable format for deployment across SMO. This information will be subsequently communicated to the model management phase during the deployment stage (step 7), as illustrated in Figure 4.8. The model management phase will coordinate the deployment of the compiled model within each module, ensuring seamless integration and operation. Furthermore, during the model management phase, the compiled model can be stored in a centralized model repository for future retrieval and deployment. The repository functions as a centralized hub where models can be accessed and managed at any stage of their lifecycle.

At this stage, the compiled model transitions from a dormant state into active operation at the ML inference host (step 8) within each module. An inference host can be a MDAF or an AI/ML Function that serves as the platform for the trained model to perform inference tasks, which may include both online learning and model execution. This specifically implies that

the MDAFs within 3GPP-NSMS, the MDAFs within NFV-MANO, and the AI/ML Function within Non-RT RIC initiate the deployment of their trained models within their respective domain-specific contexts. While in operation, the inference host can engage in online training by continuously acquiring real-time management data from data sources, enabling the ML model to adapt and improve its performance over time (step 9), as shown in Figure 4.8. It seamlessly integrates online data with an offline-trained model to dynamically augment the model's capabilities during operation. This continuous learning process ensures that the model remains adaptive and optimized to handle real-world changes and complexities.
The inference host has the capability to exchange model or data within each module. This can be facilitated using standard-complaint components and interfaces. For example, the NSSMF Termination and `NSSMF_NonRTRIC` facilitate model or data sharing between the MDAF of NSSMF and the AI/ML Function within the Non-RT RIC. This capability enables each module to generate and consume data or models from other modules, promoting seamless interoperability among the intelligent systems within the three modules. Utilizing the output of a ML model (step 10), the inference host will proactively notify the relevant actors to undertake the necessary actions (step 11) towards the subject of action, as illustrated in Figure 4.8. The subject of action can either be a NF or a MF that requires optimization through a ML-assisted solution. Within the proposed workflow, an actor can be any entity that houses a ML-driven solution, leverages the output of ML model inference, and executes a specified action.
Up to this point, the ML model has been trained and deployed within each module and is also accessible for exchange with the other modules. To further enhance the performance of the deployed model, the workflow continuously collects performance reports from the actors. Figure 4.8 shows that each module anticipates actors to provide feedback or report on the performance of the ML model (step 12). Furthermore, the inference host also provides the compiling host with reports on the performance of the model (step 13). These performance reports are then analyzed and utilized to potentially retrain, refine, optimize, update, re-select, or, in the worst-case scenario, terminate the model for execution in their respective phases (see model training and model management phases).
Finally, the workflow encompasses the continuous operation phase (step 14), as depicted in Figure 4.8. This phase incorporates a range of features for the continuous enhancement of ML models throughout their E2E lifecycle. During this phase, the ML model and its output undergo verification, monitoring, analysis, and optimization. Recommendations, predictions, and classifications are dynamically provided to the intelligent system of each module based on real-time requirements or predetermined time intervals.

4.2.7 Key Research Challenges

This section identifies key research challenges linked to the integration of intelligence into the SMO framework. Addressing these challenges necessitates substantial interdisciplinary research efforts and collaborative partnerships among various SDOs and stakeholders. This focus is crucial to meet the growing needs of developing and deploying an intelligent SMO. Ultimately, this will unlock the full potential of intelligent MANO for O-RAN components, facilitating the support of a diverse array of use cases and applications.
Data Security and Privacy: Ensuring secure and efficient management of the data pipeline while adhering to data governance regulations, such as the General Data Protection Regulation (GDPR) in the European Union or the California Consumer Privacy Act (CCPA) in California, United States, is essential within the three intelligent systems. Without a secure and protected data pipeline, achieving a scalable, reliable, trustworthy, and unified intelligent system for

SMO is unattainable. This becomes particularly critical if a centralized intelligent approach is adopted for training a ML algorithm. Furthermore, research is required to develop novel techniques for access control (both internal and external to the SMO framework), to manage data anonymization, to mitigate model inversion attacks, and to address data poisoning attacks customized to SMO. These efforts are essential to mitigate risks and ensure responsible data collection and model training practices. Distributed learning methods, such as federated learning, can be a promising approach that allows training models on distributed datasets without sharing the raw data itself. Integrating federated learning techniques could represent optimal solutions to safeguard data privacy and security during the collection and processing of sensitive management data from various internal and external data sources, as well as during the development and deployment of ML models within SMO.

Model Development and Deployment Scenarios: The SMO may support diverse use cases with the assistance of AI/ML models. However, developing and deploying AI/ML models within the Non-RT RIC may not consistently yield the desired performance. Meeting the demands of some use cases and applications might necessitate the partial or full involvement of the Non-RT RIC in AI/ML model training and inference. This implies that the decision of where to train and where to perform inference for the AI/ML model within SMO must be made on a case-by-case basis. For example, in certain scenarios, the AI/ML model may be trained within the Non-RT RIC and then transferred to the 3GPP-NSMS for inference. In other scenarios, both the training and inference processes may be conducted within the 3GPP-NSMS, while the Non-RT RIC orchestrates this and other models from other intelligent systems. Hence, there is a necessity to explore different scenarios for AI/ML model development and deployment within the three intelligent systems of SMO.

Communication Rounds: Whether the AI/ML model training paradigm is centralized or distributed, the AI/ML Function plays a central role in the intelligentization and automation of SMO within our proposed unified architectural solution. This means that the 3GPP-NSMS and NFV-MANO can transmit either raw management data or their customized trained AI/ML models to the AI/ML Function to train a centralized model (in the case of centralized learning) or a global model (in the case of federated learning), respectively. Subsequently, the AI/ML Function processes the imported data or models and provides the trained centralized model or global model back to the NFV-MANO and 3GPP-NSMS for inference. The exchange of models and/or data between the intelligent systems within the SMO framework is directly linked to the amount of required computing and communication resources. Considering these aspects, there is a need for an in-depth exploration of novel methods for optimizing the number of communication rounds between the intelligent systems and enhancing the accuracy of the training of desired AI/ML models. There is also a need to discover an optimal solution for balancing the trade-off between computing and communication resource utilization and the number of communication rounds between the intelligent systems. By optimizing local training and raw data processing within the SMO framework, one can indirectly manage resource consumption on devices and network bandwidth usage.

Resource Constraints and Scalability: The integration of AI/ML models into SMO within the resource-constrained environment of the network edge presents a significant challenge. O-RAN components residing at the network edge often operate with limited power and processing capabilities, necessitating careful consideration of resource constraints. Deploying complex AI/ML models directly on these components may exceed their processing and power capabilities, thus requiring the development of lightweight models or federated learning approaches optimized for edge deployment. Moreover, the immense volume of data generated within the O-RAN architecture demands efficient storage and transfer mechanisms. Addressing this challenge involves exploring techniques such as data compression, selective

aggregation, and in-network processing to mitigate storage and bandwidth limitations at the edge. Research efforts are needed to design and implement solutions that balance the computational demands of AI/ML algorithms with the constrained resources available at the network edge, ensuring efficient utilization and scalability of SMO.

Explainability and Transparency: The opacity and complexity inherent in AI/ML models present significant challenges in understanding their decision-making processes, particularly in critical network management and service orchestration tasks within SMO. It is crucial to enhance the explainability, interpretability, and transparency of these models within SMO to facilitate human comprehension, trust, and accountability. Achieving this involves addressing key issues such as model interpretability, feature importance, and decision rationale through the development of explainable AI/ML techniques, visualization tools, and model debugging frameworks. Furthermore, promoting transparency throughout the AI/ML model lifecycle management, including development, deployment, and governance processes, is essential for fostering stakeholder trust, ensuring regulatory compliance, and upholding ethical accountability standards. This necessitates robust mechanisms for documenting and communicating the methodologies, assumptions, and limitations underlying AI/ML models, thereby enabling informed decision-making and mitigating potential risks associated with their use in network and service management.

Data Quality and Availability: The quality, relevance, and availability of training data are paramount for effective AI/ML integration within the SMO framework. However, the SMO is expected to generate and collect vast volumes of heterogeneous and noisy data from various management and network components and interfaces. This data ranges from management data (such as configuration data and monitoring data) to network data (such as performance metrics and user behavior patterns). Addressing data quality issues such as missing values, outliers, and data incompleteness requires robust data preprocessing techniques, anomaly detection algorithms, and data augmentation strategies. Moreover, ensuring timely access to high-quality, labeled training data for AI/ML model development within SMO remains a persistent challenge, especially in dynamic network environments characterized by rapid changes and variability. Hence, collaborative efforts among the three intelligent systems are crucial to effectively address these challenges, leveraging multi-SDO and multi-vendor generated data to harness the full potential of ML models within SMO.

Model Governance and Lifecycle Management: Ensuring proper AI/ML model governance and lifecycle management is crucial for the automation and intelligentization of SMO. This entails establishing robust procedures and mechanisms for AI/ML model development, deployment, monitoring, refinement, and retirement. Given the critical role these models play in network management, it is essential to address issues related to model versioning, reproducibility, and regulatory compliance. Additionally, managing the lifecycle of AI/ML models involves continuous monitoring for performance degradation, concept drift, and model decay, especially in dynamic network environments. Implementing strategies for model explainability, interpretability, and accountability is also paramount to enhance trust and facilitate human oversight. Furthermore, integrating governance frameworks that adhere to industry standards and regulatory requirements is necessary to mitigate risks associated with biased or unreliable AI/ML models. Finally, establishing transparent and auditable processes for model selection, validation, and validation is essential to ensure the reliability and robustness of AI/ML-driven decision-making within SMO.

Human-Machine Collaboration: Promoting effective human-machine collaboration is essential for leveraging the capabilities of AI/ML models within SMO. This involves designing interfaces and interaction paradigms that facilitate seamless communication and collaboration between human operators and SMO-driven systems. Given the complex and dynamic nature

of network management tasks, integrating AI/ML models into the decision-making process requires mechanisms for human oversight, intervention, and feedback. Enhancing user interfaces with intuitive visualizations, real-time dashboards, and natural language processing capabilities can empower operators to interpret model outputs, understand underlying insights, and make informed decisions. Moreover, fostering a culture of trust and transparency is vital to encouraging collaboration between humans and machines. This entails providing explanations for AI/ML-driven recommendations, soliciting user feedback, and incorporating human domain knowledge into model training and validation processes. Additionally, developing adaptive learning systems that can dynamically adjust to user preferences and behaviors can further enhance the effectiveness of human-machine collaboration within SMO. By promoting synergy between human expertise and machine intelligence, operators can harness the full potential of AI/ML models to optimize network operations and drive innovation.

4.3 Scenarios, Solutions, and Challenges for Integrating AI/ML Models within the SMO Framework

The rapid evolution of cellular networks towards open, virtualized, cloud-native, and slicing-aware architectures, exemplified by initiatives such as O-RAN, has introduced a new era marked by flexibility, scalability, programmability, automation, and intelligence in network management and service orchestration [170, 139]. The pivotal driver of this paradigmatic transformation is SMO within O-RAN. SMO encompasses MFs introduced by the O-RAN Alliance, as well as those defined by other SDOs like the 3GPP and the ETSI [171]. These MFs, drawn from various SDOs, collectively provide MANO services for O-RAN components in a unified manner. SMO acts as a central nervous system, intelligently managing NFs, autonomously orchestrating resources, and efficiently optimizing network performance within O-RAN [1]. To that end, SMO seamlessly integrates AI/ML algorithms into its operational architecture.

The integration of ML algorithms, trained through big data analytics, into SMO holds immense promise for enhancing network and energy efficiency, improving service quality and user satisfaction, enabling E2E automation and full programmability, and facilitating proactive decision-making in dynamic and complex network environments [145, 160]. By leveraging ML algorithms, O-RAN can unlock valuable insights from vast volumes of data, predict and mitigate potential issues, automate repetitive tasks to streamline operations, and make well-informed decisions [164, 88]. The overarching objective of leveraging ML is to empower predominantly autonomous and intelligent O-RAN capable of self-configuration, self-monitoring, self-healing, and self-optimization with minimal human intervention [160, 88]. The considerable potential impact of AI/ML in O-RAN has garnered significant research interest, as evidenced by various research papers. Some studies, such as [145] and [146], propose general frameworks for the intelligentization and automation of O-RAN, addressing various use cases enhanced by data analytics and intelligence. Other studies focus on specific aspects within O-RAN, optimizing certain performance metrics. For example, research has delved into exploring scheduling policy for various types of O-RAN slices [88], designing xApps for deep reinforcement learning-based closed-loop control of O-RAN slicing [172], dynamic offloading of O-RAN disaggregation to edge sites for local data processing and faster decision-making [173], proposing a programmable traffic control service model for O-RAN [174], and addressing the placement and scaling of O-RAN virtual NF [175]. Beyond applying AI/ML to the operations of O-RAN, there have been efforts to explore specific ML models for deploying and testing optimal intelligence and automation solutions within

O-RAN. Examples of such studies include those referenced in [151] and [176]. In [151], the authors propose an automated, distributed, and AI-enabled testing framework designed to evaluate the decision-making performance, vulnerability, and security of AI models deployed within O-RAN. Finally, in [176], the authors introduce a framework that determines the data necessary for training an ML model for O-RAN, demonstrating significant performance improvement compared to traditional methods.
The analysis of the state-of-the-art reveals a predominant focus on devising general frameworks and assessing specific optimization solutions for integrating ML into O-RAN. However, we have recently studied the integration of MDA into the management frameworks of 3GPP and ETSI, as well as enhanced the Non-RT RIC AI/ML capabilities, within the context of SMO [14]. This study was conducted based on a unified and standard-compliant SMO framework proposed in [1]. To our knowledge, this was the first comprehensive study solely dedicated to the intelligentization and automation of SMO. Apart from our earlier work [14], it is crucial to note the limited focus thus far on harnessing ML algorithms and intelligence to enhance the capabilities of SMO within O-RAN.
With the aim of delving deeper into the intelligentization and automation of SMO, we build upon our previous work [14] and make the following contributions in this section:

- We **introduce three deployment scenarios for integrating ML models into SMO**. The primary objective of these scenarios is to allow SMO to select an AI/ML integration solution (tailored to its deployment and behavior) from a range of AI/ML integration options.
- We **concentrate on one of the proposed scenarios, wherein the ML model undergoes training through a centralized ML paradigm**. In this contribution, our objective is to introduce a comprehensive architectural solution, illustrating the E2E workflow of an ML model within the proposed centralized framework.
- We **identify several research challenges linked to the integration of a centralized ML paradigm within the SMO framework**. Addressing these challenges necessitates significant research endeavors to fulfill the escalating demands for enhancing the intelligence of SMO and facilitating the automation and intelligentization of various use cases related to the MANO of O-RAN components.

4.4 Deployment Scenarios for Integrating MDA into the SMO Framework

To integrate MDA into SMO, aiming to enhance its MANO capabilities, we propose the consideration of a minimum of three scenarios for AI/ML model training, hosting, deployment, and monitoring, as shown in Figure 4.9. The three scenarios are outlined as follows: Scenario A: External Data or Model Importation; Scenario B: Internal Model Training and Deployment; and Scenario C: Collaborative Data or Model Sharing. These scenarios have been formulated upon the foundation of a unified SMO we proposed in [1], complemented by our research on the intelligentization and automation of SMO in [14]. In general, the three scenarios we present highlight various approaches to training, hosting, monitoring, and integrating AI/ML models into the operations of SMO.
Each scenario offers its own set of advantages and challenges. Selecting the most suitable scenario hinges on various factors, including the specific use case, the expertise of network

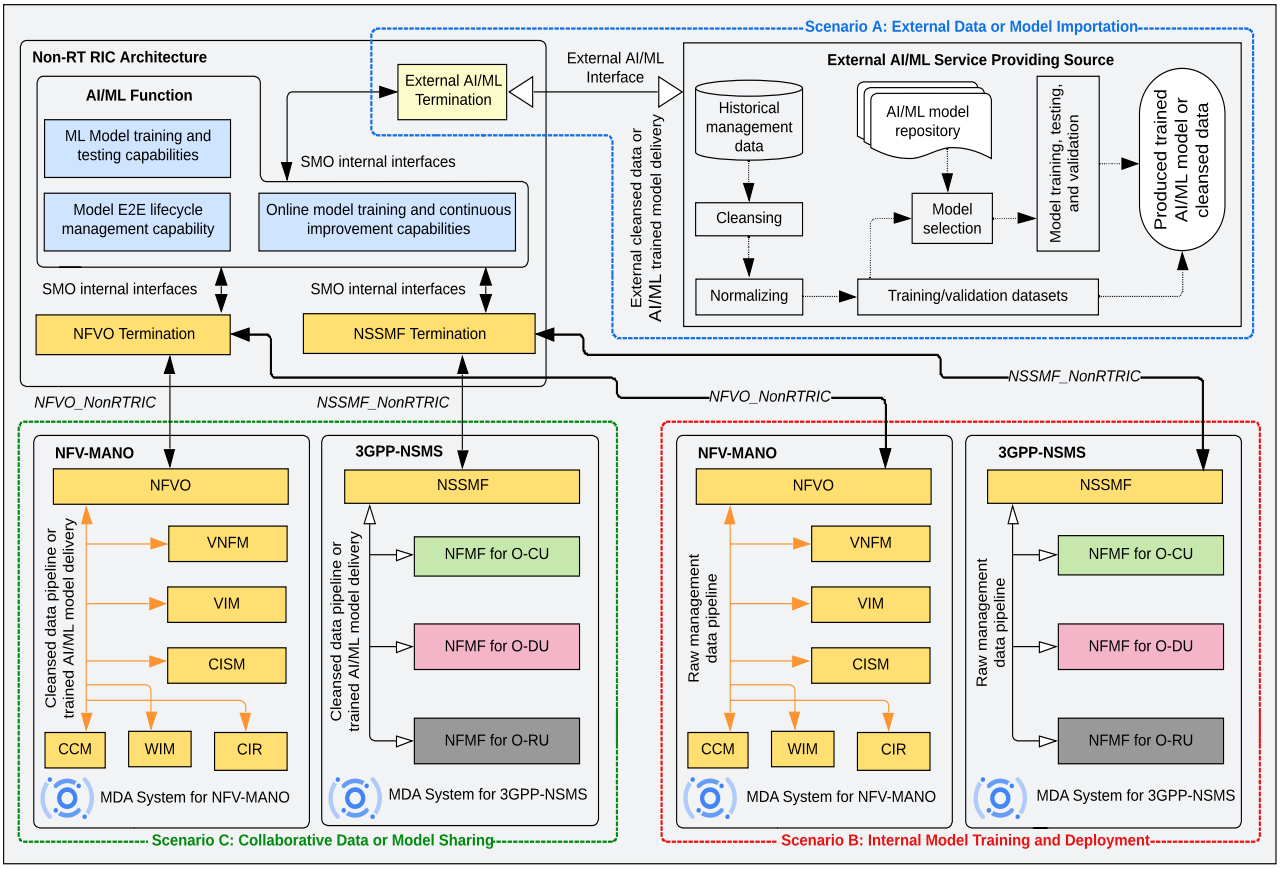

Figure 4.9: Three scenarios for incorporating AI/ML models into SMO are depicted. The scopes of Scenario A, Scenario B, and Scenario C are portrayed within the dashed blue, red, and green boxes, respectively. The legends contained within this figure have been defined in our previous work [1].

operators, the availability and confidentiality of data, as well as the volume of raw data to be exchanged between data-producing sources and model training components. Furthermore, determining an optimal AI/ML model training and deployment scenario for the automation and intelligentization of SMO also depends on the metrics used to evaluate the performance of the respective AI/ML algorithm.

The metrics used to evaluate the performance of an AI/ML model may include training time, inference time, cost and resource requirements, time-to-detection, time-to-resolution, and other relevant factors. Nevertheless, our unified and interoperable SMO framework, presented in [1], aims to provide a flexible and extensible architectural solution that supports the three previously mentioned deployment scenarios as well as other potential configurations that may be explored in the future. This capability enables seamless integration and interoperability with other standards-based AI/ML model providers, data-producing sources, and data-consuming components. In the remaining part of this section, we delve into each deployment scenario, providing a detailed discussion of its architecture, components, interfaces, and processes.

4.4.1 Scenario A: External Data or Model Importation

In this scenario, the Non-RT RIC imports cleansed data or a trained AI/ML model from an External AI/ML Service Providing Source via an External AI/ML Interface. The External AI/ML Services are provided by external components (i.e., beyond the scope of O-RAN) and can be anchored outside of SMO [1]. For example, the Non-RT RIC framework imports

a trained model or cleansed data from the ETSI ENI framework [177] or AI as a service (AIaaS) providing platform, which can be situated outside of SMO. The External AI/ML Service Provider must offer the necessary infrastructure and tools for data preprocessing, model training, and validation, aligning with best practices in machine learning operations (MLOps). It collects data from components within SMO. Data collection can be achieved via standard-compliant APIs to secure authorized access to the SMO components and retrieve the necessary management data.

By identifying the relevant data sources and understanding the API documentation, the External AI/ML Service Provider can make API requests to gather the required management data and subsequently store it within an appropriate storage system. Following these steps, as depicted in Figure 4.9, the External AI/ML Service Provider can effectively preprocess and prepare management data for training. This involves crucial data operations (DataOps) practices such as data cleansing, normalization, and the creation of training and validation datasets. Upon completion of data preparation, the External AI/ML Service Provider proceeds with the selection and training of the AI/ML model, aligning it with the specific requirements of the use case. The model training process is conducted offline, utilizing the cleansed management data in a manner widely recognized as a best practice. The trained model and cleansed data must undergo rigorous testing and validation procedures before being deployed to the Non-RT RIC. Upon completion of these processes, the cleansed data and the trained model are prepared for seamless deployment to the Non-RT RIC.

The importation of either the externally cleansed data or the trained AI/ML model to the Non-RT RIC is accomplished by utilizing the External AI/ML Termination via the External AI/ML Interface [1], as illustrated in Figure 4.9. The External AI/ML Termination is a logical function that can be anchored within the architecture of the Non-RT RIC. Upon successful importation of either the trained model or cleansed data, the AI/ML Function, which is a logical function and can also be anchored within the Non-RT RIC framework [14], takes over sole responsibility for their E2E lifecycle management. The decision of whether to import cleansed data or a trained model is a deployment choice made by the network operator.

In the event of importing cleansed data, the AI/ML Function assumes the responsibility for selecting, training, testing, and validating the appropriate AI/ML model before its execution. If the Non-RT RIC imports the model, the AI/ML Function must validate the model to ensure it meets the requirements of the use case and adheres to all privacy, security, and regulatory standards. Once the model, whether imported or trained within the Non-RT RIC, is prepared for execution, the AI/ML Function integrates it into the appropriate component(s) within SMO, aiming to enhance the performance of an optimization function. In this scenario, the Non-RT RIC may assume the responsibility for training (either online or offline), model selection, and lifecycle management, taking into consideration the imported management data. On the other hand, the Non-RT RIC's role in handling the imported AI/ML model may be limited to hosting, deployment, monitoring, and continuous improvement.

4.4.2 Scenario B: Internal Model Training and Deployment

The Non-RT RIC collects raw management data from various data-producing management systems and components within SMO via standard-compliant interfaces. The Non-RT RIC then cleanses and normalizes the collected data, as well as trains, tests, validates, executes, and monitors the selected AI/ML model. This specifically means that the AI/ML Function within the Non-RT RIC collects and utilizes management data originating from the intelligent systems of the 3GPP-NSMS, NFV-MANO, and other MANO systems and components, as

depicted in Figure 4.9. In this thesis, we adopt the MDA System as the intelligent system of the 3GPP-NSMS and NFV-MANO. Specific details regarding this adoption and the AI/ML Function are elaborated in [14].
The 3GPP-NSMS can encompass entities such as the NSSMF and the NFMFs [135]. The NFV-MANO may incorporate several FBs and MFs [135], including the NFVO, VNFM, VIM, WIM, CISM, CIR, and CCM. The Non-RT RIC may also collect data from other *de facto* and *de jure* management systems and components that operate within SMO and generate MANO services for O-RAN. The components of each intelligent system are interconnected with the components of its respective management system via standardized and interoperable interfaces [14].
The AI/ML Function of the Non-RT RIC can collect the corresponding management data from the intelligent systems of the 3GPP-NSMS and NFV-MANO through the use of NSSMF Termination and NFVO Termination via the `NSSMF_NonRTRIC` and `NFVO_NonRTRIC` interfaces, respectively. We initially introduced these two terminations and their relevant interfaces in our previous work [1]. We further expanded upon these terminations and their respective interfaces in another paper of ours referenced in [14], enhancing their capabilities to facilitate the exchange of training data and AI/ML models between the 3GPP-NSMS, NFV-MANO, and Non-RT RIC within SMO. Once the data has been collected by the two terminations, the NSSMF Termination and the NFVO Termination seamlessly facilitate the transfer of collected data to the AI/ML Function via the SMO internal interfaces.
Immediately upon receiving the relevant management data, the AI/ML Function commences the process of cleansing, normalizing, and preparing the data for validation, training, and testing. Subsequently, the AI/ML Function undertakes the training (offline or online) of the selected AI/ML model, hosts it, executes it within SMO, continuously monitors its performance, and manages it throughout its operational lifespan. If the chosen model has successfully completed training and testing, the AI/ML Function expeditiously transfers the executable AI/ML model to the NSSMF Termination and NFVO Termination via designated SMO internal interfaces.
The two terminations then take on the responsibility of conveying recommendations or predictions generated by the recently-trained AI/ML model through their respective interfaces to the management systems and components, namely the 3GPP-NSMS and NFV-MANO. These management systems utilize the outputs of the trained model to improve the performance of a certain optimization function within SMO. They also provide periodic reports to the AI/ML Function by supplying up-to-date management data. This process contributes to the improvement of the deployed AI/ML model through continuous online training. In this scenario, the Non-RT RIC holds exclusive responsibility for the intelligentization and automation of the operations and maintenance of SMO.

4.4.3 Scenario C: Collaborative Data or Model Sharing

In this scenario, the management data-producing components and systems within SMO deliver either cleansed data or trained AI/ML models to the Non-RT RIC via SMO internal interfaces. The Non-RT RIC then assumes the crucial responsibility of constructing a global model that enhances the performance of an optimization function within O-RAN. It specifically means that the AI/ML Function either collects cleansed data or trained local AI/ML models from the MDA System of the 3GPP-NSMS and the MDA System of the NFV-MANO, as shown in Figure 4.9. The AI/ML Function aggregates these individual local models or trains the cleansed data with the goal of constructing a global model.
The local models, also referred to as domain-level models in this thesis, are trained within

specific management domains (i.e., NFV-MANO, 3GPP-NSMS, and other systems within SMO), utilizing the management data or network data stored on such domains. These models guarantee that confidential information remains stored on the respective management domains and is only transmitted when absolutely required or at specified time intervals, hence protecting user privacy and data security. The global model is trained using the cleansed data or trained local models stored within the central repository of data and models of the AI/ML Function. This model incorporates the collective learnings from cleansed data or all the local models of the MDA Systems of the 3GPP-NSMS and NFV-MANO. It acts as the coordinator, periodically aggregating updates from local models and incorporating them into its own parameters.

The MDA System of the 3GPP-NSMS collects management data and network data from the NSSMF, NFMFs, and the NFs of an O-gNB. It undertakes data cleansing, conducts training for MF-level/NF-level and domain-level models, and subsequently executes these models to automate the operations of the corresponding MFs and the entire 3GPP-NSMS. MF-level and NF-level models are tailored models that are trained and executed specifically for a given MF or NF, respectively. The trained models and cleansed data of the 3GPP-NSMS can be stored within its respective MDA System for archival purposes. Depending on the configuration, the MDA System may transfer either the cleansed data or the trained models to the NSSMF Termination via the `NSSMF_NonRTRIC` interface. The NSSMF Termination may subsequently transfer the cleansed data or the trained models to the AI/ML Function via the SMO internal interface.

The MDA System of the NFV-MANO collects management data and network data from the MFs, FBs, and the underlying infrastructure. The MDA System performs data cleansing, normalization, and data preparation tasks. Additionally, it engages in the training, validation, and testing processes for the corresponding models. Once the MF-level, NF-level, and domain-level models have been trained, the MDA System executes them to intelligentize the operations of the FBs, MFs, NF, and the entire NFV-MANO. Based on the chosen design, the MDA System of the NFV-MANO may transfer either the cleansed data or the trained domain-level models to the NFVO Termination via the `NFVO_NonRTRIC` interface. The NFVO Termination then forwards them via Non-RT RIC internal interfaces to the AI/ML Function for additional processing.

If the Non-RT RIC is configured to import cleansed data from the 3GPP-NSMS and NFV-MANO, the AI/ML Function is then responsible for the validation, training, and testing of the global AI/ML model. Following the testing phase, it is anticipated that the AI/ML Function will develop an accurate and reliable global model. In the instance where the Non-RT RIC imports trained domain-level models, the AI/ML Function constructs the global model by aggregating the 3GPP-NSMS and NFV-MANO models. Once the global model (whether derived from imported cleansed data or domain-level trained models) is constructed and ready for deployment, the AI/ML Function provides it to the NFVO Termination and NSSMF Termination via the SMO internal interfaces.

The two terminations provide the global model to the MDA Systems of the NFV-MANO and 3GPP-NSMS via the `NFVO_NonRTRIC` and `NSSMF_NonRTRIC` interfaces, respectively. The two MDA Systems utilize the global model to enhance the performance and generalizability of their respective domain-level models. By providing access to a larger dataset, enforcing consistency, handling heterogeneity, correcting errors, and forming an ensemble, the global model can help domain-level models learn more accurately and generalize better to unseen data. This collaborative approach between domain-level and global models is a key enabler of federated learning, allowing us to develop AI/ML models that are both privacy-preserving and highly effective.

In Scenario C, the Non-RT RIC holds a partial role in the intelligentization and automation of the operations associated with SMO. The scope of the Non-RT RIC might be restricted to that of Scenario A. Hence, continuous interactions among the components and interfaces of SMO are essential to facilitate enhanced AI/ML model delivery and refinement.

4.5 Centralized ML Model Training and Deployment Solutions within SMO

Following the introduction of three scenarios for integrating ML models into SMO, we now focus on Scenario B. In this scenario, data is gathered at a single centralized location – the Non-RT RIC. The selected model is trained using this data, deployed across SMO, and continuously refined. In this section, we outline a detailed workflow for realizing centralized ML model training within the Non-RT RIC and its deployment across SMO. The proposed workflow consists of three phases, as shown in Figure 4.10: (a) data collection and preprocessing; (b) model training and development; and (c) model deployment and inference. By conscientiously following the three phases, robust and effective ML models can be developed that deliver valuable results for the intended optimization problem. In the following, we discuss these three phases, respectively.

4.5.1 Data Collection and Preprocessing

This phase involves gathering and organizing the requisite data for training a selected ML model within the Non-RT RIC. The data is collected at the AI/ML Function from diverse sources located within the 3GPP-NSMS, NFV-MANO, Non-RT RIC itself, and potentially other systems reside within SMO via standard-compliant interfaces. The Non-RT RIC can employ techniques such as streaming and batch collection to capture both real-time and historical data from the aforementioned sources. The collected data primarily encompasses real-time operational data pertaining to network management, service orchestration, and SMO performance.

Within the 3GPP-NSMS, the NFMFs hold a rich set of management data related to the MANO of O-CU, O-DU, and O-RU. These NFMFs collect and provide such data to the NSSMF. The NSSMF holds its own data and that of its associated NFMFs. We assume that the NSSMF, alongside its fundamental functionalities, possesses the capability to generate and utilize AI/ML services within the 3GPP-NSMS. Based on this assumption, the NSSMF, on behalf of the MDA System of the 3GPP-NSMS, provides raw management data via the `NSSMF_NonRTRIC` interface to NSSMF Termination. Subsequently, this data is transferred to the AI/ML Function via SMO internal interfaces for further processing.

The FBs and MFs within the NFV-MANO, as discussed in the preceding section, contain management data associated with the virtual and cloud-native aspects of an O-gNB and the underlying wireless infrastructure. Each MF and FB collects and holds its respective data. This data can then be transmitted to and stored within the NFVO. Assuming that the NFVO, in addition to its core functionalities, possesses the capability to produce and consume AI/ML services. Hence, under the authority of the MDA System of the NFV-MANO, the NFVO furnishes management data pertaining to the NFV-MANO via the `NFVO_NonRTRIC` interface to the NFVO Termination. Furthermore, this data is conveyed to the AI/ML Function via SMO internal interfaces for subsequent treatment.

The AI/ML Function gathers relevant data from the components of the Non-RT RIC framework and the rApps via the R1 interface and the internal interfaces of the Non-RT RIC.

In addition, the AI/ML Function can collect corresponding data from other de facto and de jure management systems within SMO via internal interfaces specific to the SMO.
Once the data has been completely gathered from the aforementioned data sources and stored within the AI/ML Function, it may contain errors, inconsistencies, and irrelevant information, necessitating preprocessing and polishing. Data preprocessing addresses these critical issues by cleansing, formatting, and transforming the collected data into a usable format for the ML model, as shown in Figure 4.10. During the cleansing process, the AI/ML Function removes duplicated data, corrects errors, and handles missing values. In the formatting process, it standardizes collected data into a common format, facilitating comparisons. In the transformation process, the AI/ML Function scales features, converts categorical data to numerical data (i.e., encoding), and performs feature engineering (i.e., creating new features from existing ones).

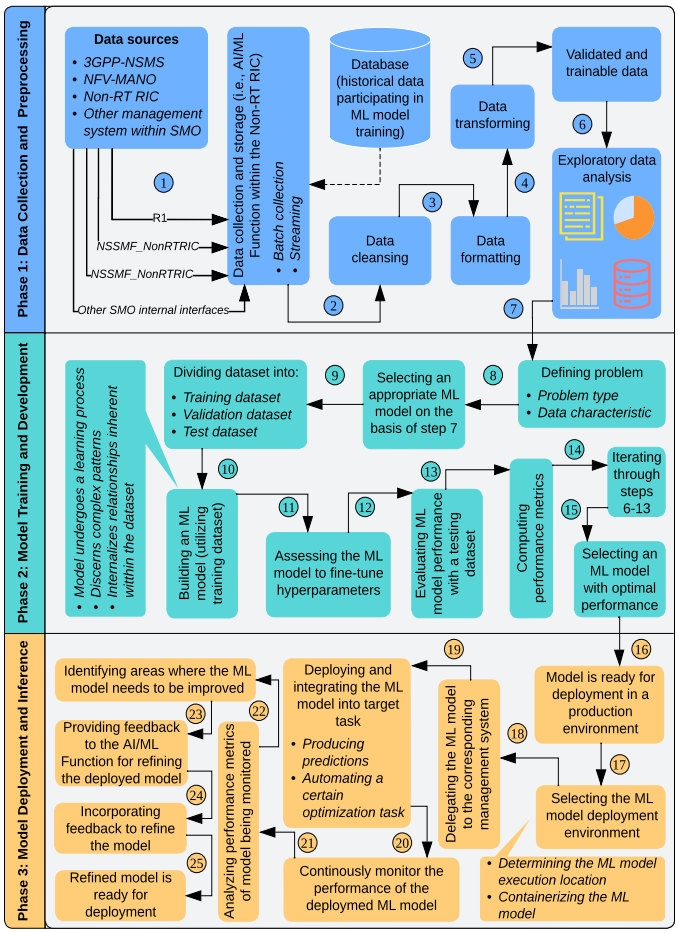

Figure 4.10: Proposed workflow for centralized ML model training, deployment, and refinement within the SMO framework of O-RAN architecture

Following the above procedures, the AI/ML Function undertakes exploratory data analysis on preprocessed data, as shown in Figure 4.10. During this process, various techniques such as visualization and statistical analysis are utilized to delve into the dataset, aiming to understand crucial variables, their correlations, magnitudes, and statistical attributes concerning the target variable. This process serves to pinpoint potential issues, guide decisions throughout model development, and distinguish between valuable features and irrelevant ones within the dataset. By executing these procedures, the data is assumed to be prepared for subsequent processes and consequently delivered to the second phase within the proposed workflow, as discussed in the following and illustrated in Figure 4.10.

4.5.2 Model Training and Development

Once the collected data has undergone preprocessing and validation, and has been successfully received by this phase, the AI/ML Function proceeds to initiate the training and development of a selected ML model. The Model Training and Development phase encompasses crucial processes in ML model lifecycle management, including appropriate model selection, training, and development, as illustrated in Figure 4.10. Each process within this phase holds significant importance in crafting resilient and efficient ML models tailored to address specific optimization problems within O-RAN.

During the model selection process, the initial crucial step is to clearly define the optimization problem that the model aims to address, as depicted in Figure 4.10. A comprehensive understanding of the optimization problem thoroughly guides the selection of appropriate ML algorithms. Depending on the problem type, data characteristics, and computational resources available, a set of candidate ML models is chosen. Common choices encompass linear regression, support vector machines, decision trees, and neural networks, among many others. Due to privacy and security concerns, the AI/ML Function is also responsible for determining whether an AI/ML model shall be deployed in a federated or centralized manner before starting to collect data from data sources.

Selecting an optimal centralized ML model requires a robust evaluation strategy. This entails dividing the data into training, validation, and test sets, as shown in Figure 4.10. The model is then trained on the training set and assessed on the validation set to fine-tune hyperparameters, which are the configurations of the model. Finally, the performance of the model is evaluated on the unseen test set. To that end, a common set of evaluation metrics includes accuracy for classification tasks or mean squared error for regression tasks.

During the process of training an ML model, the selected model is subjected to training using a training dataset. Throughout this training step, the model undergoes a learning process wherein it discerns and internalizes the complex patterns and relationships inherent within the data. Hyperparameter tuning emerges as a critical step in enhancing the performance of the selected model.

Within the context of ML, hyperparameters are configurations that govern the learning process of an ML model, such as the learning rate or the depth of a decision tree. Adjusting these hyperparameters can significantly impact the effectiveness and efficiency of the model. Techniques like grid search and randomized search are commonly employed to explore various combinations of hyperparameters, identifying the optimal settings that result in superior performance on the validation set during the training process.

In the model development process, the performance of the trained ML model is carefully evaluated on the hold-out test set. This step provides an unbiased estimate of how well the model will generalize to unseen data. Based on the evaluation metrics, the best-performing ML model from the candidate pool is selected. Furthermore, understanding how the model arrives at its predictions can be crucial. Techniques such as feature importance analysis help identify which features have the most significant influence on the model's output. This insight can provide a valuable understanding of the target optimization problem domain within O-RAN, aiding in informed decision-making and AI/ML model refinement.

4.5.3 Model Deployment and Inference

If the trained ML model performs optimally, it can be deployed into a production environment for real-time or near-real-time inference. The production environments include the 3GPP-NSMS, NFV-MANO, and other systems within the SMO. The AI/ML Function performs the model deployment and inference via the `NSSMF_NonRTRIC`, `NFVO_NonRTRIC`, R1, and

other standard-compliant interfaces. This phase entails integrating the ML model into the aforementioned management systems, where it can be utilized for real-time predictions or task automation. Additionally, this phase may involve a refinement process wherein the performance of the deployed model is continuously monitored and periodically retrained with new data to ensure its continued effectiveness in addressing optimization tasks within O-RAN.
In this phase, several critical steps must be followed, as illustrated in Figure 4.10. First, the selection of a deployment environment is paramount. This entails deciding whether the trained model will reside on on-premises servers, cloud platforms, or edge devices. The location where the model is executed depends on the requirements of the optimization problem. For specific services like URLLC, it may be necessary for the model to be executed at the edge, such as at the NFMF, to effectively optimize the management of these services. Following determining the deployment environment, containerization becomes a crucial consideration, though optional. Packaging the ML model, along with its dependencies, into a container like Docker ensures consistency across diverse environments, facilitating seamless deployment transitions. Scalability considerations form another pivotal aspect of deployment planning. Analyzing factors like anticipated workload, concurrent requests, and resource utilization aids in devising a deployment strategy that accommodates varying levels of demand [135].
Furthermore, robust monitoring and refining mechanisms are essential for optimizing the performance of the deployed model. Implementing a monitoring system within the AI/ML Function to track input data, predictions, and encountered errors provides valuable insights for continual improvement and refinement of the deployed model. Before refining the model, it is essential to assess its current performance using relevant evaluation metrics. These metrics could vary depending on the specific optimization problem as well as the characteristics and objectives of the model. Conducting a thorough analysis to identify areas where the model may be underperforming or exhibiting undesirable behavior is of vital importance. This analysis may involve examining misclassified samples, exploring feature importance, or analyzing model biases.
Once the AI/ML Function receives feedback, the model can embark on a second attempt to enhance the outcome, as depicted in Figure 4.10. In this phase, the algorithm incorporates feedback from the initial iteration and initiates improvements accordingly. As the model undergoes refinement, it yields a more polished output, thereby aiding in training the system to better comprehend real-world scenarios and expectations. Finally, by carefully navigating through these steps, a well-deployed model can fulfill its intended purpose effectively and securely, instilling confidence in its reliability and integrity.

4.6 Major Research Challenges

This section outlines several research challenges linked to implementing AI/ML solutions centrally within SMO. Addressing these challenges demands considerable research efforts to align with the escalating demands of future-oriented SMO frameworks. It also aims to facilitate the MANO of the emerging use cases and applications within O-RAN.

4.6.1 Privacy

The first technical challenge arising from the SMO-centralized AI/ML solution is data privacy. In Scenario B, the raw management data collected from various sources within SMO may contain sensitive information, such as non-access stratum signaling data. Consistently transmitting such data to the Non-RT RIC and maintaining it therewith for model training

and development poses an increased risk of data exposure and privacy breaches, particularly in scenarios where midhaul connections are established wirelessly. To address this challenge effectively, the AI/ML Function must implement robust data privacy mechanisms to ensure that sensitive information is protected. These mechanisms may include data anonymization, encryption, and secure data transmission protocols. Furthermore, the AI/ML Function must adhere to data protection regulations, such as the GDPR in the European Union, to safeguard user privacy throughout the data processing lifecycle.

4.6.2 Resilience

Ensuring the resilience of the centralized framework, particularly against data poisoning attacks, emerges as another critical concern. Centralized learning methodologies are inherently vulnerable to compromised raw data [178]. In the context of SMO, the raw data to train the AI/ML model may be tampered with, potentially leading to inaccurate predictions and suboptimal performance. Possible approaches to realizing such tampering include hijacking or manipulating user equipment to generate abnormal network behavior, or directly injecting malicious data packets into the raw management data pipeline or database. To effectively address this challenge, the AI/ML Function must implement robust data validation mechanisms to detect and filter out compromised data.

4.6.3 Elasticity

Commonly in practice, the centralized AI/ML models must be occasionally retrained to keep up with the evolving network conditions and user requirements. However, such retraining often requires significant computational resources. By sharing the resources of a Non-RT RIC with other NFs, the retraining procedures may trigger resource contention and degrade network performance. To achieve optimal elasticity, it calls for well-designed virtual resource scheduling at the Non-RT RIC that can adapt to changing workload patterns and adaptively prioritize the needs of the AI/ML Function and other NFs.

4.6.4 Agility

In addition to elasticity, the agility of the centralized AI/ML model emerges as another critical challenge, particularly concerning model retraining. While the model is trained using a vast amount of data aggregated from different domains, which usually takes a long time to converge, the network conditions, traffic patterns, and user requirements may change rapidly in individual local areas. It becomes therefore a crucial issue, how the centralized model in AI/ML Function can quickly adapt to the dynamic local environments, ensuring its continued effectiveness and accuracy in real-time scenarios. To tackle this challenge, advanced approaches are essential to enable the model to continuously and agilely learn and adapt to local variations and dynamics. This may involve incremental learning techniques, such as Stochastic Gradient Descent (SGD) [179], which allow to efficiently update a trained model based on only the differential data, instead of retraining the model from sketch on with the entire dataset. Nevertheless, this may also introduce new challenges related to incremental learning techniques, e.g., model drift and catastrophic forgetting.

4.6.5 Signaling Efficiency

Even with optimal learning efficiency that resolves the challenges of elasticity and agility, the inherent design of the centralized AI/ML solution, which involves aggregating massive amounts of raw data, gives rise to a significant signaling overhead. This overhead can

potentially lead to network congestion at the midhaul, consequently causing heightened latency and diminished QoS.

4.6.6 Robustness and Reliability

Finally, the centralized framework is vulnerable to single-point failures. If the centralized model or the AI/ML Function fails, the entire MANO procedures will be disrupted, leading to a significant degradation in network performance, or even malfunctions. To mitigate this challenge, the AI/ML Function must implement robust fault tolerance mechanisms to ensure the reliability of the centralized models and AI/ML Function. This may involve replicating the AI/ML Function across multiple Non-RT RIC instances to ensure high availability and fault tolerance. Notably, this necessitates the establishment of external data/model importation mechanisms, as discussed in Scenario A, to enable warm/hot backups. By implementing these measures, the centralized framework can withstand single-point failures and maintain operational continuity, thereby safeguarding network performance and stability.

Chapter 5

System Modeling, Problem Formulation, and Proposed Solution

Summary – In this chapter, we first present a comprehensive system model based on which we mathematically formulate the mapping problem of the vCU and vDU, internal and external VLs, joint mapping of both types of virtual components, and the VNFFG of a RAN slice onto the underlying infrastructure in the NG-RAN architecture. We next proceed with the mathematical formulation of the mapping problem of the virtual components of a RAN slice. To that aim, it is critical to define the parameters, metrics, and variables that are used in the mathematical models and equations of the proposed mapping model. In addition, we provide and mathematically formulate the main objectives of the mapping process of a RAN slice onto I-PoPs, in which we seek to find an optimal solution aimed at optimizing (maximizing, minimizing, or balancing) a set of predefined performance objectives. The mapping problem of the VNFCs, as well as the internal and external VLs, of the vCU and vDU can be limited by a number of constraints that must be stated prior to proposing the mapping solution. Hence, we provide a list of the most critical constraints that we believe have a significant impact on the quality and results of the performance objectives of the mapping process. Finally yet importantly, we discuss the proposed ML-assisted solution in order to map the requested RAN slice. We discuss various types of ML-assisted algorithms such as supervised, unsupervised, and reinforcement learning. We specifically focus on the mathematical background and application of reinforcement learning and select Q-learning algorithm in order to solve the mapping problem due to its simplicity, effectiveness, and applicability to our research problem.

5.1 System Model

The proposed system model for the mapping problem is composed of two layers: the network function layer and the infrastructure layer. The proposed mapping model is shown in Fig. 5.1. The network function layer is consisted of (among other functionalities) a vCU, a vDU, and the internal VLs and external VLs of the vCU and vDU. The infrastructure layer is comprised of (among other resource-relevant components) I-PoP #1 and I-PoP #2, which host the vCU and vDU, respectively. Each I-PoP is composed of, among other processing and networking resources, a set of VMs that are abstracted from the underlying PMs by the use of a Hypervisor. Each VM is formed of virtual compute and storage resources. The VMs are responsible for hosting the (or running the codes of their respective) VNFCs of the vCU and vDU of the requested RAN slice in both upstream and downstream directions. The I-PoP is also composed of VNs, which are abstracted from PLs of the underlying transport

infrastructure. The Hypervisor is also responsible for virtualizing the PLs and allocating them to the requested VLs of various types of RAN slices. Each VN is made up of virtual networking resources. Once the VNs are created, the Hypervisor first allocates them to the internal VLs of vCU and vDU. The VNs to the external VLs are allocated later on from the F1 and Fx interfaces, respectively.

5.1.1 Mapping the vCU and vDU

Let $\mathcal{F} = \{f_1, f_2, ..., f_{h\text{-}1}, f_h\}$ be a set of h such that $\forall h \in \mathbb{R} : > 0$ physcial and virtual functions of a RAN slice subnet, which are processed sequentially by the vCU, vDU, and RU in both upstream and downstream directions. We let $\mathcal{C} = \{f_1, f_2, f_3\}$ be a set of VNFCs assigned to the vCU such that $\mathcal{C} \subset \mathcal{F}$. In turn, let $\mathcal{D} = \{f_4, f_5, f_6, f_7, f_8\}$ be a set of VNFCs assigned to the vDU such that $\mathcal{D} \subset \mathcal{F}$. The remaining functions of $\mathcal{F}$ of a RAN slice are the members of a set $\mathcal{R}$ such that $\mathcal{R} \subset \mathcal{F}$. They are the physical functions that are processed in the RU. The mapping problem of the physical functions of the RU is beyond the scope of this thesis. It is worth mentioning that $\mathcal{C}$, $\mathcal{D}$, and $\mathcal{R}$ are said to be pairwise disjoint sets whose intersection is an empty ($\varnothing$) set.

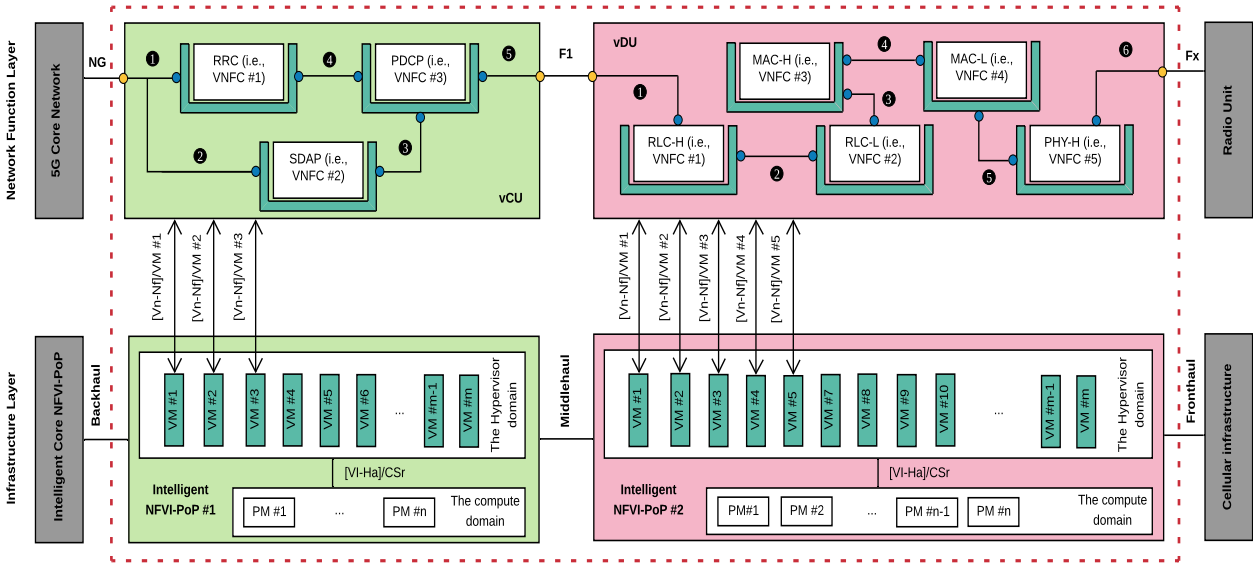

Figure 5.1: The proposed system model of the virtual compute, storage, and networking resource allocation for the VNFCs of the vCU and vDU of a RAN slice in the NG-RAN architecture. It should be noted that only the VNFCs of a gNB and the internal and external VLs, as well as their respective virtual resources in the underlying infrastructure, illustrated in the red-dotted box, are within the scope of this thesis.

The f_1, f_2, and f_3 are the RRC, the PDCP, and the SDAP functionalities of the vCU, respectively. The f_4, f_5, f_6, f_7, and f_8 are the RLC-High, RLC-Low, MAC-High, MAC-Low, and PHY-High functionalities of the vDU, respectively. Each VNFC of the vCU and vDU of a RAN slice is assumed to be hosted by a single VM (or an Unikernel or a Container). Based on this, the vCU requires three VMs and the vDU requires five VMs. These VMs belong to the vCU and vDU of a RAN slice can be hosted by a single or multiple PMs in I-PoP #1 and I-PoP #2 in the NG-RAN architecture, respectively.
Each VNFC $i \in \mathcal{F}$ (whether it belongs to vCU or vDU of a RAN slice) requires a certain amount of virtual compute and storage resources in its respective PM hosted in its corresponding I-PoP, denoted by C_i^{req} and S_i^{req}, respectively.

In light of the above, the total required virtual compute and storage resources of the vCU and vDU of a RAN slice are obtained by

$$\mathbb{T}_{com}^{req} = \underbrace{\sum_{i=1}^{3} C_i^{req}}_{\text{compute resource of vCU}} + \overbrace{\sum_{i=1}^{5} C_i^{req}}^{\text{compute resource of vDU}} \tag{5.1}$$

and

$$\mathbb{T}_{sto}^{req} = \underbrace{\sum_{i=1}^{3} S_i^{req}}_{\text{storage resource of vCU}} + \overbrace{\sum_{i=1}^{5} S_i^{req}}^{\text{storage resource of vDU}} \tag{5.2}$$

, respectively. The Hypervisor receives the order of the virtual compute and storage resources from the VIM and allocates these virtual resources based on the above two mathematical formulas. The required amount of virtual resources may vary from one type of RAN slice to another. Hence, the Hypervisor allocates different amount of virtual resources to different types of RAN slices.

It is worth noting that the relationship between the vDU and vCU of the same RAN slice is many-to-one. Therefore, $\sum_{i=1}^{3} C_i^{req} \geq \sum_{i=1}^{5} C_i^{req}$ and $\sum_{i=1}^{3} S_i^{req} \geq \sum_{i=1}^{5} S_i^{req}$, aimed at avoiding under-utilization of virtual compute and storage resources in I-PoP #1. Additionally, (de)coding and (de)modulation are the most compute and storage resource demanding functionalities in a RAN slice [68], which are processed in the upper layers of the NG-RAN architecture, namely in the vCU. Hence, the vCU must be allocated with sufficient virtual resources to process the lower layer functionalities of multiple vDUs.

Despite the above metrics, the compute and storage resource requirements of the vCU and vDU of a RAN slice are also directly proportional to the modulation and coding schemes (MCSs), frequency of the CPU of the host machine, and the allocated resource blocks (RBs) in both upstream and downstream directions, which is determined by an empirical model, described in the following quadratic polynomial:

$$\mathbb{T} = \frac{C_{\text{exp}} \cdot RB}{f_{\text{CPU}}} \times \sum_{k=0}^{2} \left(\alpha_{n,\text{DL},k} \cdot i_{s,\text{DL}}^{k} + \alpha_{n,\text{UL},k} \cdot i_{s,\text{UL}}^{k} \right) \tag{5.3}$$

where $\mathbb{T}$ indicates the virtual compute and storage resource requirements of the vCU or vDU of the requested RAN slice, C_{exp} is the computational capacity of the host machine, f_{CPU} is the frequency of the CPU of the host machine, RB_s is the number of the allocated RBs to the corresponding RAN slice, $i_{s,\text{DL}}^{k}$ is the MCS index in downlink, $i_{s,\text{UL}}^{k}$ is the MCS index in uplink, $\alpha_{n,\text{DL},k}$ is the fitting coefficient of the polynomial in downlink, and $\alpha_{n,\text{UL},k}$ is the fitting coefficient of the polynomial in uplink [68].

5.1.2 Mapping the Internal and External VLs

After the VNFCs of the vCU and vDU are modelled, we continue with the modelling of the VLs, which interconnect the VNFCs of the requested RAN slice subnet. These VLs of the RAN slice subnet are activated after the VNFCs are instantiated on their respective activated-PMs by the NFVO in the underlying infrastructure.

We let $\mathcal{L} = \{l_1, l_2, ..., l_{j\text{-}1}, l_j\}$ be a set of j such that $\forall j \in \mathbb{R} : > 0$ internal and external VLs of the requested RAN slice subnet. The internal and external VLs related to the vCU of the requested RAN slice are denoted as $\mathcal{L}_{\text{vCU}} = \{l_1, l_2, l_3, l_4, l_5, l_{\text{F1}}\}$. We denote the internal and external VLs of the vDU of the requested RAN slice as $\mathcal{L}_{\text{vDU}} = \{l_1, l_2, l_3, l_4, l_5, l_6, l_{\text{Fx}}\}$. The remaining links of the RAN slice are the PLs, which are connecting the internal functionalities of the RU. They are beyond the scope of this thesis.
Each (whether internal or external) VL of the requested RAN slice is anticipated to be hosted by a single VN in its respective I-PoP. As a result, the vCU requires six VN and the vDU requires seven VNs. Each VL demands a fixed amount of virtual networking resources (bandwidth), which is denoted as B_{vl}^{req}. In the light of this, the total required virtual bandwidth of the vCU and the vDU are obtained by

$$\mathbb{T}_{vCU}^{req} = 6xB_{vl}^{req} \tag{5.4}$$

and

$$\mathbb{T}_{vDU}^{req} = 7xB_{vl}^{req} \tag{5.5}$$

, respectively. It is worth noting that the relation between the vCU and vDU of the same RAN slice is one-to-many, therefore, the $\mathbb{T}_{vCU}^{req}$ is always greater than $\mathbb{T}_{vDU}^{req}$. During the resource configuration and allocation phases, such an in-equation must be taken into consideration in order to avoid under-and over virtual resource utilization in I-PoPs. Nevertheless, during the mapping process of the requested RAN slice, multiple internal and external VLs can be deployed on a single PL as long as, for each kind of virtual networking resource, the total bandwidth utilization of VLs (internal or external) does not exceed the total bandwidth of the respective PL.
To avoid service disruption and improve survivability, redundant VLs may also be created in advance in each of the I-PoPs and allocated if the existing VL between the VNFCs fails aimed at quickly recovering of the requested RAN slice [124]. These VLs may be activated after a failure has been detected in the existing VL (either internal or external) of at least two VNFCs (of the same or different VNFs). Since the majority of the data traffic in an I-PoP is consumed between the VM allocated for the VNFCs of their respective VNFs. As a result, we believe it is extremely important to create redundant VLs only for those VNFCs of the vCU and vDU of the requested RAN slice that are critical and may not tolerate performance degradation in order to enable flexible and efficient resource management of the underlying infrastructure in the NG-RAN architecture.
It is not necessarily important to create a redundant VL for each of the VL of the requested RAN slice. To that aim, the NFVO must calculate four metrics before deciding to allocate a redundant VL to a failed VL: (a) the quantity of bandwidth allocated for the existing VL, (b) the quantity of the bandwidth reserved for the redundant VL, (c) the total quantity of bandwidth allocated for active VL on the PL, and (d) the total quantity of bandwidth allocated for redundant VL on the PL. To manage the underlying resources efficiently, we thus keep the number of reserved list as minimum as possible. Hence, the following mathematical formula is used to calculate such a number in an I-PoP:

$$\min \sum_{l \in L} \hat{f}_l \tag{5.6}$$

, where $l \in L$ is the VL between the two VNFCs of similar or different VNFs of the requested RAN slice and $\hat{f}_l \geq 0$ is the total quantity of bandwidth reserved on VL $l \in L$ for traffic that has been failed over after its respective PM has failed [124].

5.1.3 Joint Mapping of the vCU, vDU, and Their Internal and External VLs

The mapping of a RAN slice is known to be an NP-hard optimization problem. To efficiently map both types of virtual components of the requested RAN slice, we believe it is essential to study their joint mapping onto the underlying infrastructure in NG-RAN architecture. In this thesis, we provide a unified, autonomous, and ML-assisted solution to the joint mapping of both VNFs and VLs problems of the requested RAN slice. We believe it is critical to note that the VNFs must be mapped onto the underlying infrastructure in the first stage. In the second stage, the internal and external VLs are mapped onto the underlying physical networking infrastructure, respectively.

To that aim, the available bandwidth between the PMs that host the VNFCs needs to be tackled first and foremost. The VNFCs of a VNF maybe hosted by the same PM or different PMs in an I-PoP. In the former case, if the VNFCs of either vCU or vDU are hosted by the same PM then the bandwidth of the internal VLs among the VNFCs can be infinite. However, if the host PM is unavailable, then the internal VLs as well as the hosted VNFCs will not be able to provide their respective functionalities in a RAN slice. In the latter case, the VNFCs hosted on peer PMs will consume a large amount of bandwidth. In either case, we believe it is extremely important to have backup VLs and VNFCs for critical VLs and VNFCs, respectively. We assume the former use case scenario for any type of the requested RAN slice in this thesis.

In order to proceed with the joint mapping of both types of virtual components, the VNFCs of the vCU and vDU of the requested RAN slice must be sorted in a descending manner. It specifically means that the RRC should be the first VNFC and the PHY-high should be the last VNFC in the list of VNFCs after being sorted. Subsequently, the internal VL #1 of the vCU should be the first VL and Fx of the vDU should be the last VL in the list of internal and external VLs after being sorted. Both lists of VLs and VNFCs are sorted according to the priority of the service provider and the ability of the network operator. Once these lists have been prepared, each I-PoP continues with the placing of the virtual components onto their respective physical counterparts.

At the start of a joint mapping, the VIM must search for an available PM in I-PoP #1 to host the VNFCs of the vCU. In parallel, the VIM must also search in I-PoP #2 for an available PM in order to host the VNFCs of the vDU. Furthermore, the VIM must also find available external VLs on Middle-haul and Front-haul in order to host the F1 and Fx, respectively. If the available virtual compute, storage, and networking resources for the vCU and vDU have been found on the underlying infrastructure in the NG-RAN architecture, the VIM places these logical components onto their corresponding virtual components and the virtual components are placed onto their physical components. With this, the joint mapping of the requested RAN slice is completed.

For the sake of argument, if we assume that the VIM is unable to find available virtual compute, storage, and networking resources in an I-PoP. It may search for an I-PoP that is located in close proximity in terms of geographical location to the primary I-PoP. If the VIM could not able to find the closest I-PoP, it searches for other I-PoPs in the underlying infrastructure until the available virtual resources are found. The VNFs and their internal and external VLs of the requested RAN slice may be hosted in such an I-PoP temporarily. In parallel, while the RAN slice is being hosted, the VIM continues searching for the virtual resources in the I-PoP that received the virtual resource allocation request at the beginning. Once the virtual resources are found in the primary I-PoP, the VIM migrates the VNFs and VLs of the hosted RAN slice to the primary I-PoP. This will lead to two critical advantages:

(a) the service request of the requested RAN slice will not be rejected by the underlying infrastructure; and (b) the virtual resources in the I-PoP that temporarily hosts the requested RAN slice will be efficiently managed.
The VIM must repeat this loop of searching for available virtual resources for every single RAN slice request if the virtual resources have not been found in the primary I-PoP. With this loop, the I-PoP also learns (using relevant ML-assisted algorithms) about the available, reserved, and allocated virtual resources. In addition, the VIM learns the best possible way in order to temporarily and permanently host the virtual components of the requested RAN slice. Lastly, it also finds an efficient VM migration solution in order to intelligently migrate the VMs related to vCU and vDU from a temporary host I-PoP to a primary host I-PoP. Likewise, it also learns to migrate VNs related to internal and external VL of the vCU and vDU of the requested RAN slice.
Furthermore, if the request of a RAN slice is accepted by the NFVO, the joint mapping of both the VNFs and VLs related to the accepted RAN slice obtain a specific amount of revenue to the network operator. In reference [180], the authors have mathematically defined such a revenue for an I-PoP at a given time slot t, which is as follows:

$$R_t(r) = \sum_{v \in V_r} \theta C_{v,r} + \sum_{e \in E_r} \beta B_{e,r} \tag{5.7}$$

where $R_t(r)$ is the total revenue of a RAN slice obtained at a give time slot t, $v \in V_r$ is the node that host the VNFCs, θ is the unit cost for a VM, $C_{v,r}$ is a specific number of VMs which are created by the underlying node, β is the unit cost for network bandwidth, $B_{e,r}$ is bandwidth of a link that is used to transmit data between two nodes. Nevertheless, an I-PoP can host a large number of RAN slices at a given time slot t. If we denote such a set of RAN slices by $\mathcal{G}_t$, then the total revenue of the network operator from hosting such RAN slices is defined by the following mathematical equation.

$$R_t(\mathcal{G}_t) = \sum_{r \in \mathcal{G}_t} \left(\sum_{v \in V_r} \theta C_{v,r} + \sum_{e \in E_r} \beta B_{e,r} \right). \tag{5.8}$$

In order for a network operator to maximize the total obtained revenue and at the same time efficiently utilize both virtual networking as well as virtual storage and compute resources, the virtual resource utilization and an average revenue must be mathematically defined at the first place. The former has been extensively studied for both VNFs and VLs of a RAN slice, while the latter can be obtained from the following equation [180]:

$$R = \lim_{T \to \infty} \frac{\sum_{t=1}^{T} R_t(\mathcal{G}_t)}{T} \tag{5.9}$$

where R is denoted as average revenue of a network operator from an I-PoP. Since the VNFs of RAN slices are hosted by two I-PoPs, therefore, the average revenue for the entire underlying virtual infrastructure can be obtained as follows:

$$R_v = 2 \lim_{T \to \infty} \frac{\sum_{t=1}^{T} R_t(\mathcal{G}_t)}{T}. \tag{5.10}$$

5.1.4 Mapping the VNFFG of the RAN Slice

Each requested RAN slice has a unique VNFFG, which defines the number of VNFs, the number of internal and external VLs, the amount of the required virtual resources, and the

flow of data traffic (through VNFCs and VLs) from source to destination in both downstream and upstream directions. These components of the requested RAN slice might be configured and instantiated based on the type (i.e., SST) of the requested RAN slice and the number of the supported end-users. Once both the vCU and vDU of the requested RAN slice is deployed on a PM that has available virtual resources, the subsequent VNF must also be deployed and QoS requirements must be met in order to successfully established a VL among them. Then, the traffic must be able to flow in both upstream and downstream directions. As a result, the VNFFG of the requested RAN slice is successfully mapped onto the underlying infrastructure in the NG-RAN architecture and the data traffic is flown based on the routing requirements specified in the RAN NSST.
As discussed in the previous chapter, as of this writing, there are three types of RAN slices standardized by the relevant SDOs, each must have its own customized VNFFG. They are the eMBB type VNFFG, URLLC type VNFFG, and mMTC type VNFFG. In the eMBB type VNFFG, those VNFCs must be customized and allocated with more virtual resources, which are bandwidth sensitive. In the URLLC type VNFFG, those functionalities must be prioritized and provided with enough virtual resources that are delay sensitive in order to support the requested services. Lastly, in the mMTC type VNFFG only those VNFCs must be prioritized, which are customized mostly for supporting a higher number of end-users and the requested traffic is also periodic.

5.2 Problem Formulation

In this section, we provide several conditions and assumptions that are reasonable to simplify the mapping problem of the vCU and vDU, as well as their internal and external VLs, of the requested RAN slice subnet onto I-PoPs in the NG-RAN architecture. In order to accomplish this, we will first mathematically define the mapping problem of both the VNFs as well as the VLs, respectively. Following that, we will present the main objectives, from a mathematical point of view, of the proposed architectural solution for mapping the VNFs and VLs of the requested RAN slice. Lastly, we will mathematically define the major constraints that limit the proposed mapping solution.

5.2.1 Definition

This subsection defines the optimization of the mapping problem of a RAN slice. To that end, it splits the mapping problem into the VM and VL placement sub-problems. Both of these sub-problems are covered in the following, respectively.

The VM Placement Problem Definition

Let $\mathcal{V} = \{v_1, v_2, ..., v_{m\text{-}1}, v_m\}$ be a set of m such that $\forall m \in \mathbb{R} : > 0$ VMs in the Hypervisor domain of an I-PoP and $\mathcal{P} = \{p_1, p_2, ..., p_{n\text{-}1}, p_n\}$ be a set of n such that $\forall n \in \mathbb{R} : > 0$ PMs in the compute domain of the same I-PoP. The $\mathcal{V}$ is abstracted from the $\mathcal{P}$ in both I-PoP #1 and I-PoP #2 in the NG-RAN architecture.
Each VM $j \in \mathcal{V}$ is characterized by a vector of available virtual resources. It specifically means that each VM $j \in \mathcal{V}$ has a maximum virtual compute capacity, denoted by C_j^{max}, and a maximum virtual storage capacity, denoted by S_j^{max}. The former is the virtual resource capacity to compute the VNFC $i \in \mathcal{F}$ and the latter is the virtual resource capacity to store information related to the VNFC $i \in \mathcal{F}$. We consider that the VNFC $i \in \mathcal{F}$ hosted on the PM $j \in \mathcal{V}$, if activated, demands a certain amount of C_j^{max} and S_j^{max}. Such a required

amount of the virtual compute and storage resources of the VNFC $i \in \mathcal{F}$ are denoted by C_i^{req} and S_i^{req}, respectively.
Likewise, each PM $k \in \mathcal{P}$ in an I-PoP has a maximum physical compute capacity, denoted by C_k^{max}, and a maximum physical storage capacity, denoted by S_k^{max}. The former is the physical resource capacity to compute and the latter is the physical resource capacity to store the VM $j \in \mathcal{V}$ at a given time in an I-PoP. If we assume for the sake of argument that $k \in \mathcal{P}$ hosts only a single VM at a given time, the C_k^{max} and S_k^{max} are virtualized and then fully dedicated to the C_j^{max} and S_j^{max}, respectively. On the contrary, if $k \in \mathcal{P}$ hosts more than a single VM at a given time in an I-PoP, the C_k^{max} and S_k^{max} are virtualized and then dynamically shared – based on an optimal resource allocation mechanism – between the C_j^{max} and S_j^{max} of the hosted VMs.
We proceed to define a variable, which represents the placement of VM(s) onto PM(s) for both vCU and vDU in I-PoP and I-PoP, respectively. To accomplish this, let's us assume that $X_{m,n}$ is a matrix representation to the VM placement problem in both of the I-PoPs, defined as follows:

$$X_{m,n} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n-1} & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n-1} & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{m-1,1} & x_{m-1,2} & \cdots & x_{m-1,n-1} & x_{m-1,n} \\ x_{m,1} & x_{m,2} & \cdots & x_{m,n-1} & x_{m,n} \end{bmatrix}$$

where $x_{j,k}$ is the matrix variable with a double subscript notation that indicates the position of an element in $X_{m,n}$. The first subscript (j) refers to the row that represents the position of a VM in a list of m VMs. The second subscript (k) refers to the column that represents the position of a PM in a list of n PMs. This applies to both vCU and vDU in I-PoP #1 and I-PoP #2, respectively.
In such a case, if a single VM belong to the VNFC (of either vCU and vDU in I-PoP #1 or I-PoP #2) is placed onto its respective single PM, the VM placement problem can be defined as follows:

$$\sum_{k=1}^{n} x_{j,k} = 1, \quad \forall j : 1 \leq j \leq m \tag{5.11}$$

where $x_{j,k} \in \{0, 1\}, 1 \leq j \leq m, 1 \leq k \leq n$ is 1 if VM j is placed on PM k in the same I-PoP, and 0 otherwise. However, if multiple VMs are assigned to a single PM in the same I-PoP, the placement problem is presented in the following:

$$\sum_{j=1}^{m} x_{j,k} \leq t_k, \quad \forall k : 1 \leq k \leq n \tag{5.12}$$

where t_k is the maximum number of VMs, PM k can host in an I-PoP. Hence, the placement of $j \in \mathcal{V}$ onto $k \in \mathcal{P}$ can be presented as a mapping function $f : \mathcal{V} \to \mathcal{P}$ in such a way that the optimization objectives given in Subsection 5.2.2 are achieved and the constrains discussed in Subsection 5.2.3 are satisfied. Based on this statement, we formulate the VM placement problem of the vCU and vDU of the requested RAN slice onto VNFCs in both of the I-PoPs as follows:

$$x_{j,k} = \begin{cases} 1, & \text{if VM } j \text{ is placed onto PM } k \\ 0, & \text{otherwise } \forall k \in \mathcal{P}, \forall j \in \mathcal{V}. \end{cases} \tag{5.13}$$

In both of the above cases, the virtual compute resource demand of VM(s) must not exceed the physical compute resource capacity of the host PM:

$$\sum_{j=1}^{m} C_j^{max} x_{j,k} \leq C_k^{max} y_k, \quad \forall k : 1 \leq k \leq n \tag{5.14}$$

where C_j^{max} is the virtual compute resources demand by VM j, C_k^{max} is the physical compute capacity offered by PM k, and y_k is 1 if PM k is active and 0 otherwise. Likewise, the virtual storage resource demand of VM(s) must not exceed the physical storage resource capacity of the host PM:

$$\sum_{j=1}^{m} S_j^{max} x_{j,k} \leq S_k^{max} y_k, \quad \forall k : 1 \leq k \leq n \tag{5.15}$$

where S_j^{max} is the virtual storage resources demand by VM j in an I-PoP, S_k^{max} is the physical storage capacity offered by PM k in the same I-PoP, and y_k is 1 if PM k is active and 0 otherwise.

Defining the virtual compute and storage workloads of VM(s) belong to the VNFCs of the requested RAN slice are vital metrics in the efficient placement of the VMs and the partitioning of the physical compute and storage resource of a PM in an I-PoP, which is defined in the following mathematical equation [181]:

$$W_j = \frac{1}{1 - C_j^{max}} \times \frac{1}{1 - S_j^{max}} \tag{5.16}$$

where the W_j is the total workload, C_j^{max} is the virtual compute demand, and S_j^{max} is the virtual storage demand of VM $j \in \mathcal{V}$. It is worth noting that the virtual compute and storage workloads of a VM belong to a specific VNFC of the vCU and vDU of the requested RAN slice must not exceed (or overloaded) over 100%. Hence, the $C_j^{max} < 1$ and the $S_j^{max} < 1$. Based on (5.16), the workloads of the vCU and vDU of a RAN slice on I-PoP #1 and I-PoP #2 are defined as follows:

$$W_{RAN} = 3 \cdot (W_j) + 5 \cdot (W_j). \tag{5.17}$$

Likewise, defining the workloads of the PM(s) in an I-PoP for either of the VNFs of the requested RAN slice, after placing the VM(s), is also a vital metric for optimal resource allocation and efficient management of the physical resources of the respective I-PoP, which is defined as follows:

$$W_k = \frac{1}{1 - \left(C_k^{max} + \sum_{j=1}^{m} C_j^{max} \right)} \times \frac{1}{1 - \left(S_k^{max} + \sum_{j=1}^{m} S_j^{max} \right)} \tag{5.18}$$

where W_k is the total workload, C_k^{max} is the current state of the compute resource workload, and S_k^{max} is the current state of the storage resource workload of PM $k \in \mathcal{P}$ in an I-PoP. The $\sum_{j=1}^{m} C_j^{max}$ is the the total compute resource utilization and $\sum_{j=1}^{m} S_j^{max}$ is the total storage resource utilization of m VMs hosted by $k \in \mathcal{P}$ PM in its respective I-PoP. It should be noted that the total compute resource load and storage resource load of PM k in an I-PoP must not exceed (or overload) over 100% of the total available resources. Hence, the $C_k^{max} + \sum_{j=1}^{m} C_j^{max} < 1$ and $S_k^{max} + \sum_{j=1}^{m} S_j^{max} < 1$.

Defining the workloads of $j \in \mathcal{V}$ before placing onto $k \in \mathcal{P}$ is also extremely critical to avoid the virtual resource wastage of a PM and the total virtual resource wastage of an I-PoP. To efficiently accomplish this, we must define the virtual resource wastage of PM $k \in \mathcal{P}$ in an I-PoP using the following mathematical formula:

$$PF_k = w_1 \times \frac{C_k^{ava}}{C_k^{cap}} + w_2 \times \frac{S_k^{ava}}{S_k^{cap}} \tag{5.19}$$

such that w_1 and w_2 are weighting coefficients, C_k^{ava} are the available compute resources and S_k^{ava} are the available storage resources on $k \in \mathcal{P}$ in an I-PoP after the requested resources of the RAN slice are assigned to its respective VMs, and C_k^{cap} is the compute resource capacity and S_k^{cap} is the storage resource capacity of the $k \in \mathcal{P}$ in an I-PoP. Likewise, the resource wastage of VM $j \in \mathcal{V}$ assigned for a VNFC while taken PF_k into account is calculated using the following equation:

$$VF_j = w_1 \times \frac{C_j^{ava}}{C_j^{cap}} + w_2 \times \frac{S_j^{ava}}{S_j^{cap}} + PF_k \cdot j \tag{5.20}$$

such that C_j^{ava} are the available virtual compute resources and S_j^{ava} are the available virtual storage resources after $j \in \mathcal{V}$ hosts its respective VNFC $i \in \mathcal{F}$, and C_j^{cap} is the virtual compute resource capacity and S_j^{cap} is the virtual storage resource capacity of the $j \in \mathcal{V}$. These statements are valid for both the vCU and vDU of the requested RAN slice in the underlying infrastructure in NG-RAN architecture.
At this stage, with these procedures and mathematical equations and analysis of the mapping process, we assume that we have mathematically formulated the required amount of the virtual resources of the VNFCs and have also mathematically formulated the available virtual resources of the VMs that host these VNFCs in both I-PoPs. With this, we assume that we have successfully mapped the VNFCs of the requested RAN slice onto the underlying VMs. In the next stage, we must mathematically formulate the virtual networking resources for both VNs in the underlying I-PoPs and the internal and external VLs of the logical components of the requested RAN slice to map them efficiently throughout its life time. We provide this discussion in the next section.

The VL Placement Problem Definition

We Let $\mathcal{N} = \{n_1, n_2, ..., n_{r\text{-}1}, n_r\}$ be a set of r such that $\forall r \in \mathbb{R} : > 0$ the internal and external VNs in the Hypervisor domain of an I-PoP and $\mathcal{M} = \{m_1, m_2, ..., m_{s\text{-}1}, m_s\}$ be a set of s such that $\forall s \in \mathbb{R} : > 0$ PLs in the networking domain of the same I-PoP as well as in the transportation network that is connected with the host I-PoP. The $\mathcal{N}$ is abstracted from the $\mathcal{M}$ in both I-PoP #1 and I-PoP #2 in the NG-RAN architecture.
The $\mathcal{N}$ is divided into two parts: $\mathcal{N}_1$ and $\mathcal{N}_2$. The former contains the VNs that are assumed to host internal VLs of a VNF and the latter comprises of VNs that are used to host external VLs of a VNF. Since, we assumed that the internal VL of a VNF are hosted on the same PM, hence, we skip the $\mathcal{N}_1$. For simplicity, we assume the $\mathcal{N}_2$ as $\mathcal{N}$ in this thesis.
Likewise, $\mathcal{M}$ is also contains two subsets: $\mathcal{M}_1$ and $\mathcal{M}_2$. The former subset contains the PLs that are assumed to host the internal VLs of the vCU and vDU. The latter is a subset of PLs that are allocated to the external VLs of both of the VNFs of the requested RAN slice. In the previous section, we assumed that the internal VLs of a VNF are hosted by the same PM which hosts the VNFCs, hence, we skip the $\mathcal{M}_1$ and focus on $\mathcal{M}_2$ in this thesis. For the sake of simplicity, we assume $\mathcal{M}_2$ as $\mathcal{M}$ in this thesis.

Each VN $r \in \mathcal{N}$ has a predefined available virtual networking resources, denoted by N_r^{max}, which are used to process its respective VL of a VNF of the requested RAN slice. We assume that if VN $r \in \mathcal{N}$ hosted on PL $s \in \mathcal{M}$, activated, demands a certain amount of N_r^{max}. Such a required amount of virtual networking resources of a VN is denoted by the N_r^{req}. Likewise, each PL $s \in \mathcal{M}$ in an I-PoP has a maximum physical networking bandwidth, which is denoted by N_s^{max}. If we assume for the sake of argument that N_s^{max} hosts only a single VN at a given time, the N_s^{max} is virtualized using Hypervisor and then fully dedicated to the respective VN in an I-PoP. Nevertheless, if the PL $s \in \mathcal{M}$ hosts more than a single VN at a given time in an I-PoP, the N_s^{max} is first virtualized and subsequently shared – based on a dynamic virtual networking resource allocation algorithm – between the N_r^{max} of all hosted VNs of vCU and vDU in both I-PoPs.
Based on the above information, the required amount of the internal and external VLs' networking resources of the vCU and vDU of the requested RAN slice are 6X N_r^{max} (five for internal VLs and one for F1) and 7XN_r^{max} (six for intenral VLs and one for Fx), respectively. Since we assumed that a single vCU hosts multiple vDUs, therefore, the 6X $N_r^{max} > 7\text{X}N_r^{max}$. Likewise, the F1 external VL (i.e., Middlehaul) must also be also be allocated with enough bandwidth in order to host multiple Fx external VLs (i.e., Fronthaul) in the NG-RAN architecture.

5.2.2 Main Objectives

In the process of mapping the VNFs and VLs of the requested RAN slice, we set a number of main performance objectives that are expected to be minimized, maximized, or balanced throughout its lifetime. These performance objectives might or might not be in conflict with one another at a given time. In any of these two use case scenarios, the proposed mapping solution is expected to provide an optimal solution to the performance objectives we are attempting to achieve. Additionally, the mapping process (of both components, the VLs and VNFs of a RAN slice) can be a single-objective or multiple-objectives optimization problems. In the former scenario, a single performance objective can be optimized at a given time interval, while in the latter scenario, multiple performance objectives can be optimized during a specified time interval.
In light of the above, we provide a list of common performance objectives related to the VNFs and VLs of the requested RAN slice, along with their mathematical descriptions, that are expected to be maximized, minimized, or balanced by the proposed mapping solution. In addition to the following list, we arranged a complete list of performance objectives in the previous chapter. These performance objectives can be optimized in order to evaluate the performance of a successful mapping solution of a RAN slice.

The Number of PMs

One of the most common performance objectives that is intended to be optimized is the minimization of the number of active PMs that host the VMs of a RAN slice in an I-PoP. While minimizing the number of active PMs, the power consumption of the hosted RAN slice is also minimized in an I-PoP. This led to a reduction in total power consumption of the underlying infrastructure. This performance objective of the requested RAN slice can be expressed mathematically as follows:

$$minimize \sum_{k=1}^{m} y_k \quad \forall j : 1 \leq j \leq n \tag{5.21}$$

where y_k is 1 if PM k is active and 0 otherwise. In addition to reducing power consumption, the minimization of active PMs can also lead to avoiding under-utilization and over-utilization of virtual resources in an I-PoP and minimizing resource wastage (at the VM, PM, and I-PoP levels). For example, if we assume that each VM of a vCU is hosted by a single PM. Then, there is a need for five active PMs in I-PoP #1. However, all five active PMs may not be fully utilized by the respective VNFCs of the vCU. In order to better utilize the underlying active PMs, there may be a need to activate only one PM for hosting the entire vCU. Hence, the rest of the four PMs may be turned off.

Link Utilization

One of the most common performance objectives that can be optimized is to minimize the total bandwidth (i.e., networking resources) consumption of the underlying PLs in a transportation network. To accomplish this goal, first and foremost, the required virtual networking resources of the internal and external VLs of the requested RAN slice must be predicted using certain ML-assisted algorithms. Following that, the PL must be partitioned in such a way that the total bandwidth of the VLs of the requested RAN slice does not exceed the underlying PL's total bandwidth. Based on the prediction, VNs should be allocated to those VLs where the quantity of the virtual resources of the VN is equal to or greater than those of the hosted VL. Finally, the VL of the RAN slice must be mapped onto the most suitable PL in the transportation network.

VM and VN Migrations

If a VM or a VN of the vCU or vDU is hosted on a physically over-utilized node or link in an I-PoP, the Hypervisor must migrate it to a physically under-utilized node or link in the same or different I-PoP. Such a migration of VM or VN consumes power, complicates the management and orchestration of virtual resources, and places an additional burden on the VIM to manage the life cycle of the virtual resources of the requested RAN slice, among other tasks. Reducing the number of VM and VN migrations from an over-loaded PM and PL to a less-loaded PM and PL is another common objective that can be achieved by the proposed unified functioning architecture. To accomplish this goal, a real-time ML-assisted migration algorithm is needed that can predict the future resource demands of VMs or VNs based on their historical resource requirements and then migrate these virtual components.

Power Consumption

The power consumption reduction is a critical performance objective for an I-PoP. Among other factors, the energy consumption of an I-PoP can be increased primarily due to the high volume of communication traffic, the number of active PMs and PLs, and cooling equipment. In I-PoPs, the PMs and cooling systems consume most of the electricity, followed by storage devices and networking equipment. To reduce the consumption of such energy in an I-PoP, utilizing ML-assisted algorithms in these three areas can have a significant impact.

5.2.3 Constraints

In order to map the VNFCs and VLs of the vCU and vDU of the requested RAN slice, there are a number of intrinsic and use-case-specific constraints that limit the mapping process. These constraints must be stated prior to proposing the mapping solution for a requested RAN slice onto the underlying infrastructure. Some of the most critical constraints that

we believe have a significant impact on the quality of the performance objectives and the results of the mapping process of a RAN slice are discussed briefly in the previous chapter. Additionally, the performance objectives that we discussed in the previous chapter can also be used as constraints during the formulation of the mapping problem of the requested RAN slice. For instance, reducing the number of active PMs in an I-PoP can be a performance objective, while satisfying the inter-communication traffic among the VMs can be a constraint to the mapping process of a RAN slice. This example is also valid vice versa during the mapping problem formulation. Below we present two common constraints of the mapping problem of a RAN slice in the NG-RAN architecture.

Physical Node Resource Constraint

One of the most common constraints encountered during the mapping process of the requested RAN slice is the limitation of the underlying physical compute and storage resources of the host PM. To put this into argument, it is critical to state that the virtual compute and storage resource demands of the VNFC of a VNF must not exceed the virtual compute and storage resource capacity of the host VM in a PM. Likewise, the sum of the virtual compute and storage resource demands of all VMs that belong to the VNFCs of a VNF (either vCU or vDU) must not exceed the physical compute and storage resource capacity of the host PM in an I-PoP. This constraint can be applied to both I-PoPs and is mathematically described as follows:

$$\sum_{j=1}^{n} d_{rj} x_{ji} \leq c_{ri} y_i, \quad \forall i : 1 \leq i \leq m \tag{5.22}$$

such that d_{rj} is the r type virtual resource demand (i.e., virtual compute or storage) of VM j, cri is the capacity of r type physical resource (i.e., physical compute and storage) offered by i PM in an I-PoP, and yi equals to 1 if PM i is in active mode and 0 if it is in inactive mode.

Physical Link Capacity Constraint

The second most common constraint during the mapping process of a RAN slice is the limitation of the physical networking resources of the underlying transportation network in the NG-RAN architecture. To that end, the virtual networking resource demands of a VL should not exceed the virtual networking resource capacity of its host VN. Similarly, the sum of the virtual networking resource demands of the internal and external VLs of a VNF (either vCU or vDU) should not exceed the total available physical bandwidth of the underlying host PL in the transportation network. It should be noted that a VL can only be hosted by a single physical path in the underlying PL. This constraints can be applied to both I-PoPs and mathematically shown as follows [116]:

$$\forall l_s \in E_S : \tag{5.23}$$

$$u(l_s) = \sum_{i=1}^{r} \sum_{f_l \in F_l} \sum_{\iota_V \in f_l} \sum_{p \in P_M(l_V)} x^p(l_V) y^p(l_s) b(l_V) \tag{5.24}$$

$$\forall l_s \in E_S . u(l_s) \leq D(l_s) \tag{5.25}$$

where $D(l_s)$ is the bandwidth that a PL can provide, $x^p(l_v)$ is a binary variable (if a VL is successfully mapped onto PL, the value is 1 otherwise the value is 0), and $y^p(l_s)$ is also a binary variable (if the PL is along the physical path in the transportation network the value is 1 otherwise the value is 0), $b(l_V)$ is the bandwidth demand of a VL, $u(l_s)$ is the bandwidth that the PL has allocated, and $P_M(l_V)$ is a set of alternative physical paths on which the VL can be mapped in the transportation network.

5.3 Proposed Solution

In this section, we first present key concepts related to the utilization of ML in the field of mobile communication engineering. Next, we provide a detailed discussion of various categories of the ML algorithms, such as supervised learning, unsupervised learning, and reinforcement learning. Then, we focus on reinforcement learning and provide a detailed mathematical background and its application in the field of communication engineering. Lastly, we select Q-learning as the most appropriate ML algorithm to map the VNFs, as well as the internal and external VLs, of the requested RAN slice onto the underlying infrastructure in the NG-RAN architecture.

5.3.1 Machine Learning in Wireless Communication

In his paper [182], Arthur Samuel (an American computer scientist) defined ML, for the first time, as a sub-field of computer science that enables "computers to learn without being explicitly programmed" in 1959. This definition of ML is still widely accepted nearly seven decades later. ML algorithms are used in different fields, including computer science, telecommunication engineering, robotics, medicine, biology, chemistry, economics, and many others, where utilizing state-of-the-art algorithms cannot perform well enough to execute the required tasks of a given system. In each of these fields, there are numerous complex tasks that can be extremely difficult for a technician to perform manually or create the necessary algorithms using conventional methods. In the real world, assisting a machine in developing its own algorithms may prove to be more effective than having a technician specify each necessary step for performing such complex tasks. To that end, ML has two distinct goals: the first goal is to classify data using certain developed models, and the second goal is to forecast future outcomes using these models.

One of the most important characteristics of ML is the concept of "self-learning," which refers to the utilization of statistical modeling aimed at detecting patterns and improving the performance of a given system by using empirical information and relevant raw data. There are various types of input data in ML that are injected into a given system with different values, structure levels, and formats in order to obtain the required results. In the beginning, these different types of input data are fed to a machine. Then, from a set of already developed algorithms, a relevant algorithm is chosen to perform the task(s). Lastly, the parameters of the given system are configured and adjusted according to the assumed model, and the machine is instructed to perform analysis during a specified time interval. The machine then uses trial and error to decipher patterns found in the data that is currently under process. After analyzing data patterns, the model can be used by the machine to forecast future values for the task(s) of a given system.

For instance, ML can be used extensively in the field of medicine to assist physicians in diagnosing and treating their patients' diseases quickly and accurately. Let us assume that a large set of data has been obtained through clinical trials or from real-world patients in a hospital using a specific data collection mechanism. This data set may contain information about the age, gender, weight, height, symptoms, genetic information, blood group type, and other variables associated with a specific disease of a group of individuals. Once such a data set is loaded into a machine, the machine determines the relationship and identifies the patterns between the different variables of the data set. Next, these variables are efficiently analyzed in relation to the disease that is being modeled by the machine. Then, the appropriate ML-assisted algorithm is selected and its parameters are configured accordingly to obtain the results. The selected ML algorithm is also loaded into a machine that already contains

the input data. At this point, the machine begins training the algorithm for a specified time interval using the input data. Lastly, the machine makes the final forecasts through data modeling and self-learning of medical information and patients' history. The physicians use these predictions in order to diagnose and, furthermore, treat the diseases of the patients in a timely, accurate, efficient, and cost-effective manner.

Additionally, another real-world application of ML algorithms could be in the social media platforms offered by social networking sites, such as Facebook, Instagram, LinkedIn, Twitter, and many others, in order to improve their services for the benefit of their end-users and optimize (or automate) the execution of the operations and management of their own tasks. The algorithms available on a social networking site first collect limitless data related to the end-users' interactions, such as comments, likes, views, mutual friends, and others. Such data may be gathered only from those end-users who agree to participate in the data collection. The algorithms then analyze the collected data from the interactions of the end-users, spot behavioral patterns hidden behind the collected data, and transfer the collected data to knowledge. Based on such an analysis, the social networking sites recommend the most relevant content to their target end-users. In addition, the service providers can make data-driven and fact-based decisions as well as produce accurate predictions about those services that they offer to their end-users. Lastly, using ML algorithms, these social networking sites can also hide content from their end-users. For instance, if an end-user reports content (such as videos, images, text, etc.) and does not want them to appear on their profile, the ML-associated algorithms on the social networking site will record this behavior of the end-user. In addition to disappearing the reported content, all other content from the same instances will not be visible to the end-user in the future.

The preceding two examples determine the critical nature of the availability of the relevant data in the field of ML. They also demonstrate how critical it is to select the optimal ML-assisted algorithm for the task we are intending to optimize for a given system in a specific duration of time. There are thousands of ML-assisted algorithms and many others are being developed every single year. Each ML-assisted algorithm has its own application domain, which is designed to execute only those tasks that have the same configuration parameters, behavior, and environment. When deploying the right ML-assisted algorithm to the right task, it must have at least three distinct components: the representation, the evaluation, and the optimization. The representation component illustrates the mechanisms that show the representation of the relevant knowledge. Decision trees could be an example of the representation component. They represent (or visualize) the decisions and the decision-making process in order to make predictions about the value of a given variable. The evaluation component describes the mechanisms of the evaluation of the ML-assisted algorithm that is being considered to execute the task. For example, squared error can be one of the instances of the evaluation component of ML-assisted algorithm, which acts as the measure of the quality of an estimator. It is used to measure the average error square during the estimation of a variable. The optimization component describes the mechanisms by which the candidate ML-assisted algorithm is generated to optimize the performance objective of a specific task of a system. For instance, convex optimization could be one of the best examples of this component of a ML-assisted algorithm, which minimizes convex functions over convex sets of a given problem.

It can be observed that, in addition to other service sectors and vertical industries, ML-assisted algorithms can also be used in various aspects of wireless communication systems and the next-generation of mobile networks. TheseÂ next-generation of wireless communication networks must offer ultra-reliable, high-speed, high throughput, and low-latency communication services to end-user devices in an autonomous and intelligent manner. To accomplish these benefits,

the wireless communication network generates enormous amounts of data that must be collected and processed in real time and intelligently by an ML-assisted architectural and algorithmic solution. These various types of data, among other aspects, are related to the traffic data from end-users, sensors installed in various industrial users' equipment, and other service sector-related devices located in a third party. These requirements for wireless communication and core intelligence requirements can only be met by incorporating ML-assisted techniques into wireless infrastructure and end-user devices. These AI-driven or ML-assisted algorithms enable a network operator to facilitate resource orchestration and wireless network management efficiently, as well as assist in managing the growing number of communication, computation, and networking applications. Nonetheless, the application of ML-assisted algorithms in heterogeneous wireless networks continues to be a point of contention. Additional effort is required to bridge the divide between the fields of ML and wireless communication networks.

5.3.2 Training, Validation, and Testing Data Sets in ML

In the previous subsection, we have noticed that data is the most significant aspect of the AI-driven and ML-assisted algorithms in wireless communication networks. Large corporations are spending significant sums of money to amass as much specific data as possible. This data is critical to the network operator in order to improve the tasks related to the services and network. This data must be collected, converted into a unified format, manipulated, and interpreted in order to provide meaningful inferences and conclusions about the end-users. Lastly, the obtained conclusions (which is called information), experiences, learning, and insights are combined in order to give a machine, system, or network the ability to perform the tasks intelligently without/partial human involvement.
Each data set in ML has the following five properties:

1. **Volume:** It shows the scale of the data for a network operator. With the increasing number of user-devices, a vast amount of data is generated worldwide that contains information about numerous parameters associated with the profile and behavior of the vertical industry. This data requires appropriate tools and techniques of data collection, archiving, and data preparation while ensuring that the privacy of the end users is not violated. For instance, a network operator has 10 million active end-user devices. The network operator must handle the data associated with the profiles, behavior, preferences, and billing of such a large number of user devices every single minute, uploading petabytes of data into its database.

2. **Variety:** The data collected (structured, semi-structured, and unstructured) by the operator has a variety of formats (i.e., types). They are collected from a number of data sources, mainly including humans, machines, and applications. The most common forms of collected data may be voice, video, text, pictures, and so on. To efficiently manage the collected data, it is important that the operator have the ability to classify the raw data into multiple categories according to predefined classification rules. The collected data from multiple sources helps the operator analyze the tasks and subsequently make an intelligent, smarter, and more informed decision in a system. Therefore, clear, fast, and easy access to a variety of formats of data is significantly important in order to improve the efficiency and productivity of an ML-assisted system.

3. **Velocity:** The speed (a.k.a., how quickly) of data collection, data generation, data moving, data labeling, data distribution, and all operations in real-time related to the

collected data is one of the important properties of an ML-assisted algorithm. The main advantage of this aspect of an algorithm is that it helps a network operator process the data flow quickly, make the right predictions, and take the most accurate possible decision at the right time by using specific processing techniques. If data were collected and processed more quickly, it would become more valuable and retain its value for a longer period of time. Nonetheless, the network operator's data systems must be capable of performing all data-related tasks. The more quickly a user can integrate data into a network operator's data and analytics platform, the more flexibility end users will have when it comes to reports and other necessary information. On the basis of these facts, high-velocity data requires advanced processing tools, both at system and algorithmic levels, in order to reveal necessary information for better decision-making.

4. **Value:** Another critical feature of an ML-assisted algorithm is the value (i.e., meaningfulness or usefulness) of the collected data set. It should be useful and valuable enough to be used by the analytics system of the network operator in order to infer the right conclusions at the right time from it. Using customized data processing software can help network operators gain a thorough understanding of the collected data, derive valuable inferences, and add value to the network operator's decision-making process based on facts, trends, and more direct terms and numbers. On the basis of this, a network operator must first set several objectives prior to data collection aimed at defining the type of data the operator is attempting to collect. This brings significant value for the network operator and end users, who then use the data in a specific way in order to enable the owner of a vertical industry and the network operator to properly derive the highest amount of that value. The useful collection and proper utilization of the collected data helps the network operator to minimize the cost, time, and resources on the one hand, and on the other hand, maximize the efficiency of the network operator's analytics system. Lastly, the value of the data depends on two outputs: the ability to generate flow and the ability to solve a specific problem. The former determines the basis on which the data should be generated, while the latter aspect specifies the efficiency of the problem-solving of the collected data. Both of these outputs are equally important to the network operator.

5. **Veracity:** The last but one of the important characteristics of the collected data set is its veracity, which is used to define the accuracy and truthfulness of the collected data set. However, in the field of ML, the veracity (or the reliable and significant) is significant not just for the data's quality but also for the trustworthiness of the data's source, type, and processing by the respective model. To improve the veracity of the data and produce actionable and relevant results, researchers typically remove biases, noise, inconsistencies, abnormalities, and duplication from the collected data. To that aim, when developing a data strategy for an ML-assisted algorithm, the operator should involve an expert team in efforts to keep the collected data clean and put processes in place to prevent the accumulation of "noisy data" in the analytics systems. Hence, in the data analyzing phase, the operator should again ensure that when, how, and where the data has been collected. In the field of ML, low-veracity data typically includes a high proportion of unimportant, "noisy," and meaningless data that is useless for the data analytics system of the operator. However, high-veracity data includes a large number of records that are extremely valuable for the analytics system of the operator, and it significantly contributes to the overall results and performance of the model.

The aforementioned five "Vs" represent the five most fundamental and inherent characteristics of ML in wireless communication. Understanding these five "Vs" enables network operators

to extract more value from the collected data, produce accurate predictions, efficiently communicate and articulate the critical characteristics of the collected data set, and take the right decisions. If such a large amount of data related to the behavior and profile of the end users has been collected effectively and purposefully, it enables the operator to make more informed decisions about overall operation efficiency, profitability, and customer satisfaction. We assume that the network operator's analytics system has captured all five "Vs" and is ready to derive the desired objectives systematically. Additionally, the network analytics system has the appropriate software and tools in order to proceed with further processes within an acceptable time for both parties. Prior to proceedings with more process, the network operators typically divide the collected data into three sets: the training data set, the validation data set, and the testing data set. They are discussed in the following.

Training Data Set

The primary collected data set that is utilized to train the ML model is referred to the "training data set," which is used during the learning process. The training data set is used to teach the ML-assisted algorithms to produce predictions, recognize the hidden features and patterns in data, and execute a desired task using the training data sets. The training data set should contain a diverse set of inputs to ensure that the ML-assisted model is trained in all possible use case scenarios and is capable of predicting any previously unseen data sample. In the field of ML, the training of a model is processed in three steps: the feeding step, the tagging step, and the testing step. In the feeding step, the input data is fed to the machine learning model. In the tagging step, the model converts the training data set into numbers (a.k.a., text vectors) that must represent the features of the data. In the testing step, the model is trained in order to associate the text vectors with the tags on the basis of tagged samples. Once the model has been trained, it uses the training data set throughout its lifetime, aimed at building an efficient, high-performing, and accurate ML-assisted solution.

Validation Data Set

During the training period, a specific data set is created (out of the training data set) and is used to validate the performance of the ML-assisted model. This data set is called the validation data set, which is different from the training data set. The validation procedure of the collected data set provides data that enables the analytics system of the network operator to tune the hyperparameters and configurations of the model appropriately, as well as provide an unbiased evaluation and final estimated structure of the model. The validation of the data set is significant to the system analytics, telling the administrator whether the training of the model is on track or not. Throughout the validation process, the ML-assisted model is trained utilizing the primary training data set, while the validation of the model is evaluated using the validation data set in every single epoch. The grand objective of dividing the primary data set into validation data sets is to avoid over-fitting in the model. Over-fitting is a situation where the model is performing well at categorizing the samples in the data set. However, it is unable to produce accurate and valuable classifications of the collected data. Lastly, during the validation (evaluation) of the algorithm, the validation data sample is typically obtained randomly and must not be used to fine-tune the model.

Testing Data Set

The last type of data set in the field of ML is the testing data set, which is completely separate from the training and validation data sets. This data set is used by the model after

the training and validation processes have been successfully completed. The testing procedure provides a final and unbiased ML-assisted model from the accuracy, thrust-worthiness, and precision perspectives. The quality of the training data set is critical for the performance of the ML-assisted model. If we assume that the performance of the model is not satisfactory, then it will not perform well during the deployment phase. In addition, the ML-assisted models are highly sensitive to the training data sets. Hence, even a minor error might result in a significant performance degradation during the implementation. Therefore, care must be taken during the training phase of the model, aiming to produce a highly accurate model. It should be noted that there are some misunderstandings between the validation data set and the testing data set. The former tunes the model, while the latter confirms that the mode is performing well. That will give the administrator the confidence to deploy it in the real world.

5.3.3 The Types of ML-assisted Algorithms

The field of ML is composed of hundreds of statistical algorithms. Selecting the optimal ML-assisted algorithm (or a combination of algorithms) for a given task is a continuing problem for researchers. However, before delving into individual algorithms, it is necessary to grasp the four broad categories of ML-assisted algorithms: supervised learning-assisted algorithms, unsupervised learning-assisted algorithms, semi-supervised learning-assisted algorithms, and reinforcement learning-assisted algorithms. These four high-level categories of ML algorithms are discussed in the following subsections, respectively.

5.3.4 Supervised Learning

The supervised learning algorithms are the most frequently used and well-known algorithms in the field of ML. They are also the simplest algorithms to comprehend (at both theoretical and application levels) and deploy in the real world. In supervised learning, the algorithms are trained using labeled data. While accurate labeling of data is extremely important for these types of algorithms to perform the task, if utilized appropriately, they are the most powerful algorithms in the field of ML.

In supervised learning, at the start, the ML algorithm is provided with a small training data set in order to perform the task during the training period. As stated earlier, the training data must be a subset of the collected data set that is of a large size. In addition, the algorithm is provided with a fundamental understanding of the given task, a possible solution to the problem, and the data to be processed during the training and implementation periods. The training data set may share many parameters, configurations, and characteristics with the original collected data set. Finally, yet importantly, supervised learning labels the parameters of the algorithm in order to solve the task.

Subsequently, the algorithm establishes the relationship between the parameters provided in order to find the relationship between the data set and the given variables. Once the training of the algorithm has been completed, the algorithm will have a general understanding of the way data works as well as the relationship between the input data (typically represented by "X") and the desired output (typically represented by "Y"). Lastly, the solution should be deployed along with the finalized data set from which the model derives its knowledge as it learned during the training period. It specifically means that the supervised learning algorithms continuously optimize their behavior and solutions after being implemented in the real world.

Focusing on the mechanism by which the algorithm executes the process related to the

solution, the algorithm must first recognize patterns in the data set, make conclusions (or inferences) from the patterns, and produce the desired predictions. While generating the predictions, the data analytics system of the network operator must continuously correct the algorithm if it does not perform well. Such a type of monitoring of the algorithm is repeated on a regular basis until a certain level of accuracy in prediction or the desired performance has not been reached. Once such an objective has been accomplished, the algorithm is said to be perfectly trained and is expected to achieve the desired performance in any other circumstance in any environment.
Supervised learning can be used in various areas, such as the price prediction of a product, the recognition of a face, the classification of spam email systems, and many others. Overall, the application of supervised learning can be categorized into three aspects: classification, regression, and forecasting. They are discussed in the following.

- In classification, the data is grouped into several categories. Once the model has looked into the data, it must draw clear conclusions. For example, spam can be one of the binary classification problems, where the emails are going to be categorized into legitimate emails and spam emails.

- In regression, the model is built and then used to comprehend and estimate the relationship between different variables. It focuses on a single dependent variable (usually denoted as "Y") and a number of independent variables (usually denoted as "X") during a specific duration of time aimed at the prediction of a certain task in the model.

- In forecasting, the model generates predictions on the basis of historical data. It is usually used to analyze the trends in a model. In this type of model, an input is fed into the machine with a specific number of variables. The machine analyzes and generates predictions about a desired parameter. For example, forecasting weather conditions is one of the problems that fall into this category.

To this end, we have discussed supervised learning in great detail. We have also provided an insight into their classifications and categories. In the following, we will discuss a number of the most well-known supervised learning algorithms that are important and have been widely used in the wireless communication system.

- **Linear Regression:** is the most basic algorithm in the field of ML, where the model finds the optimal linear relationship between the continuous independent variables and the dependent variable. Linear Regression is divided into two types: the Simple Linear Regression and the Multiple Linear Regression. In the former, the model is used when there is only a single independent variable and it is tasked with determining its linear relationship with the dependent variable. In the latter, the model consists of more than a single independent variable and needs to determine their relationship. To name a few, each model in the linear regression must have the following characteristics: linearity, normality, no multicollinearity, homoscedasticity, and no autocorrelation.

- **The K-Nearest-Neighbour Algorithm:** is also referred to as the "lazy learner algorithm" due to the fact that it does not necessarily learn but rather classifies the data points in a data set. In addition, the K-NN Clustering algorithm is efficient when the size of the data set is smaller, but it is inefficient when a larger data set is given to the algorithm. This algorithm is said to be applied to classification problems and linear regression problems. Nevertheless, in the wireless communication field, to the best of

our knowledge, the K-Nearest algorithm has been significantly utilized to address the classification problems. In this algorithm, all available cases in the data set are first stored. Furthermore, it classifies all cases that are newly arrived and takes into account their K neighbors. Subsequently, the algorithm assigns the class to which it shares a significant number of similarities in terms of features and characteristics. For instance, if the model is attempting to learn about the group type of a specific variable in the data set. It would start looking around that specific variable and will also learn from its close neighboring variables that have already been stored and the model has full knowledge of them.

The K-Nearest algorithm performs based on the above principles and assumptions. In addition, the model must calculate the distance between the points in the graph. Once the distance has been calculated, using a specific distance calculation method, the model will then organize and add this information to its memory. Furthermore, the algorithm sorts all the collected information in an ascending manner. It specifically means that the distances are sorted from smallest to largest in the table. The ascending order of the distances helps the model quickly select the K entries from the table. On the basis of this principle, the model then takes the first entry from the ascending collection. It then labels the selected entries. At this stage, the model faces choices: if the problem the algorithm is attempting to solve is linear regression in nature, it should return the mean of the Ks; if the problem is classification, then it has to return the mode of Ks. With this, the model ends.

- **Decision Trees Algorithm:** This algorithm is a flow-chart-like model, which is typically used for solving classification problems in wireless communication systems. Nevertheless, it has also been employed to solve linear regression problems at some point. In the literature, the algorithm has shown significant performance in the classification of categorical and continuous variables. In principle, this algorithm divides the population into a minimum of two or more homogeneous collections. Such a distribution is performed on the basis of the most significant variables in the population. In order to perform a decision analysis, the model utilizes a so-called decision tree, which significantly assists the algorithm in making and taking proper decisions.

  The decision tree is designed in such a way that its root is located at the top and its nodes are available at the bottom using a top-down approach. There are three types of nodes in a Decision Tree algorithm: the decision node, which is shown by a square; the chance node, which is illustrated by a circle; and the end node, which is depicted by a triangle. Once the root has been determined, the model splits the decision tree into edges until it reaches the desired goal. The process continues until all the nodes in the population have split into the edges. At this stage, when the nodes can no longer split into edges, they are called the leaves. With this, the procedure of the Decision Tree algorithm has been ended. The aforementioned processes of this model apply to both classification problems and linear regression problems in a similar fashion.

- **Support Vector Machines Algorithm:** Similar to the Decision Tree model, the Support Vector Machines algorithm can also be used to solve both linear regression and classification problems in wireless communication systems. However, according to the literature, it is widely used in classification tasks. The support vector algorithm is one of the most well-known algorithms in ML due to its low power consumption, extremely high performance, high speed, and very high accuracy. However, this algorithm has a number of significant disadvantages: (a) When there is more noise in the data set, the

performance of the algorithm degrades. (b) This algorithm does not directly calculate the estimation of the probability. (c) It also degrades its performance as the number of features associated with each point increases.

In this algorithm, once the model has received the training data for each of the labeled data categories, it first starts with categorizing. The labeled data is then plotted on the plane. The model takes this input data and generates a hyperplane that divides the tags of the labeled data. A hyperplane is a line that has two dimensions on the plot. It serves as the plot's decision boundary. The data on one side of the hyperplane will also be classified as belonging to that category, while the data on the other side will also be classified as belonging to that category. It should be noted that in some complicated optimization problems, the data points will be plotted in different heterogeneous locations. It means that labeled data is not always linear. Therefore, the hyperplane may not be as easy to draw as a linear plot with the data points of both categories of data. To address this issue, a third-dimensional space is added to the hyperplane aimed at separating nonlinear labeled data.

- **The Naive Bayes Classifier Algorithm:** is one of the famous supervised learning algorithms that is used for solving problems that are related to the classification category. Its name consists of two terms: the Naive and the Bayes. The former refers to a situation where the existence of a feature is completely independent of other features. The latter is associated with the principle of Bayes' rules. This algorithm is based on the Bayes theorem, which was proposed by an English mathematician in the 18th century. The Bayes theorem is used for determining (or calculating) conditional probability in mathematics. It is used when we are attempting to find a probability that a few other certain probabilities are known to us in the earlier stages. On the basis of this theory, the Naive Bayes algorithm predicts the probable value of a variable. Recent studies have shown that it performs more effectively and quickly in text classification with a large number of high-dimensional data sets. To this end, there are three types of Naive Bayes models: the Gaussian model, the multinomial model, and the Bernoulli model. The Gaussian model is used to ensure that all features in the model are normally distributed. The multinomial is utilized when there is multinomial distributed data. Lastly, the Bernoulli model is similar in terms of functionality to the multinomial mode. However, the predictors are independent in this case.

- **Random Forests Algorithm:** is one of the most applied algorithms for both linear regression and classification problems due to its diversity and simplicity. It means that the input data sets can be both continuous variables and categorical variables. However, studies have shown that it performs very well in the case of classification tasks. It is a collection of an extremely large number of decision trees that function in a similar ensemble. Every single decision tree in the system generates a prediction about the variable. The model selects only that value which has a higher number of votes in comparison to other values in the random forest. With this combination learning of the model, the overall prediction power (performance) of the algorithm is increased with respect to the value that is expected to be predicted. At the start, the model takes a large number of records from the data set. The model then creates an individual decision tree for each of the samples. The decision tree will produce an output. Lastly, the final outcome (output) is decided based on the votes. If the output is decided based on the majority of the votes, then it is a classified task. If it is decided based on averaging, then it is a linear regression task.

5.3.5 Semi-supervised Learning

Semi-supervised learning is one of the fields of the ML that utilizes a hybrid data set, which contains both labeled data and unlabeled data. However, the amount of labeled data is much less than the amount of unlabeled data in the data set on which the model has been created. In such a scenario, where the labeled data points are very small, it is extremely difficult for the model to produce labeled data. These types of ML problems, which do not fit into either supervised nor unsupervised categories, fall into semi-supervised ML algorithms. The grand objectives of semi-supervised learning are similar to the objectives of the supervised learning algorithm. On the one hand, the semi-supervised ML algorithm uses the advantages of both supervised and unsupervised algorithms, while on the other hand, it avoids the disadvantage of searching for a larger collection of labeled data sets. It specifically means that when a semi-supervised learning model is being trained, there is no need for a large amount of labeled data as required for the training of supervised learning algorithms.
There are a number of techniques that enable semi-supervised learning algorithms to perform very well and produce high-quality output in wireless communication systems. They are self-training, co-training, and graph-based methods, among other algorithms. They are discussed in the following.

- The self-training in semi-supervised learning is a technique where a small amount of labeled data is used to train the model and, additionally, to predict a small amount of unlabeled data. This procedure is repeated until all the data has been labeled. To begin with, the model first divides the data into a training subset and a test subset. The former subset is then used to train a classification algorithm utilizing labeled data during the training period. In the second stage, the model uses the classifier to generate predictions associated with class labels for all unlabeled data points. During this prediction, the model labels the highest correct value with "pseudo." In the third stage, the model retrains both the labeled data and the "pseudo" data. Lastly, the training process is repeated until the model labels the entire data set. It should be noted that the model can specify a certain time (or a set of conditions) for retraining and labeling the data set.

- The co-training technique divides the data set into two completely separate views. Once this has been done, the classifier trains each view of the data set individually. The model then checks a probability confidence value against another value that has already been defined and set as a threshold by the analytics system (or the model itself). This process is executed for both views of the data set independently. During this training period, any view that has a higher confidence value is added to the training set of the opposite view in order for the classifier to train the unlabeled data of that view as well. Based on this, the selection of views of a data set is extremely important to any analytics system or model used in wireless communication systems.

- The graph representation method is one of the most efficient and effective techniques in semi-supervised machine learning algorithms that is used for representing a data set. The graph network is composed of a large set of edges and a large set of nodes. The former are used to represent the relationship, while the latter are used to represent the points on a graph. Graph-based learning has three major advantages: (a) it is applicable to a wide range of problems, both for labeled and unlabeled data; (b) it is highly scalable to large sets of data; and (c) it has a unique structure for representing the data. Due to these advantages, graph-based techniques are widely used in the field of semi-supervised learning.

5.3.6 Unsupervised Learning

In unsupervised learning, the majority of the data patterns and variables are not classified (or labeled). Thus, a model (s) should be developed to first discover patterns in the collected data set, then uncover those patterns, and lastly, label the data set. On the basis of this methodology, it is worth noting that one of the benefits of unsupervised learning is that it does not require a data scientist to prepare a large-scale collected data set readable for a machine. The machine itself does this job much more efficiently in terms of time, resources, and cost in the context of unsupervised learning. In traditional algorithms, the machine analyzes (or predicts) a specific variable based on historical data provided as input that has been previously trained and tested. However, in the case of unsupervised learning, the input data is neither trained nor tested, and the machine itself finds the relationships and correlations among the variables, sets the rules for predictions, and then makes predictions while the system is functioning online. Unsupervised learning can be used in various areas, such as the analysis of social networks, cyber-security, customer segmentation, detection of similarities, and many others.
Unsupervised machine learning algorithms can be broadly classified into two types: clustering-based machine learning algorithms and association-based machine learning algorithms. They are described in detail in the sections that follow, respectively.

Clustering

Clustering is an unsupervised learning technique in which data points in a collected data set are classified into multiple categories based on their characteristics and a certain degree of similarity. There are two major types of clustering in unsupervised learning: hierarchical clustering and partitioning clustering.
In hierarchical clustering, data points in a data set are clustered using a tree-like structure. When the structure of the tree-like clustering follows a bottom-up approach in a data set, it is referred to as agglomerative clustering. Each element is considered a cluster in this approach, and then it is step-by-step merged into a larger cluster in the data set. However, if the tree-like structure follows an up-bottom approach in a data set, this is referred to as divisive clustering. This approach considers the whole set first and then breaks it down into smaller clusters and elements.
Partitioning clustering is a technique that is used to categorize data patterns within a data set into various categories on the basis of the similarities that exist between the data patterns of the respective data set. However, the number of the clusters, which are required to be generated, is specified manually by the data analyst. In the partitioning clustering, each group must contain at least one object. On the other hand, each object must be associated with at least one group.
In the following, we will provide a well-known clustering algorithm called the K Means.

- **K Means Clustering Algorithm:** is an iterative algorithm that partitions the collected data set into K predefined different clusters, each of which contains at least a single data point. For instance, if the K=4 there will be four clusters, and K=6, there will be six clusters, and so on. The K means algorithm is attempting to collect as much homogeneous data points as possible. The primary goal of clustering data into distinct clusters is to determine the most efficient way to locate categories within the data set. This algorithm makes an attempt to maintain the similarity of intra-cluster data points while maintaining the distinctness at the same time. To accomplish this, the algorithm must clusters the unlabeled data points in such a way that the sum of the distance

of squared between them and the centroid of the cluster is as small as possible. It is worth noting that the smaller the variation between clusters, the more similar the data points will be to the cluster. The cluster to which a data point is closest to the centroid is determined by the shortest distance between the data point and the centroid of that specific cluster. This must ensures that the clusters do not overlap. The centroid is typically selected randomly. They serve as the center of the clusters. The centrorid must result in a cluster that can be labeled as desired. K Means algorithm can be used in various use cases, such the deployment of sensors in the IoT networks, diagnostics systems in the medical sector, performance evaluation, etc.

Association

The association is another technique in unsupervised learning that finds the dependency of a data item on another data item in a data set. In addition, it maps the data items with the same dependencies in order to increase profit. Recent use cases in wireless communications have shown that association learning has found interesting (and hidden) relations between the variables of a data set. Association-related algorithms are deployed in various areas, such as offering and analysis of products, medical diagnosis, analysis of market baskets, and many others. The primary philosophy of association learning is founded on two simple statements: If and Else. The former is referred to as the antecedent, while the latter is referred to as the consequent. The algorithm can use these two statements to determine the relationship between two items in a data set. To that end, association learning makes use of a number of metrics to discover relationships between variables. These metrics are lift, confidence, and support. The support metric shows the frequency of a data item in the data set. The confidence refers to the truthiness of the rule between data items. The lift metrics describe the strength of the relationship between two items.
In the following, we will discuss the most well-know algorithm that applies association rules.

- **The Apriori Algorithm:** was proposed by Srikant and Agrawal in 1994. It is an algorithm that is used to find the most frequent items in a data set and then utilizes them to produce association rules. To that end, the items should first be searched, and to do so, it uses a breadth-first technique. The algorithm then calculates the items efficiently by using the Hash Tree algorithm. As its name suggests, this algorithm uses the prior knowledge of the most frequent items in order to search for and calculate them. This algorithm is executed in a number of steps that each follow another in order to determine the most frequent items in a data set. (1) The algorithm takes each and every single item in the data and considers them as 1-item candidates. The algorithm subsequently counts how many times each item is repeated in the given data set. (2) At this stage, the algorithm takes into account the minimum support threshold, which is set by the administrator or that has already been given during the definition phase of the problem. The algorithm determines the set of items in the data set whose frequent occurrence satisfies the threshold level of the minimum support. The algorithm takes only those values into further iteration which are equal to or more than the threshold value of the minimum threshold. (3) The algorithm at this stage discovers 2-item in the frequent items of the data set. (4) The algorithm at each step compares the 2-item with the threshold value of minimum support. It only moves those 2-item to the next iteration that equals or exceeds that threshold value. (5) The algorithm then moves to 3-item and discovers the most frequent items and compares them with the threshold value. (6) Finally, the algorithm will stop when the last frequent item in the data set is achieved.

5.3.7 Reinforcement Learning in Wireless Communication

Reinforcement learning is the most sophisticated class of ML-assisted algorithms. In contrast to supervised and unsupervised algorithms, reinforcement learning enhances the performance of its model by using feedback from its previous iterations on a regular basis. Reinforcement learning does not require any training or testing data sets. Unlike supervised and unsupervised learning, the reinforcement learning model does not also require learning prior to its deployment in a real environment. In literature, reinforcement learning is referred to as learning from mistakes. Once the model is deployed in an environment, it will make a lot of mistakes in the beginning. The model will learn from its mistakes during a given time period. The longer it is deployed in an environment, the more experience it will gain and the fewer mistakes it will make. Hence, the experience of a model in an environment is directly proportional to the number of mistakes a reinforcement algorithm makes. In addition to its wide application in wireless communication, reinforcement learning has also been deployed in psychology, medicine, neuroscience, transportation, manufacturing, logistics, resource management, and various other fields.

In reinforcement learning, a machine is given several parameters and their quantitative values. It is provided with a set of possible actions that the said machine may take (or execute) in the environment. In this case, the reinforcement learning algorithm should be modeled in such a way that it must first interact with the environment by exploring different possibilities and options as well as monitoring and assessing the results of the interactions. This interaction is called iteration. If the selected algorithm performs very well in the environment, it gets a positive reward in every single iteration. However, if the algorithm performs poorly, it gets a negative reward in every iteration. In the latter case, where the outcome is not favorable, the reinforcement learning algorithm continues to perform well through continued interaction with the environment and learning from past experiences. This learning from the environment and giving positive and/or negative rewards in iterations is called the trial-and-error method in the field of ML.

One of the primary challenges in reinforcement learning is striking a balance between environmental experience and the use of the appropriate algorithm. For instance, the reinforcement learning agent always gives priority to the actions it has previously performed and discovered to be the most effective in terms of obtaining a large number of rewards. One example of an agent is one that is used to monitor and control the charge level of the electric battery in a robot. Such an agent checks the charge level on a regular basis and notifies the controlling system of the robot if this level decreases or increases. However, the agent must also attempt the actions that they have never previously performed. Therefore, the agent faces two challenges at the same time: (a) in order to obtain a large number of positive rewards, it must utilize previous actions; and (b) in order to make better decisions in the future, it must explore new actions in the environment. In both of these scenarios, neither the first types of actions nor the second types of actions are performed without receiving negative rewards. In any of these scenarios, the agent is responsible for acting in such a way as to continuously give priority to the actions that are considered the best among other actions in the environment.

However, in both of the preceding scenarios, each action in the environment must be repeated several times in order to obtain a reliable estimation of the rewards. To date, such a trade-off between exploring the environment and using an appropriate algorithm has been extensively explored by researchers, both from pure theoretical and application perspectives. However, such a trade-off has still not been resolved, and requires further study. On the basis of these discussions, we can conclude that the trade-off between exploring the environment

and appropriate algorithms for making better decisions (a) is not utilized in supervised and unsupervised learning and (b) requires additional theoretical research to fully exploit its benefits in reinforcement learning.

One of the distinguishing features of reinforcement learning is that it takes into account the entire problem without considering a particular environment. This characteristic makes reinforcement learning distinct from other supervised and unsupervised learning algorithms, which focus exclusively on a certain sub-set without considering their application to a larger set of similar practical problems. For instance, supervised learning has been explored to a greater extent without elaborating on how the capabilities of these types of algorithms might be beneficial in the long run. Numerous planning theories in the context of supervised and unsupervised learning have been investigated, but it is unclear how they can be deployed in dynamic scenarios (or use-cases) to make real-time predictions. While the state-of-the-art supervised and unsupervised solutions have produced numerous useful and practical results in the field of ML, their emphasis and application on a single sub-problem is considered a significant drawback.

In light of the aforementioned constraints, the scope of a reinforcement learning algorithm is always in the complete opposite direction. It begins with a fully interactive and goal-oriented agent that can be applied to a broader range of real-world situations in the field of ML. Two critical features that all reinforcement learning algorithms share are their ability to (a) sense all aspects of the environment in which they are located and (b) select the best possible actions in order to positively influence those environments. In addition, before deploying any reinforcement learning algorithm into a real-world environment, it is typically taken into account that its agents must continue playing despite the fact that they face uncertainty and do not receive a significant number of positive rewards from the environment. If reinforcement learning is combined with planning, the algorithm must also cover the interaction between the planning and the selection of a real-time action, as well as the acquisition and improvement of the environment for the model. However, if reinforcement learning algorithms are combined with supervised learning algorithms, which is done for a legitimate reason, such as indicating critical and non-critical features or capabilities, to advance the learning process in the environment, the critical sub-problems must be addressed first and foremost. However, they have to be those problems that contribute to acquiring positive rewards for the agent aimed at improving the overall performance of the respective algorithm.

One of the most attractive and alluring characteristics of the cutting-edge reinforcement learning algorithms is that they interact with other fields of engineering and natural science, such as optimization, statistics, mathematics, and others. In addition, reinforcement learning algorithms have a wide range of applicability in neuroscience and psychology. It is believed that reinforcement learning is the most similar type of learning that animals and humans engage in. On the basis of this, the majority of the reinforcement learning algorithms were developed by the biological systems of learning of animals and humans. In addition, reinforcement learning has contributed in two ways to the field of machine learning: (a) by developing a psychological model that humans and animals can learn in order to closely match empirical data; and (b) through the development of a reward system for the brain on the basis of influential models. These characteristics and features of reinforcement learning algorithms define the agent of an algorithm as being interactive with a variety of domains and fields, comprehensive, goal-oriented, dynamic, and productive in a real-time environment. It must also be applicable to complex problems and environments involving large amounts of data.

The Elements of the Reinforcement Learning Algorithms

Each reinforcement learning algorithm is composed of a number of elements that participate in a reinforcement learning algorithm throughout its lifetime, from input raw data until the production of output processed data. In reinforcement learning, these elements are extremely important to be considered by an algorithm to produce the best possible results in a certain situation. These elements include input data, agent, policy, environment, reward, action, state, algorithm, and output data. They are illustrated in Fig. 5.2 and discussed in greater detail in the following.

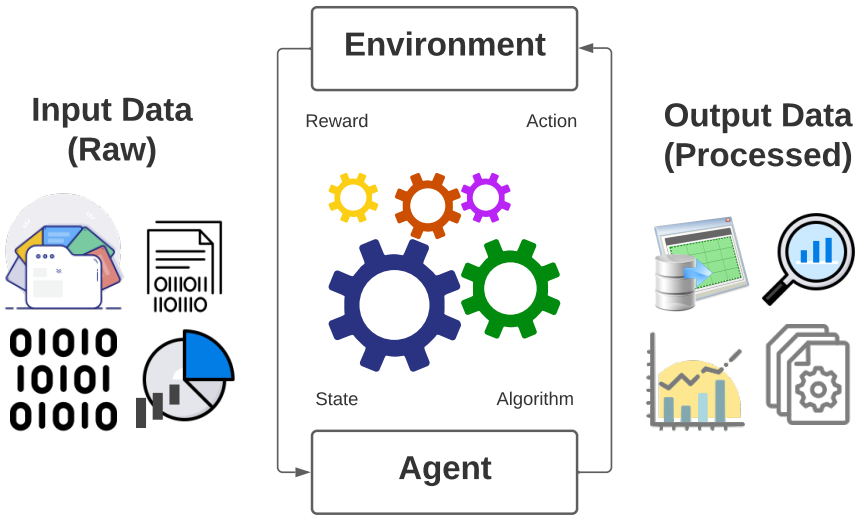

Figure 5.2: The elements of a reinforcement learning algorithm.

Input Data: In principle, reinforcement learning does not require predefined or labelled data, as supervised and unsupervised learning do. However, we assume that the algorithm is provided with some information about the desired goal that it is expected to accomplish in the environment.
Agent: The agent is an experimental tool in an environment that makes the best possible decisions in response to rewards and punishments in a given situation based on its experience.
Policy: A policy is a set of rules that an agent uses to determine which actions to take. These rules can be deterministic or stochastic. The determinism policy is the policy in which states are linked to specific actions. The policy receives the state and returns an action to perform. These policies are used in deterministic environments, where the performed actions determine the outcomes. In stochastic policy, a state does not have a clearly defined action to take but rather a probability distribution of possible actions. The stochastic policy can be divided into categorical policies and diagonal Gaussian policies.
Environment: Environment in reinforcement learning is the world in which the agent is locates and interacts. By executing specific actions, the agent interacts with the environment. However, it has no effect on the rules defined by the environment. Additionally, the environment cannot be changed by the actions taken by the agent. The environment could be physical or simulated.
Reward: When an agent interacts with its environment, it is essentially carrying out certain tasks. And with each action that a reinforcement algorithm's agent performs, it obtains a reward. The reward may be either positive or negative. Always, the agent's primary goal is to maximize the total number of positive rewards.
Action: The action of an agent is the method by which it interacts with its environment. It enables the agent to interact dynamically with its environment and change its location in order

to maximize positive rewards. Each action taken by an agent is significant in reinforcement learning. The policy determines whether or not to assist an agent in performing a particular action in an environment.

State: In reinforcement learning, state refers to the current physical or simulated location of an agent within a given environment. The state can assist the agent in selecting from a limited set of available actions.

Algorithm: There are various types of algorithms in reinforcement learning. Each has its own set of policies, rules, and working conditions. In order to solve a certain problem efficiently, an algorithm (out of a large collection of algorithms) must be chosen to provide an optimal solution by achieving the maximum number of certain rewards.

Output Data: There is no such thing as a reinforcement learning output data set. The output of this type of algorithm is a list of the positive actions taken by the agent that resulted in a maximum number of possible rewards. For example, this could be a prediction about a specific object.

Different Types of Reinforcement Learning Algorithms

Generally, all reinforcement learning algorithms can be classified into two categories: model-free algorithms and model-based algorithms.

The critical aspect of the reinforcement learning algorithm is the agent's access to an environment model or the agent's ability to learn an environment model. The environment model is a function that is used to forecast the rewards and the state transitions in reinforcement learning. The primary advantage of using an environment model in a reinforcement learning algorithm is that it enables the agent to make future plans by looking further, examining the consequences of a variety of possible actions, and making precise choices between alternative actions. If this approach is successful, it will result in a significant increase in the efficiency of samples over the currently used methods, which typically do not include any environmental model. However, the main disadvantage of having an environment model is that the agent is rarely provided with real-time information and truth from the environment. For example, if the agent is willing to learn by utilizing an environment model, it must learn it entirely through previous experience, which presents a number of difficulties. Due to this difficulty, even though the agent exerts considerable effort in experiencing the environment, it may fail to produce the desired results.

Given the foregoing, model-based reinforcement algorithms employ a model of the environment, whereas model-free reinforcement algorithms do not. Model-free algorithms are typically easier to deploy and tune than model-based algorithms. However, model-based algorithms are difficult to implement in real-world situations. In the following, we will provide a technical definition of model-free and model-based algorithms.

Model-based reinforcement learning algorithms: Algorithms based on model-based reinforcement learning develop transitions and outcomes within the internal model through the use of prior experiences. It subsequently takes proper actions, such as searching or planning techniques, in order to make systematic choices within the model. Because model-based reinforcement learning algorithms are out of the scope of this thesis, we intentionally skip providing further details on the types of this algorithm.

Model-free reinforcement learning algorithms: Algorithms based on model-free reinforcement learning make use of prior experience and knowledge to learn either a single or both of the quantities, state and value, at a given time interval. It is expected that such a model will achieve optimal behavior without the use of an estimation function. In the following, we briefly introduce several model-based reinforcement learning algorithms.

- **state-action-reward-state-action (SARSA):** The SARSA algorithm, which was proposed by Rummery and Niranjan, is based on an on-policy mechanism and is used to learn a Markov decision process policy. It updates the Q-table with current and future states and actions of the agent. It specifically means that the SARSA updates the Q-value based on the current state of the agent, the optimal action that the agent takes, the possible (negative or positive) reward that the agent receives for taking the action, the next state that the agent will enter after taking the first action, and lastly, the next action that the agent may possibly take in the second state.

- **Q-Learning:** The Q-learning algorithm, which will be discussed in the next section, is based on an off-policy technique that is used to discover an optimal policy aimed at maximizing the values of all the rewards of the actions that are expected to be taken in an environment. Q-learning is used to learn about the value of a possible action in a given state in an environment. The distinction between the Q-learning algorithm and the SARSA algorithms is that the SARSA algorithm selects optimal action consistent with the existing policy and then updates the Q-values. The Q-learning algorithm, on the other hand, first selects the greedy actions that produce the highest Q-value and then proceeds with following an optimal policy.

- **Deep Q Network (DQN):** The Q-learning algorithms are said to be inefficient in complex problems and environments with multiple outcomes and numerous possibilities. In order to address this problem, DeepMind proposed the DQN in 2015. The DQN algorithm is obtained by mixing deep neural network techniques with reinforcement learning techniques with the goal of solving a large number of use cases in the field of ML. More specifically, the grand objective of the proposed DQN algorithm was to improve the performance of Q-learning algorithms by merging the experience replay technique used in deep neural networks into Q-learning. The Q-learning algorithm optimizes the performance of an agent in an environment in an indirect manner. When applied to an environment, the Q-learning algorithm has the advantage of being significantly more efficient in terms of sample size due to its effective reuse of data.

- **Deep Deterministic Policy Gradient (DDPG):** The DDPG is a well-known algorithm in reinforcement learning, which learns the policy and the Q-function at the same time in a given environment. In order to learn the Q-function, the DDPG utilizes the data related to the off-policy, and in order to learn the policy, it uses the Q-function. The DDPG is based on an off-policy technique. This algorithm is used specifically in environments with continuous action spaces.Therefore, in the literature, the DDPG is also considered to be a deep Q-learning algorithm used for continuous action spaces in reinforcement learning.

- **Soft Actor-Critic (SAC):** The SAC is an off-policy optimization technique used in reinforcement learning for stochastic policies. The stocahstic policy optimization concept and DDPG approaches are combined in this algorithm. Furthermore, the SAC algorithm makes use of target policy in a given environment. The SAC algorithm is said to be capable of handling both continuous and discrete action spaces. In the SAC algorithm, recurrence regularizing the entropy is considered a critical element. In this technique, the policy is trained in the environment in order to optimize the trade-off between entropy and return, which is also called the measure of randomness. Such a trade-off stops the policy from quickly convergent to a less-than-ideal local optimum in SAC algorithm.

5.4 Q-Learning in Wireless Communication System

We have briefly defined Q-learning in the previous section. In this section, we delve deeper into its theoretical and application aspects. Among other reinforcement learning algorithms, we select Q-Learning due to its simplicity, effectiveness, and applicability to the problem we are attempting to solve in this thesis. We employ this algorithm in the mapping process and virtual resource allocation of eMBB, URLLC, and mMTC types of RAN slices in the underlying I-PoPs in the NG-RAN architecture.

In Q-learning, the algorithm learns the value of a particular action at a given state and then determines the appropriate action to be taken based on the current state and value. Q-learning algorithms are not always dependent on models interacting with their environments. They use stochastic transitions to solve optimization problems. Therefore, the Q-learning algorithms are referred to as model-free algorithms. The main goal in Q-learning is to achieve an optimal policy through the maximization of the total rewards' values. It first starts from the current state and value, interacts with a certain environment, maximizes the rewards, and takes further actions based on the previous experiences.

To that end, there is a function in the Q-learning algorithm, which is referred to as the Q function, that accepts a state and an action as an input. The Q function processes the aforementioned inputs and returns the expected rewards (whether they are negative or positive) for this particular action in this particular state. The Q function proceeds with all subsequent actions using the same process, and for every action, it receives a reward. Prior to exploring a particular environment by the model, the Q function is given a fixed value. However, with additional exploration of the environment, the Q function can be used to determine the optimal value for a particular action at a given state. With more experience in the environment, the Q function gradually increases its optimality. The model updates the Q function with any new changes in its value.

To better explain the above concepts, we provide here a well-known example of the application of Q-learning. For example, we assume that a certain object is moving along an obstacle-filled path from a starting point ("Point A") to an ending point ("Point B"). The main goal of the movement of this object is to take the shortest path possible in order to reach Point B without colliding with any obstacles that exist along the way. The object must accomplish this while remaining within the obstacle-filled boundary. In this example, the object is essentially the agent. The starting point ("Point A") is the current state of the agent, where it currently exists. The movement of the object is referred to as the action that the agent takes. The ending point ("Point B") is the goal of the agent that he is expected to achieve. The pathway is the environment in which the agent is operating.

Similarly, the aforementioned concepts and terminologies can be realized in the mapping problem for the three types of RAN slices in the NG-RAN architecture. The underlying infrastructure, which virtualizes the physical networking and compute resources into VMs and VNs, could be considered the environment for the implementation of the Q-learning model. The configuration, allocation, reservation, deactivation, and other functions associated with resource management and orchestration that the VNFM and VIM perform could be considered the actions of the Q-learning model. The configuration (or perhaps resource instantiation) phase of the mapping process could be considered the current state in the context of Q-learning. The preparation phase, activation phase, operation phase, decommissioning phase, and so on could be thought of as the next steps. If the Q-learning successfully completes the aforementioned phases, it receives a positive reward; otherwise, a negative reward is issued. In order for Q-learning to receive a maximum number of positive rewards, it must perform the aforementioned tasks in an optimal manner. Finally, the algorithm that is used

to manage and orchestrate the virtual resources of the respective RAN slice is referred to as the agent of Q-learning.
In Q-learning, the "Q" is referred to as "quality". The notion of "quality" in Q-learning refers to the effectiveness of a certain action in achieving total rewards in a certain environment. In order to define a Q-learning model, we need to first define the following terms and terminologies that are very useful in understanding its theoretical aspects:

- In order to perform a particular action in a particular environment, the model expects to receive a specific value (also known as a discounted reward). In the context of Q-learning, this value is defined as Q (s, a). The Q value could belong to the new state of the action or the next state of the action. They can be expressed by various indexes in their mathematical formula.

- In the context of Q-learning, the concept of temporal differences is employed to predict the expected value that Q (s, a) may achieve. The temporal difference refers to an agent acquiring knowledge about an environment through episodes without prior knowledge of this environment.

- In Q-learning, the agent always captures a table of discounted values for Q (S, A), where S denotes the set of all the states(s) and A denotes the set of all actions(a) of the corresponding model.

Based on the aforementioned definitions of the concepts and terminologies, the Q-learning function can be expressed by the following mathematical formula:

$$\mathrm{Q(s,a)} = \mathrm{Q(s,a)} + \alpha[\mathrm{r} + \gamma \ \max \mathrm{a}, \ \ \mathrm{Q(a^{'}, s^{'})\text{-}Q(s,a)}] \tag{5.26}$$

where Q denotes the expected value (i.e., discounted rewards) of an action "a" that has been taken in a state "s". The "s" indicates the state vector in which the action(s) are performed. The vector of an action that an agent takes is denoted by "a". The reward (whether negative or positive) that an agent obtains is denoted by "r" in the environment. The α denotes the rate of learning, which is what determines convergence.
In Q-learning, the α is typically set between 0 and 1. By setting the value of α to 0, the values of Q are no longer updated, and hence the model does not learn any new information about the environment. Nevertheless, if the value of α is set to a large number, such as 0.99 or 0.999, then it indicates that the agent is capable of learning rapidly in its environment. A learning rate of $\alpha_t = 1$ is an ideal rate in the Q-learning.
The γ is referred to as the discount factor. Thanks to the discount factor in Q-learning, the rewards that are earned earlier are more valuable to the model than the rewards that are earned later. Instead of focusing exclusively on the optimal policy in Q-learning, the method of discount factor learns the estimated values of all actions that are taken by the agent. On the one hand, this knowledge is costly from the perspective of the amount of data that must be stored; on the other hand, it does have numerous advantages.
In the Q-learning algorithm, prior to the agent performing the first action in the environment, an initial condition is always assumed. If the initial values are positive and high, the agent is always encouraged to explore the environment and find the optimal policy. To accomplish this, the initial conditions are established using the first reward in the Q-learning algorithm. On the basis of these principles, the first reward is critical and is always used to set the value of Q in the Q-learning algorithm. Lastly, it also enables instantaneous learning in the presence of fixed deterministic rewards.

5.4.1 Q-Function

In Q-learning, which is a value-based algorithm in reinforcement learning, the Q-value function (or simply the Q-function) is utilized to determine the optimal policy for the selection of an action. The Q-value is a function that is used to map a pair of observation-based actions onto a scalar value in order to represent the total long-term rewards that are expected to be obtained by the agent from the time it begins with a particular observation until it completes a particular action. The main purpose of this value in Q-learning is to always maximize the Q-function. It also estimates for a Q-learning model how optimal it is for an agent to exist in a particular state. This will enable the anticipated rewards to be maximized by choosing the best of all optimal actions in the environment (which is always deterministic). The entire set of values generated by a Q-learning algorithm in an environment is stored in a table (which will be discussed in the next section) referred to as the Q table, which enables the system administrator to determine the optimal course of action for every single state. In Q-learning, we always begin by examining the environment of the model and updating the Q table. When the Q table is complete, the agent will begin exploiting the environment and taking more appropriate actions.
In order to achieve this value, the Q-function uses, which is based on the Bellman equation, accepts two key parameters: state (s) and action (a). The Bellman equation is one of the key elements of a number of reinforcement learning algorithms. It is used to calculate the value of a Q-function in the Q-learning algorithm. This function, which in its simplest form has also been described in the preceding subsection, can be achieved through the following equation.

$$\underbrace{\text{New}Q(s,a)}_{\text{New Q-value}} = \overbrace{Q(s,a)}^{\text{Current Q-value}} + \alpha[\underbrace{R(s,a)}_{\text{Reward}} + \gamma \overbrace{\max Q'(s',a')}^{\substack{\text{Maximum reward that is predicted by the model in a} \\ \text{given new state and all actions that are possible}}} - Q(s,a)] \quad (5.27)$$

Learning rate

Discount rate

In the aforementioned equation, as we have also defined in the preceding subsection, $NewQ(s,a)$ is the new value of the Q-function that is obtained recently by an agent in the environment, $Q(s,a)$ is the current value of Q-function in the existing state, α is the learning rate of the Q-learning model, $[R(s,a)]$ is the reward that is obtained by the model, γ is the discount factor $(0 < \gamma < 1)$ of the Q-learning model, and $maxQ'(s',a')$ is the maximum reward that has been predicted by the Q-learning model in a given state by a given possible action in an environment. The above function for obtaining the Q-value is used to achieve the Q-values for every single cell in the Q table of a model. In the beginning, the Q-values for all the cells are set to zero. An iterative process is followed by the model to update these values in an environment. Utilizing this process, the agent explores the environment on a regular basis and updates the values of the Q-functions in each and every cell by obtaining optimal approximations.

5.4.2 Creating the Q-Table

The term "Q-Table" refers to a simple and straightforward method of displaying a table in which the maximum expected future rewards for actions taken at each state are calculated. Essentially, this table directs data scientists to the optimal course of action for each state in a particular environment. The Q-table is extremely important for proceeding the Q-learning algorithm. It must be built to store the values of Q-functions. In this table, there are n

number of columns and and m number of rows. The former is referred to as the number of actions. The latter is referred to as the number of states. In the following, we will depict what a Q-table looks like in the Q-learning algorithm.

Table 5.1: The structure of the Q-table in Q-learning. The rows represent the states, while the columns represent the actions of the agent.

	$\mathbf{A}_1$	$\mathbf{A}_2$	**...**	$\mathbf{A}_M - 1$	$\mathbf{A}_M$
S_1	$Q(S_1, A_1)$	$Q(S_1, A_2)$	...	$Q(S_1, A_M - 1)$	$Q(S_1, A_M)$
S_2	$Q(S_2, A_1)$	$Q(S_2, A_2)$	...	$Q(S_2, A_M - 1)$	$Q(S_2, A_M)$
...	...	...	...	...	..
$S_N - 1$	$Q(S_N - 1, A_1)$	$Q(S_N - 1, A_2)$	...	$Q(S_N - 1, A_M - 1)$	$Q(S_N - 1, A_M)$
S_N	$Q(S_N, A_1)$	$Q(S_N, A_2)$	...	$Q(S_N, A_M - 1)$	$Q(S_N, A_N)$

In the above table, given the current state of the agent state, the Q-values can be simply, effectively, and easily reproduced. The state in the table corresponds to the agent being in the middle and having just begun. Naturally, the agent is initially unaware of its surroundings. As it performs actions, it is aware of the action values, and the cells of the Q-table are updated at each step. After a few trials, we anticipate that the corresponding cell of the Q-table will converge to the desired values.

Q-learning Algorithm for Estimating $\pi \approx \pi^*$

Data: γ = discount factor, α = learning rate, and e a small number
Result: A Q-table which must contain $Q(A, A)$ in order to define the optimal anticipated policy
Parameters: step size $\alpha \in (0, 1]$ where the value of α must be btween 0 and 1, small *epsilon* > 0;
Initialization phase;
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\text{terminal}, \cdot) = 0$;
foreach *episode* **do**
    Initialize S;
    foreach *step of episode* **do**
        Choose A from S using policy derived from Q (e.g., ε-greedy);
        it basically selects an action from a set of states. Take action A, observe R, S';
        it then takes the selected action.
        $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$;
        $S \leftarrow S'$;
    end foreach
end foreach

Once the Q-table has been initiated, the next step is to determine the path that an agent has to take in order to reach the desired goal. For instance, if an agent goes in the left direction, right direction, up direction, and down direction, it is basically taking the actions that could be described in the column on Tab. 5.1. Each of the above actions has five aspects: start, idle, correct path, wrong path, and end. In the beginning, the agent starts. When it takes its first move, it obtains a reward referred to as an "ideal case." The agent could take either the correct or wrong path. It could also reach the final desired destination.

5.4.3 The Steps of the Q-Learning Algorithm

The Q-learning algorithm is shown in the following. The algorithm is executed in five steps, which are discussed in the following respectively.

- **The Initialization of Q-table:** In this phase, the Q-table is built by the model. The size of the Q-table is built according to the number of actions and states of the problem. If there are seven actions (a = 7) and eight states (s = 8), a Q-table with seven columns and eight rows must be designed.
- **Choose an Action:** The primary goal of the Q-learning algorithm during this process is to teach the agent how to behave in a variety of situations within a given environment. The agent determines which actions to choose and then performs them during the training process. It chooses an action (a) that existed in a state (a) from the Q-table that was generated in the first process. This action should always be chosen on the basis of its reward. In order to generate maximum positive rewards, the agent must also prefer to choose the optimal actions in a particular environment.
- **Perform an Action:** Once an optimal action has been selected either on the basis of Epsilon Greedy Strategy or Exploration and Exploitation Trade-off, the algorithm performs that action. In literature, the previous (Choose an Action) step is performed with this step for a limited amount of time. It specifically means that both of the steps are running until the administrator stors the training of the Q-learning. It could also be defined in advance in the loop of the time frame in the code.
- **Measure the Reward:** At this point, the action has been taken and observed by the algorithm. The model obtains a reward - either positive or negative - on the basis of its efficiency.
- **Update the Q-table:** Finally, the values of Q-functions are in the Q-table. The algorithm iteratively repeats the preceding four steps until the learning process is halted. This strategy optimizes the value function Q by updating the Q-Table. As a result, the Q-value represents the anticipated future benefit of that action in that state.

5.4.4 Double Q-Learning

This algorithm decouples action selection from action evaluation, thereby reducing the overestimation that affects conventional Q-learning algorithms [183]. Double Q-learning maintains two separate estimators, denoted as Q^A and Q^B, and updates them alternately. At each update step, one of the estimators is chosen based on the step index. If Q^A is selected for an update, the greedy action is identified using Q^A, but its value is taken from Q^B. Conversely, if Q^B is selected for update, the greedy action is chosen using Q^B, but its value is taken from Q^A. In this way, the maximization operator does not act on the same estimator for both selection and evaluation, thereby reducing the bias inherent in conventional Q-learning methods.
In double Q-learning, the update rules for Q values are formally given as follows [183]. If updating Q^A:

$$Q^A(s,a) \leftarrow Q^A(s,a) + \alpha\Big[r + \gamma Q^B(s', \arg\max_{a'} Q^A(s',a')) - Q^A(s,a)\Big],$$

and if updating Q^B:

$$Q^B(s,a) \leftarrow Q^B(s,a) + \alpha\Big[r + \gamma Q^A(s', \arg\max_{a'} Q^B(s',a')) - Q^B(s,a)\Big],$$

where, s is the current state, a is the selected action, r is the observed reward, and s' is the next state. The parameters α and γ denote the learning rate and discount factor, respectively. By alternating between these two updates, double Q-learning generates more accurate and less optimistic estimates of the true action-value function.

Double Q-learning Algorithm for Estimating $\pi \approx \pi^*$

Data: γ = discount factor, α = learning rate, and e a small number
Result: Two Q-tables, Q^A and Q^B, which are used jointly in order to define the optimal anticipated policy
Parameters: step size $\alpha \in (0, 1]$ where the value of α must be between 0 and 1, small $\epsilon > 0$;
Initialization phase;
Initialize $Q^A(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q^A(\text{terminal}, \cdot) = 0$;
Initialize $Q^B(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q^B(\text{terminal}, \cdot) = 0$;
foreach *episode* **do**
  Initialize S;
  foreach *step of episode* **do**
    Choose A from S using policy derived from $\frac{1}{2}\Big(Q^A(S, \cdot) + Q^B(S, \cdot)\Big)$ (e.g., ε-greedy);
    it basically selects an action from a set of states. Take action A, observe R, S';
    it then takes the selected action. **if** *the step is even* **then**
      Choose $a^* = \arg\max_a Q^A(S', a)$;
      $Q^A(S, A) \leftarrow Q^A(S, A) + \alpha[R + \gamma Q^B(S', a^*) - Q^A(S, A)]$;
    end if
    else
      Choose $a^* = \arg\max_a Q^B(S', a)$;
      $Q^B(S, A) \leftarrow Q^B(S, A) + \alpha[R + \gamma Q^A(S', a^*) - Q^B(S, A)]$;
    end if
    $S \leftarrow S'$;
  end foreach
end foreach

The pseudocode of the double Q-learning algorithm is presented in above [183]. The considered inputs include the state space $\mathcal{S}$, the action space $\mathcal{A}(s)$, and the reward space $\mathcal{R}(s, a)$. Additionally, the key parameters such as the discount factor γ, exploration rate ϵ, and learning rate α, and the optional horizon T_{max} are included. A predetermined value $m \in \mathbb{N}$ (e.g., $m = 100$), which defines the maximum number of episodes before the algorithm terminates, is also taken into account. Given these input parameters, we initialize the state-action value with the initial state-action estimators $Q^A(s, a) = 0$, and $Q^B(s, a) = 0$ for all state-action pairs (s, a). We also initialize an initial ϵ-greedy policy π_0. Ultimately, for the output, we expect double Q-learning algorithm to learn an optimal Q-value, Q^*, and an optimal policy, π^*, that maximizes the cumulative rewards.

Double Q-learning algorithm proceeds by iterating over the triples (A_t, R_{t+1}, S_{t+1}). For each episode, the system begins from an initial state. At each step, an action is selected using an ϵ-greedy strategy with probability ϵ, a random action is chosen to ensure exploration, while with

probability $1-\epsilon$, the action that maximizes the combined estimates $Q^A(s,a') + Q^B(s,a')$ is selected. After executing the action, the agent observes the immediate reward and transitions to the next state. Depending on the index of the current state, one of the two estimators is chosen for update. The action-value estimator Q^A is updated via

$$Q^A(S_t, A_t) := Q^A(S_t, A_t) + \alpha\,[\,y - Q^A(S_t, A_t)],$$

where $a^\star = \arg\max_{a \in \mathcal{A}(S_{t+1})} Q^A(S_{t+1}, a)$, and

$$y := R_{t+1},$$

if S_{t+1} is the terminal state, and

$$y := R_{t+1} + \gamma\, Q^B(S_{t+1}, a^\star)$$

otherwise. The state-action estimator Q^B is updated via

$$Q^B(S_t, A_t) := Q^B(S_t, A_t) + \alpha\,[\,y - Q^B(S_t, A_t)],$$

where $a^\star := \arg\max_{a \in \mathcal{A}(S_{t+1})} Q^B(S_{t+1}, a)$, and

$$y := R_{t+1},$$

if S_{t+1} is the terminal state, and

$$y := R_{t+1} + \gamma\, Q^A(S_{t+1}, a^\star)$$

otherwise.
To put things into words, if Q^A is updated, then the greedy action is determined from Q^A, but its value is evaluated using Q^B. If Q^B is updated, the roles are reversed. This process repeats until the episode terminates, and the cycle continues across multiple episodes.

5.4.5 Deep Q-Learning

Deep Q-learning is an off-policy RL method that uses a deep neural network to approximate the Q-function, the expected long-term reward of taking an action in a given state and then following a policy thereafter [184]. It scales the classic Q-learning function approximation method to handle high-dimensional inputs by letting a neural network do the function approximation. Instead of storing a huge Q-table, the network takes a state and outputs Q-values for all possible actions, then selects actions greedily. Training minimizes the Bellman error between current Q-value estimates and a target built from the reward plus the discounted max next-state value. Two key tricks make it stable, namely experience replay (learn from a shuffled buffer of past transitions) and a target network (a periodically updated copy used to compute targets). Because it is off-policy, the agent can learn from old data and even from the behavior of other policies [184].

Deep Q-learning Algorithm for Estimating $\pi \approx \pi^*$

Data: γ = discount factor, α = learning rate, and e a small number
Result: A trained Q-network $Q(s, a; \theta)$ and a target network $\hat{Q}(s, a; \theta^-)$ used to define the optimal anticipated policy
Parameters: step size $\alpha \in (0, 1]$, small $\epsilon > 0$, replay memory size D, minibatch size M;
Initialization phase;
Initialize $Q(s, a; \theta)$ with random weights θ;
Initialize target network $\hat{Q}(s, a; \theta^-)$ with weights $\theta^- = \theta$;
Initialize replay memory D;
foreach *episode* **do**
  Initialize S;
  foreach *step of episode* **do**
    Choose A from S using policy derived from $Q(s, a; \theta)$ (e.g., ε-greedy);
    it basically selects an action from a set of states. Take action A, observe R, S';
    it then takes the selected action. Store transition (S, A, R, S') in replay memory D;
    if *replay memory D contains at least M transitions* **then**
      Sample random minibatch $(S_j, A_j, R_{j+1}, S'_j)$ from D;
      foreach *sample in minibatch* **do**
        if S'_j *is terminal* **then**
          $y_j \leftarrow R_{j+1}$;
        end if
        else
          $y_j \leftarrow R_{j+1} + \gamma \max_a \hat{Q}(S'_j, a; \theta^-)$;
        end if
      end foreach
      Perform a gradient descent step on loss;
      $L(\theta) = \sum (y_j - Q(S_j, A_j; \theta))^2$;
      Update network weights θ;
    end if
    Periodically update target network $\theta^- \leftarrow \theta$;
    $S \leftarrow S'$;
  end foreach
end foreach

The pseudocode of the deep Q-learning algorithm is shown in double Q-learning algorithm [184]. The considered inputs are the state space $\mathcal{S}$, the action space $\mathcal{A}(s)$, and the reward space $\mathcal{R}(s, a)$. As input parameters, the hyperparameter γ is chosen (user-defined). Replay memory $\mathcal{D}$ is established to store a fixed number of transitions, up to a capacity N. A minibatch of size M is selected. Moreover, to comply with the off-policy setting of this algorithm, a behavior policy π_b defined for all state-action pairs (s, a) is also provided as input. The initial Q-value Q_0 is initialized to 0. An initial policy π_0 is taken into account. The function $Q(s, a; \theta)$ is initialized with random weights θ, while a target action-value function $\hat{Q}(s, a; \theta^-)$ is also initialized, with its weights θ^- set equal to θ. This target network is periodically updated to stabilize training by decoupling target generation from the learning

updates.
The algorithm operates over multiple episodes, each beginning with an initial state S. During each step $t \in \{1, \ldots, N\}$ within an episode, an action A_t is selected from the current state S_t based on a policy derived from $Q(s, a; \theta)$, typically using a greedy strategy. The agent executes the action A_t, observes the immediate reward R_{t+1}, and transitions to the next state S_{t+1}. The transition tuple $(S_t, A_t, R_{t+1}, S_{t+1})$ is stored in the replay memory $\mathcal{D}$, enabling the algorithm to leverage past experiences during training.
Once the replay memory contains at least M transitions, a random minibatch of M transitions $(S_j, A_j, R_{j+1}, S_{j+1})$ is sampled for training. For each transition in the minibatch, the target value y_j is computed as follows:

$$y_j = \begin{cases} R_{j+1} & \text{if } S_{j+1} \text{ is terminal,} \\ R_{j+1} + \gamma \max_a \hat{Q}(S_{j+1}, a; \theta^-) & \text{otherwise,} \end{cases}$$

where γ is the discount factor that prioritizes immediate rewards over future rewards. Using these targets, the network parameters θ are updated by minimizing the loss function:

$$L(\theta) = \sum_{j=1}^{m} (y_j - \hat{Q}(S_j, A_j; \theta))^2.$$

Finally, the optimal policy, $\pi^*(s) = \arg\max_a \hat{Q}(s, a)$, is returned.

Chapter 6

Simulation Results and Discussions

Summary – In the previous chapter, we mathematically formulated the problem of the mapping of the VNFs and VLs of three different types of RAN slices onto the underlying I-PoPs and transportation networks, respectively. Additionally, we defined the underlying virtual compute, storage, and networking resources that are required to be allocated to the virtual components of the three different types of RAN slices in the NG-RAN architecture. The performance objectives of the mapping problem are also clearly defined, and a set of constraints of the system model that limit the possible values and decision variables are also discussed in detail. In light of this formulation, we begin this chapter by describing the simulation environment (which includes the simulation setup, network topology, and simulation parameters) that is considered for simulating the virtual components of the eMBB, URLLC, and mMTC types of RAN slices in the NG-RAN architecture. Following that, we thoroughly discuss the results obtained during the simulation period employing the Q-learning algorithm in the mapping process of RAN slices. The obtained results demonstrate the mapping of three distinct types of RAN slices onto the underlying infrastructure under different conditions. It also includes an evaluation of the performance objectives we are attempting to achieve through the use of the Q-learning algorithm against the SLA signed between the tenant and the service provider during the lifetime of the requested RAN slice. Additionally, the chapter discusses the findings regarding the global resource allocation required for hosting a large number of RAN slices in the underlying infrastructure. Finally, the chapter explains the advantages of employing the Q-learning algorithm in the mapping process of a RAN slice over other state-of-the-art algorithms and thoroughly provides an in-depth analysis of the obtained results of this thesis.

6.1 Simulation Environment

We commence by providing necessary information about the environment we have selected for the simulation of the mapping problem of the RAN slices onto the underlying infrastructure in the NG-RAN architecture. The simulation environment considered for the mapping problem includes the simulation setup, network topology, and simulation parameters. These aspects are discussed in the following subsections, respectively.

6.1.1 Simulation Setup

The experiments regarding the mapping problem of the RAN slices presented in this thesis are implemented and evaluated in MATLAB Release 9.0 and Python Release 3.6 programming languages. The basic requirements for the system model considered in the previous chapter

is mainly built in PyTorch. We have used various Python libraries for data mining, data processing and modeling, and data visualization. For data visualization, we have used pydot, Plotly, Seaborn, Bokeh, and Matplotlib. For data processing and modeling, we have mainly used the Pytorch. However, TensorFlow has also been used on several occasions. For data mining, we have mainly used Scrapy. Additionally, we have also utilized NumPy, which is a well-known Python library to support comprehensive mathematical functions, multi-dimensional arrays and matrices, linear algebra, and other mathematical operations in the field of ML. Lastly, the results generated for this thesis are obtained using an Intel Core i7 processor having 64 GB RAM, Microsoft Windows 10 operating system, and 1000 GB solid state drive (SSD).

6.1.2 Network Topology

Implementing the proposed mapping solution on real network topology and measuring the performance objectives of the deployment of the virtual components of RAN slices onto the underlying infrastructure in the NG-RAN architecture are extremely critical for achieving higher certainty and accuracy in the simulation results. To that aim, we have considered three types of network topologies for three types of RAN slices. These network topologies have been carefully selected based on the number of I-PoPs, I-PoP's location, number of links, link capacity, complexity of the architecture of the respective topologies, and the characteristics of the respective types of RAN slices from a large-scale collections (i.e., 232) of Internet topologies in the The Internet Topology Zoo [2]. The Internet Topology Zoo is a collection of real network topologies from several well-known network operators around the world. The Zoo contains various network topologies with different characteristics in terms of transportation links (i.e., optical fiber, virtual connectivity, etc.) and the underlying compute and storage nodes (i.e., node capacity, isolation level, etc.).

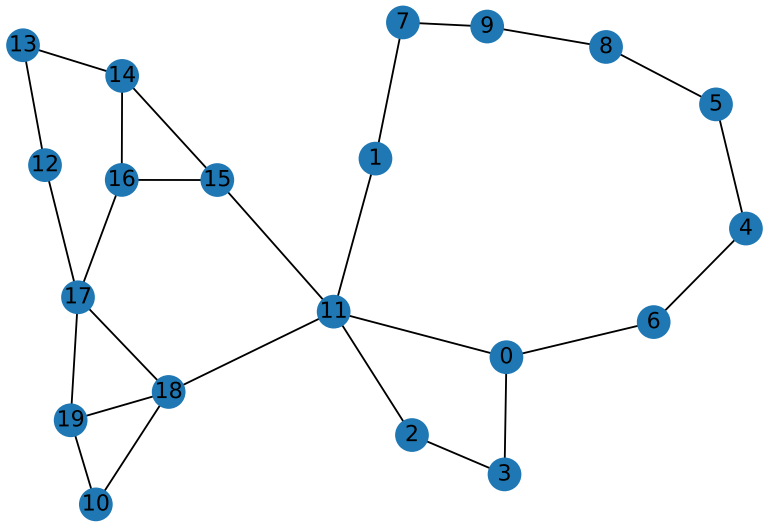

Figure 6.1: The underlying network topology (i.e., Oxford) considered for the deployment of the eMBB types of RAN slices onto the underlying I-PoPs and transportation networks in the NG-RAN architecture [2].

To conduct experiments on the eMBB type of RAN slices, we have considered the Oxford network topology, which is depicted in Fig. 6.1 [2]. The Oxford network topology consists of 20 I-PoPs and 26 links, which can be virtualized. We consider the IBM network topology for the URLLC types of RAN slices. It consists of 18 I-PoPs and 26 links. These links, which

connect the I-PoPs located in different geographical locations, can be virtualized along with their respective I-PoPs. Due to high density and lower required bandwidth, we consider the UniNet network topology for the mMTC types of RAN slices. The UniNet network topology is consisted of 13 I-PoPs and 25 links.

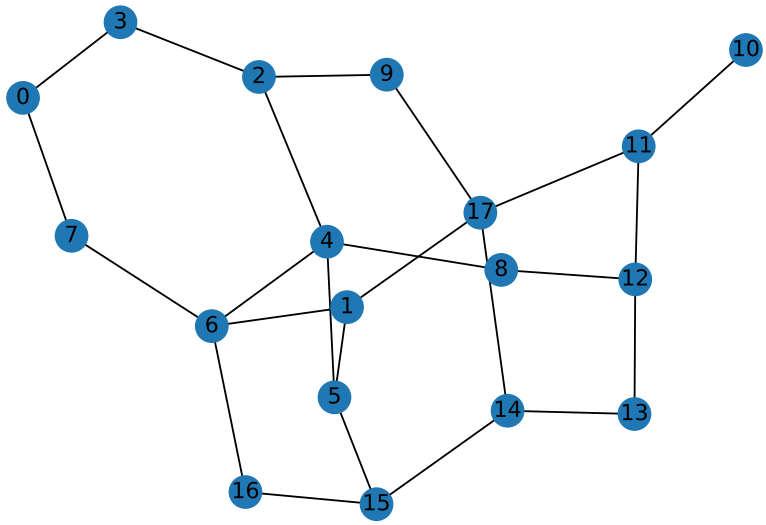

Figure 6.2: The underlying network topology (i.e., IBM) considered for the deployment of the URLLC types of RAN slices onto the underlying I-PoPs and transportation networks in the NG-RAN architecture [2].

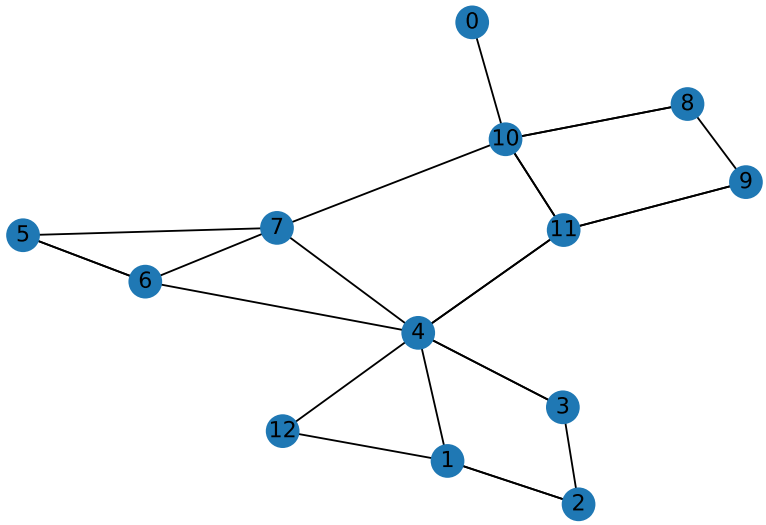

Figure 6.3: The underlying network topology (i.e., UniNet) considered for the deployment of the mMTC types of RAN slices onto the underlying I-PoPs and transportation networks in the NG-RAN architecture [2].

6.1.3 Simulation Parameters

In this subsection, we provide a list of network simulation parameters and their corresponding values, which are taken into consideration for the simulation environments of the mapping problem of the virtual components of the eMBB, URLLC, and mMTC types of RAN slices in the NG-RAN architecture. These network simulation parameters are used to evaluate the

Table 6.1: The network simulation parameters, which are considered for the evaluation of the performance objectives of the eMBB types of RAN slices of the proposed mapping solution in this thesis.

Simulation Parameter	Value
Topology	Oxford
Number of I-PoPs	20
Number of links	26
Communication model	0.01
Computation model	0.001
No. of hidden layers	2
Hidden layer	256 Neurons
Mini-batch size	64
Discounting factor	0.99
Minimum memory size	80
Size of replay memory	100000
Optimizer	Adam
Beta	0.5
Epsilon	0.95
Loss function	mean square error (MSE)
Activation function	ReLu
Maximum number of episode	50000
Minimum data rate for eMBB end user	1000 Kbps

performance objectives of the mapping system model, which are discussed in the previous chapter. Most of the network simulation parameters considered in this thesis are in compliance with the 3GPP specifications, which have recently standardized in Release 15, Release 16, Release 17, and Release 18. The RAN slice specific parameters are provided in their respective tables. However, some network simulations parameters are mutual in all of these three use-case scenarios. Among them, we assume that each end-user is connected with only one gNB at a give time interval, each end-user access only a single type of a RAN slice at a given time, each gNB covers a 500 m circular area, there are totally five gNBs in the selected geographical region, and the total coverage area considered in our simulation is 2500 square meters. In addition, we considered two fully connected layers, which are hidden, for the training of the Q-learning algorithms in the simulation of the three types of RAN slices in the NG-RAN architecture. Lastly, we also assume that all end-users in the selected coverage area of each type of a RAN slice are uniformly distributed.

In the following, we discuss the parameters of the eMBB, URLLC, and mMTC types of RAN slices, respectively.

The eMBB types of RAN slices support a stable high data rate connection with high reliability, moderate mobility, high device density, and low packet error rate on the order of 10^{-3}. The eMBB devices send large data payloads in the uplink direction to a given gNB at a certain time. Based on these requirements, we considered a list of simulation parameters, along with their corresponding values, related to the eMBB types of RAN slices in Tab. 6.1. These parameters are used to train the mapping model in the context of the eMBB types of RAN slices.

The URLLC types of RAN slices support data traffic with very high reliability (e.g., packet error rate around or lower than 10^{-5}) and very low latency transmission of small size data

Table 6.2: The network simulation parameters, which are considered for the evaluation of the performance objectives of the URLLC types of RAN slices of the proposed mapping solution in this thesis.

Simulation Parameter	Value
Topology	IBM
Number of I-PoPs	18
Number of links	26
Communication model	0.01
Computation model	0.001
No. of hidden layers	2
Hidden layer	256 Neurons
Mini-batch size	64
Discounting factor	0.96
Minimum memory size	80
Size of replay memory	10000
Optimizer	Adam
Beta	0.5
Epsilon	0.98
Loss function	MSE
Maximum number of episode	2500
Activation function	ReLu
Maximum latency for URLLC end user	10 ms

payloads. Due to these requirement, the URLLC traffic must be accommodated at the edge. On the basis of these requirements, we considered a list of simulation parameters, along with their corresponding values, related to the URLLC types of RAN slices in Tab. 6.2. These parameters are used to train the mapping model in the context of the URLLC types of RAN slices.

The mMTC types of RAN slices provide connections to an extremely large number of devices that are intermittently active and send a small size of data payloads to a gNB. These devices are either stationary or move in a very low speed, requiring a low data rate and a low packet error rate on the order of 10^{-1}. According to these performance requirements, the we considered a list of simulation parameters, along with their corresponding values, related to the mMTC types of RAN slices in Tab. 6.3. These parameters are used to train the mapping model in the context of the mMTC types of RAN slices.

6.2 Simulation Results

In this subsection, we discuss the results we obtained through the use of the Q-learning algorithm in the mapping process of the virtual components of the eMBB, URLLC, and mMTC types of RAN slices onto the underlying virtual components in the NG-RAN architecture. To accomplish this, we consider varying numbers of episodes for each simulation of each type of RAN slice. We also discuss the results related to the allocation of virtual resources of different types for the three types of RAN slices in different scenarios. Following that, we discuss the results obtained related to the SLA, with respect to various metrics, signed between the tenant and the service provider. The SLA defines the metrics and imposes penalties if the service provider fails to provide the desired performance.

Table 6.3: The network simulation parameters, which are considered for the evaluation of the performance objectives of the mMTC types of RAN slices of the proposed mapping solution in this thesis.

Simulation Parameter	Value
Topology	UniNet
Number of I-PoPs	13
Number of links	25
Communication model	0.01
Computation model	0.001
No. of hidden layers	2
Hidden layer	256 Neurons
Mini-batch size	32
Discounting factor	0.93
Minimum memory size	70
Size of replay memory	1000
Optimizer	Adam
Beta	0.5
Epsilon	0.91
Loss function	MSE
Activation function	ReLu
Maximum number of episode	2500
Minimum number of UEs of each mMTC RAN slice	100

6.2.1 eMBB Types of RAN Slices

To this end, we thoroughly defined the simulation environment, network topology, and simulation parameters for each type of RAN slice in the NG-RAN architecture. In this subsection, we discuss the results we obtained related to the eMBB RAN slices. Hence, we begin by discussing the results regarding the four scenarios of the VNFs that are shared or dedicated among the eMBB RAN slices. Following that, we discuss how these four scenarios impact on the number of instantiated VNFs, the number of eMBB RAN slices, and the amount of virtual resources required for each of the eMBB RAN slices in the NG-RAN architecture. Then, we discuss the results obtained by employing Q-learning algorithm, considering various levels of episode, to the mapping of the VNFs of an eMBB RAN slice. Finally, we compare the QoS metrics to the penalty imposed by the service provider.

- **Fully Dedicated Scenario for eMBB RAN Slices:** In this scenario, each eMBB RAN slice has its own dedicated vCU and vDU in the I-PoP #1 and I-PoP #2, respectively. For example, if an NG-RAN infrastructure hosts ten eMBB RAN slices, then they will require ten vCUs and ten vDUs. From the view point of isolation and customization, this scenario is performing well, extremely easy to implement, and is in the best interest of the tenant. Nevertheless, it raises several concerns regarding the inefficiency of virtual resource allocation in the NG-RAN architecture. For some tenants of the eMBB RAN slices, which requires strict isolation, this scenario may fit very well. Hence, both the service provider and tenant must agree on the cost of the underutilized virtual resources in the I-PoPs. However, for those eMBB RAN slices that are hosted by the underlying infrastructure and require standards-defined isolation, this scenario may not optimal due to its under-utilization of a significant amount of

virtual compute and storage resources in the NG-RAN architecture. Finally, since each of the VNFCs may require a customized VM, therefore, the higher the number of slices are the harder their management and orchestration will be for the network operator. On the basis of the above advantages and disadvantages, as well as considering the sufficient bandwidth required for a large number of the end-users of the eMBB RAN slices, it is up to the service provider and tenant to decide whether or not this scenario is fulling their demands and requirements.

- **Shared vCU Scenario for eMBB RAN Slices:** In this scenario, the vCU is shared among several eMBB RAN slices in I-PoP #1, however, the vDUs are customized to all the eMBB RAN slices in I-PoP #2, in the NG-RAN architecture. For example, if there are ten eMBB RAN slices to be hosted in the NG-RAN architecture, then there is a need for one vCU and ten vDUs to be instantiated. In comparison to the previous scenario, this scenario is less efficient in terms of customization and isolation in I-PoP #1 and similar to that of I-PoP #2. Because, a single VNF is shared among several eMBB RAN slices in I-PoP #1 and similar number of vDUs are installed in I-PoP #2. However, it may be resource-efficient at least in I-PoP #1 in comparison to the previous scenario due to the fact that all the resources allocated to the vCU are shared among the slices. It also reduces the management and orchestration burden on the NFV-MANO in the I-PoP #1. Because only a single VNF is instantiated and its management and orchestration require less operations in comparison to several vCUs instantiated in the previous scenario. However, the management and orchestration of virtual resources, as well as the vDUs, in I-PoP #2 is similarly complicated to the first scenario. Based on the above characteristics, the MVNO may decide whether or not to agree with the operator on the instantiating of an eMBB RAN slice for its end-users.

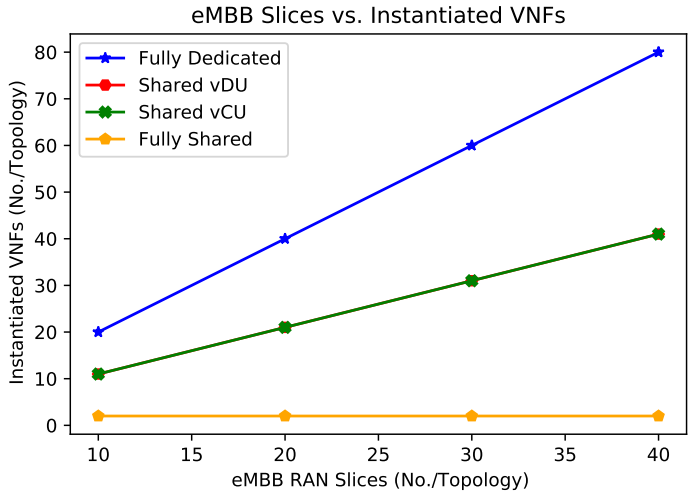

Figure 6.4: The number of instantiated VNFs in the Oxford network topology versus the number of eMBB RAN slices, taking into account fully dedicated, shared vCU, shared vDU, and fully shared scenarios.

- **Shared vDU Scenario for eMBB RAN Slices:** This scenario is similar to the previous (second) scenario. In this scenario, every eMBB RAN slice has its own customized vCU, however, the vDU is shared among all the eMBB RAN slices in the NG-RAN architecture. For instance, if an NG-RAN architecture is expected to host ten

eMBB RAN slices, then it shall instantiates ten vCUs and one vDU. Based on this, the customization and isolation levels are higher in the I-PoP #1 and less in I-PoP #2 due to the fact that multiple vDUs and and a single vCU are instantiated in both of the I-PoPs for a number of eMBB RAN slices, respectively. Furthermore, the management and orchestration operations in I-PoP #1 are more complicated than those of I-PoP #2. In contrary to the second scenario, this scenario is less resource-efficient in the I-PoP #1 and high resource-efficient in the I-PoP #2. Finally, considering the above pros and cons, it is up to the operator and tenant to decided whether or not to agree on instantiating of an eMBB RAN slice in such a scenario.

- **Fully Shared Scenario for eMBB RAN Slices:** In this scenario, both the vCU and vDU are shared among the eMBB RAN slices. For instance, if an NG-RAN architecture hosts ten eMBB RAN slices, then only one vCU and one vDU shall be instantiated by the NFV-MANO. From the viewpoint of customization and isolation, this scenario is less optimal in comparison to all three scenarios discussed in the above. Because, only two VNFs will be instantiated and shared among a number of RAN slices. Nevertheless, it is a high resource-efficient scenario when compares to the previous three scenarios due to the fact that all virtual resources of the two VNFs are optimally shard among the eMBB RAN slices. This scenario might be in the favor operator due to its efficiency in the resource allocation. However, it may not perform very well due to isolation and customization. Similar to the above scenarios, we let the operator and tenant decide whether or not to agree on instantiating of such an eMBB RAN slice in the NG-RAN architecture. Finally, the management and orchestration operations are complicated in both of the I-PoPs in the NG-RAN architecture.

Discussion on the Four Scenarios for eMBB RAN Slices

To demonstrate the efficiency of the above four scenarios for the eMBB types of RAN slices, we performed a simulation in order to study the number of instantiated VNFs versus the number of supported RAN slices in the NG-RAN architecture. The results of the simulation are illustrated in Fig. 6.4. It demonstrates that there is a linear relationship between two variables. The results reveals that, from the viewpoint of the number of instantiated VNFs, the "fully shared" scenario is the most efficient and the "fully dedicated" is the less efficient scenario. The remaining two scenarios can be interpreted accordingly in Fig. 6.4.

- **Instantiated VNFs for eMBB RAN Slices:** The number of instantiated VNFs in an I-PoP is extremely important parameter to be considered when mapping a RAN slice onto the underlying infrastructure. The more VNFs are instantiated by an I-PoP, the more complicated it becomes for a VIM to manage and orchestrate the required resources throughout their life cycles. In addition, the fewer VNFs are instantiated the fewer virtual resources will be required, thereby resulting in efficiently allocating of the underlying virtual resources. These advantages lead to reduce the energy consumption, lower the total cost and time of deployment of a RAN slice, and increase the performance of a RAN slice, among other tings. However, this parameter also comes at the cost of reducing the level of customization and isolation in the NG-RAN architecture.

- **Number of eMBB RAN Slices:** We also consider the number of instantiated RAN slices in NG-RAN as one of the parameters to be simulated during the simulation period. The more RAN slices an NG-RAN hosts with limited resources the more

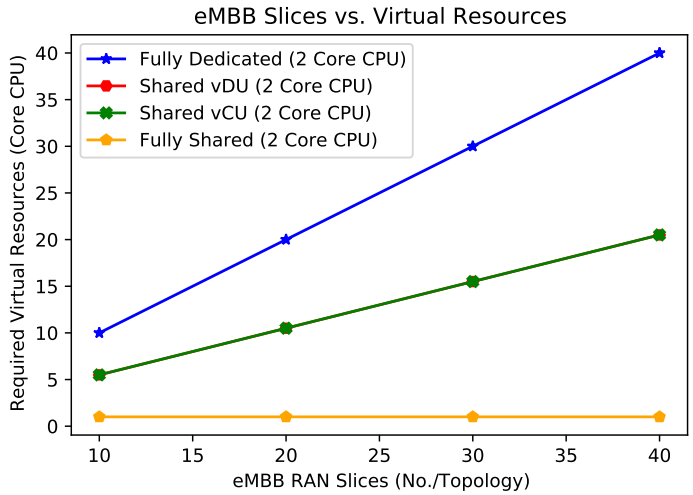

Figure 6.5: The required amount of virtual resources in the Oxford network topology versus the number of eMBB RAN slices, taking into account fully dedicated, shared vCU, shared vDU, and fully shared scenarios for a two-core CPU.

resource efficient it will be considered for a 5G mobile network. Furthermore, it will also serve a large number of end-users or devices of a vertical industry. This parameter has been evaluated in the context of Oxford topology for eMBB RAN slices against several parameters that will be discussed later in NG-RAN.

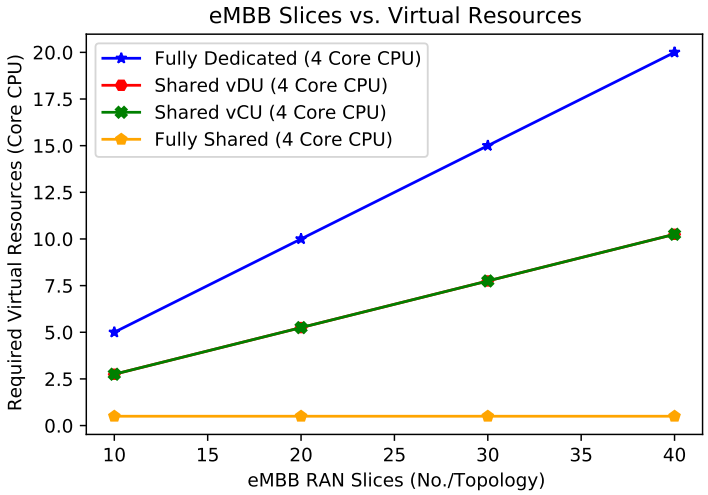

Figure 6.6: The required amount of virtual resources in the Oxford network topology versus the number of eMBB RAN slices, taking into account fully dedicated, shared vCU, shared vDU, and fully shared scenarios for a four-core CPU.

- **Required Virtual Resources for eMBB RAN Slices:** One of the most important parameters that we are attempting to consider during the simulation period is the required amount of virtual resources for the eMBB RAN slices in the Oxford topology in the NG-RAN architecture. This parameter is evaluated in the context of four scenarios, discussed above in the thesis, against the number of eMBB RAN slices in the NG-RAN

architecture. These virtual resources are considered in 2 Core CPU, 4Core CPU, 8 Core CPU, and 16 Core CPU scenarios.

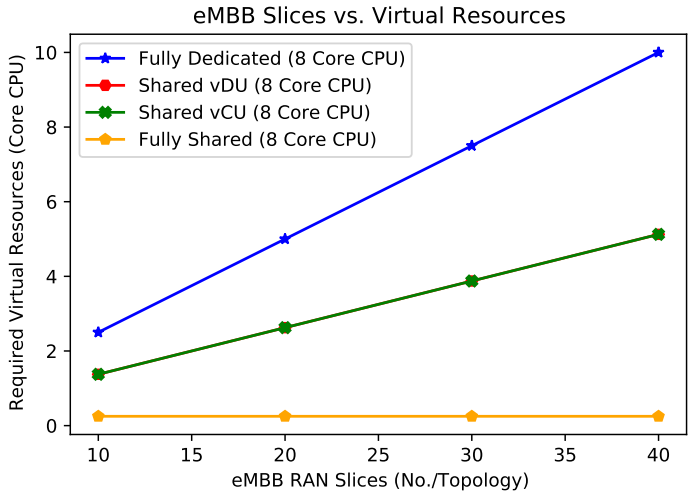

Figure 6.7: The required amount of virtual resources in the Oxford network topology versus the number of eMBB RAN slices, taking into account fully dedicated, shared vCU, shared vDU, and fully shared scenarios for an eight-core CPU.

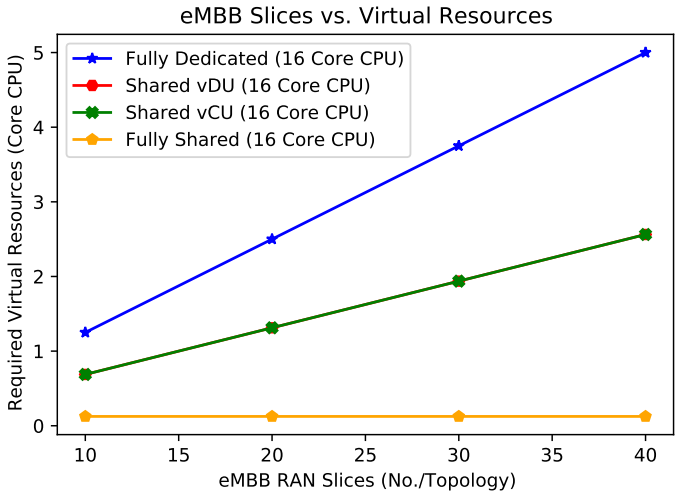

Figure 6.8: The required amount of virtual resources in the Oxford network topology versus the number of eMBB RAN slices, taking into account fully dedicated, shared vCU, shared vDU, and fully shared scenarios for a sixteen-core CPU.

Discussion on the Four Scenarios for eMBB RAN Slices

Considering the above two parameters (namely, number of eMBB RAN slices and required virtual resources), we conduct simulations in fully dedicated, shared vDU, shared vCU, and fully shared scenarios. The four types of simulations are executed against various types of underlying virtual resources, including 2 Core CPU (Fig. 6.5), 4 Core CPU (Fig. 6.6), 8 Core CPU (Fig. 6.7), and 16 Core CPU (Fig. 6.8). The results obtained in these four figures

indicate that the more a CPU contains Cores the less amount of servers we need in an I-PoP for hosting the RAN slices. And the less Cores a CPU contain in server the more servers we need to host the eMBB types of RAN slices.

6.2.2 The Q-Learning Algorithm for the eMBB RAN Slices

Considering the values of the simulation parameters, provided in the previous sections, we discuss the results we obtained following the simulations we have executed for the eMBB RAN slices in the context of Oxford topology in the NG-RAN architecture. We first discuss the training of the Q-learning model customized for eMBB RAN slices. Following that, we discuss the obtained results and evaluate the performance of the mapping of the eMBB RAN slices in the NG-RAN architecture.

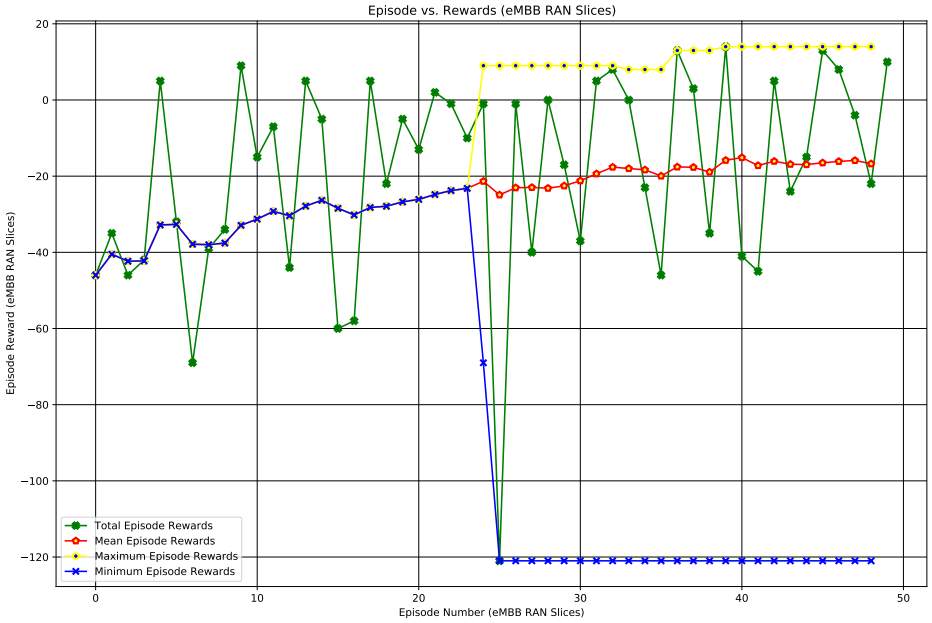

Figure 6.9: Total rewards collected versus episode number (e.g., 50) during the training phase of the agent for the eMBB RAN slices in the Oxford topology.

Model Training for eMBB RAN Slices

To train the Q-learning algorithm for the eMBB RAN slices, we consider two important parameters of a Q-learning algorithm, the episode number and the episode reward. These two parameters are evaluated in four different scenarios for eMBB RAN slices: total episode rewards, mean episode rewards, maximum episode rewards, and minimum episode rewards. We have evaluated the proposed Q-learning algorithm for the two above parameters under five different episode numbers in order to achieve an optimal rewards for the mapping of the eMBB RAN slice. These five different episode numbers are: 50, 250, 500, 2500, and 50000. The results obtained for the five types of episode number are shown in Fig. 6.9, Fig. 6.10, Fig. 6.11, Fig. 6.12, and Fig. 6.13, respectively.

Model Evaluation for eMBB RAN Slices

Once the simulation has been executed for the eMBB type of RAN slices in the Oxford topology, we realized that in the case where the episode number is 50, the performance of

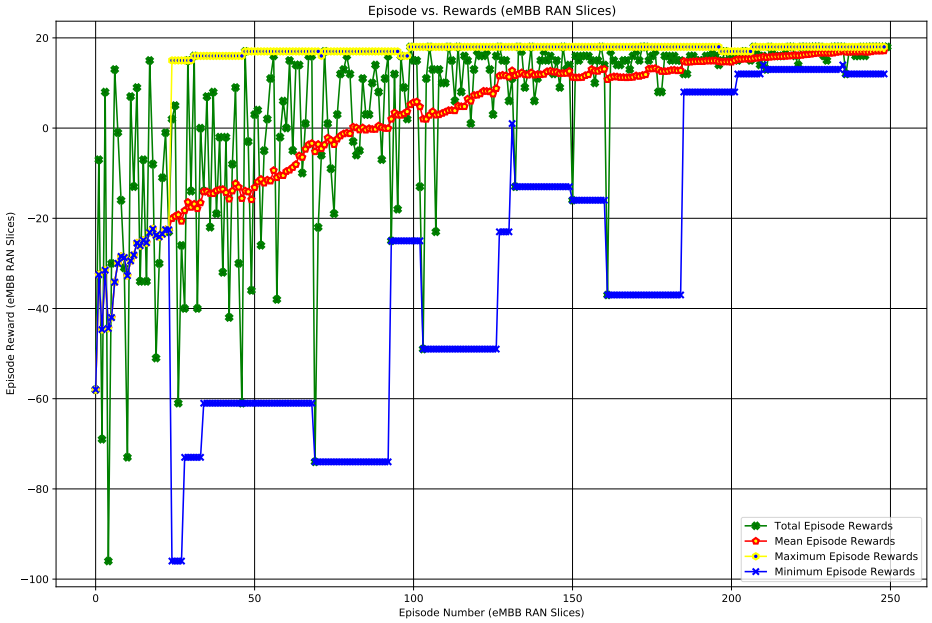

Figure 6.10: Total rewards collected versus episode number (e.g., 250) during the training phase of the agent for the eMBB RAN slices in the Oxford topology.

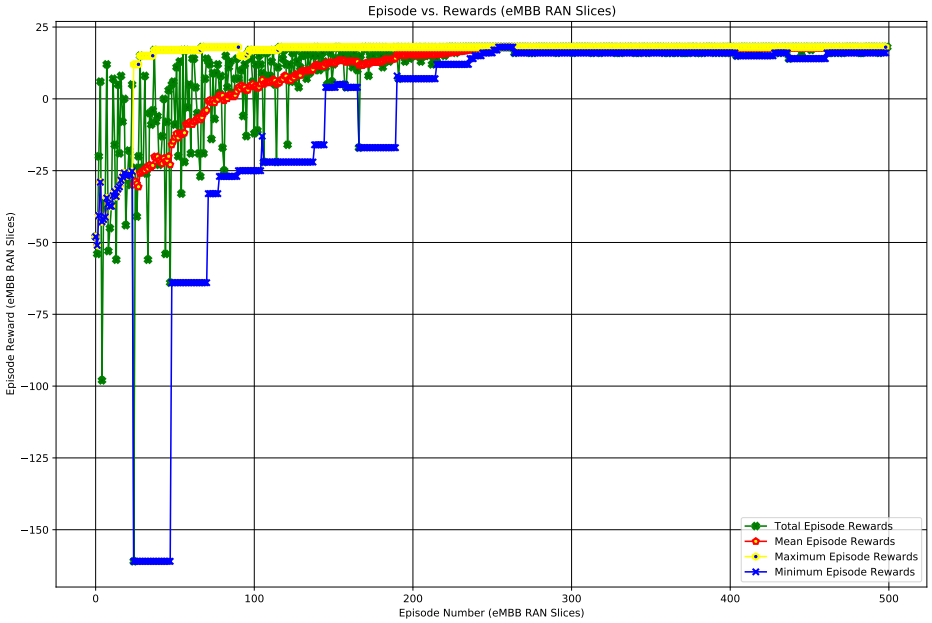

Figure 6.11: Total rewards collected versus episode number (e.g., 500) during the training phase of the agent for the eMBB RAN slices in the Oxford topology.

the learning process of the model is extremely worse, as illustrated in Fig. 6.9. The reason for this is that the learning process in this period of time is mostly based on the random exploration with a large epsilon value, which is how this period of learning is thought of. This fact is applicable to four types of episodes we assumed for the eMBB type of RAN slices. Following 50 episode, we exceed the number of episode to 250, as illustrated in Fig. 6.10. The results in this figure demonstrate that performance of the model has slightly increased in comparison of that previous figure. Furthermore, we continue increasing the number of episodes to 500. The results obtained for this simulation is illustrated in Fig. 6.11 show that the model perform better than that of illustrated in the previous period. We also set the number of episodes to 2500 as shown in Fig. 6.12. The obtained results shown in this figure

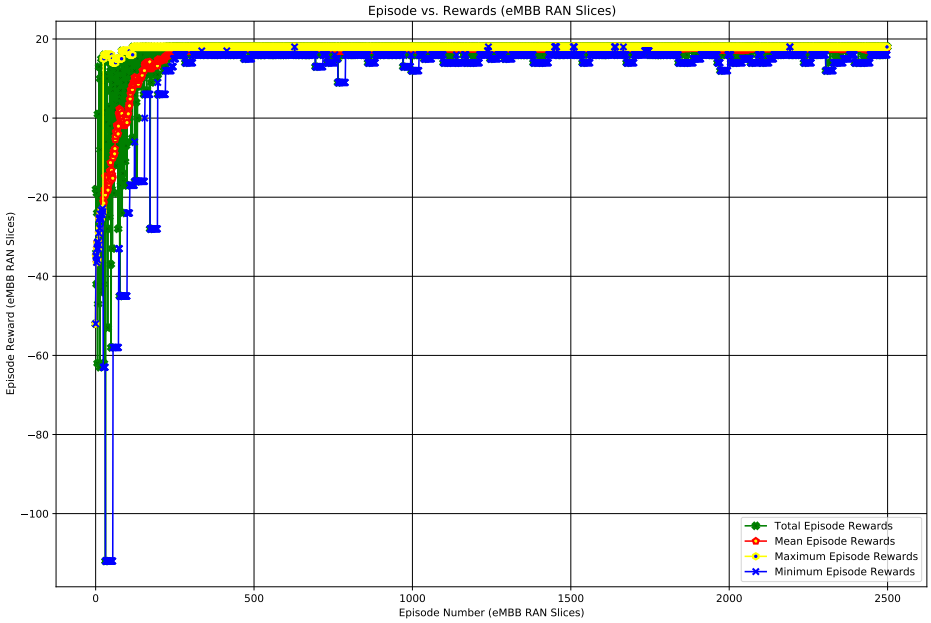

Figure 6.12: Total rewards collected versus episode number (e.g., 2500) during the training phase of the agent for the eMBB RAN slices in the Oxford topology.

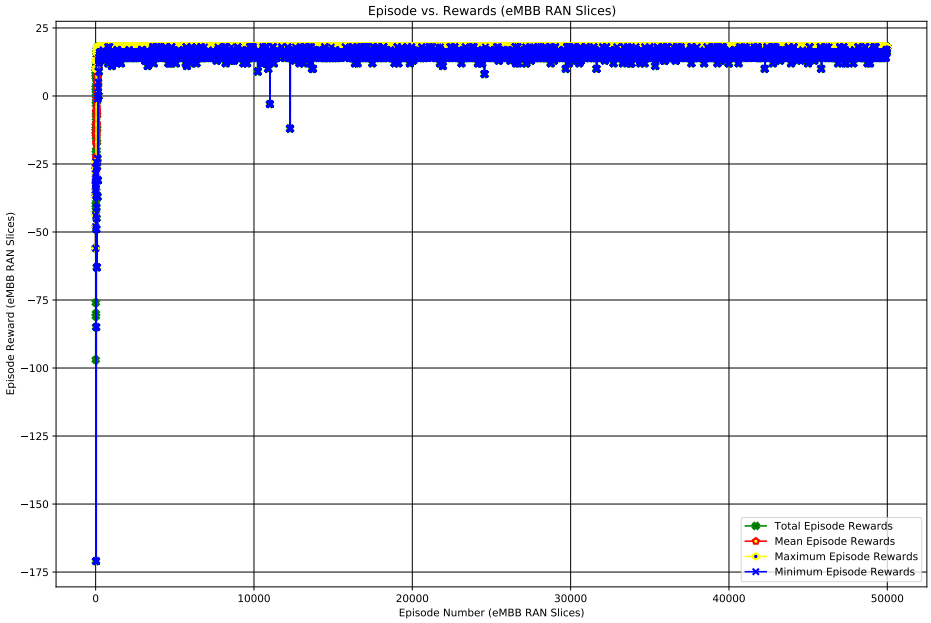

Figure 6.13: Total rewards collected versus episode number (e.g., 50000) during the training phase of the agent for the eMBB RAN slices in the Oxford topology.

illustrate that the model outperforms than all previous number of episodes. Finally, the Q-learning algorithm related to the mapping problem of eMBB type of RAN slices in the NG-RAN architecture converges to an optimum model after setting the number of episodes to 50000, as illustrated in Fig. 6.13.

6.2.3 The Evaluation of SLA Metrics for eMBB RAN Slices

Finally, we also evaluate the performance of a number of QoS metrics against the penalty during the lifetime of an eMBB RAN slice in the Oxford topology. The performance metrics include availability, device density, reliability, bandwidth, and latency. We set different

performance downgrade values for each of the QoS metrics. The highest value is 100 while the lowest performance downgrade value is 98.4, which is shown in Fig. 6.14. The figure demonstrates that when the performance of a metric decreases the model imposes a certain degree of penalty on the service provider. This value is almost certain till 99.2. However, if the performance of the QoS metric decreases more than this value, the model doubles the penalty and impose it on the service provider. Once it reaches 98.4 %, it is considered the worse case scenario where the performance of an eMBB RAN slice is at its lowest possible level. The tenant shall regularly checks the frequency of performance downgrade. If it continuously decreases, the tenant may decided to change the service provider for the sake of a better QoS.

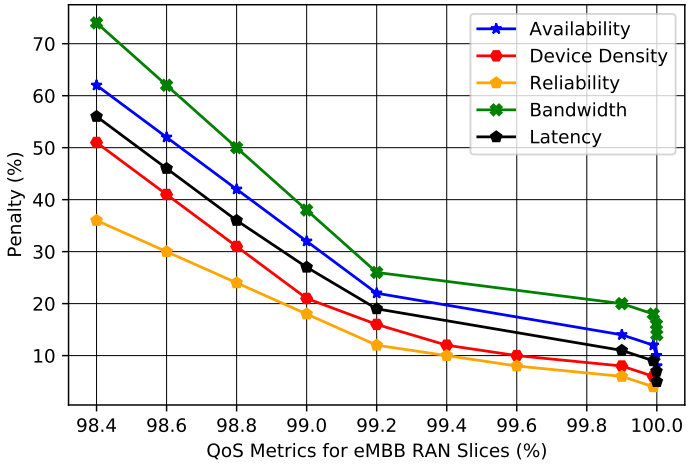

Figure 6.14: The evaluation of QoS metrics in the Oxford network topology for eMBB RAN slices versus penalty imposed on the service provider, taking into account fully dedicated, shared vCU, shared vDU, and fully shared scenarios.

6.2.4 URLLC Types of RAN Slices

In addition to the eMBB RAN slices, we have also defined the environment, network topology, and parameters for the simulation of URLLC type of RAN slices in the NG-RAN architecture. On the basis of these simulation metrics and parameters, in this subsection, we discuss the results we obtained regarding the URLLC type of RAN slices. To that end, we first start by discussing the results we obtained regarding the four types of scenarios of the VNFs that are shared or dedicated among the URLLC type of RAN slices in the NG-RAN architecture. Then, we discuss how these four types of scenarios have a significant impact on the number of instantiated VNFs, the number of URLLC type of RAN slices, and the amount of virtual resources required for each of the URLLC RAN slices in the NG-RAN architecture. Following that, we discuss the results we have obtained by employing Q-learning algorithm, considering various levels of episode, to the mapping of the VNFs of an URLLC type of a RAN slice. Finally, we compare the QoS metrics to the penalty imposed by the service provider in order to deeply study the SLA signed between the two parties.

- **Fully Dedicated Scenario for URLLC RAN Slices:** This is the first scenario, where each URLLC type of RAN slice has its own dedicated vCU and vDU in the I-PoP #1 and I-PoP #2, respectively. For example, if an NG-RAN infrastructure hosts ten URLLC type of RAN slices, then these slices will require ten vCUs and ten vDUs in the underlying infrastructure. From the view point of resource isolation and customization, this scenario is performing extremely optimal, quite easy to deploy, and is in the

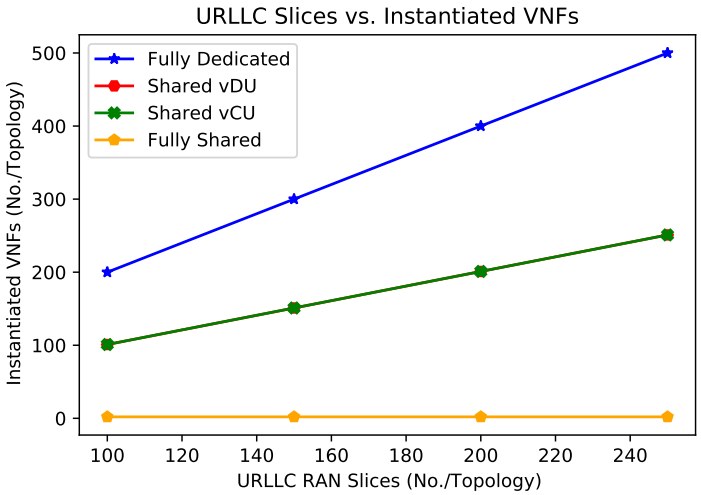

Figure 6.15: The number of instantiated VNFs in the IBM network topology versus the number of URLLC RAN slices, taking into account fully dedicated, shared vCU, shared vDU, and fully shared scenarios.

best interest of the tenant and MVNO. However, this scenario may raise a number of concerns regarding the inefficiency of virtual resource allocation in NG-RAN. For some tenants of the URLLC type of RAN slices, which demands extremely low latency and higher reliability, this scenario may seem to be an optimal option. Therefore, both the service provider and tenant (or MVNO) need to agree on the cost of the underutilized virtual resources in the I-PoPs. Nevertheless, for those URLLC type of RAN slices that are hosted by the underlying infrastructure and require standards-defined isolation and resource customization, this scenario may not seem to be optimal thanks to its under-utilization of a significant amount of virtual compute and storage resources in the underlying infrastructure in NG-RAN. Finally, since each of the VNFCs may require a customized VM in the PMs, hence, the higher the number of URLLC RAN slices are the difficult their management and orchestration tasks will be for the network operator. Based on these advantages and disadvantages of this scenario, as well as considering the required latency for delay-sensitive end-users of the URLLC type of RAN slices, it is up to the service provider and tenant to decide whether or not this scenario is fulfilling their business demands and technical requirements.

- **Shared vCU Scenario for URLLC RAN Slices:** This is the second scenario, in which the vCU is shared among a number of URLLC type RAN slices in I-PoP #1. Nevertheless, the vDUs are customized to all URLLC type RAN slices that are hosted by the I-PoP #2 in the NG-RAN architecture. For instance, if there are ten URLLC type RAN slices to be hosted in NG-RAN in the underlying infrastructure, then there is a strong need for a single vCU and ten vDUs to be instantiated. In comparison to the first scenario, this scenario is less efficient in terms of customization and isolation in I-PoP #1. However, it is similar in terms of customization and isolation to that of I-PoP #2. Because, a single VNF is shared among several URLLC RAN slices in I-PoP #1 and similar number of vDUs are installed in I-PoP #2. In addition, it may be resource-efficient at least in I-PoP #1 in comparison to the first scenario. This is because all the virtual resources allocated to the vCU in I-PoP #1 are shared among the URLLC type RAN slices. It also reduces the management and orchestration burden on the NFV-MANO in the I-PoP #1. Since only a single VNF is instantiated and its

management and orchestration require less operations in comparison to several vCUs instantiated in the first scenario. However, the management and orchestration of virtual compute and storage resources, as well as the vDUs, in I-PoP #2 is similarly complicated to the first scenario. Based on the above advantages and disadvantages, the MVNO may decide whether or not to agree with the network operator on the instantiating of an URLLC type of RAN slices for its latency-demanding user equipment.

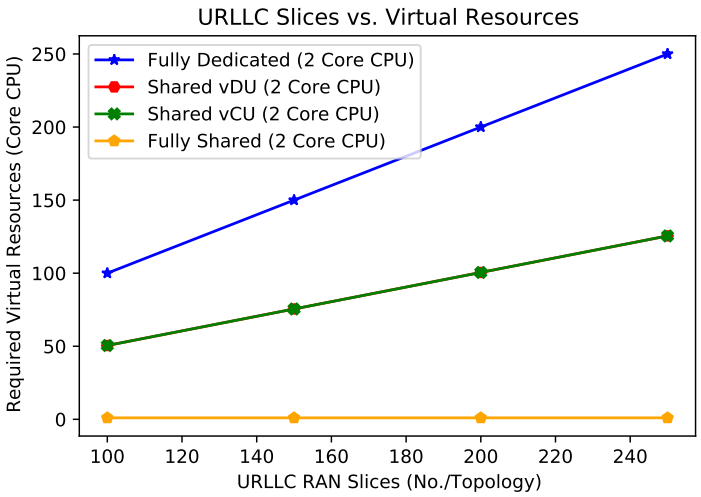

Figure 6.16: The required amount of virtual resources in the IBM network topology versus the number of URLLC RAN slices, taking into account fully dedicated, shared vCU, shared vDU, and fully shared scenarios for two-core CPU.

- **Shared vDU Scenario for URLLC RAN Slices:** This is the third scenario, which is similar to the second scenario. In this scenario, every URLLC RAN slice has its own customized vCU, however, the vDU is shared among all the URLLC type RAN slices in the NG-RAN architecture. For example, if an NG-RAN architecture is expected to host ten URLLC type RAN slices, then it shall instantiate ten vCUs and one vDU. Based on this assumption, the customization and isolation levels are higher in the I-PoP #1 and less in I-PoP #2. This is because that multiple vDUs and and a single vCU are instantiated in both of the I-PoPs for a number of URLLC RAN slices, respectively. Furthermore, the management and orchestration operations in I-PoP #1 are more complicated than those of I-PoP #2. In contrary to the second scenario, this scenario is less resource-efficient in the I-PoP #1 and high resource-efficient in the I-PoP #2. Finally yet importantly, considering the above advantages and disadvantages, it is up to the network operator and tenant to decided whether or not to agree on instantiating of an URLLC RAN slice in such a scenario.

- **Fully Shared Scenario for URLLC RAN Slices:** This is the fourth scenario, in which both the vCU and vDU are shared among the URLLC RAN slices in the underlying infrastructure. For example, if an NG-RAN hosts ten URLLC type RAN slices, then only one vCU and one vDU shall be instantiated by the NFV-MANO in the I-PoPs. From the viewpoint of customization and isolation, this scenario is less optimal in comparison to all three scenarios discussed in the above. Because, only two VNFs will be instantiated and shared among a number of RAN slices. Nevertheless, it is a high resource-efficient scenario when compares to the previous three scenarios due

to the fact that all virtual resources of the two VNFs are optimally shard among the URLLC RAN slices. This scenario might be in the favor operator due to its efficiency in the resource allocation. However, it may not perform very well due to isolation and customization. Similar to the above scenarios, we let the operator and tenant decide whether or not to agree on instantiating of such an URLLC RAN slice in NG-RAN. Finally, the management and orchestration operations are complicated in both of the I-PoPs in NG-RAN.

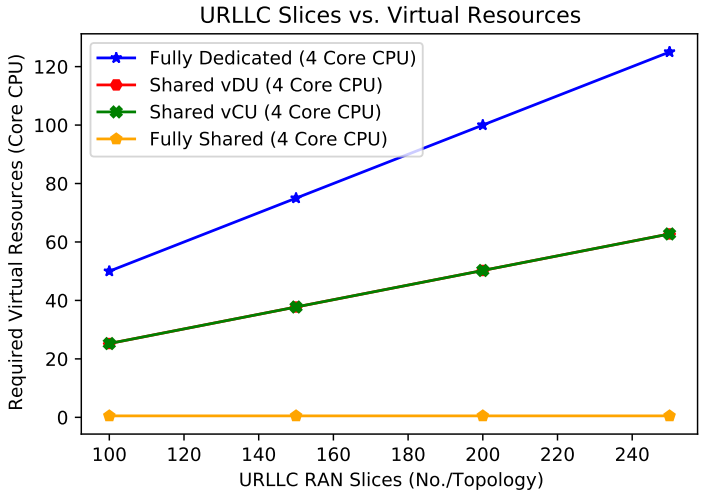

Figure 6.17: The required amount of virtual resources in the IBM network topology versus the number of URLLC RAN slices, taking into account fully dedicated, shared vCU, shared vDU, and fully shared scenarios for a four-core CPU.

Discussion on the Four Scenarios for URLLC RAN Slices

Following the discussion of the above four scenarios, in this section, we demonstrate the efficiency and effectiveness of the above four scenarios for the URLLC types of RAN slices in the underlying intelligent infrastructure. We performed a simulation in order to study the number of instantiated VNFs versus the number of supported URLLC type RAN slices in NG-RAN. The results of the simulation are illustrated in Fig. 6.15. It demonstrates that there is a linear relationship between two variables. The results reveals that, from the viewpoint of the number of instantiated VNFs, the "fully shared" scenario is the most efficient and the "fully dedicated" is the less efficient scenario for the hosted URLLC type RAN slices. The remaining two scenarios can be interpreted accordingly as shown in Fig. 6.15.

- **Instantiated VNFs for URLLC RAN Slices:** The number of instantiated VNFs for an URLLC RAN slice in an I-PoP is extremely vital parameter to be considered when mapping an URLLC type RAN slice onto the underlying infrastructure in the NG-RAN architecture. The more VNFs of an URLLC RAN slice are instantiated by an I-PoP, the more complicated it becomes for a VIM to manage and orchestrate the required resources throughout their life cycles in NG-RAN. Additionally, the fewer VNFs are instantiated for URLLC type RAN slices the fewer virtual resources will be required, thereby resulting in efficiently allocating of the underlying virtual resources. These advantages lead to reducing the energy consumption, lower the total cost and time of deployment of URLLC type RAN slices, and increase the performance of URLLC

type RAN slice, among other tings. However, this parameter also comes at the cost of reducing the level of customization and isolation in the NG-RAN architecture.

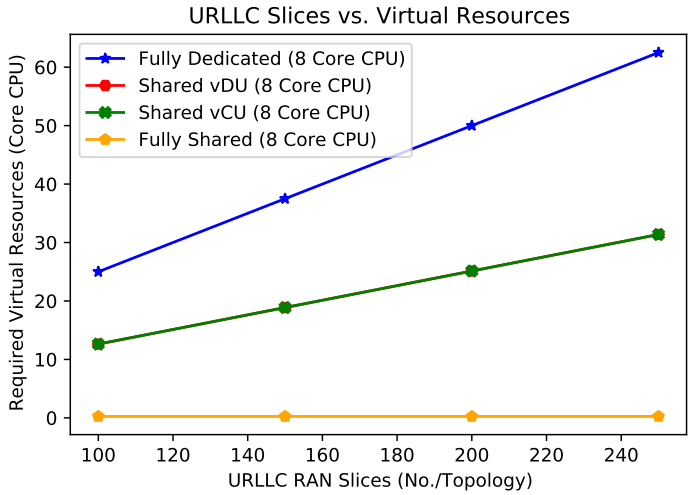

Figure 6.18: The required amount of virtual resources in the IBM network topology versus the number of URLLC RAN slices, taking into account fully dedicated, shared vCU, shared vDU, and fully shared scenarios for an eight-core CPU.

- **Number of URLLC RAN Slices:** In this simulation related to the URLLC type RAN slice, we also consider the number of instantiated URLLC type RAN slices in the NG-RAN architecture as one of the parameters to be simulated during the simulation period. It is a fact that the more URLLC type of RAN slices an NG-RAN architecture hosts with limited virtual resources the more resource efficient it will be considered for a 5G mobile network. Furthermore, it will also serve a large number of latency-demanding end-users or devices of a vertical industry. This parameter has been evaluated in the context of IBM topology for URLLC RAN slices against several parameters that will be discussed later in the NG-RAN architecture in this section.

- **Required Virtual Resources for URLLC RAN Slices:** In the context of latency-demanding RAN slices, one of the critical parameters that we are trying to consider during the simulation period is the required amount of virtual resources for the URLLC RAN slices in the IBM topology in the NG-RAN architecture. This parameter has been evaluated in the context of four scenarios, as discussed above in this thesis, against the number of URLLC type of RAN slices in the NG-RAN architecture. These virtual resources are considered in 2 Core CPU, 4Core CPU, 8 Core CPU, and 16 Core CPU scenarios similar to those of eMBB RAN slices.

Discussion on the Four Scenarios for URLLC RAN Slices

While considering the aforementioned two simulation parameters (namely, number of URLLC type RAN slices and the required amount virtual resources), we conduct our simulations in fully dedicated, shared vDU, shared vCU, and fully shared scenarios for the URLLC type RAN slices. These four types of simulations are executed against various types of underlying virtual resources in the underlying I-PoPs, including 2 Core CPU (Fig. 6.16), 4 Core CPU

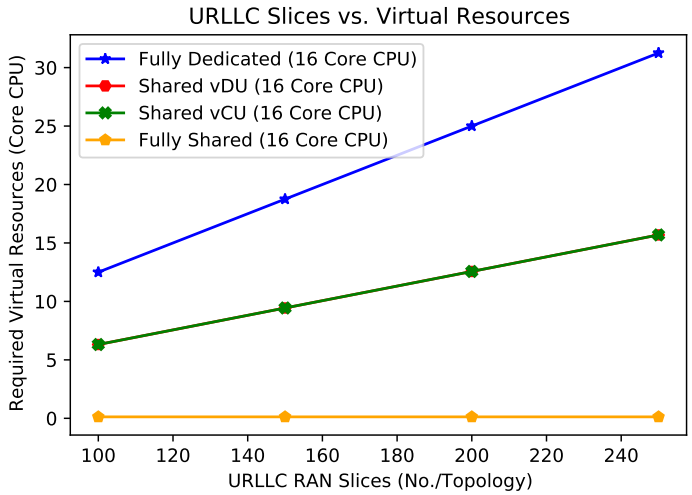

Figure 6.19: The required amount of virtual resources in the IBM network topology versus the number of URLLC RAN slices, taking into account fully dedicated, shared vCU, shared vDU, and fully shared scenarios for a sixteen-core CPU.

(Fig. 6.17), 8 Core CPU (Fig. 6.18), and 16 Core CPU (Fig. 6.19). The results obtained for the URLLC type RAN slices in these four figures indicate that the more a CPU contains Cores the less amount of servers we need in an I-PoP for hosting the URLLC type RAN slices. And the less Cores a CPU contain in server the more servers we need to host the URLLC types of RAN slices.

6.2.5 The Q-Learning Algorithm for the URLLC RAN Slices

Following the conditions and scenarios for the allocation of virtual resources of URLLC type of RAN slices in the NG-RAN architecture, in this section, we discuss the mapping aspects of the URLLC RAN slices by employing the Q-learning algorithms. While considering the values of the simulation parameters, provided in the previous sections of this chapter, we discuss the results we obtained following the simulations we have executed for the URLLC type of RAN slices in the context of IBM topology in the NG-RAN architecture. To accomplish this, we first discuss the training of the Q-learning model customized for URLLC RAN slices. Then, we discuss the obtained results and evaluate the performance of the mapping of the URLLC RAN slices in the NG-RAN architecture.

Model Training for URLLC RAN Slices

In order to optimally mapp the URLLC type of RAN slices, we first and foremost need to train the mapping model. To accomplish the training of the Q-learning algorithm for the URLLC type of RAN slices, we consider two important parameters of a Q-learning algorithm, namely the episode number and the episode reward. These two parameters are evaluated in four different scenarios for URLLC RAN slices: total episode rewards, mean episode rewards, maximum episode rewards, and minimum episode rewards. We have evaluated the proposed Q-learning algorithm for the two above parameters under five different episode numbers in order to achieve an optimal rewards for the mapping of the URLLC RAN slice. These five different episode numbers for URLLC type of RAN slices are: 50, 250, 500, 2500, and 50000. The results obtained for the five types of episode number are shown in Fig. 6.20, Fig.

6.21, Fig. 6.22, Fig. 6.23, and Fig. 6.24, respectively. We will interpret these results in the following sections.

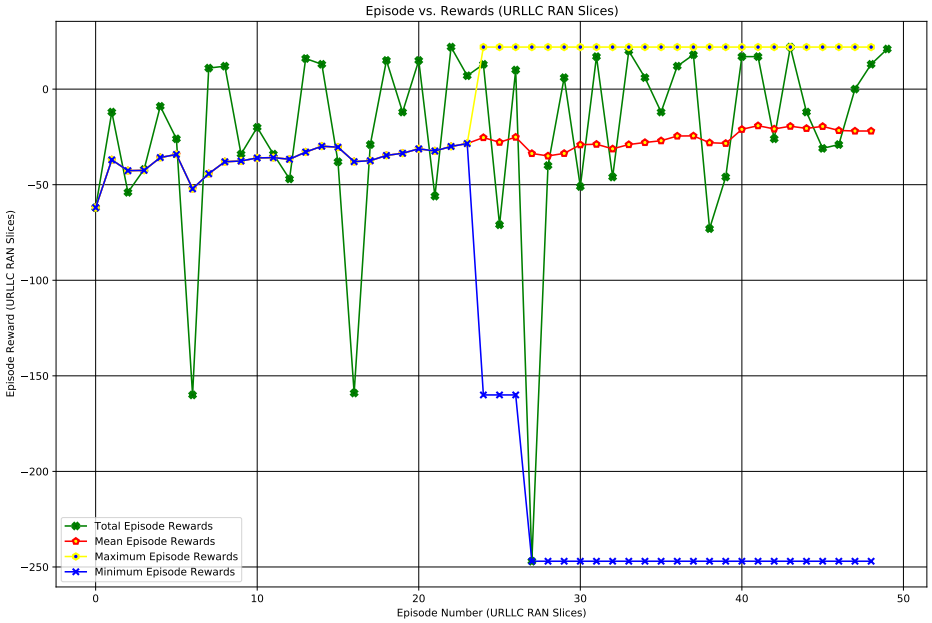

Figure 6.20: Total rewards collected versus episode number (e.g., 50) during the training phase of the agent for the URLLC RAN slices in the IBM topology.

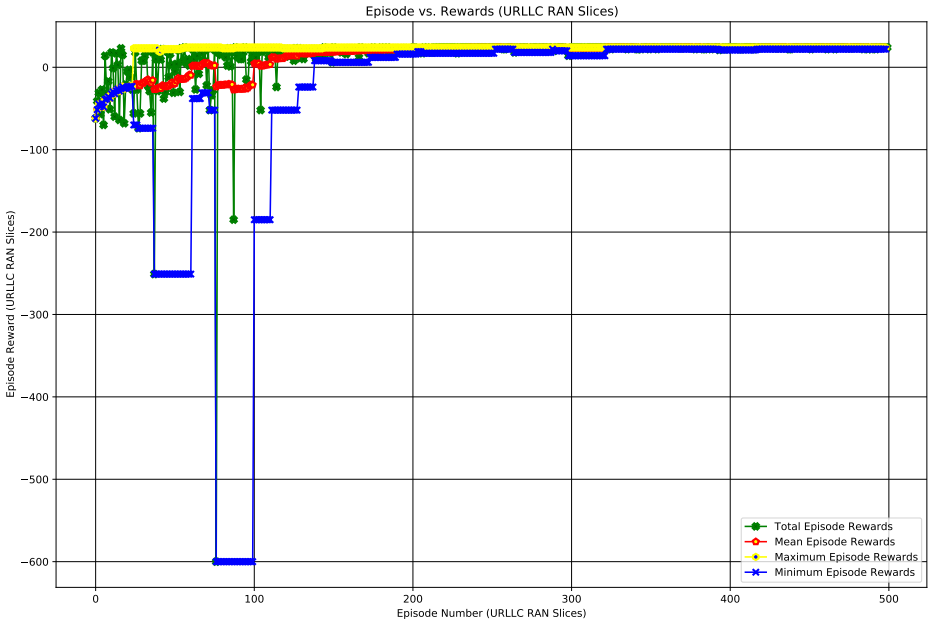

Figure 6.21: Total rewards collected versus episode number (e.g., 250) during the training phase of the agent for the URLLC RAN slices in the IBM topology.

Model Evaluation for URLLC RAN Slices

After the simulation has been executed for the URLLC type of RAN slices in the IBM topology, we realized that in the case where the episode number is 50, the performance of the learning process of the model is worse, as illustrated in Fig. 6.20. The reason for this is that the learning process in this period is mostly based on the random exploration with a large

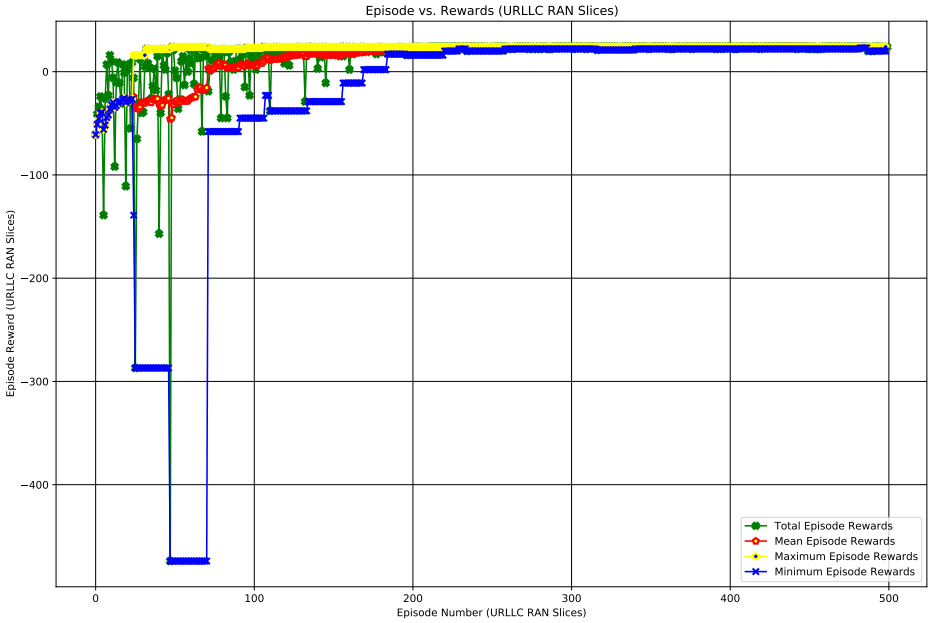

Figure 6.22: Total rewards collected versus episode number (e.g., 500) during the training phase of the agent for the URLLC RAN slices in the IBM topology.

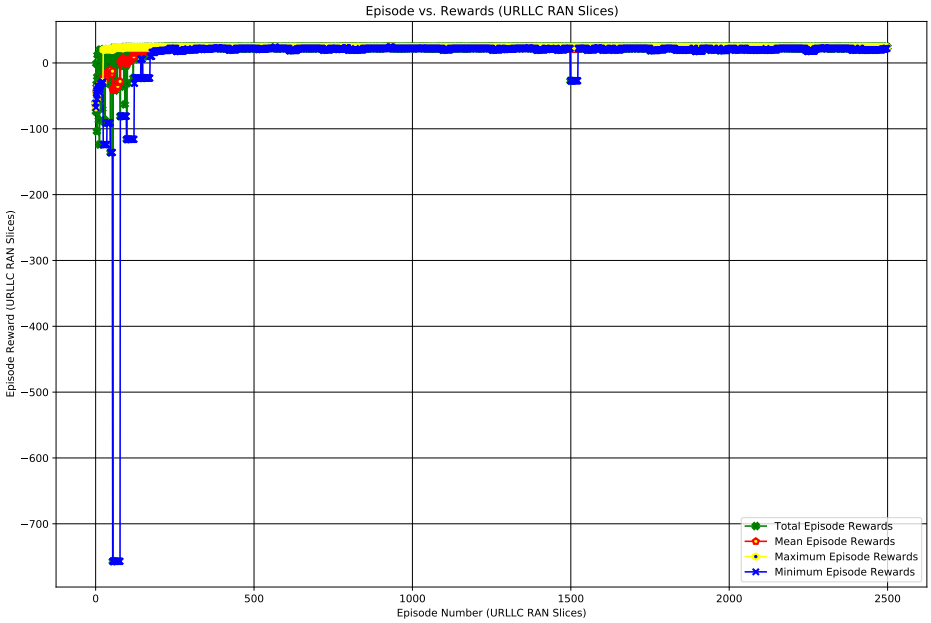

Figure 6.23: Total rewards collected versus episode number (e.g., 2500) during the training phase of the agent for the URLLC RAN slices in the IBM topology.

epsilon value. This fact is applicable to all four types of episodes we assumed for the URLLC type of RAN slices. Following that, we exceed the number of episode to 250, as illustrated in Fig. 6.21. The results in this figure demonstrate that performance of the model has slightly increased in comparison of that previous figure. Furthermore, we continuously increasing the number of episodes to 500. The results obtained for this simulation is illustrated in Fig. 6.22 show that the model performs better than that of illustrated in the previous period. We also set the number of episodes to 2500 as shown in Fig. 6.23. The obtained results shown in this figure illustrate that the model outperforms than all previous number of episodes. Finally, the Q-learning algorithm related to the mapping problem of URLLC type of RAN slices in the NG-RAN architecture converges to an optimum model after setting the number

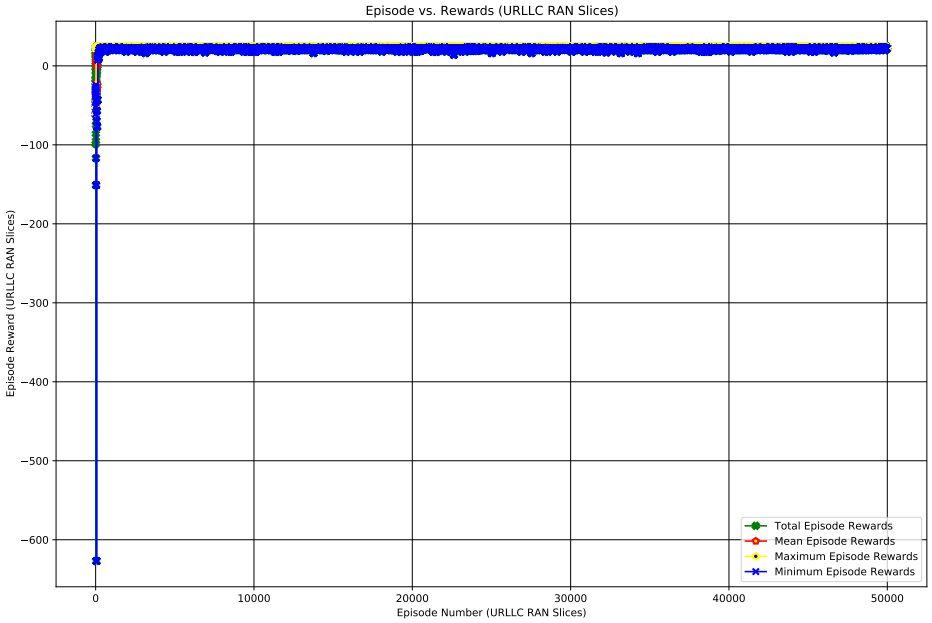

Figure 6.24: Total rewards collected versus episode number (e.g., 50000) during the training phase of the agent for the URLLC RAN slices in the IBM topology.

of episodes to 50000, as illustrated in Fig. 6.24.

6.2.6 The Evaluation of SLA Metrics for URLLC RAN Slices

Lastly, we also evaluate the performance of several QoS metrics against the penalty during the lifetime of an URLLC type of RAN slice in the IBM topology. The performance metrics include availability, device density, reliability, bandwidth, and latency, as shown in Fig. 6.25. We set different performance downgrade values for each of the QoS metrics. The highest value is 100 while the lowest performance downgrade value is 98.4, which is shown in Fig. 6.25. The figure demonstrates that when the performance of a metric decreases the model imposes a certain degree of penalty on the service provider. This value is almost certain till 99.2. However, if the performance of the QoS metric decreases more than this value, the model doubles the penalty and impose it on the service provider. Once it reaches 98.4 %, it is considered the worse case scenario where the performance of an URLLC RAN slice is at its lowest possible level. The tenant shall regularly checks the frequency of performance downgrade. If it continuously decreases, the tenant may decided to change the service provider for the sake of a better QoS.

6.2.7 mMTC Types of RAN Slices

In the previous section of this chapter, we have also defined the simulation environment, network topology, and network parameters for the simulation of the mMTC type of RAN slices in the NG-RAN architecture. Based on these simulation metrics and parameters, we thoroughly discuss the results we obtained regarding the mMTC type of RAN slices in this section. To accomplish this goal, we first begin by discussing the results we obtained regarding the four types of scenarios we have considered for the VNFs of the mMTC type of RAN slices in the NG-RAN architecture. Then, we discuss how these four types of scenarios have a significant impact on the number of instantiated VNFs, the number of mMTC type of RAN slices, and the amount of virtual resources required for each of the mMTC RAN

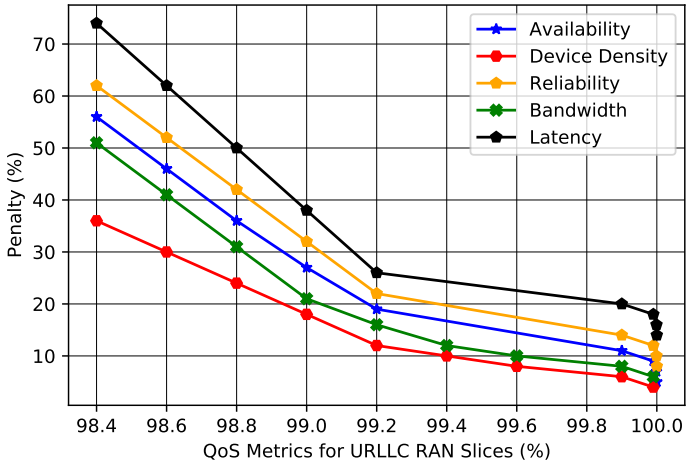

Figure 6.25: The evaluation of QoS metrics in the IBM network topology for URLLC RAN slices versus the penalty imposed on the service provider, taking into account fully dedicated, shared vCU, shared vDU, and fully shared scenarios.

slices in the NG-RAN architecture. Then, we discuss the results we have obtained through the use of Q-learning algorithm, considering various levels of episode, to the mapping of the VNFs of an mMTC type of a RAN slice. Finally, we compare the QoS metrics to the penalty imposed by the service provider in order to deeply study the SLA signed between the two parties during the lifetime of the requested mMTC type of RAN slice.

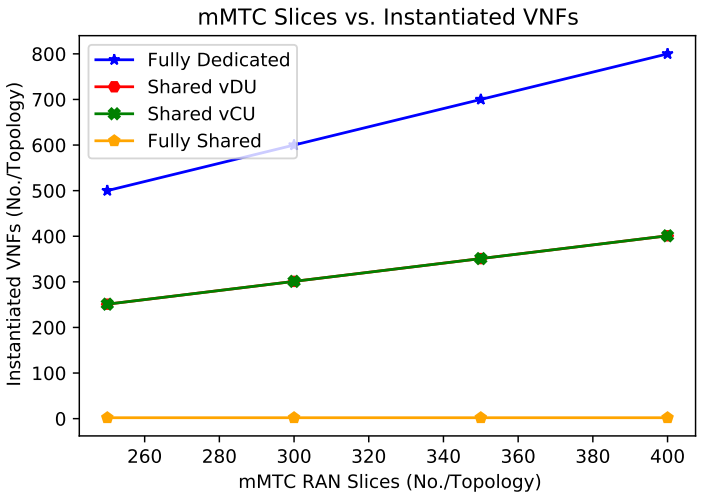

Figure 6.26: The number of instantiated VNFs in the UniNet network topology versus the number of mMTC RAN slices, taking into account fully dedicated, shared vCU, shared vDU, and fully shared scenarios.

- **Fully Dedicated Scenario for mMTC RAN Slices:** It is the first scenario, in which every mMTC type of RAN slice has a dedicated vCU and a dedicated vDU in I-PoP #1 and I-PoP #2, respectively. For example, if an NG-RAN infrastructure hosts ten mMTC type of RAN slices, then these RAN slices will require ten vCUs and ten vDUs in the underlying I-PoPs. From the resource isolation and customization perspectives, this scenario is performing extremely optimal, quite easy to deploy, and is in the best interest of the tenant and MVNO. Nevertheless, this scenario will raise

a number of concerns regarding the inefficiency of virtual resource allocation in the NG-RAN architecture. For those tenants of the mMTC type of RAN slices, which demands extremely large number of supporting services, this scenario may seem to be an optimal option. Therefore, both the service provider and tenant (or MVNO) need to agree on the total cost of the underutilized virtual resources in the I-PoPs. Nevertheless, for those mMTC type of RAN slices that are hosted by the underlying infrastructure and require standards-defined isolation and resource customization, this scenario may not seem to be optimal thanks to its under-utilization of a significant amount of virtual compute and storage resources in the underlying infrastructure in the NG-RAN architecture. Finally yet importantly, since each of the VNFCs may require a customized VM in the PMs, hence, the higher the number of mMTC RAN slices are the difficult their management and orchestration tasks will be for the network operator. Based on these advantages and disadvantages of this scenario, as well as considering the required latency for delay-sensitive end-users of the mMTC type of RAN slices, it is up to the service provider and tenant to decide whether or not this scenario is fulling their business demands and technical requirements for their supportive use-case.

- **Shared vCU Scenario for mMTC RAN Slices:** Among the four scenarios, it is the second choice, where the vCU is shared among a number of mMTC type RAN slices in I-PoP #1. Nevertheless, the vDUs are customized to all mMTC type RAN slices that are hosted by the I-PoP #2 in the NG-RAN architecture. For example, if there are ten mMTC type RAN slices to be hosted in the NG-RAN architecture in the underlying infrastructure, then there is a strong need for a single vCU and ten vDUs to be instantiated. In comparison to the first scenario, this scenario is less efficient in terms of customization and isolation in I-PoP #1. However, it is similar in terms of customization and isolation to that of I-PoP #2. Because, a single VNF is shared among several mMTC RAN slices in I-PoP #1 and similar number of vDUs are installed in I-PoP #2. In addition, it may be resource-efficient at least in I-PoP #1 in comparison to the first scenario. This is because all the virtual resources allocated to the vCU in I-PoP #1 are shared among the mMTC type RAN slices. It also reduces the management and orchestration burden on the NFV-MANO in the I-PoP #1. Since only a single VNF is instantiated and its management and orchestration require less operations in comparison to several vCUs instantiated in the first scenario. However, the management and orchestration of virtual compute and storage resources, as well as the vDUs, in I-PoP #2 is similarly complicated to the first scenario. Based on the above advantages and disadvantages, the MVNO may decide whether or not to agree with the network operator on the instantiating of an mMTC type of RAN slices for its latency-demanding user equipment in such a scenario.

- **Shared vDU Scenario for mMTC RAN Slices:** It is the third scenario, which is similar to the second scenario. In this scenario, every mMTC RAN slice has its own customized vCU, however, the vDU is shared among all the mMTC type of RAN slices in the NG-RAN architecture. For example, if an NG-RAN architecture is expected to host ten mMTC type of RAN slices, then it must instantiate ten vCUs and one vDU. Based on this assumption, the customization and isolation levels are higher in the I-PoP #1 and less in I-PoP #2. This is because that multiple vDUs and and a single vCU are instantiated in both of the I-PoPs for a number of mMTC RAN slices, respectively. Furthermore, the management and orchestration operations in I-PoP #1 are more complicated than those of I-PoP #2. In contrary to the second scenario, this

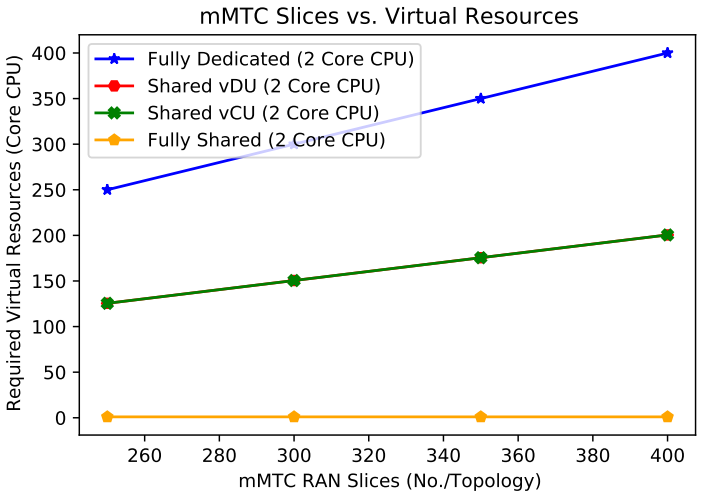

Figure 6.27: The required amount of virtual resources in the UniNet network topology versus the number of mMTC RAN slices, taking into account fully dedicated, shared vCU, shared vDU, and fully shared scenarios for a two-core CPU.

scenario is less resource-efficient in the I-PoP #1 and high resource-efficient in the I-PoP #2. Lastly, considering the above advantages and disadvantages, it is up to the network operator and tenant to decided whether or not to agree on instantiating of an mMTC RAN slice in such a scenario.

- **Fully Shared Scenario for mMTC RAN Slices:** This is the fourth scenario, in which both the vCU and vDU are shared among the mMTC RAN slices in the underlying infrastructure. For example, if an NG-RAN architecture hosts ten mMTC type RAN slices, then only one vCU and one vDU shall be instantiated by the NFV-MANO in the I-PoPs. From the viewpoint of customization and isolation, this scenario is less optimal in comparison to all three scenarios discussed in the above. Because, only two VNFs will be instantiated and shared among a number of RAN slices. Nevertheless, it is a high resource-efficient scenario when compares to the previous three scenarios due to the fact that all virtual resources of the two VNFs are optimally shard among the mMTC RAN slices. This scenario might be in the favor operator due to its efficiency in the resource allocation. However, it may not perform very well due to isolation and customization. Similar to the above scenarios, we let the operator and tenant decide whether or not to agree on instantiating of such an mMTC RAN slice in the NG-RAN architecture. Finally, the management and orchestration operations are complicated in both of the I-PoPs in the NG-RAN architecture.

Discussion on the Four Scenarios for mMTC RAN Slices

Once we have discussed the above four VNFs scenarios, in this section, we demonstrate the efficiency and effectiveness of the above four scenarios for the mMTC types of RAN slices in the underlying intelligent infrastructure. We performed a simulation in order to study the number of instantiated VNFs versus the number of supported mMTC type RAN slices in the NG-RAN architecture. The results of the simulation are illustrated in Fig. 6.26. This figure demonstrates that there is a linear relationship between two variables. The results reveals that, from the viewpoint of the number of instantiated VNFs, the “fully shared” scenario is

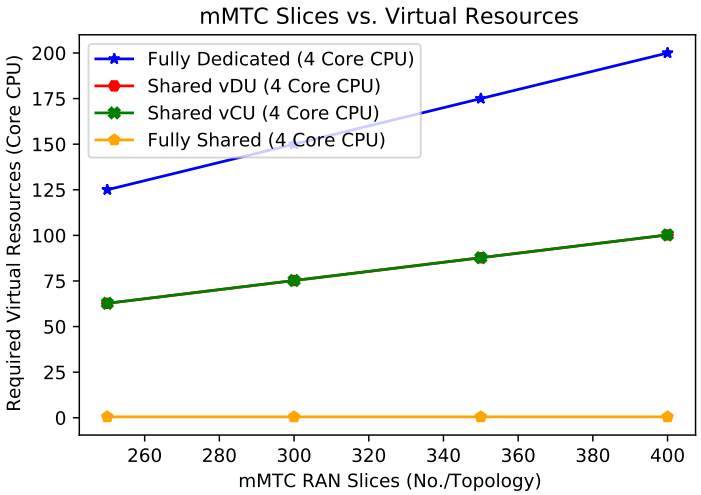

Figure 6.28: The required amount of virtual resources in the UniNet network topology versus the number of mMTC RAN slices, taking into account fully dedicated, shared vCU, shared vDU, and fully shared scenarios for a four-core CPU.

the most efficient and the "fully dedicated" is the less efficient scenario for the hosted mMTC type RAN slices. The remaining two scenarios can be interpreted accordingly as shown in Fig. 6.26.

- **Instantiated VNFs for mMTC RAN Slices:** Another parameter that we consider is the number of instantiated VNFs for an mMTC RAN slice in an I-PoP, which is extremely vital parameter to be considered while mapping an mMTC type RAN slice onto the underlying infrastructure in the NG-RAN architecture. The more VNFs of an mMTC RAN slice are instantiated by an I-PoP, the more complicated it becomes for a VIM to manage and orchestrate the required resources throughout their life cycles in NG-RAN. Additionally, the fewer VNFs are instantiated for mMTC type RAN slices the fewer virtual resources will be required, thereby resulting in efficiently allocating of the underlying virtual resources. These advantages lead to reduce the energy consumption, lower the total cost and time of deployment of mMTC type RAN slices, and increase the performance of mMTC type RAN slice, among other tings. However, this parameter also comes at the cost of reducing the level of customization and isolation in the NG-RAN architecture.

- **Number of mMTC RAN Slices:** The fifth parameter we consider during this simulation related to the mMTC type RAN slice is the number of instantiated mMTC type RAN slices in the NG-RAN architecture, which is one of the important parameters. It is a fact that the more mMTC type of RAN slices an NG-RAN architecture hosts with limited virtual resources the more resource efficient it will be considered for a 5G mobile network. Furthermore, it will also serve a large number of end-users or devices of a vertical industry in a specific geographical area. This parameter has been evaluated in the context of IBM topology for mMTC RAN slices against several parameters that will be discussed later in the NG-RAN architecture in this section.

- **Required Virtual Resources for mMTC RAN Slices:** The final parameter we considered in the context of large number-demanding end-users type of RAN slices,

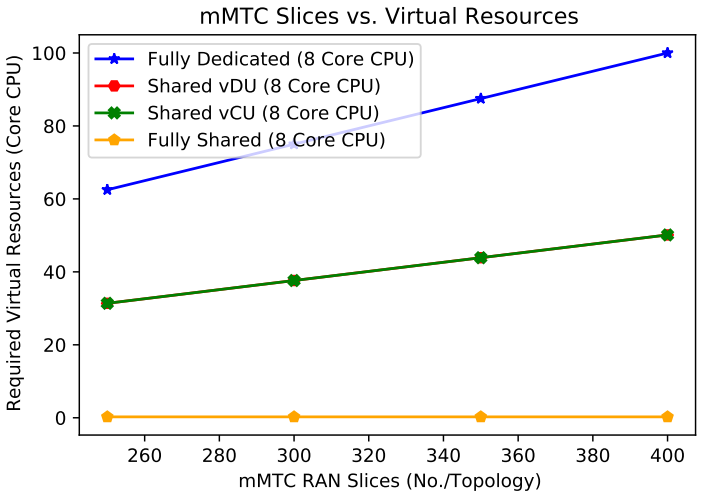

Figure 6.29: The required amount of virtual resources in the UniNet network topology versus the number of mMTC RAN slices, taking into account fully dedicated, shared vCU, shared vDU, and fully shared scenarios for an eight-core CPU.

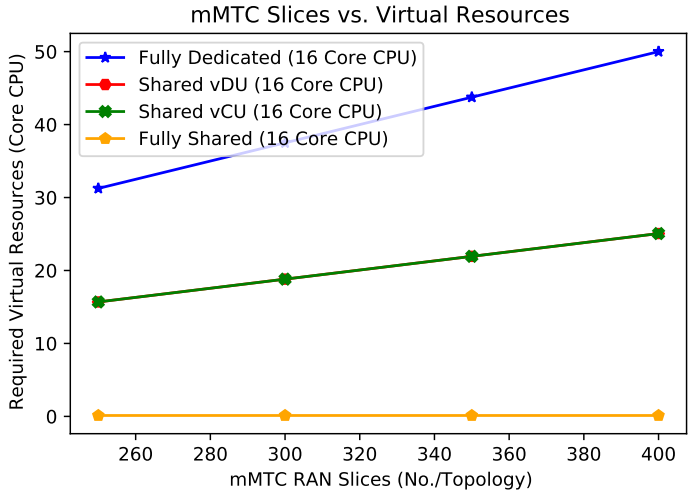

Figure 6.30: The required amount of virtual resources in the UniNet network topology versus the number of mMTC RAN slices, taking into account fully dedicated, shared vCU, shared vDU, and fully shared scenarios for a sixteen-core CPU.

one of the critical parameters, which we are trying to consider during the simulation period is the required amount of virtual resources for the mMTC RAN slices in the UniNet topology in the NG-RAN architecture. This parameter has been evaluated in the context of four scenarios, as discussed above in this thesis, against the number of mMTC type of RAN slices in the NG-RAN architecture. These virtual resources are considered in 2 Core CPU, 4Core CPU, 8 Core CPU, and 16 Core CPU scenarios similar to those of eMBB RAN slices.

Discussion on the Four Scenarios for mMTC RAN Slices

When we take into account the aforementioned two simulation parameters (namely, number of mMTC type RAN slices and the required amount virtual resources), we conduct our

simulations in fully dedicated, shared vDU, shared vCU, and fully shared scenarios for the mMTC type RAN slices. These four types of simulations are executed against various types of underlying virtual resources in the underlying I-PoPs, including 2 Core CPU (Fig. 6.27), 4 Core CPU (Fig. 6.28), 8 Core CPU (Fig. 6.29), and 16 Core CPU (Fig. 6.30). The results obtained for the mMTC type RAN slices in these four figures indicate that the more a CPU contains Cores the less amount of servers we need in an I-PoP for hosting the mMTC type RAN slices. And the less Cores a CPU contain in server the more servers we need to host the mMTC types of RAN slices.

6.2.8 The Q-Learning Algorithm for the mMTC RAN Slices

Once the conditions and scenarios for the allocation of virtual resources of mMTC type of RAN slices in the NG-RAN architecture have been addressed, in this section, we discuss the mapping aspects of the mMTC RAN slices by employing the Q-learning algorithms. While considering the values of the simulation parameters, provided in the previous sections of this chapter, we discuss the results we obtained following the simulations we have executed for the mMTC type of RAN slices in the context of UniNet topology in the NG-RAN architecture. To that end, we first discuss the training of the Q-learning model customized for mMTC RAN slices. Then, we discuss the obtained results and evaluate the performance of the mapping of the mMTC RAN slices in the NG-RAN architecture.

Model Training for mMTC RAN Slices

To optimally map the mMTC type of RAN slices in the NG-RAN architecture, we first and foremost need to train the mapping model. To accomplish the training of the Q-learning algorithm for the mMTC type of RAN slices, we consider two important parameters of a Q-learning algorithm, namely the episode number and the episode reward. These two parameters are evaluated in four different scenarios for mMTC RAN slices: total episode rewards, mean episode rewards, maximum episode rewards, and minimum episode rewards. We have evaluated the proposed Q-learning algorithm for the two above parameters under five different episode numbers in order to achieve an optimal rewards for the mapping of the mMTC RAN slice. These five different episode numbers for mMTC type of RAN slices are: 50, 250, 500, 2500, and 50000. The results obtained for the five types of episode number are shown in Fig. 6.31, Fig. 6.32, Fig. 6.33, Fig. 6.34, and Fig. 6.35, respectively. We will interpret these results in the following sections.

Model Evaluation for mMTC RAN Slices

Once the simulation has been executed for the mMTC type of RAN slices in the UniNet topology, we realized that in the case where the episode number is 50, the performance of the learning process of the model is worse, as illustrated in Fig. 6.31. The reason for this is that the learning process in this period is mostly based on the random exploration with a large epsilon value. This fact is applicable to all four types of episodes we assumed for the URLLC type of RAN slices. Following that, we exceed the number of episode to 250, as illustrated in Fig. 6.32. The results in this figure demonstrate that performance of the model has slightly increased in comparison of that previous figure. Furthermore, we continuously increasing the number of episodes to 500. The results obtained for this simulation is illustrated in Fig. 6.33 show that the model performs better than that of illustrated in the previous period. We also set the number of episodes to 2500 as shown in Fig. 6.34. The obtained results shown in this figure illustrate that the model outperforms than all previous number of episodes.

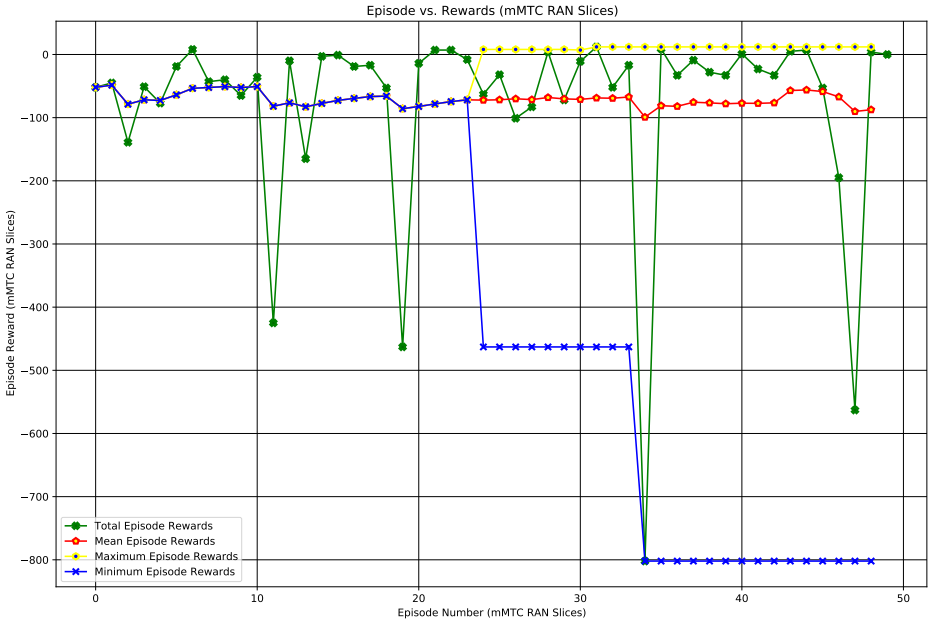

Figure 6.31: Total rewards collected versus episode number (e.g., 50) during the training phase of the agent for the mMTC RAN slices in the UniNet topology.

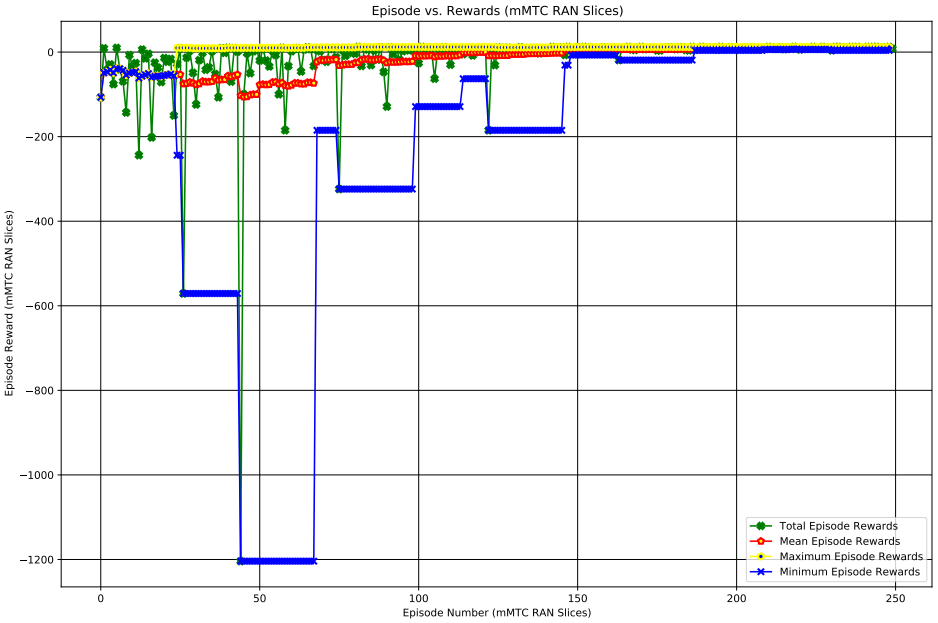

Figure 6.32: Total rewards collected versus episode number (e.g., 250) during the training phase of the agent for the mMTC RAN slices in the UniNet topology.

Finally, the Q-learning algorithm related to the mapping problem of mMTC type of RAN slices in the NG-RAN architecture converges to an optimum model after setting the number of episodes to 50000, as illustrated in Fig. 6.24.

6.2.9 The Evaluation of SLA Metrics for mMTC RAN Slices

Finally yet importantly, we also evaluate the performance of several QoS metrics against the penalty during the lifetime of an mMTC type of RAN slice in the UniNet topology. The performance metrics include availability, device density, reliability, bandwidth, and latency, as shown in Fig. 6.36. We set different performance downgrade values for each of the QoS

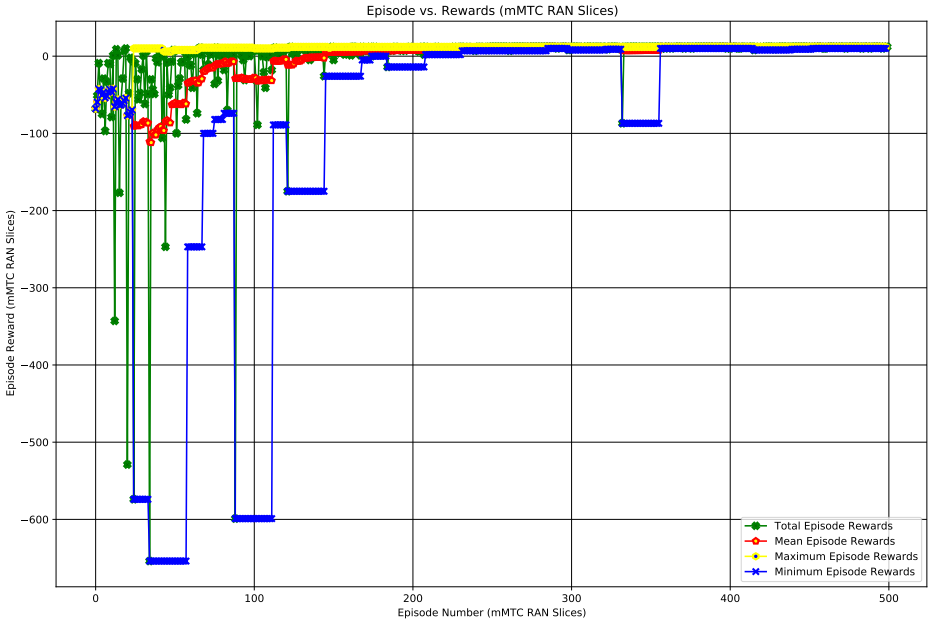

Figure 6.33: Total rewards collected versus episode number (e.g., 500) during the training phase of the agent for the mMTC RAN slices in the UniNet topology.

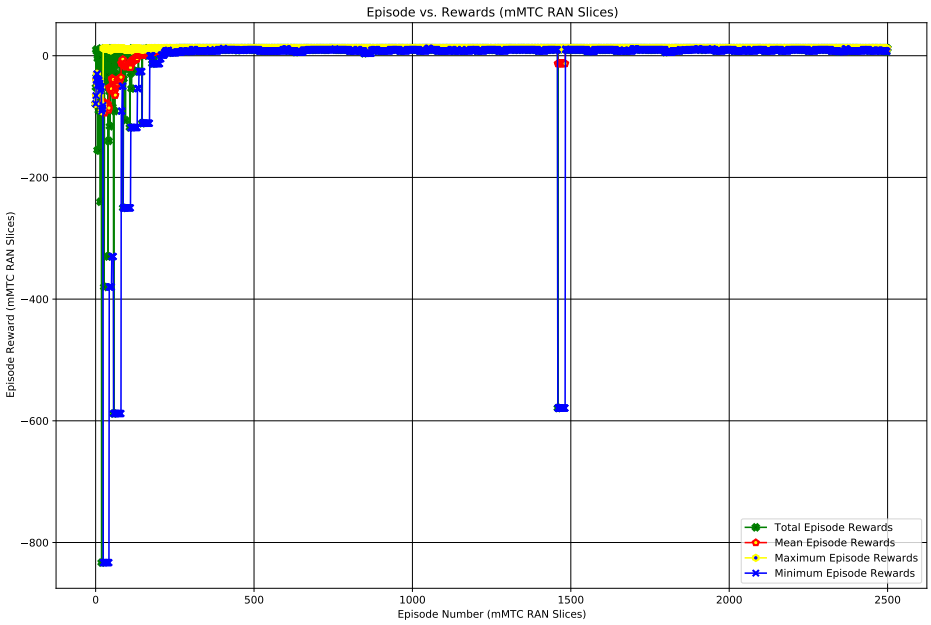

Figure 6.34: Total rewards collected versus episode number (e.g., 2500) during the training phase of the agent for the mMTC RAN slices in the UniNet topology.

metrics. The highest value is 100 while the lowest performance downgrade value is 98.4, which is shown in Fig. 6.36. The figure demonstrates that when the performance of a metric decreases the model imposes a certain degree of penalty on the service provider. This value is almost certain till 99.2. However, if the performance of the QoS metric decreases more than this value, the model doubles the penalty and impose it on the service provider. Once it reaches 98.4 %, it is considered the worse case scenario where the performance of an mMTC RAN slice is at its lowest possible level. The tenant shall regularly checks the frequency of performance downgrade. If it continuously decreases, the tenant may decided to change the service provider for the sake of a better QoS.

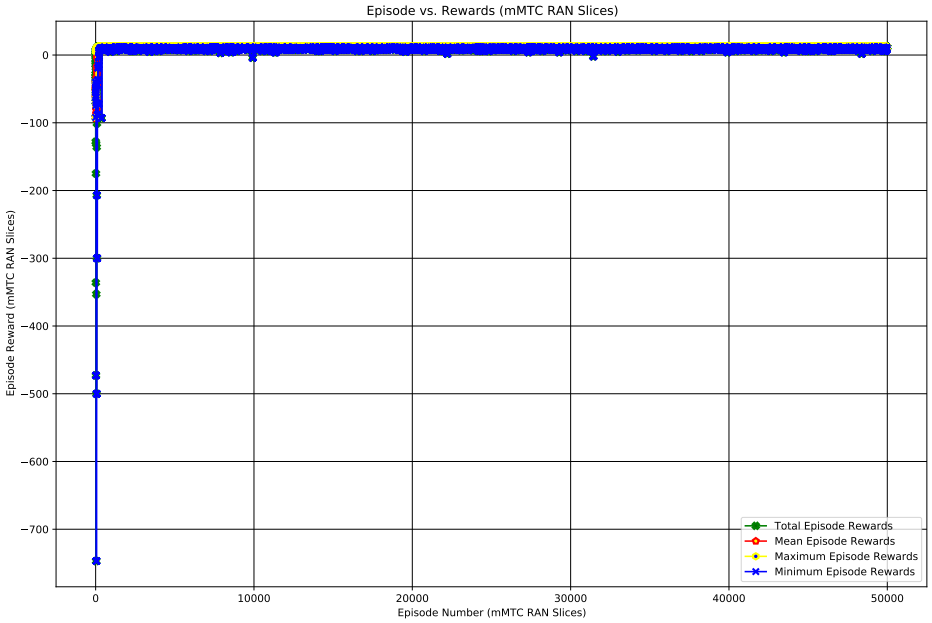

Figure 6.35: Total rewards collected versus episode number (e.g., 50000) during the training phase of the agent for the mMTC RAN slices in the UniNet topology.

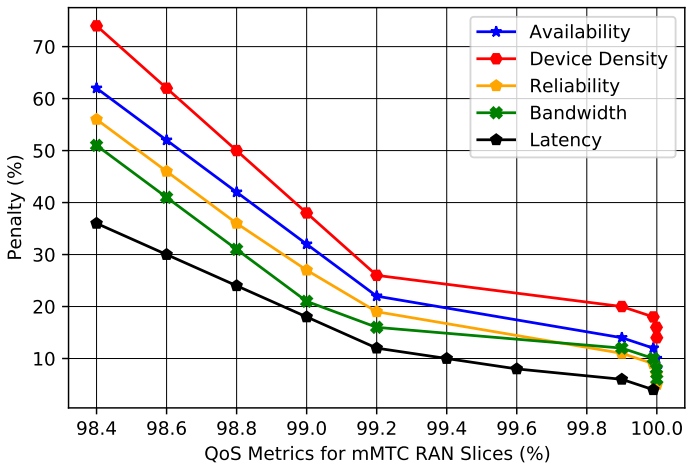

Figure 6.36: The evaluation of QoS metrics in the UniNet network topology for mMTC RAN slices versus the penalty imposed on the service provider, taking into account fully dedicated, shared vCU, shared vDU, and fully shared scenarios.

6.3 Global Resource Allocation

The results related to the mapping process of the eMBB, URLLC, and mMTC types of RAN slices have been thoroughly discussed in the previous sections. Each RAN slice has been individually mapped onto the underlying infrastructure considering its assumed network topology and requirements. The three types of RAN slices are mapped in four different resource allocation scenarios in both I-PoP #1 and I-PoP #2. The VNFs associated to these three types of RAN slices have been mapped in such a way that considered the four resource allocation scenarios we defined in the previous section. However, these RAN slices are not allocated with a customized underlying infrastructure. Rather, they share a single underlying infrastructure, which shall host all the VNFs of different types with different requirements at a give time. Therefore, in this section, we discuss the global mapping of all these three types of RAN slices considering the amount of links and nodes required for each of the RAN slices

in the underlying infrastructure.

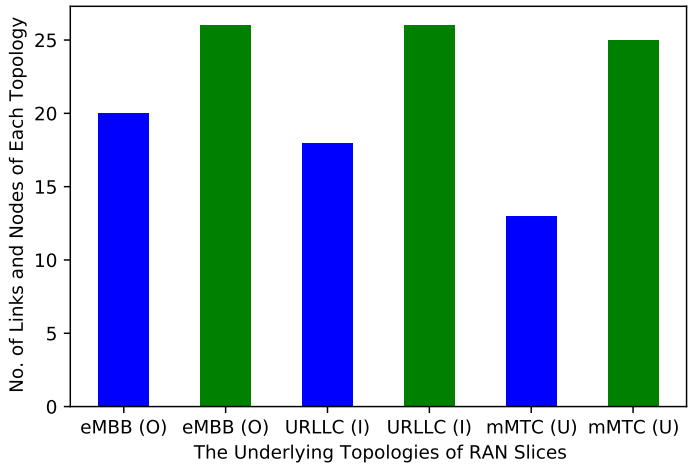

Figure 6.37: The underlying network topologies considered for the deployment of eMBB, URLLC, and mMTC types of traffic of RAN slices versus the number of links and nodes required for each topology [2]. In this figure, O, I, and U stand for Oxford, IBM, and UniNet, respectively.

To accomplish this, Fig. 6.37 illustrates the global resource allocation of nodes and links for the three types of RAN slices in the NG-RAN architecture. As shown, the number of nodes and links are evaluated against the Oxford, IBM, and UniNet topologies. The results reveal that the required number of links in mMTC types of RAN slices are utilizing UniNet topology is less than in comparison to the other two types of RAN slices. Therefore, from the virtual networking resource allocation perspective, this topology in the context of mMTC type of RAN slices is considered the most efficient choice. From the virtual compute and storage resource perspective, although there is not a significant difference between the three types of RAN slices, however, the results reveal that mMTC types of RAN slices in the UniNet topology also consumes less nodes in the underlying infrastructure in the NG-RAN architecture. Finally, both the eMBB and URLLC types of RAN slices are demanding more virtual compute, storage, and networking resources in the context of their respective topologies in the underlying infrastructure.

6.4 Performance Evaluation of Double and Deep Q-Learning

Following the theoretical discussion of the system model and the mathematical formulation presented in the previous sections, we employ double and deep Q-learning methods to evaluate the VNFC-to-VM mapping problem with respect to selected performance metrics. We first describe the simulation settings, including the considered VNFC-VM assignment scenarios, the adopted hyperparameters, and the evaluation criteria. Subsequently, we present the results obtained under the defined performance metrics.

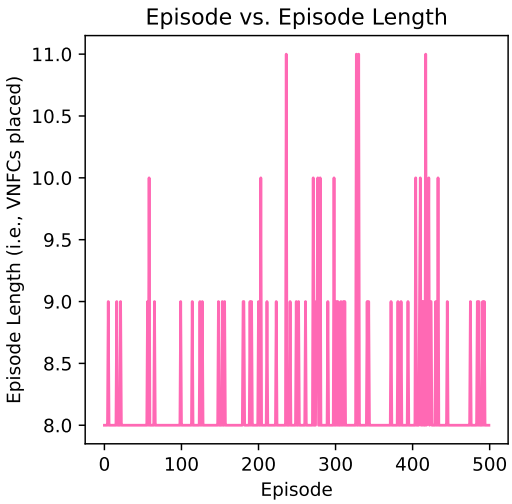

Figure 6.38: Episode versus episode length for the double Q-learning agent with the function approximation method

6.4.1 Simulation Settings

The proposed methods were implemented using several *Python* packages, namely *Random*, *Matplotlib*, *NumPy*, and *Collections*. The implementation employs the following set of hyperparameters: number of episodes set to 500, discount factor $\gamma = 0.99$, replay buffer size of $50,000$, batch size of 128, warm-up steps equal to 500, target network update frequency of 500, initial exploration rate $\epsilon = 1.0$, final exploration rate $\epsilon = 0.1$, and ϵ-decay steps equal to $10,000$. The dataset under consideration consists of 100 VMs and 8 VNFCs. The framework is designed to support user-defined inputs as well as synthetic data generation. In the performance evaluation, we consider a predefined set of VM-VNFC pairs. Each pair (a, b) represents the computational capacity a and storage capacity b, respectively, in the case of VM. It also represents the computational requirement and storage requirement, respectively, in the case of VNFC. The learning agent updates its policy using an ϵ-greedy strategy.
Following the methodology adopted in, the performance of the proposed methods is evaluated using the following metrics: episode versus total reward, episode versus episode length, exploration versus episode, cumulative reward versus episodes, and exploration ratio versus episodes. A comparative analysis is then conducted to assess the performance of the methods proposed in this work against those presented in within the context of the VNFC-to-VM placement problem.
Furthermore, all six performance plots are analyzed. Based on this unified representation, a detailed comparative analysis is conducted using the area under the curve (AUC) of each method, with an emphasis on the average reward, standard deviation, and convergence episode.

6.4.2 Results Analysis

Double Q-learning Results

With the aforementioned discussion of the double Q-learning method in mind, we now analyze its performance with respect to the selected evaluation metrics. The learning behavior of the double Q-learning agent over 500 episodes is reported in Figure 6.38 (episode versus episode length), Figure 6.39 (episode versus total reward), Figure 6.40 (episode versus exploratory

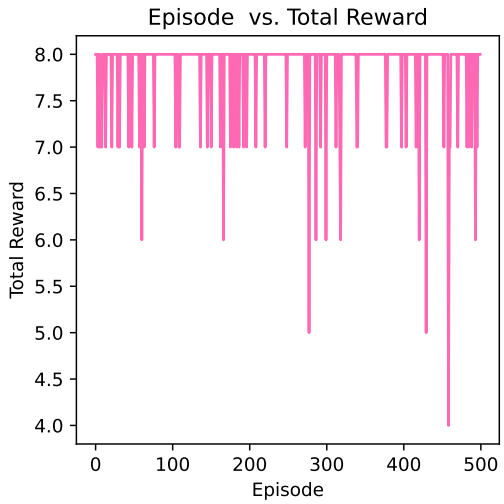

Figure 6.39: Episode versus total reward for the double Q-learning agent with the function approximation method

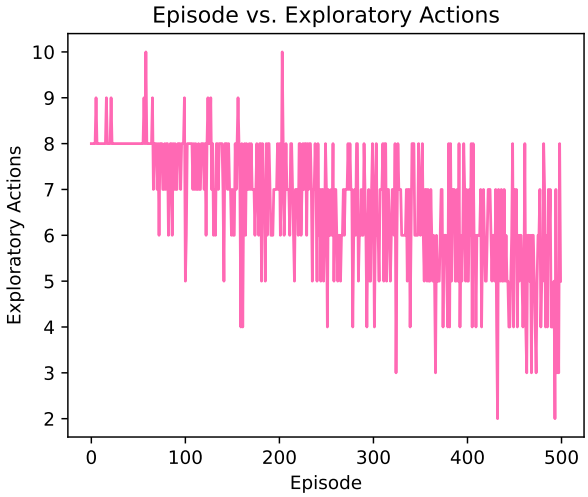

Figure 6.40: Episode versus exploratory actions for the double Q-learning agent with the function approximation method

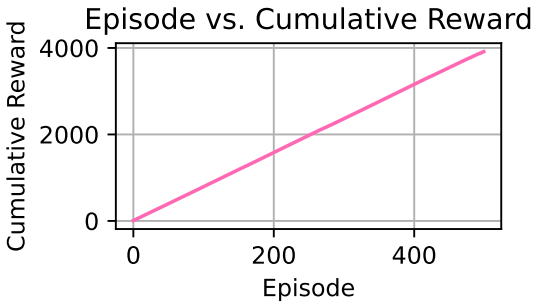

Figure 6.41: Episode versus cumulative reward for the double Q-learning with the function approximation method

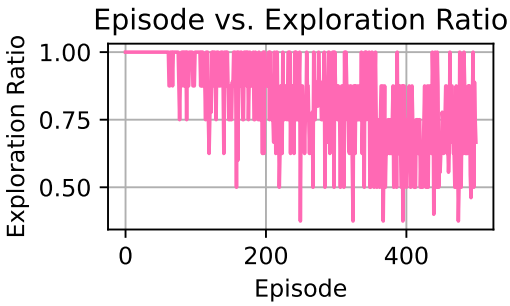

Figure 6.42: Episode versus exploration ratio for the double Q-learning with the function approximation method

actions), Figure 6.41 (episode versus cumulative reward), and Figure 6.42 (episode versus exploration ratio).
As shown in Figure 6.38, the episode length fluctuates between 8 and 11 steps. In most episodes, the agent converges to an episode length of 8 steps, while values in the range of 9-11 steps occur only infrequently. This behavior indicates that the agent consistently learns efficient policies that complete the task in a reduced number of steps.
As illustrated in Figure 6.39, the total reward per episode exhibits limited fluctuations, remaining approximately within the range of 4 to 8. The agent achieves the maximum reward value of 8 in the vast majority of episodes, indicating near-optimal performance. Episodes with reduced rewards occur only sporadically, suggesting that the learned policy stabilizes effectively across episodes.
Figure 6.40 depicts the number of exploratory actions per episode. Although noticeable fluctuations are observed, the values generally remain within the range of 2 to 10 and follow a stochastic pattern. This behavior indicates persistent exploration throughout the training process, helping the agent avoid premature convergence while maintaining stable performance.
Figures 6.41 and 6.42 provide further insight into the agent's learning behavior. Figure 6.41 illustrates the cumulative reward per episode, which exhibits a near-linear increase with noticeable acceleration over time, indicating that the agent consistently accumulates rewards without significant performance regressions. Finally, Figure 6.42 depicts the exploration ratio per episode. Although the exploration ratio fluctuates significantly throughout training, a clear decaying trend is observed, suggesting that the ϵ-greedy exploration strategy gradually diminishes as learning progresses.

Deep Q-learning Results

Moving forward, we now analyze the performance of the deep Q-learning algorithm under the same performance evaluation metrics. The results obtained from the learning behavior of the deep Q-learning agent over 500 episodes is illustrated in Figure 6.43 (episode versus episode length), Figure 6.44 (episode versus total reward), Figure 6.45 (episode versus exploratory actions), Figure 6.46 (episode versus cumulative reward), and Figure 6.47 (episode versus exploration ratio).
As shown in Figure 6.43, the episode length fluctuates between 8 and 12 steps. In most episodes, the agent converges to an episode length of 8 steps, while values in the range of 9â€“12 steps occur only infrequently. This behavior indicates that the agent consistently learns efficient policies that complete the task in a reduced number of steps.
As illustrated in Figure 6.44, the total reward per episode exhibits limited fluctuations, remaining approximately within the range of 4 to 8. The agent achieves the maximum reward

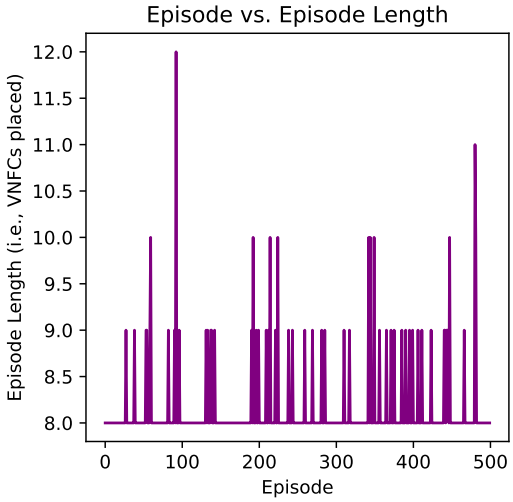

Figure 6.43: Episode versus episode length for the deep Q-learning agent with the function approximation method

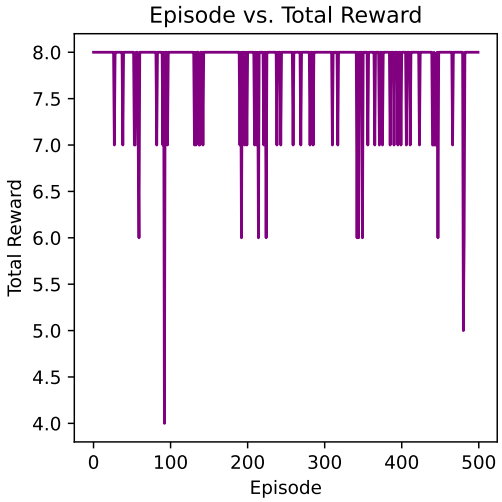

Figure 6.44: Episode versus total reward for the deep Q-learning agent with the function approximation method

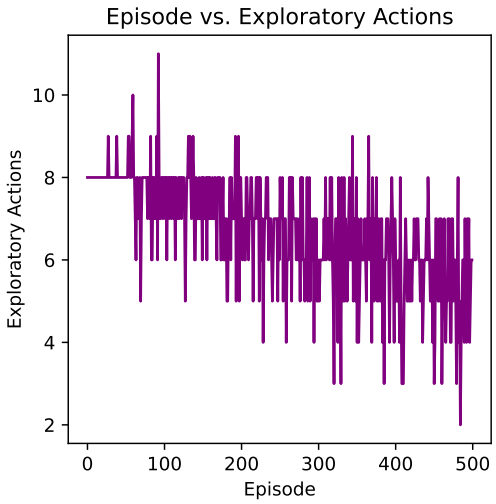

Figure 6.45: Episode versus exploratory actions for the deep Q-learning agent with the function approximation method

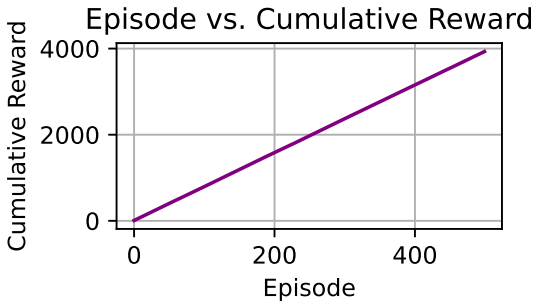

Figure 6.46: Episode versus cumulative reward for the deep Q-learning with the function approximation method

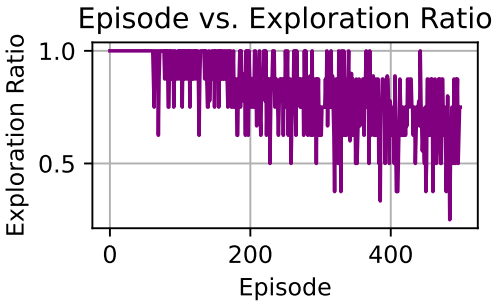

Figure 6.47: Episode versus exploration ratio for the deep Q-learning with the function approximation method

value of 8 in the majority of episodes, indicating near-optimal performance. This observation suggests that the learned policy stabilizes effectively across episodes.
Figure 6.45 depicts the number of exploratory actions per episode. Similar to the behavior observed in Figure 6.40, noticeable fluctuations are present throughout training. In general, the number of exploratory actions remains within the range of 2 to 12 and follows a stochastic pattern, indicating persistent exploration during the learning process.
Figure 6.46 illustrates the cumulative reward per episode, which exhibits a near-linear increase with comparable acceleration over time. This trend indicates that the agent consistently accumulates rewards without significant performance regressions.
Finally, Figure 6.47 shows the exploration ratio per episode. Although the exploration ratio fluctuates markedly throughout training, a clear decaying trend is observed, suggesting that the ϵ-greedy exploration strategy gradually diminishes as learning progresses.

Chapter 7

Conclusions and Future Work

Summary – In this chapter, we provide a summary of the key discussions and major contributions highlighted in the previous chapters of this thesis, as well as future research directions that require further investigations and subsequent research efforts in the area of RAN slicing in the NG-RAN architecture, respectively.

7.1 Concluding Remarks

In this thesis, we first provided a comprehensive overview and critical analysis of the virtual and physical components of RAN slicing in the NG-RAN architecture, as well as the physical and virtual aspects of the underlying infrastructure layer at the edge of 5G and beyond mobile communication networks. We then extended the architectural framework of network slicing, defined by the NGMN Alliance, towards the NG-RAN architecture in order to provision the eMBB, URLLC, and mMTC types of RAN slices. Based on this framework, we addressed the management and orchestration of the operations and resources of RAN slices by leveraging the unified functioning architecture of the NFV-MANO, 3GPP-NSMS, and the underlying compute and transportation network infrastructure. Then, we integrated the ENI System into the NFV-MANO FBs, 3GPP-NSMS entities, and the components of the underlying infrastructure in order to automate their operations and bring intelligence to the edge of 5G and beyond cellular networks. These contributions were primarily aimed at laying the groundwork for an autonomous and intelligent mapping of the virtual components of a RAN slice. To that end, we defined and modeled the mapping problem of the vCU, vDU, and VLs, taking into account a number of performance objectives that are limited by a set of constraints. We also proposed an ENI-enabled architectural solution that is both autonomous and intelligent for mapping the VNFCs of vCU and vDU onto their respective VMs, as well as the internal and external VLs of the vCU and vDU onto their corresponding VNs, in I-PoPs and transportation links, respectively.

In addition to the above, we presented a comprehensive system model on the basis of which we mathematically formulated the mapping problem of the vCU and vDU, internal and external VLs, joint mapping of both types of virtual components, and the VNFFG of a RAN slice onto underlying infrastructure in the NG-RAN architecture. Following that, we discussed various types of ML-assisted algorithms such as supervised, unsupervised, and reinforcement learning. We specifically focused on the mathematical background and application of the reinforcement learning and selected Q-learning to solve the mapping problem. Lastly yet importantly, we discussed the results we obtained during the simulation period through the utilization of Q-learning algorithm in the mapping process of RAN slices. The obtained results demonstrated the mapping process of the three distinct types of RAN slices onto

the underling infrastructure under different conditions in the NG-RAN architecture. It also included an evaluation of the performance objectives we achieved through using Q-learning algorithm against the SLA signed between the tenant and the service provider during the lifetime of the requested RAN slice.

7.2 Future Research Directions

Regarding the future works, there are numerous research challenges associated with different types of RAN slices, which require substantial research efforts in order to provision efficient RAN slicing in the NG-RAN architecture for futuristic networks. Some of the most important research challenges related to this topic are described as follows.

- We would like to extend our work on virtualizing and partitioning the physical resources of the I-PoPs and transportation network on top of the proposed ENI-enabled framework of the 3GPP-NSMS and NFV-MANO in the NG-RAN architecture.
- We are interested in looking into how virtual compute, storage, and networking resources are allocated for various types of RAN slices using cutting-edge autonomous and intelligent tools with the goal of optimizing resource utilization and enhancing RAN slice performance while also taking into account a set of predefined constraints, such as the trade-off between resource utilization ratio and isolation level, the harmonization of inter-RAN and intra-RAN slice resource allocation algorithms, and the management of inter-RAN and intra-RAN slice priority.
- The physical and virtual compute, storage, and networking resources of different types of RAN slices should be managed and orchestrated dynamically and opportunistically across the underlying compute and transport network infrastructure. Therefore, we are also interested in applying cutting-edge ML-assisted algorithms to resource management and orchestration in the near future, as well as exploring the impact of automation and intelligence on these two areas.
- Most existing studies related to RAN slicing are now focusing on radio and virtual resource slicing, leaving open a significant gap to a full network slicing in the NG-RAN architecture, i.e., to slice infrastructural resources. Differing from physical resource blocks, transmission power, storage, and computing resources - which are relatively easy to split and assign to different RAN slices - bare-metal resources such as antennas are hard to be flexibly shared among different RAN slices while guaranteeing satisfactory isolation. This challenge can be further amplified by the introduction of emerging 6G infrastructures, such as high-altitude platforms, unmanned-aerial-vehicles, and intelligent reflecting surfaces.
- Finally, RAN slicing isolation can be carried out on different levels. The higher the isolation level is, the better can the QoS of prioritized RAN slices (e.g. URLLC) be guaranteed, at a price of reduced spectral flexibility and utilization efficiency. Furthermore, even within the same RAN slice, multiple isolation approaches can be applied on different layers of its protocol stack. It is a critical challenge to find the optimal isolation scheme that maximizes resource efficiency while guaranteeing the isolation requirement. Therefore, we are also interested in investigating this aspect of RAN slicing in the near future.

Appendix A

Network Slicing for Multiple Use Cases of a Single Vertical Industry

Abstract: There is a significant number of vertical industries that consist of multiple use cases. Each use case of such a vertical industry is characterized by diverging service, network, resource, and connectivity requirements, such as automobile, manufacturing, public transportation, power grid, and many others. Such heterogeneity cannot be effectively managed and efficiently mapped onto a single type of NSI in the tenant and operator domains. The tailored provisioning of an E2E network slicing solution to a vertical industry implementing multiple use cases is thus an important research problem. This chapter is, therefore, aimed to explore this never-addressed and challenging issue by proposing the use case-specific network slicing and the sub-network slicing concepts that enable the provisioning of the E2E use case-specific NSI (US-NSI) and generic NSI (GN-NSI), respectively. Both proposed concepts tackle the same research problem in order to provide, manage, and orchestrate per-vertical per-use-case NSIs aiming to improve resource allocation, enhance network performance, and increase user experience. The chapter also explores the architectural frameworks to manage US-NSI and GN-NSI. Both proposed frameworks extend the service deployment concept and system architecture of network slicing for vertical industries which are supported by 5G and beyond mobile networks.

A.1 Background on Key Concepts

In the previous chapters, we defined that an NSI is instantiated based on a machine-processable template (a.k.a. blueprint, service profile, and descriptor) which a set of predefined attributes with values/ranges, known as the NST. It defines the required network resources, functionalities, and the interconnection and configuration of these functionalities in order to meet the performance, functional, and operational requirements of a use case [7]. We have also discussed that an NSI is provided by an operator to a tenant as a service [8]. The tenant and operator are obligated to sign a formal SLA in order to clarify any possible misunderstanding throughout the business period, assess the characteristics of an NSI, and define the responsibilities and priorities of both sides.

So far, a significant number of industrial experiments, theoretical researches, and standardization efforts have been executed aiming to explore various aspects of network slicing, such as admission control, resource allocation, performance optimization, management, orchestration, and others in the context of a single NSI. Referred to the previous chapter, an NSI could be configured to support multiple services of the same type of service category, such as eMBB, mMTC, and ultra-reliable low latency communications (uRLLC) of a vertical industry.

However, such a scenario may not always be optimal. For example, multiple service instances within a vertical industry may represent different types of use cases, each with diverging QoS, resource, functionalities, and configuration requirements [185] [7]. Let's assume the power grid industry which contains a group of use cases, such as distribution automation, net metering, renewable power supply, and others. Each use case (out of a family of use cases of a power grid industry) has its own specific performance, functional, and operational requirements. One NSI per such a vertical industry is thus not able to fulfill the diverse service requirements of its use cases.
This issue has partially raised by GSMA in [50], the International Telecommunication Union (ITU) in [186], and the Open Networking Foundation (ONF) in [187]. The ETSI ISG NFV, 3GPP, NGMN Alliance, IETF, and other SDOs have not provided solutions for supporting divergent use cases of a vertical in their standardization documents so far. Providing network slicing solutions – mainly the management and orchestration of NSIs, and the allocation of their corresponding physical and virtual resources – to such type of verticals is still an open research problem in the research community. Addressing them in SDOs, especially in 3GPP and ETSI ISG NFV, are thus essential to design a complete interoperable E2E network slicing paradigm.
Motivated by the solution gap described above, we propose two concepts namely the use case specific network slicing and the sub-network slicing, and their architectural frameworks that enable the provisioning of US-NSI and GN-NSI. The objective is to provide solution proposals on this crucial issue of network slicing in SDOs (see Sec. A.3 and Sec. A.4, respectively). Both approaches are in compliance with the E2E service deployment concept, and their architectures are built on top of architectural frameworks of network slicing, defined by the NGMN Alliance in [7] and standardized by the 3GPP in [86]. The main objectives of the proposed solutions are to provide an insight into the concepts, use cases and their integration in 3GPP networks, procedures, and mechanisms required to make the deployment of independent NSI to each of the use cases of such verticals in a cost/resource-efficient, flexible, and agile manner from the perspective of both operator and tenant. Before addressing these issues, we commence by providing a background on key concepts in the following section.

A.2 Network Slicing for Multiple Use Cases of a Single Vertical Industry

A vertical is owned by at least a single tenant. It consists of at least a single use case. Each use case requires at least one service instance and is represented by its own architected NSI with a customized NST. Each NSI – dedicated to a specific use case – is isolated in terms of resources, security, policies, configuration, etc. from other NSIs and may contain specific capabilities in 5GC, NG-RAN, TN, and terminal [188]. Each NSI is managed by its own manager and has operation and maintenance (O&M) requirements unique to the NSI. The O&M system is responsible to provide monitoring and self-control capabilities to support business requirements throughout the life-cycle of an NSI. Therefore, the interactions between tenant and operator during the business period is constructively handled in order for both parties to negotiate their conflicting interests and to achieve satisfaction.
The design, the negotiation of SLA, and the life cycle management (LCM) of an NSI dedicated to a vertical that consists of a single use case are straightforward processes. The technical requirements of such verticals can be mapped onto the E2E network slicing concept easily and efficiently. However, verticals that contain more than one use case are usually characterized by diverse service requirements. For example, a V2X communication system consists of

various use cases in four communication modes (see Fig. A.1) – vehicle to vehicle (V2V), vehicle to infrastructure (V2I), vehicle to pedestrian (V2P), and vehicle to network (V2N) – defined by 3GPP [189]. Among them, autonomous driving, infotainment applications, and remote diagnostic are the most well-known use cases. Every use case demands its own set of QoS requirements tailored to satisfy specific business requirements of one or more V2X services (see Tab. A.1). Therefore, each use case requires an individual and isolated NSI. These NSIs are intended for different scenarios; for instance, uRLLC for autonomous driving, eMBB for infotainment applications, and mMTC for remote diagnostics.

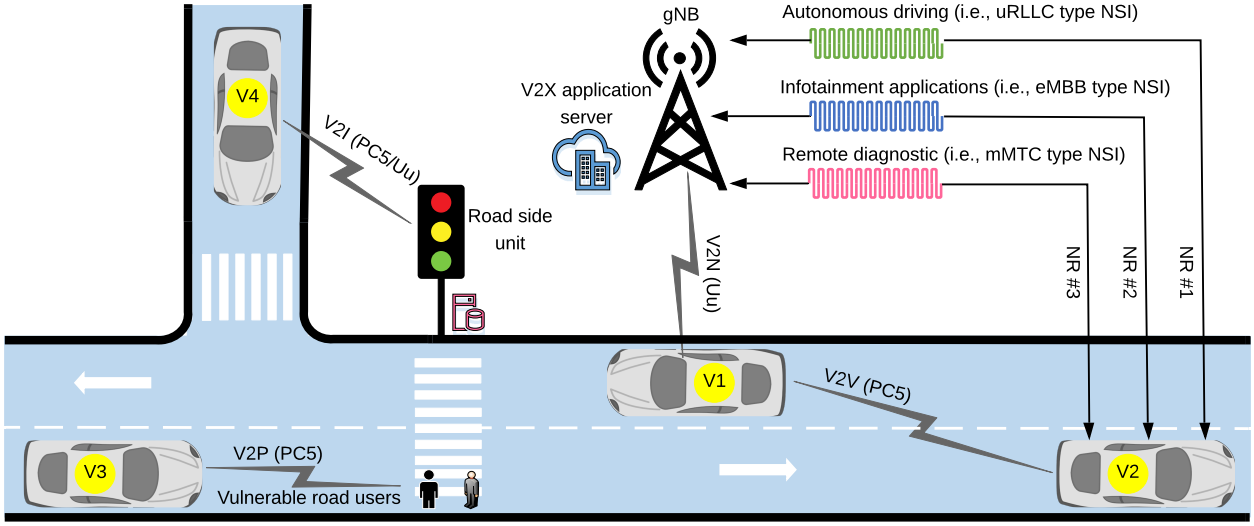

Figure A.1: Three use cases and their corresponding NSIs of V2X communication system. Each NSI is individually deployed with configurable characteristics and different levels of isolation to only one use case out of a group of use cases.

- The NSI, which is dedicated to supporting autonomous driving, requires ultra-high reliability (99.999%) and ultra-low latency (below 10 ms) on V2V communication mode over sidelink air-interface (PC5). Moreover, full-road network coverage (99.9999%) is needed to provide communication services for autonomous vehicles in all types of geographical locations under high vehicle-density condition (see Tab. A.1). This NSI, in some scenarios, may also provide driving efficiency and safety services, such as using a see-through application to detect hazards on road and to trigger emergency messages.

- NSI supporting infotainment applications shall be configured to offer a short-lived high data rate connection of up to 100 Mbps (for mobile broadband in vehicles) between a vehicle and nearby gNBs in order to support services, such as the broadband transmission of video, files/apps downloading/uploading, online gaming, access to social media, audio/video conference streaming, etc. This NSI shall support connection under high mobility of up to 250 Km/hr on V2I and V2N communication modes over the air interface (see Tab. A.1).

- NSI designed for vehicle management and remote diagnostic shall provide connectivity between a massive number of vehicles (hundreds per square kilometer, see Tab. A.1) and a remote server outside a communication network. This NSI is functioning on V2I and V2N communication modes over long term evolution (LTE)-Uu/5G new radio (NR) air interfaces and meanwhile demands extreme power saving mechanism. It is usually owned by a V2X service provider or a diagnostic center that continuously collects

information from sensors installed on a vehicle to track the position of the vehicle, the status of the vehicle, perform remote diagnostic troubleshooting, etc.

The example of instantiating three E2E isolated NSIs to three use cases of the V2X communication system proves that providing network slicing solutions to verticals that consist of multiple use cases is an important practical research problem. Identifying the existing use cases and fulfilling the QoS requirements of such verticals, by providing per-use case NSIs, brings a number of key advantages to both tenant and operator. From the tenant's perspective, the proposed approach simplifies the operations by enforcing the QoS requirements of individual use cases in isolation from others. In this way, the tenant's QoE is improved and the business flexibility is also empowered. Whereas, from the operator's perspective, both physical and virtual resources are efficiently allocated to the individual use case of a vertical. The per-use-case NSIs can be dynamically offered by either a single or multiple operators. In this chapter, a single operator is assumed. To offer these NSIs, either of the following proposed approaches – the use case specific network slicing or the sub-network slicing – shall be chosen.

- E2E use case Specific Network Slicing: in this approach, a per-use-case NSI (known as the US-NSI) is provisioned, where the number of the US-NSIs equals the number of use cases.
- E2E sub-network slicing: in this approach, a per-vertical GN-NSI is provisioned, which contains a family of sub network slice instances (S-NSIs), where every S-NSI is dedicated to a use case.

In both approaches, all existing use cases have to be identified first. In order to ensure specific needs and to meet diverse service demands of every use case, all of its performance, functional, and operational attributes should be then individually defined and quantified. Subsequently, the NST of every use case shall be designed and uploaded into the network slice catalogue (NSC).

A.3 Use Case Specific Network Slicing

In this concept, a per-use case NSI namely the US-NSI is instantiated to support a specific use case (out of a group of use cases) of a vertical. The US-NSIs are provided according to the number of the use cases of a vertical. For example, if there are three use cases in the V2X communication system, then three US-NSIs must be offered. The LCM of a US-NSI is isolated from other US-NSIs of the same vertical. In other words, every US-NSI is individually activated, operated, and deactivated. Therefore, every US-NSI must also have its own SLA.

A.3.1 The Architectural Framework of Use Case Specific Network Slicing

Fig. A.2 illustrates Vertical #1, which is owned by Tenant #1. We assume that it consists of $n \in \mathbb{N}^+$ use cases (i.e., Use-case #1 – Use-case #n). Tenant #1 requests n US-NSIs (i.e., US-NSI #1 – US-NSI #n, respectively). The number of slice requests in the northbound interface thus equals the number of use cases of Vertical #1. Once requests are admitted, the operator allocates the required shared and dedicated PNFs/VNFs, resources, radio access technologies (RATs) settings, and per slice tailored user/control-plane splits to every

Table A.1: The quantitative and qualitative comparison of performance, functional, and operational requirements of three NSIs of the V2X communication system.

Category	Requirements	Three NSIs supporting three use cases of the V2X system		
		uRLLC type NSI for autonomous driving (V2I, V2N, V2V)	**eMBB type NSI for infotainment (V2N)**	**mMTC type NSI for remote diagnostics (V2I, V2N)**
Performance	Latency	1-10 ms	<20 ms	<100 ms
	Reliability	99.999%	99.99%	95%
	Availability	99.9999%	99.999%	99%
	Mobility	0-250 Km/hr	0-250 Km/hr	0-250 Km/hr
	Device density	High	High	Very high
	Data rate	50 Mbps	1-100 Mbps	0.55 Mbps
Functional	Isolation	Very high	High	Medium
	Security	Very high	High	Not a concern
	server positioning	Not required	Edge/remote-cloud	Remote cloud
	Scheduling	Semi-persistent	Dynamic	Semi-persistent
	Priority	Very high	High	Medium
	Battery life	High	High	Very high
Operational	Coverage type	Nationwide	Global	Nationwide
	Supported APIs	Yes	Yes	Yes
	Energy efficiency	High	High	Very high
	Resources/policies	Self management	Self management	Self management
	Monitoring	Real	Real/non-real	Real/non-real
	Communication mode	PC5	LTE-Uu/NR	LTE-Uu/NR
	Communication primitive	Broadcast	Unicast	Unicast

Note: The performance requirements are the most demanding requirements, which specify the characteristics and/or type of an NSI. The functional requirements are some of the basic functions that are used to describe the intended capabilities and interactions of an NSI. The operational requirements are associated with the operation, mission, and management of an NSI. Please also note that all values in Tab. A.1 are in compliance with 3GPP Releases 14-16.

US-NSI according to its NST. These multiple US-NSIs' requests may or may not be granted concurrently. Based on this, a dynamic, automated, and synchronized strategy of interactions is anticipated in the northbound to fulfill the technical requirements of each of the US-NSIs.

The interactions between operator and tenant (such as the dynamic and automated negotiation over SLA, slice request, and network resources) are accomplished by allowing the service capability exposure function (SCEF) to securely expose required services and capabilities to every US-NSI. The SCEF, specified by the 3GPP in Release 13, is one of the nodes in LTE core network that acts as a mediator between operator and tenant. In 5GC, the network exposure function (NEF) was introduced to evolve the capabilities of SCEF with special emphasis on network slicing. The SCEF/NEF abstracts services from the 3GPP network's interfaces and protocols to a tenant (see Fig. A.2). Moreover, it allows a tenant to configure, monitor, control and change certain policies, QoS, and status of the UE attached to a US-NSI. These unique capabilities of SCEF/NEF provide operators the opportunity to fulfill the requirements of innovative use cases and more flexibly offer services that they could not support prior to its deployment.

The operator offers these capabilities to a tenant over a set of dedicated northbound APIs or T8, which is standardized by 3GPP in [190]. By means of API, the tenant can configure resources, communication services, and NFs tailored to every US-NSI. In the proposed architecture, an independent set of APIs is considered to every use case. Fig. A.2 depicts that for n use cases of Vertical #1, n sets of independent APIs are provided. These APIs connect the SCEF/NEF and the service capability server/application server (SCS/AS).

The SCS/AS acts as a gateway between operator and tenant. It may or may not reside at

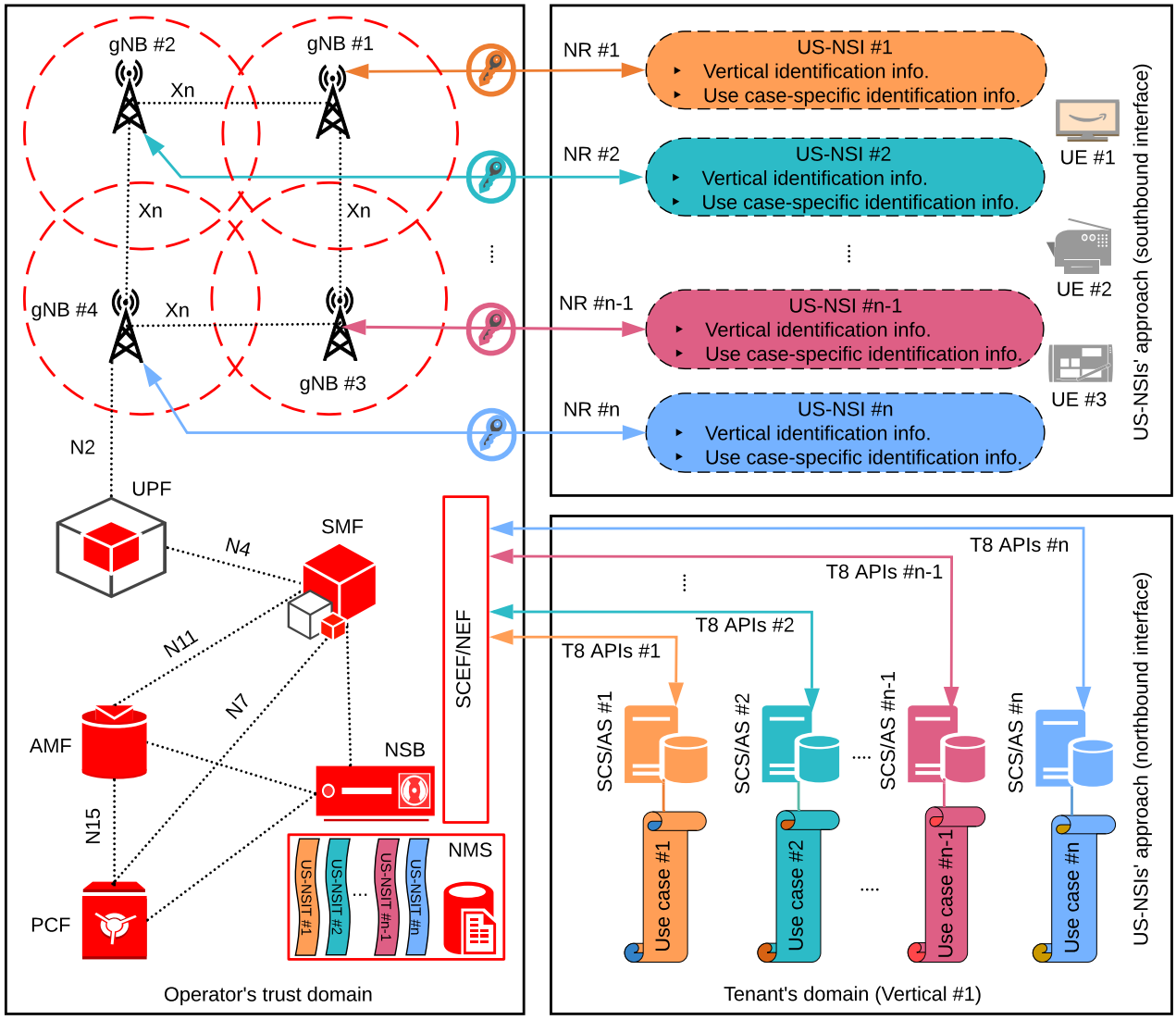

Figure A.2: The architectural framework of the use case specific network slicing.

the operator's trust domain. In this architecture, we assume that the SCS/AS is provided by the tenant. Fig. A.2 illustrates that every use case has been provided with its own SCS/AS. The SCEF/NEF provides the SCS/AS access to a rich set of information related to US-NSI. The SCS/AS exposes this information to obtain access to a significant amount of insight into UE that its corresponding US-NSI supports, including as user density, status and location of UE, connectivity, reachability, failure in the communication link, retainability, etc. This brings a number of advantages including energy efficiency, spectrum efficiency, simplicity in UE management, etc.

The 5GC does also consist of network slice broker (NSB), which enables the tenant to dynamically request and lease resources form the operator, based on US-NSIs' service requirements that are translated into technical requirements in their NSTs [191]. The NSB is located in the operator's network management system (NMS) and connected with the SCEF/NEF through APIs to tenants (see Fig. A.2). It dynamically allocates network resources, admission control, and NG-RAN scheduler configuration to every US-NSI.

Moreover, there is a number of new components introduced by 3GPP in both control and user planes for an efficient NFV, dynamic resource allocation, and flexible multi-tenant deployment (see Fig. A.2). On the control plane, the access and mobility management function (AMF)

supports authentication and registration of UE, the session management function (SMF) manages multi sessions contexts, and the policy control function (PCF) provides policy rules to control plane functions [192]. On the user plane, user-plane function (UPF) forwards and routes packets between UEs of US-NSI and data network through NG-RAN [192]. These components are interconnected using 3GPP defined interfaces (see Fig. A.2).
The NG-RAN components are also virtualized in terms of application, cloud, spectrum, and cooperation. The main objectives of virtualization in the NG-RAN are to map and implement US-NSIs in an efficient manner from the perspectives of energy, cost, radio resources, spectrum, and security; while guaranteeing the agreed SLA for each US-NSI. The NG-RAN (specifically the gNB) interacts over N2 or NG interface with the 5GC (see Fig. A.2) in order to provide required NFs and resources to each US-NSI [188]. The gNBs are interconnected over Xn interfaces, and deliver full NG-RAN functionality to interact with the UEs using the NR interface.

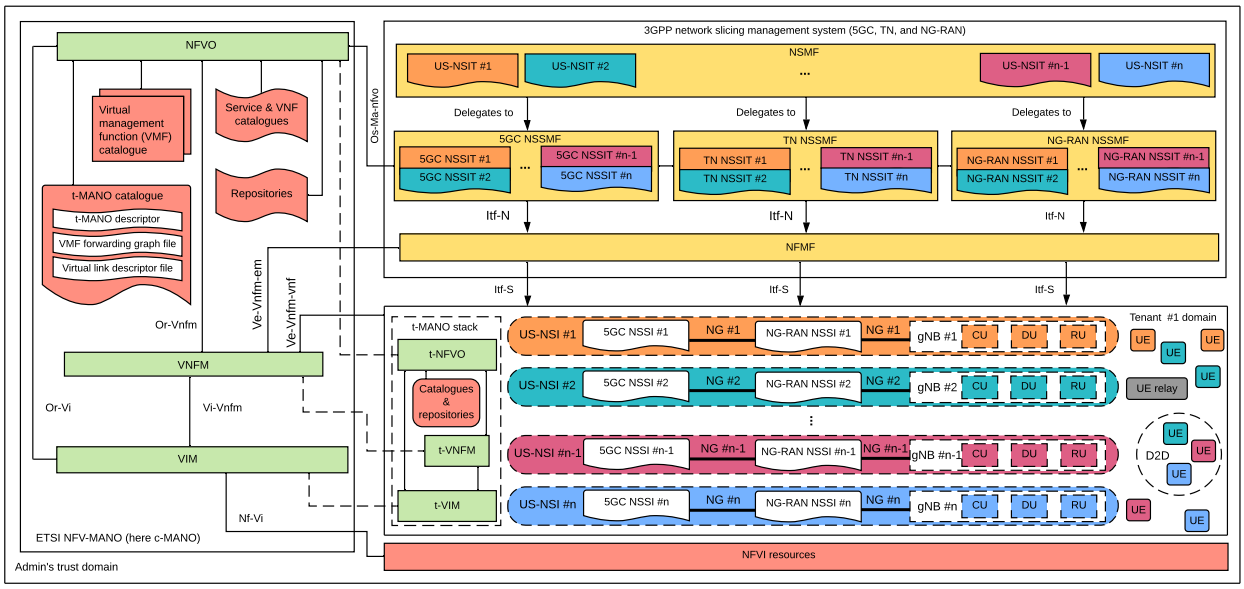

Figure A.3: The management and orchestration of the US-NSI over 3GPP/NFV-based slicing framework.

A.3.2 The Management and Orchestration of the US-NSIs

The Tenant #1 requests the CSMF, which is beyond the scope of this chapter, for the required communication services of its vertical. The CSMF translates these requirements to the US-NSIs related requirements and delivers them to the NSMF. The NSMF either selects predefined NSTs from NSC or creates new NSTs for requested US-NSIs. The US-NSI template (US-NSIT) makes the US-NSI network-wide available and identifies – using vertical/use case identification information (see the upper-right side of Fig. A.2) – it uniquely in the tenant's domain. The NSMF is also responsible for the orchestration and LCM of n US-NSIs, through n US-NSITs, using certain self-organizing network (SON) algorithms (see Fig. A.3).
Every US-NSI is composed of 5GC network subnet slice instance (NSSI) and NG-RAN NSSI, which are connected over TN NSSI. They are instantiated based on their corresponding templates; namely the 5GC NSSIT, NG-RAN NSSIT, and TN NSSIT, respectively, which are derived from their associated US-NSIT. The 5GC NSSI is a set of shared/dedicated VNFs/PNFs, which provides a particular 5GC behavior. The NG-RAN NSSI is a set of configured gNBs (including PNFs and VNFs), which provides a particular NG-RAN behavior.

The TN NSSI (illustrated by NG interface in Fig. A.3) is a set of networking and computing resources, which interconnects the 5GC and NG-RAN subnets.
To manage fault, configuration, accounting, performance, and security (FCAPS), and life-cycles of three aforementioned subnets, the 3GPP defines NSSMF. Every subnet has its own NSSMF, which reports to NSMF using management interfaces. The NSSMF maps the NSSI's requirements to communication, computation, and storage resources in 5GC, TN, and NG-RAN. Each NSSMF is connected to NFMF over Itf-N interface. Each type of component in each of the subnets has its own NFMF. For example, the gNB has three components namely the CU, DU, and RU. Each component of each gNB has its own NFMF in each NG-RAN NSSI and is responsible for the FCAPS of that particular component. The NFMF and the component under its control are interconnected over the Itf-S interface. To avoid duplicated components and redundant information, we put one NFMF in Fig. A.3.
To effectively manage and orchestrate PNFs/VNFs and their resources, the 3GPP network slicing management system (NSMS) is not enough, because the LCM of the VNFs and the components implementing them are in the scope of the ETSI NFV MANO [94]. The ETSI and 3GPP have thus jointly proposed a unified framework for the E2E management and orchestration of the 5G network slicing system illustrated in Fig. A.3. The NFV-MANO (hereinafter MANO) is composed of NFVO, VNFM, and VIM [94]. These three functional blocks are interconnected over standard interfaces, which are shown in Fig. A.3. Moreover, they are integrated with 3GPP NSMS entities over standard interfaces for effectively managing and orchestrating the US-NSIs. Further insights into the functional blocks and interfaces of MANO and their tentative integration with 3GPP management entities are out of the scope of this chapter. The interested readers are advised to refer to the relevant publications.
The current design of MANO provides a centralized management approach, which involves a discrete number of challenges including performance, management, monitoring and processing overload on central (c)-MANO, etc. To overcome these challenges and provide the tenant with the required autonomy for managing and orchestrating the US-NSIs and their corresponding resources, services, and policies, we propose the tenant (t)-MANO in the tenant's domain [94]. In this concept, a customized t-MANO instance is abstracted from a c-MANO to a tenant through negotiation and enforcement of management level agreement (MLA), which provides a significant level of control and capabilities to the tenant on requested US-NSIs. This enables the tenant to exercise MANO's functions over US-NSIs with minimum reliance on the c-MANO. However, the c-MANO has full administrative rights and controls on the t-MANO stack, such as monitoring MLA compliance and provide services, features, and capabilities.
The US-NSIs can be offered to the UEs from the same or different gNB(s)[188]. The 3GPP has also specified that a UE can be served by at most eight NSIs (a.k.a. the US-NSIs in this section) concurrently over a single NG-RAN, sharing a common AMF in all US-NSIs and an individual SMF to every US-NSI [86] in the following two scenarios (see the bottom-right side of Fig. A.3):

- **The UEs served by a dedicated US-NSI:** are the UEs of the same service category that are served by their corresponding US-NSI (differentiated by the same color in Fig. A.3) over their respective NR interfaces throughout an agreed business duration. Such UEs can also act as a relay to facilitate device-to-device (D2D) communication with other UEs of the same service category.

- **The UEs served by multiple US-NSIs:** are the UEs that are served by multiple US-NSIs (maximum eight) of different service categories in both intra-frequency and inter-frequency modes, concurrently. Each US-NSI is offered to such a UE through an

individual NR interface. These UEs can also facilitate D2D communication with other UEs of their service categories. Due to space limitations, these types of UEs are not illustrated in Fig. A.3.

A.3.3 The Identification and Selection of an US-NSI

The identity management of a US-NSI is crucial for both operator and tenant. To identify a US-NSI at the UE, NG-RAN, and 5GC; two categories of identification information are proposed: the vertical and the use case specific.

- The vertical category of identification information includes information related to the tenant, such as vertical type (e.g., V2X system, power grid, healthcare, etc.), tenant's ID (e.g., Tenant #1, Tenant #2, Tenant #3, etc.), and service providing locations (e.g., urban areas, rural areas, highways, cell #n, etc.) in order to distinguish a selected vertical from other verticals.

- The use case specific category of identification information is containing more specific information to distinguish a US-NSI among a family of US-NSIs of a vertical, such as NSI type (e.g., eMBB, uRLLC, mMTC, and non-standard NSIs), US-NSI's ID (e.g., US-NSI #1, US-NSI #2, US-NSI #3, etc.), and the LCM of each US-NSI.

Both sets of identification information are used to offer customized performance to corresponding US-NSI and to produce measurement reports. Moreover, they are utilized by the UE to request a US-NSI from a nearby gNB. This process takes place during the registration of UE. The 5GC authorizes the UE to attach to its corresponding US-NSI(s). Both sets must be coherent with the identity management strategy, more specifically with the NSSAI specified by 3GPP in [86].
The NSSAI is a set (up to eight) of S-NSSAIs, which are sent in signalling messages between UE and 5G network in order to assist the 5G network in the selection of an NSI. The S-NSSAI is comprised of two components: the SST and the SD. The SST is a mandatory 8-bits component, which identifies the behaviour (in terms of services and features) of an NSI [193]. The SD is an optional 24-bits component, which complements the SST by distinguishing multiple homogeneous NSIs that serve different tenants from each other [193]. Some of the identification information of US-NSI have already been proposed by 3GPP. For example, expected NSI's behaviour in terms of services and features, which is the SST, have already been defined, known as eMBB, uRLLC, and mMTC. Whereas, some other identity information, such as the ID of US-NSI, have not been considered by the 3GPP so far. In order to include proposed additional identification information in the S-NSSAI, two options are available:

- In standardized S-NSSAI (which has only SST and not SD), both vertical and use case specific identification information of a US-NSI, shall be included in the standardized-SST.

- In non-standardized S-NSSAI, both vertical and use case specific identification information of a US-NSI, are either included only in non-standardized-SST or split into SST and SD.

A.4 Sub-network Slicing

The sub-network slicing is an alternative approach to providing network slicing solution to multiple use cases of a vertical. In this concept, a number of NSIs, called the S-NSI (or Micro NSIs), are designed to fulfill the service requirements of a vertical that consists of multiple use cases. Every S-NSI is dedicated to a use case. These S-NSIs are then logically clustered or bundled (see Fig. A.4) and subsequently provided as a single business product to the tenant. This family (or group) of S-NSIs is called the GN-NSI. The GN-NSI is a per-vertical (or Macro) NSI, which is designed for a specific type of vertical. It does not necessarily convey the meaning of one NSI per vertical, but rather, it is a logical cluster (or a set) of n S-NSIs. There are two types of GN-NSIs:

- **Standard GN-NSIs:** is a GN-NSIs, which is by default defined, designed, and configured with all of its S-NSIs through joint efforts between operators, vendors, verticals, regulators, and SDOs, and its NST is already inserted into the NSC. When the tenant requests a GN-NSIs, the type of its vertical has to be identified, its corresponding NST has to be selected from the NSC, and the GN-NSIs must be subsequently offered. This approach makes the whole business process, more specifically the preparation phase of the LCM of a GN-NSIs, simpler and easier. Both sides can reach a win-win agreement in a short time.

- **Non-standard GN-NSIs:** is a GN-NSIs, which is not available in the NSC or is not yet standardized by the five aforementioned parties. In this case, the operator and tenant privately negotiate to define the attributes, design its NST, and subsequently insert it into the NSC. Both parties equally participate in the configuration of the requirements, the allocation of resources and NFs, and the instantiation of S-NSIs. In this approach, the tenant has more influence over the configuration of the GN-NSIs and its corresponding S-NSIs as well as the allocation of required NFs and resources during the whole business period.

A.4.1 The Architectural Framework of the Sub-network Slicing

In Fig. A.4, Vertical #1 (exemplified in the previous section) is assumed again. Each use case is provisioned with an independent S-NSI (i.e., S-NSI #1 – S-NSI #n, respectively). These S-NSIs are then clustered in the form of GN-NSIs #1. Tenant #1 requests a GN-NSIs in the northbound interface (see the bottom-right side of Fig. A.4). The GN-NSIs's request coming through a set of APIs indicates its requested NST. The NSB is in charge of granting or denying the GN-NSIs's request, and dynamically allocating of resources to the tenant based on a pre-defined SLA. Once the request is granted, the operator provides the desired GN-NSIs back through its own set of APIs from SCEF/NEF to SCS/AS.
In order for the tenant to interact with the operator and manage its S-NSIs, we propose the Master/Slave (which is also called the Primary/Secondary) communication protocol in the northbound interface. In this architecture, a single Master-SCS/AS and multi Slave-SCS/ASs (i.e., Slave-SCS/AS #1 – Slave-SCS/AS #n), controlled and managed by a system administrator, are taken into consideration (see Fig. A.4). Both Master/Slave-SCS/ASs must have their own application systems. The Master-SCS/AS requests the GN-NSIs from the operator and controls n Slave-SCS/ASs. Each Slave-SCS/AS is customized to a use case and runs a single S-NSI. Once the tenant's request for the GN-NSIs is admitted, and the Master-SCS/AS and the Slave-SCS/AS relationship is established, the direction of the S-NSI's request, resource allocation, and control is always from Master to Slave(s).

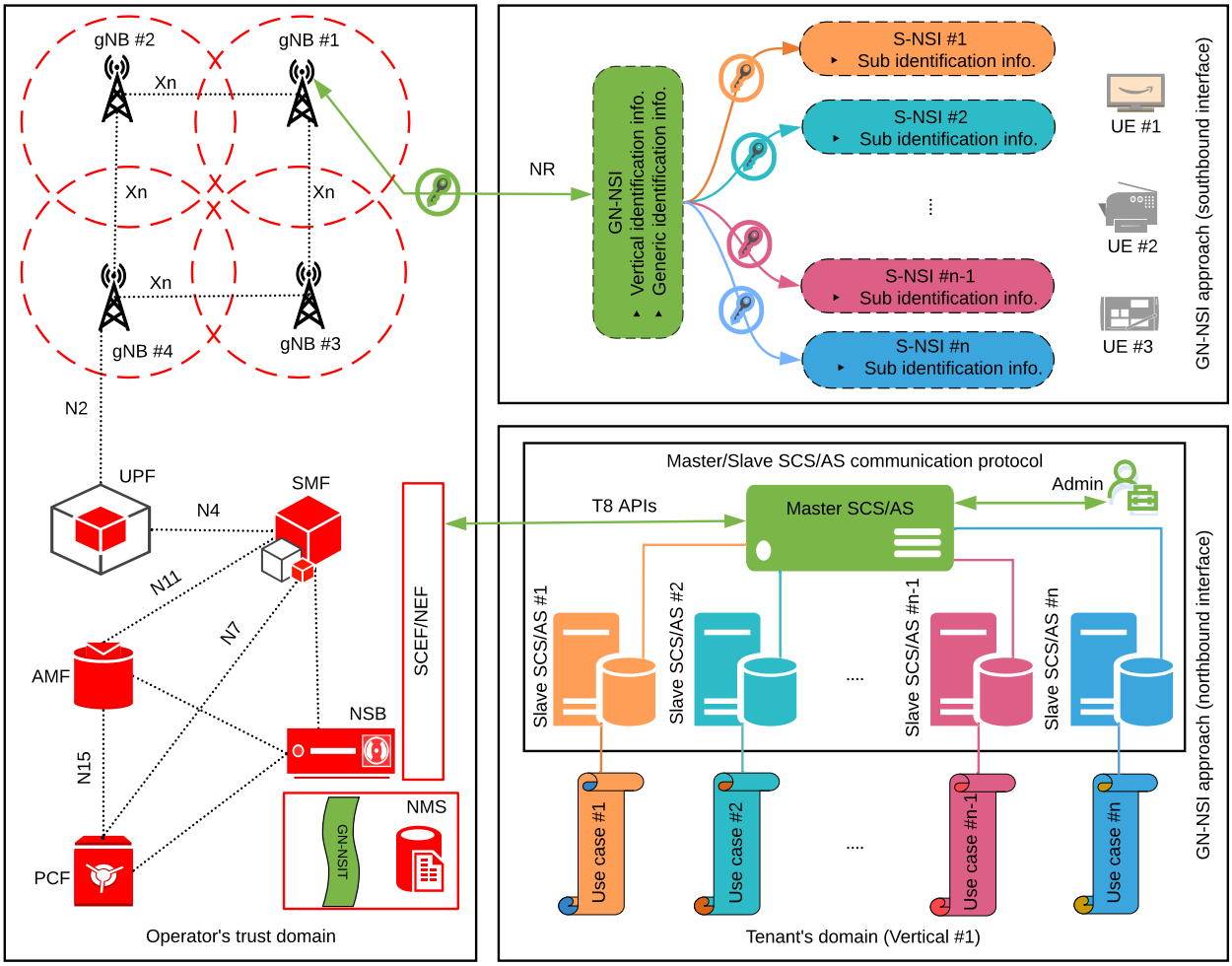

Figure A.4: The architectural framework of the sub-network slicing.

Through the southbound interface (see the upper-right side of Fig. A.4), the tenant accesses a GN-NSIs, containing S-NSIs, over their corresponding NR interfaces. The GN-NSIs and its associated S-NSIs can be offered from a single or multiple gNB(s). Some radio processing functions and resources are shared among S-NSIs, whereas some are dedicated to S-NSI [194]. The configuration and allocation of resources and NFs of S-NSIs in a logical cluster of GN-NSIs in the NG-RAN is a challenging issue. Further research is thus required to analyze the feasibility and efficiency of sub-network slicing in the NG-RAN.
The UEs may or may not access the S-NSIs concurrently. For example, if a vehicle connected to a V2X GN-NSIs is parked in a parking area. Two S-NSIs (i.e., related to infotainment applications and autonomous driving) are presumably disconnected, whereas the third S-NSI (i.e., related to vehicle management) may be operating. However, if a vehicle is in the driving mode, then all three S-NSIs, in a cluster of GN-NSIs, are simultaneously provided to the vehicle. The UE can not access more than eight S-NSIs concurrently in the NG-RAN [193]. Therefore, a GN-NSIs may consist of up to eight S-NSIs. The UE access scenarios to the S-NSIs of a GN-NSIs (see Fig. A.5) are also the same as access to the US-NSI discussed in section A.3.2.

A.4.2 The Management and Orchestration of the S-NSIs

The CSMF receives a list of service requirements from Tenant #1, translates it to the GN-NSIs's requirements, and delivers it to the NSMF. The NSMF either selects a predefined

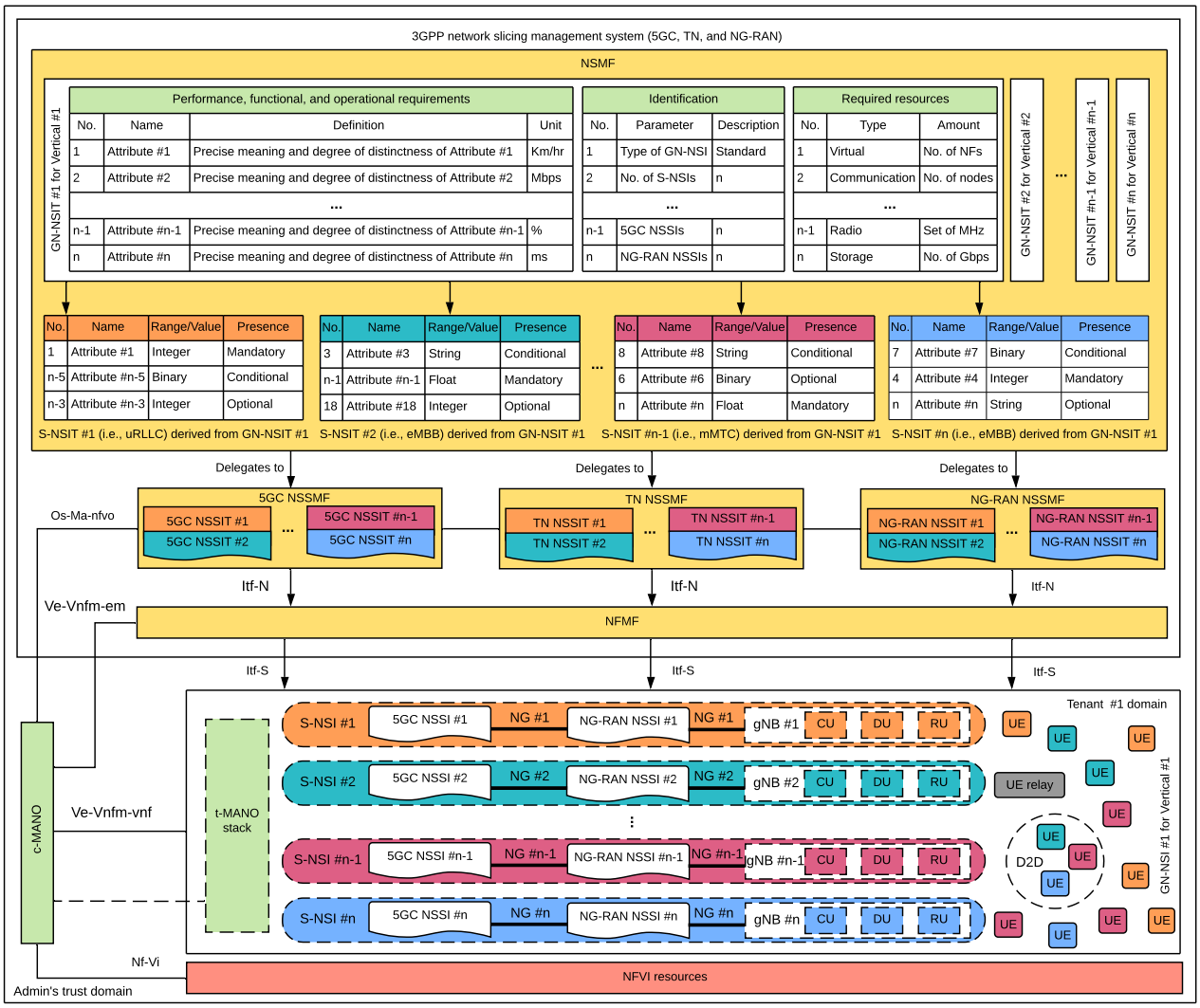

Figure A.5: The management and orchestration of S-NSIs in a logical cluster of GN-NSI over 3GPP/NFV-based slicing framework.

NST from NSC or creates a new NST for requested GN-NSIs. The NST of a GN-NSIs is called the GN-NSI template (GN-NSIT). The NSMF is also responsible for the LCM of n GN-NSIs through their corresponding n GN-NSITs (see Fig. A.5).
The GN-NSIT is composed of functional, performance, and operational attributes (each attribute with its own definition and unit of measurement), required resources information, and identification information. The two sets of information are in compliance with the 3GPP definition of NST. However, to further identify the S-NSIs and their corresponding components, we propose an identification set of information in the GN-NSIT – vertical/generic on the GN-NSIs level and Sub on the S-NSIs level. These three sets of information, which define a GN-NSIs from technical and business perspectives, are subsequently used to obtain a group of S-NSIs, according to the number of the use cases of a vertical, (see Fig. A.5).
Each S-NSI has also an NST with industry-accepted characteristics, which is called the S-NSI template (S-NSIT). The attributes that define an S-NSIT are taken from GN-NSIT (see Fig. A.5). The S-NSIT only includes necessary information related to the range/value (i.e., integer, binary, string, float, etc.) and presence (i.e., mandatory, optional, and conditional) of attributes required by the S-NSI. Once the selection/design of GN-NSIT and its associated S-NSITs are completed. The NSMF is responsible for FCAPS, management, and orchestration throughout their life-cycles. The FCAPS, LCM, and templates of the components of the

S-NSI are the same as those of US-NSI. The t-MANO has also introduced here to effectively manage and orchestrate the virtualized part of the S-NSIs in the tenant's domain.

A.4.3 The Identification and Selection of a GN-NSI and its S-NSIs

To identify the GN-NSIs and its corresponding S-NSIs, we propose three types of identification information: vertical, generic, and sub. According to their purposes, these three types of identification information may be arranged into two categories: external (or global), which contains vertical and generic types of information; and internal (or domestic), which contains sub type of information. The former category is used to identify a GN-NSIs and broadcasted in the 3GPP network for interaction between operator and tenant over resource request, slice request, resource allocation, etc. The latter category is used by tenant (in its domain) in order to identify and distinguish its various S-NSIs. In this way, an operator utilizes a relatively small portion of information (from the total size of identification information) for the selection of a GN-NSIs in the 3GPP network; and the remaining portion is used only in the tenant's domain for the selection of the S-NSI. This approach thus avoids inefficient utilization of predefined size of bits dedicated to SST and SD in the S-NSSAI.

- The vertical type of identification information of a GN-NSIs is similar to that of the US-NSI's ones, which includes information related to a tenant, such as vertical type, tenant ID, and service providing location in order to distinguish a selected vertical from other homogeneous verticals.
- The generic type of identification information is used to recognize a GN-NSIs from other homogeneous GN-NSIs. It contains information related to the number of the S-NSIs that a GN-NSIs consists (e.g., a GN-NSIs dedicated to vertical X contains n S-NSIs), the overall SLA duration and the LCM of a GN-NSIs (e.g., from DD/MM/YYYY to DD/MM/YYYY), and the GN-NSIs's ID (e.g., GN-NSIs #1, GN-NSIs #2, GN-NSIs #3, etc.).
- The sub type of the identification information contains more specific information on the S-NSIs' level, such as the NSI type (e.g., eMBB, uRLLC, and mMTC), the S-NSI's ID (e.g., S-NSI #1, S-NSI #2, S-NSI #3, etc.), the LCM of every S-NSI, etc. in order to identify a S-NSI among other S-NSIs of the same vertical.

Some of the above-mentioned identification information are already proposed by the 3GPP, whereas some are not yet taken into consideration. These three types of information must be coherent with the NSSAI and should be included either only in SST (in the case of standard S-NSSAI) or in both SST and SD (in the case of non-standard S-NSSAI).

A.5 Conclusions Remarks

In this chapter, we have addressed the solution gap in terms of the flexible and dynamic provisioning and efficient management of per-vertical per-use-case NSIs tailored to the respective use case service and resource requirements in order to ensure its functional and operational integrity in isolation from other use cases of a single vertical. In this context, we have presented two solution concepts namely the use case Specific Network Slicing and the sub-network slicing, and their architectural frameworks, while placing a special emphasis on their management and orchestration in 5G mobile networks. It is important to mention that both concepts and their corresponding architectures are in compliance with the definition of

network slicing, as defined by the NGMN Alliance and the 3GPP. Due to this, it is planned to introduce these two solution concepts in the relevant SDOs, such as in the relevant technical specification groups of 3GPP and ETSI NFV in order to address the solution gap to this important requirement.

In the future, we are interested to investigate the qualitative and quantitative comparison of a number of technical (such as resource allocation, isolation, security, admission control, etc.) and business (such as LCM, capital and operational expenditure, SLA, etc.) related parameters of both concepts in order to figure out their potential deployment implications on 5G and beyond mobile networks. The comparative study of both approaches will give interested readers further insights into the applicability and feasibility of the US-NSI and the GN-NSIs architectural frameworks in terms of efficiency, complexity, and flexibility. There is no doubt that the implementation of both concepts is also going to raise a number of technical challenges for the UE, the NG-RAN, and the 5GC. Exploring these research problems in the future is thus essential with the purpose of motivating new advances and potential solutions related to the US-NSIs and the GN-NSIs.

Appendix B

List of Publications

1. P. K. Kakani, **M. A. Habibi**, M. R. C. Balannagari, *et al.*, "Mitigating ML-Driven Adversarial Attacks on xApps Using Dynamic Defense Mechanisms," in *IEEE Open Journal of the Communications Society*, vol. 6, pp. 6912-6929, 2025.

2. S. B. Mallikarjun, **M. A. Habibi**, *et al.*, "Exploring Y1 Communications and Services in O-RAN: Background, Privacy, and Security," in *IEEE Open Journal of the Communications Society*, vol. 6, pp. 4638-4666, 2025.

3. P. Porambage, M. Christopoulou, B. Han, **M. A. Habibi**, *et al.*, "Security, Privacy, and Trust for Open Radio Access Networks in 6G," in *IEEE Open Journal of the Communications Society*, vol. 6, pp. 332-361, 2025.

4. **M. A. Habibi**i, Bin Han, Merve Saimler, *et al.*, "Towards an AI/ML-driven SMO Framework in O-RAN: Scenarios, Solutions, and Challenges," in *IEEE Future Networks World Forum 2024*, 1-8, 2024.

5. **M. A. Habibi**, G. M. Yilma, U. Fattore, X. Costa-Pérez, and H. D. Schotten, "Unlocking O-RAN Potential: How Management Data Analytics Enhances SMO Capabilities?," Accepted in *IEEE Open Journal of the Communications Society*, July 2024.

6. K. Alam, **M. A. Habibi**, *et al.*, "A Comprehensive Overview and Survey of O-RAN: Exploring Slicing-aware Architecture, Deployment Options, and Use Cases," Submitted to *IEEE Communications Surveys & Tutorials*, June 05, 2024.

7. **M. A. Habibi**, G. M. Yilma, X. Costa-Pérez and H. D. Schotten, "Unifying 3GPP, ETSI, and O-RAN SMO Interfaces: Enabling Slice Subnets Interoperability," 2023 *IEEE Future Networks World Forum (FNWF)*, Baltimore, MD, USA, 2023, pp. 1-8.

8. W. Jiang, Q. Zhou, J. He, **M. A. Habibi**, *et al.*, "Terahertz Communications and Sensing for 6G and Beyond: A Comprehensive Review," Accepted in *IEEE Communications Surveys & Tutorials*, April 08, 2024.

9. B. Han, **M. A. Habibi**, B. Richerzhagen, K. Schindhelm, F. Zeiger, F. Lamberti, A. Cannavo, M. M. Upadhya, C. Korovesis, I.-P. Belikaidis, P. Demestichas, S. Yuan, and H. D. Schotten, "Digital Twins for Industry 4.0 in the 6G Era," in *IEEE Open Journal of Vehicular Technology*, vol. 4, pp. 820-835, October 17, 2023.

10. **M. A. Habibi**, B. Han, A. Fellan, W. Jiang, I. L. Pavón, Adrián Gallego Sánchez, A. Boubendir, and H. D. Schotten, "Towards an Open, Intelligent, and End-to-End Architectural Framework for Network Slicing in 6G Communication Systems," in *IEEE Open Journal of the Communications Society*, vol. 4, pp. 1615-1658, July 11, 2023.

11. **M. A. Habibi**, A. G. Sánchez, I. L. Pavón, B. Han, P. Serrano, J. P.-Valero, and A. Virdis, and Hans D. Schotten, "The Architectural Design of Service Management and Orchestration in 6G Communication Systems," in *IEEE INFOCOM 2023*, New York, USA, May 2023.

12. **M. A. Habibi**, A. G. Sánchez, I. L. Pavón, B. Han, G. Landi, B. Sayadi, C. Ntogkas, I.-P. Belikaidis, H. D. Schotten, P. Serrano, J. P.-Valero, and A. Virdis, "Enabling Network and Service Programmability in 6G Mobile Communication Systems," in *IEEE Future Networks World Forum (FNWF)*, Montreal, Canada, pp. 320-327, 2022.

13. N. P. Kuruvatti, **M. A. Habibi**, S. Partani, B. Han, A. Fellan, and H. D. Schotten, "Empowering 6G Communication Systems With Digital Twin Technology: A Comprehensive Survey," in *IEEE Access*, vol. 10, pp. 112158-112186, 2022.

14. **M. A. Habibi**, F. Z. Yousaf, and H. D. Schotten, "Mapping the VNFs and VLs of a RAN Slice onto Intelligent PoPs in Beyond 5G Mobile Networks," in *IEEE Open Journal of the Communications Society*, vol. 3, pp. 670-704, April 18, 2022.

15. **M. A. Habibi**, B. Han, M. Nasimi, N. P. Kuruvatti, A. Fellan, and H. D. Schotten, "Towards a Fully Virtualized, Cloudified, and Slicing-aware RAN for 6G Mobile Networks," In. Yulei Wu *et al. 6G Mobile Wireless Networks*. Springer, March 2021, ch. 15, pp. 327-358.

16. W. Jiang, B. Han, **M. A. Habibi**, and H. D. Schotten, "The Road Towards 6G: A Comprehensive Survey," in *IEEE Open Journal of the Communications Society*, vol. 2, pp. 334-366, February 08, 2021.

17. B. Han, W. Jiang, **M. A. Habibi**, and H. D. Schotten, "An Abstracted Survey on 6G: Drivers, Requirements, Efforts, and Enablers," in the 2020 *Workshop on Next Generation Networks and Applications*, Kaiserslautern, Germany, December 15, 2020.

18. **M. A. Habibi**, B. Han, F. Z. Yousaf, and H. D. Schotten, "How Should Network Slice Instances be Provided to Multiple Use Cases of a Single Vertical Industry?," in *IEEE Communication Standards Magazine*, vol. 4, no. 3, pp. 53-61, September 23, 2020.

19. M. Nasimi, **M. A. Habibi**, and H. D. Schotten, "Platoon-assisted Vehicular Cloud in VANET: Vision and Challenges," in *European Symposium on Computer and Communications (ESCC)*, November 20-22, 2019, Paris, France.

20. **M. A. Habibi**, M. Nasimi, B. Han, and H. D. Schotten, "A Comprehensive Survey of RAN Architectures Toward 5G Mobile Communication System," in *IEEE Access*, vol. 7, pp. 70371-70421, May 28, 2019.

21. M. Nasimi, **M. A. Habibi**, B. Han, and H. D. Schotten, "Edge-Assisted Congestion Control Mechanism for 5G Network Using Software-Defined Networking," in *15th International Symposium on Wireless Communication Systems (ISWCS)*, August 28-31, 2018, Lisbon, Portugal.

22. **M. A. Habibi**, B. Han, M. Nasimi, and H. D. Schotten, "The Structure of Service Level Agreement of Slice-based 5G Network," in *29th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, September 9 – 12, 2018, Bologna, Italy.

23. **M. A. Habibi**, B. Han, and H. D. Schotten, "Network Slicing in 5G Mobile Communication: Architecture, Profit Modeling, and Challenges," in *14th International Symposium on Wireless Communication Systems (ISWCS)*, September 28 – October 1, 2017, Bologna, Italy.

24. B. Han, **M. A. Habibi**, and H. D. Schotten, "Optimal Resource Dedication in Grouped Random Access for Massive Machine Type Communications," in *IEEE Conference on Standards for Communications and Networking (CSCN)*, September 18 – 20, 2017, Helsinki, Finland.

Bibliography

[1] M. A. Habibi *et al.*, "Unifying 3GPP, ETSI, and O-RAN SMO interfaces: Enabling slice subnets interoperability," in *IEEE FNWF*, 2023. Baltimore, MD, USA, pp. 1-8.

[2] S. Knight *et al.*, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.

[3] W. Jiang *et al.*, "The road towards 6G: A comprehensive survey," *IEEE Open Journal of the Communications Society*, vol. 2, pp. 334–366, 2021.

[4] M. A. Habibi *et al.*, "A comprehensive survey of RAN architectures toward 5G mobile communication system," *IEEE Access*, vol. 7, pp. 70371–70421, 2019.

[5] M. A. Habibi *et al.*, "How should network slice instances be provided to multiple use cases of a single vertical industry?," *IEEE Communications Standards Magazine*, vol. 4, no. 3, pp. 53–61, 2020.

[6] W. Jiang *et al.*, "Terahertz communications and sensing for 6G and beyond: A comprehensive review," *IEEE Communications Surveys & Tutorials*, vol. 26, no. 4, pp. 2326–2381, 2024.

[7] NGMN Alliance, "Description of network slicing concept." Version 1.0, January 2016.

[8] X. Zhou *et al.*, "Network slicing as a service: enabling enterprises' own software-defined cellular networks," *IEEE Communications Magazine*, vol. 54, pp. 146–153, July 2016.

[9] 3GPP, "3GPP TS 28.530 V16.0.0, Technical Specification Group Services and System Aspects; Management and Orchestration; Concepts, Use Cases and Requirements. Release 16." September 2019.

[10] GSMA, "Road to 5G: Introduction and migration." April 2018.

[11] K. B. Letaief *et al.*, "The roadmap to 6G: AI empowered wireless networks," *IEEE Communications Magazine*, vol. 57, no. 8, pp. 84–90, 2019.

[12] M. Bagaa *et al.*, "Collaborative cross system AI: toward 5G system and beyond," *IEEE Network*, pp. 1–9, 2021.

[13] J. Du *et al.*, "Machine learning for 6G wireless networks: Carrying forward enhanced bandwidth, massive access, and ultrareliable/low-latency service," *IEEE Vehicular Technology Magazine*, vol. 15, no. 4, pp. 122–134, 2020.

[14] M. A. Habibi *et al.*, "Unlocking O-RAN potential: How management data analytics enhances SMO capabilities?," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 4710–4730, 2024.

[15] M. Condoluci and T. Mahmoodi, "Softwarization and virtualization in 5G mobile networks: Benefits, trends and challenges," *Computer Networks*, vol. 146, pp. 65–84, 2018.

[16] R. F. Moyano *et al.*, "NFV and SDN-based differentiated traffic treatment for residential networks," *IEEE Access*, vol. 8, pp. 34038–34055, 2020.

[17] H. Yang *et al.*, "Artificial-intelligence-enabled intelligent 6G networks," *IEEE Network*, vol. 34, no. 6, pp. 272–280, 2020.

[18] Y. Xiao *et al.*, "Toward self-learning edge intelligence in 6G," *IEEE Communications Magazine*, vol. 58, no. 12, pp. 34–40, 2020.

[19] 3GPP, "3GPP TS 28.801, Technical Specification Group Services and System Aspects; Telecommunication management; Study on management and orchestration of network slicing for next generation network (Release 15). V15 1.0." January 2018.

[20] L. M. Contreras *et al.*, "Orchestration of crosshaul slices from federated administrative domains," in *2016 European Conference on Networks and Communications (EuCNC)*, pp. 220–224, 2016.

[21] 5GNORMA, "Deliverable D3.2, 5G NORMA network architecture - intermediate report." Version 1.0, January 2017.

[22] 5GESSENCE, "Deliverable D3.2, Final report on network embedded cloud, 5G SDRAN controller and network slicing." Version 1.0, May 2019.

[23] 5GCHARISMA, "Deliverable D1.1, CHARISMA intelligent, distributed low-latency security C-RAN/RRH architecture." Version 1.1, June 2016.

[24] X. Li *et al.*, "5G-crosshaul network slicing: Enabling multi-tenancy in mobile transport networks," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 128–137, 2017.

[25] F. Giannone *et al.*, "Impact of virtualization technologies on virtualized ran midhaul latency budget: A quantitative experimental evaluation," *IEEE Communications Letters*, vol. 23, no. 4, pp. 604–607, 2019.

[26] H. Hirayama *et al.*, "RAN Slicing in Multi-CU/DU Architecture for 5G Services," in *IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, pp. 1–5, 2019.

[27] ETSI, "ETSI GR NFV-EVE 012 V3.1.1, Network Functions Virtualisation (NFV) Release 3; Evolution and Ecosystems; Report on Network Slicing Support with ETSI NFV Architecture Framework." December 2017.

[28] 3GPP, "3GPP TS 28.533; Technical Specification Group Services and System Aspects; Management and orchestration; Architecture framework (Release 16). V16.3.0." March 2020.

[29] O-RAN Alliance, "O-RAN.WG1.V03.00, O-RAN Architecture Description." November 2020.

[30] U. U. Rahman *et al.*, "Nutshell-simulation toolkit for modeling data center networks and cloud computing," *IEEE Access*, vol. 7, pp. 19922–19942, 2019.

[31] L. Ruiz *et al.*, "Genetic algorithm for holistic VNF-mapping and virtual topology design," *IEEE Access*, vol. 8, pp. 55893–55904, 2020.

[32] NGMN Alliance, "A Deliverable by the NGMN Alliance: NGMN 5G White Paper." February 2015.

[33] NGMN Alliance, "NGMN Overview on 5G RAN Functional Decomposition v 1.0." February 2018.

[34] NGMN Alliance, "5G RAN CU – DU Network Architecture, Transport Options and Dimensioning v 1.0." April 2019.

[35] 3GPP, "3GPP TS 22.261 V17.1.0, Technical Specification Group Services and System Aspects; Service requirements for the 5G system; Stage 1. (Release 17)." December 2019.

[36] 3GPP, "3GPP TS 38.401 V16.1.0, Technical Specification Group Radio Access Network; NG-RAN; Architecture description (Release 16)." March 2020.

[37] ETSI, "ETSI GR MEC 024 V2.1.1, Multi-access Edge Computing (MEC); Support for network slicing." November 2019.

[38] ETSI, "ETSI GR NGP 011 V1.1.1, Next Generation Protocols (NGP); E2E Network Slicing Reference Framework and Information Model." September 2018.

[39] ETSI, "ETSI GS NFV-MAN 001 V1.1.1, Network Functions Virtualisation (NFV); Management and Orchestration." December 2014.

[40] ETSI, "ETSI GS NFV-SWA 001, Network Functions Virtualisation (NFV); Virtual Network Functions Architecture." December 2012.

[41] ETSI, "ETSI GS NFV 002 V1.2.1, Network Functions Virtualisation (NFV); Architectural Framework." December 2014.

[42] ETSI, "ETSI GS NFV-INF 001 V1.1.1, Network Functions Virtualisation (NFV); Infrastructure Overview." January 2015.

[43] ETSI, "ETSI GS NFV-INF 003 V1.1.1, Network Functions Virtualisation (NFV); Infrastructure; Compute Domain." December 2014.

[44] ETSI, "ETSI GS NFV-INF 004 V1.1.1, Network Functions Virtualisation (NFV); Infrastructure; Hypervisor Domain." January 2015.

[45] ETSI, "ETSI GS NFV-INF 005 V1.1.1, Network Functions Virtualisation (NFV); Infrastructure; Network Domain." December 2015.

[46] ETSI, "ETSI GS NFV-IFA 014 V2.1.1, Network Functions Virtualisation (NFV); Management and Orchestration; Network Service Templates Specification." October 2016.

[47] ETSI, "ETSI GS ENI 005 V1.1.1, Experiential Networked Intelligence (ENI); System Architecture." September 2019.

[48] O-RAN Alliance, "O-RAN-WG6.CAD-V01.00.00 Technical Report, Cloud Architecture and Deployment Scenarios for O-RAN Virtualized RAN." October 2019.

[49] A. Garcia-Saavedra and X. Costa-Pérez, "O-RAN: Disrupting the Virtualized RAN Ecosystem," *IEEE Communications Standards Magazine*, vol. 5, no. 4, pp. 96–103, 2021.

[50] GSMA, "Network slicing use case requirements." Version 1.0, April 2018.

[51] GSMA, "From Vertical Industry Requirements to Network Slice Characteristics." August 2018.

[52] GSMA, "Generic Network Slice Template Version 1.0." May 2019.

[53] IETF, "Considerations on Network Virtualization and Slicing." November 2017.

[54] IETF, "Problem Statement of Common Operation and Management of Network Slicing." March 2018.

[55] IETF, "COMS Architecture." March 2018.

[56] IETF, "Interconnecting (or Stitching) Network Slice Subnets." March 2020.

[57] IETF, "Gateway Function for Network Slicing." March 2018.

[58] O. Sallent *et al.*, "On radio access network slicing from a radio resource management perspective," *IEEE Wireless Communications*, vol. 24, no. 5, pp. 166–174, 2017.

[59] Y. L. Lee *et al.*, "A new network slicing framework for multi-tenant heterogeneous cloud radio access networks," in *2016 International Conference on Advances in Electrical, Electronic and Systems Engineering (ICAEES)*, pp. 414–420, 2016.

[60] D. H. Kim *et al.*, "Distributed radio slice allocation in wireless network virtualization: Matching theory meets auctions," *IEEE Access*, vol. 8, pp. 73494–73507, 2020.

[61] V. N. Ha *et al.*, "End-to-end network slicing in virtualized OFDMA-based cloud radio access networks," *IEEE Access*, vol. 5, pp. 18675–18691, 2017.

[62] S. Vural *et al.*, "Dynamic preamble subset allocation for RAN slicing in 5G networks," *IEEE Access*, vol. 6, pp. 13015–13032, 2018.

[63] D. Marabissi *et al.*, "Highly flexible RAN slicing approach to manage isolation, priority, efficiency," *IEEE Access*, vol. 7, pp. 97130–97142, 2019.

[64] S. E. Elayoubi *et al.*, "5G RAN slicing for verticals: Enablers and challenges," *IEEE Communications Magazine*, vol. 57, pp. 28–34, January 2019.

[65] H. Xiang *et al.*, "Network slicing in fog radio access networks: Issues and challenges," *IEEE Communications Magazine*, vol. 55, no. 12, pp. 110–116, 2017.

[66] A. Laghrissi *et al.*, "Towards edge slicing: VNF placement algorithms for a dynamic realistic edge cloud environment," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pp. 1–6, 2017.

[67] I. Benkacem *et al.*, "Optimal VNFs placement in CDN slicing over multi-cloud environment," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 616–627, 2018.

[68] A. D. Domenico *et al.*, "Optimal virtual network function deployment for 5G network slicing in a hybrid cloud infrastructure," *IEEE Transactions on Wireless Communications*, vol. 19, no. 12, pp. 7942–7956, 2020.

[69] H. Gupta *et al.*, "Apt-RAN: A flexible split-based 5G RAN to minimize energy consumption and handovers," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 473–487, 2020.

[70] R. L. Gomes *et al.*, "Energy-aware slicing of network resources based on elastic demand through daytime," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pp. 604–608, 2019.

[71] S. Ravindran *et al.*, "EESO: Energy efficient system-resource optimization of multi-sub-slice-connected user in 5G RAN," in *2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, pp. 1–6, 2020.

[72] J. Mei *et al.*, "An intelligent self-sustained RAN slicing framework for diverse service provisioning in 5G-beyond and 6G networks," *Intelligent and Converged Networks*, vol. 1, no. 3, pp. 281–294, 2020.

[73] T. C. Chuah *et al.*, "Intelligent RAN slicing for broadband access in the 5G and big data era," *IEEE Communications Magazine*, vol. 58, no. 8, pp. 69–75, 2020.

[74] T. Pamuklu *et al.*, "Reinforcement learning based dynamic function splitting in disaggregated green open RANs," *https://arxiv.org/abs/2012.03213*, Feb. 2021.

[75] H. Yu *et al.*, "Isolation-aware 5G RAN slice mapping over WDM metro-aggregation networks," *Journal of Lightwave Technology*, vol. 38, no. 6, pp. 1125–1137, 2020.

[76] F. Z. Morais *et al.*, "Placeran: Optimal placement of virtualized network functions in the next-generation radio access networks," *https://arxiv.org/pdf/2102.13192.pdf*, March 2021.

[77] R. Riggio *et al.*, "Scheduling wireless virtual networks functions," *IEEE Transactions on Network and Service Management*, vol. 13, no. 2, pp. 240–252, 2016.

[78] L. Diez *et al.*, "Flexible Functional Split and Fronthaul Delay: A Queuing-Based Model," *IEEE Access*, vol. 9, pp. 151049–151066, 2021.

[79] C. Mei *et al.*, "5G network slices embedding with sharable virtual network functions," *Journal of Communications and Networks*, vol. 22, no. 5, pp. 415–427, 2020.

[80] H. A. Alameddine *et al.*, "On the interplay between network function mapping and scheduling in VNF-based networks: A column generation approach," *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 860–874, 2017.

[81] J. Martín-Pérez *et al.*, "Multi-domain VNF mapping algorithms," in *2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pp. 1–6, 2018.

[82] M. A. Tahmasbi Nejad *et al.*, "vSPACE: VNF simultaneous placement, admission control and embedding," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 542–557, 2018.

[83] M. Gamal *et al.*, "Mapping and scheduling for non-uniform arrival of virtual network function (VNF) requests," in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, pp. 1–6, 2019.

[84] Z. Shaoping *et al.*, "Virtual network function instantiation and service function chaining mapping in wide area network," in *2016 IEEE/CIC International Conference on Communications in China (ICCC)*, pp. 1–6, 2016.

[85] M. Jalalitabar *et al.*, "Service function graph design and mapping for NFV with priority dependence," in *2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–5, 2016.

[86] 3GPP, "3GPP TS 23.501; Technical Specification Group Services and System Aspects; System architecture for the 5G System (5GS); Stage 2 (Release 17)." V17.2.0, September 2021.

[87] W. Diego, "Evolution Toward the Next Generation Radio Access Network," in *IFIP Networking Conference (Networking)*, pp. 685–685, 2020.

[88] L. Bonati *et al.*, "Intelligence and Learning in O-RAN for Data-Driven NextG Cellular Networks," *IEEE Communications Magazine*, vol. 59, no. 10, pp. 21–27, 2021.

[89] E. Zeydan *et al.*, "Service based virtual RAN architecture for next generation cellular systems," *IEEE Access*, vol. 10, pp. 9455–9470, 2022.

[90] L. M. P. Larsen *et al.*, "A survey of the functional splits proposed for 5G mobile crosshaul networks," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 146–172, 2019.

[91] ITU-T, "ITU-T Series G, Supplement 66: 5G wireless fronthaul requirements in a passive optical network context." Edition 2.0, July 2019.

[92] J. S. Wey *et al.*, "5G wireless transport in a PON context: An overview," *IEEE Communications Standards Magazine*, vol. 4, no. 1, pp. 50–56, 2020.

[93] C. Chang *et al.*, "Slice orchestration for multi-service disaggregated ultra-dense RANs," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 70–77, 2018.

[94] F. Z. Yousaf *et al.*, "MANOaaS: A multi-tenant NFV MANO for 5G network slices," *IEEE Communications Magazine*, vol. 57, pp. 103–109, May 2019.

[95] J. Wang *et al.*, "A machine learning framework for resource allocation assisted by cloud computing," *IEEE Network*, vol. 32, no. 2, pp. 144–151, 2018.

[96] D. M. Gutierrez-Estevez *et al.*, "Artificial intelligence for elastic management and orchestration of 5G networks," *IEEE Wireless Communications*, vol. 26, no. 5, pp. 134–141, 2019.

[97] A. U. Rehman *et al.*, "Network functions virtualization: The long road to commercial deployments," *IEEE Access*, vol. 7, pp. 60439–60464, 2019.

[98] J. van de Belt *et al.*, "Defining and surveying wireless link virtualization and wireless network virtualization," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1603–1627, 2017.

[99] VMWare, "Containers on virtual machines or bare metal? deploying and securely managing containerized applications at scale." December 2018.

[100] Y. Wang *et al.*, "Network management and orchestration using artificial intelligence: Overview of ETSI ENI," *IEEE Communications Standards Magazine*, vol. 2, no. 4, pp. 58–65, 2018.

[101] D. Gligoroski *et al.*, "Expanded Combinatorial Designs as Tool to Model Network Slicing in 5G," *IEEE Access*, vol. 7, pp. 54879–54887, 2019.

[102] ETSI, "ETSI GS NFV 003 V1.3.1, Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV." January 2018.

[103] F. B. Lopes *et al.*, "VNFAccel: An FPGA-based Platform for Modular VNF Components Acceleration," in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 250–258, 2021.

[104] M. Pattaranantakul *et al.*, "A First Step Towards Security Extension for NFV Orchestrator," in *Proceedings of the ACM International Workshop on Security in Software Defined Networks and Network Function Virtualization*, (New York, NY, USA), pp. 25–30, Association for Computing Machinery, 2017.

[105] R. Mijumbi *et al.*, "Topology-Aware Prediction of Virtual Network Function Resource Requirements," *IEEE Transactions on Network and Service Management*, vol. 14, no. 1, pp. 106–120, 2017.

[106] ETSI, "ETSI GR NFV-SEC 009 V1.2.1, Network Functions Virtualisation (NFV) Release 3; NFV Security; Report on use cases and technical approaches for multi-layer host administration." January 2017.

[107] V. Stocker *et al.*, "The state of network neutrality regulation," *ACM SIGCOMM Computer Communication Review*, vol. 50, no. 1, pp. 45–59, 2020.

[108] M. Otokura *et al.*, "Evolvable Virtual Network Function Placement Method: Mechanism and Performance Evaluation," *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 27–40, 2019.

[109] J. Cao *et al.*, "VNF-FG design and VNF placement for 5G mobile networks," *Science China Information Sciences*, vol. 60, no. 4, 2017.

[110] F. Alvarez *et al.*, "An Edge-to-Cloud Virtualized Multimedia Service Platform for 5G Networks," *IEEE Transactions on Broadcasting*, vol. 65, no. 2, pp. 369–380, 2019.

[111] H. Hantouti *et al.*, "Service Function Chaining in 5G and Beyond Networks: Challenges and Open Research Issues," *IEEE Network*, vol. 34, no. 4, pp. 320–327, 2020.

[112] E. Datsika *et al.*, "Software Defined Network Service Chaining for OTT Service Providers in 5G Networks," *IEEE Communications Magazine*, vol. 55, no. 11, pp. 124–131, 2017.

[113] V. A. Cunha *et al.*, "An SFC-enabled approach for processing SSL/TLS encrypted traffic in Future Enterprise Networks," in *IEEE Symposium on Computers and Communications (ISCC)*, pp. 01013–01019, 2018.

[114] T. A. Khan *et al.*, "GAN and DRL Based Intent Translation and Deep Fake Configuration Generation for Optimization," in *International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 347–352, 2020.

[115] I. Sarrigiannis *et al.*, "Online VNF Lifecycle Management in an MEC-Enabled 5G IoT Architecture," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4183–4194, 2020.

[116] J. Cao *et al.*, "VNF placement in hybrid NFV environment: Modeling and genetic algorithms," in *2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 769–777, 2016.

[117] D. Camps-Mur *et al.*, "5G-XHaul: A Novel Wireless-Optical SDN Transport Network to Support Joint 5G Backhaul and Fronthaul Services," *IEEE Communications Magazine*, vol. 57, no. 7, pp. 99–105, 2019.

[118] J. Xia *et al.*, "Optimized virtual network functions migration for NFV," in *2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 340–346, 2016.

[119] J. Zhang *et al.*, "Towards Virtual Machine Image Management for Persistent Memory," in *35th Symposium on Mass Storage Systems and Technologies (MSST)*, pp. 116–125, 2019.

[120] C. Xu *et al.*, "On Multiple Virtual NICs in Cloud Computing: Performance Bottleneck and Enhancement," *IEEE Systems Journal*, vol. 12, no. 3, pp. 2417–2427, 2018.

[121] Y. Nakajima *et al.*, "High-Performance vNIC Framework for Hypervisor-Based NFV with Userspace vSwitch," in *Fourth European Workshop on Software Defined Networks*, pp. 43–48, 2015.

[122] B. Tak *et al.*, "Block-Level Storage Caching for Hypervisor-Based Cloud Nodes," *IEEE Access*, vol. 9, pp. 88724–88736, 2021.

[123] O. Osanaiye *et al.*, "From Cloud to Fog Computing: A Review and a Conceptual Live VM Migration Framework," *IEEE Access*, vol. 5, pp. 8284–8300, 2017.

[124] J. Xu *et al.*, "Survivable virtual infrastructure mapping in virtualized data centers," in *2012 IEEE Fifth International Conference on Cloud Computing*, pp. 196–203, 2012.

[125] H. D. Chantre *et al.*, "Redundant Placement of Virtualized Network Functions for LTE Evolved Multimedia Broadcast Multicast Services," in *IEEE International Conference on Communications (ICC)*, pp. 1–7, 2017.

[126] B. Brik *et al.*, "Deep Learning for B5G Open Radio Access Network: Evolution, Survey, Case Studies, and Challenges," *IEEE Open Journal of the Communications Society*, vol. 3, pp. 228–250, 2022.

[127] M. A. Habibi *et al.*, "The structure of service level agreement of slice-based 5G network." in *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, 09-12 September, 2018, Bologna, Italy.

[128] M. Maule *et al.*, "5G RAN slicing: Dynamic single tenant radio resource orchestration for eMBB traffic within a multi-slice scenario," *IEEE Communications Magazine*, vol. 59, no. 3, pp. 110–116, 2021.

[129] A. Chagdali *et al.*, "Slice Function Placement Impact on the Performance of URLLC with Multi-Connectivity," *Computers*, vol. 10, no. 5, pp. 1–18, 2021.

[130] ETSI, "ETSI GR NFV-MAN 001; Network Functions Virtualisation (NFV); Management and Orchestration; Report on Management and Orchestration Framework." V1.2.1, December 2021.

[131] A. Alleg *et al.*, "Joint Diversity and Redundancy for Resilient Service Chain Provisioning," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 7, pp. 1490–1504, 2020.

[132] H.-G. Kim *et al.*, "Machine Learning-Based Method for Prediction of Virtual Network Function Resource Demands," in *IEEE Conference on Network Softwarization (NetSoft)*, pp. 405–413, 2019.

[133] O-RAN Alliance, "Working Group 1; Study on O-RAN Slicing; Technical Report; V02.00." Alfter, Germany, Apr. 2020.

[134] O-RAN Alliance, "Working Group 2; Non-RT RIC Architecture; Technical Specification; V02.01." Alfter, Germany, Oct. 2022.

[135] M. A. Habibi *et al.*, "Mapping the VNFs and VLs of a RAN Slice Onto Intelligent PoPs in Beyond 5G Mobile Networks," *IEEE Open J. Commun. Soc.*, vol. 3, pp. 670–704, 2022.

[136] 3GPP, "3GPP TR 28.801; Technical Specification Group Services and System Aspects; Telecommunication management; Study on Management and Orchestration of Network Slicing for Next Generation Network (Release 15); V15.1.0 ." January 2018.

[137] O-RAN Alliance, "Working Group 4 (Open Fronthaul Interfaces); Control, User and Synchronization Plane Specification; Technical Specification; V12.00." Alfter, Germany, Jun. 2023.

[138] ETSI, "ETSI GS NFV 006; Network Functions Virtualisation (NFV) Release 4; Management and Orchestration; Architectural Framework Specification; V4.4.1." Sophia Antipolis, France, December 2022.

[139] K. Alam *et al.*, "A comprehensive tutorial and survey of O-RAN: Exploring slicing-aware architecture, deployment options, use cases, and challenges," *IEEE Communications Surveys & Tutorials*, vol. 28, pp. 1637–1678, 2026.

[140] K. Samdanis *et al.*, "AI/ML Service Enablers and Model Maintenance for Beyond 5G Networks," *IEEE Net.*, vol. 37, no. 5, pp. 162–172, 2023.

[141] M. Q. Hamdan *et al.*, "Recent Advances in Machine Learning for Network Automation in the O-RAN," *Sensors*, vol. 23, no. 21, 2023.

[142] P. H. Masur *et al.*, "Artificial Intelligence in Open-Radio Access Network," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 37, no. 9, pp. 6–15, 2022.

[143] F. Rezazadeh *et al.*, "Sliceops: Explainable mlops for streamlined automation-native 6G networks," *IEEE Wirel. Commun.*, pp. 1–7, 2024.

[144] A. Arnaz *et al.*, "Toward Integrating Intelligence and Programmability in Open Radio Access Networks: A Comprehensive Survey," *IEEE Access*, vol. 10, pp. 67747–67770, 2022.

[145] X. Lin *et al.*, "Embracing AI in 5G-Advanced Toward 6G: A Joint 3GPP and O-RAN Perspective," *IEEE Commun. Mag.*, vol. 7, no. 4, pp. 76–83, 2023.

[146] Q. Sun *et al.*, "Intelligent RAN Automation for 5G and Beyond," *IEEE Wirel. Commun.*, vol. 31, no. 1, pp. 94–102, 2024.

[147] Hoejoo Lee *et al.*, "Hosting AI/ML Workflows on O-RAN RIC Platform," in *IEEE Globecom Workshops*, pp. 1–6, 2020.

[148] O-RAN Alliance, "WG 2; AI/ML Workflow Description and Requirements; Technical Report; V01.03." Alfter, Germany, Oct. 2021.

[149] ETSI, "ETSI GR NFV-IFA 041; Network Functions Virtualisation (NFV) Release 4; Management and Orchestration; Report on Enabling Autonomous Management in NFV-MANO." Sophia Antipolis, France, Aug. 2021.

[150] 3GPP TSG SA5, "3GPP TR 28.809; Technical Specification Group Services and System Aspects; Management and Orchestration; Study on Enhancement of Management Data Analytics (MDA); (Release 17); V17.0.0 ." Mar. 2021.

[151] B. Tang *et al.*, "AI Testing Framework for Next-G O-RAN Networks: Requirements, Design, and Research Opportunities," *IEEE Wirel. Commun.*, vol. 30, no. 1, pp. 70–77, 2023.

[152] A. Giannopoulos *et al.*, "Supporting Intelligence in Disaggregated Open Radio Access Networks: Architectural Principles, AI/ML Workflow, and Use Cases," *IEEE Access*, vol. 10, pp. 39580–39595, 2022.

[153] S. D'Oro *et al.*, "OrchestRAN: Orchestrating Network Intelligence in the Open RAN," *IEEE Trans. Mob. Comput.*, pp. 1–16, 2023.

[154] 3GPP TSG SA5, "3GPP TS 28.533; Technical Specification Group Services and System Aspects; Management and Orchestration; Architecture Framework; (Release 18); V18.1.0 ." Mar. 2024.

[155] 3GPP TSG SA5, "3GPP TS 28.530; Technical Specification Group Services and System Aspects; Management and orchestration; Concepts, use cases and requirements (Release 18) V18.0.0 ." Dec. 2023.

[156] 3GPP TSG SA5, "3GPP TS 28.531; Technical Specification Group Services and System Aspects; Management and orchestration; Provisioning; (Release 18) V18.5.0 ." Mar. 2024.

[157] M. A. Garcia-Martin *et al.*, "Network Automation and Data Analytics in 3GPP 5G Systems," *IEEE Net.*, pp. 1–1, 2023.

[158] A. Imran, "Challenges in 5G: How to Empower SON with Big Data for Enabling 5G," *IEEE Net.*, vol. 28, no. 6, pp. 27–33, 2014.

[159] E. Pateromichelakis *et al.*, "End-to-End Data Analytics Framework for 5G Architecture," *IEEE Access*, vol. 7, pp. 40295–40312, 2019.

[160] R. Ferrus *et al.*, "Data Analytics Architectural Framework for Smarter Radio Resource Management in 5G Radio Access Networks," *IEEE Commun. Mag.*, vol. 58, no. 5, pp. 98–104, 2020.

[161] ETSI, "ETSI GS NFV-IFA 007; Network Functions Virtualisation (NFV) Release 4; Management and Orchestration; Or-Vnfm reference point - Interface and Information Model Specification." Sophia Antipolis, France, Mar. 2023.

[162] ETSI, "ETSI GR NFV-IFA 046; Network Functions Virtualisation (NFV) Releas.e 4; Network Functions Virtualisation (NFV) Release 5; Architectural Framework; Report on NFV support for virtualisation of RAN." Sophia Antipolis, France, May. 2023.

[163] ETSI, "ETSI GR NFV-MAN 001; Network Functions Virtualisation (NFV); Management and Orchestration; Report on Management and Orchestration Framework." Sophia Antipolis, France, Dec. 2021.

[164] M. A. Habibi *et al.*, "Toward an Open, Intelligent, and End-to-End Architectural Framework for Network Slicing in 6G Communication Systems," *IEEE Open J. Commun. Soc.*, vol. 4, pp. 1615–1658, 2023.

[165] ETSI, "ETSI GR NFV-IFA 047; Network Functions Virtualisation (NFV) Release 4; Management and Orchestration; Management data analytics Service Interface and Information Model Specification." Sophia Antipolis, France, Mar. 2023.

[166] "ETSI NFV Release 5 Kicks off with Increased Support for Cloud-enabled Deployment," 2021.

[167] Y. Roh *et al.*, "A Survey on Data Collection for Machine Learning: A Big Data - AI Integration Perspective," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 4, pp. 1328–1347, 2021.

[168] M. Liu *et al.*, "Task-Oriented ML/DL Library Recommendation Based on a Knowledge Graph," *IEEE Trans. Softw. Eng.*, vol. 49, no. 8, pp. 4081–4096, 2023.

[169] C. Chai *et al.*, "Data Management for Machine Learning: A Survey," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 5, pp. 4646–4667, 2023.

[170] M. Polese *et al.*, "Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges," *IEEE Commun. Surv. Tutor.*, vol. 25, no. 2, pp. 1376–1411, 2023.

[171] M. A. Habibi *et al.*, "Enabling Network and Service Programmability in 6G Mobile Communication Systems," in *IEEE FNWF*, pp. 320–327, 2022.

[172] M. P. *et al.*, "ColO-RAN: Developing Machine Learning-Based xApps for Open RAN Closed-Loop Control on Programmable Experimental Platforms," *IEEE Trans. Mob. Comput.*, vol. 22, no. 10, pp. 5787–5800, 2023.

[173] E. A. *et al.*, "Edge-AI Empowered Dynamic VNF Splitting in O-RAN Slicing: A Federated DRL Approach," *IEEE Comm. Lett.*, vol. 28, no. 2, pp. 318–322, 2024.

[174] M. Irazabal *et al.*, "TC-RAN: A Programmable Traffic Control Service Model for 5G/6G SD-RAN," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 2, pp. 406–419, 2024.

[175] K. Ali *et al.*, "Proactive VNF Scaling and Placement in 5G O-RAN Using ML," *IEEE Trans. Netw. Serv. Manag.*, vol. 21, no. 1, pp. 174–186, 2024.

[176] J. Martin-Perez *et al.*, "Choose, Not Hoard: Information-to-Model Matching for Artificial Intelligence in O-RAN," *IEEE Commun. Mag.*, vol. 61, no. 4, pp. 58–63, 2023.

[177] ETSI, "ETSI GS ENI 005; Experiential Networked Intelligence (ENI); System Architecture." Sophia Antipolis, France, Aug. 2021.

[178] A. Shabbir *et al.*, "Resilience of Federated Learning Against False Data Injection Attacks in Energy Forecasting," in *GECOST*, 2024. Miri Sarawak, Malaysia, pp. 245-249.

[179] S. Santra *et al.*, "Gradient Descent Effects on Differential Neural Architecture Search: A Survey," *IEEE Access*, vol. 9, pp. 89602–89618, 2021.

[180] T. Truong-Huu *et al.*, "Markov chain based algorithm for virtual network embedding in optical data centers," in *2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pp. 899–906, 2016.

[181] T. Wood *et al.*, "Sandpiper: Black-box and gray-box resource management for virtual machines," *Computer Networks*, vol. 53, no. 17, pp. 2923–2938, 2009.

[182] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, vol. 3, no. 3, pp. 210–229, 1959.

[183] L. Doring, *The Mathematics of Reinforcement Learning*. Mannheim, Germany: University of Mannheim, 2024.

[184] S. Zhao, *Mathematical Foundations of Reinforcement Learning*. New York, NY, USA: Springer, 2025.

[185] X. Foukas *et al.*, "Network slicing in 5G: Survey and challenges," *IEEE Communications Magazine*, vol. 55, pp. 94–100, May 2017.

[186] ITU-T, "G.7702: Architecture for SDN control of transport networks." Edition 1.0, March 2018.

[187] ONF, "TR-502: SDN architecture." Issue 1, June 2014.

[188] Ferrús *et al.*, "On the automation of RAN slicing provisioning: Solution framework and applicability examples," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, p. 167, Jun 2019.

[189] 3GPP, "Feasibility study on architecture enhancements for LTE support of V2X services (Release 14)." V1.1.0, July 2016.

[190] 3GPP, "Architecture enhancements to facilitate communications with packet data networks and applications (Release 15)." V15.5.0, July 2018.

[191] K. Samdanis *et al.*, "From network sharing to multi-tenancy: The 5g network slice broker," *IEEE Communications Magazine*, vol. 54, pp. 32–39, July 2016.

[192] 3GPP, ""5G System; Policy and Charging Control signalling flows and QoS parameter mapping (Release 15)"." V15.0.0, June 2018.

[193] 3GPP, "3GPP TS 38.300 V16.1.0, Technical Specification Group Radio Access Network; NR; NR and NG-RAN Overall Description; Stage 2. Release 16." March 2020.

[194] C. Sexton *et al.*, "Customization and trade-offs in 5G RAN slicing," *IEEE Communications Magazine*, vol. 57, pp. 116–122, April 2019.

Appendix C

Curriculum Vitae

Personal Details

Name: Mohammad Asif Habibi

Place of birth: Baghlan, Afghanistan

Education

January 2017 - Present Doctor of Engineering (Dr. -Ing.) in Electrical and Computer Engineering | Department of Electrical and Computer Engineering | Rheinland-Pfälzische Technische Universität (RPTU) | Kaiserslautern, Germany

September 2014 - May 2016 Masters of Science (M.Sc.) in System Engineering and Informatics | Department of Information Technology | Czech University of Life Sciences | Prague, Czech Republic

March 2008 - December 2011 Bachelor of Science (B.Sc.) in Information and Communication Technology | Institute of Information and Communication Technology | Kabul University | Kabul, Afghanistan